Babak Rahmani

Distributed Model Predictive Control Strategies for Constrained Multi-Agent Systems Moving in Uncertain Environments

This dissertation is submitted for the degree of Doctor of Philosophy

Department of Computer Science, Modeling, Electronics and Systems Engineering (DIMES)
University of Calabria

Supervisor: Prof. Giuseppe Franzè

May 30, 2021

I am dedicating this thesis to some beloved people who have meant and continue to mean so much to me. A special feeling of gratitude to my loving parents, for their love....

Preface

In the last years, coordination and control of multi-agent systems have assumed an increasing relevance in virtue of their capability because they are expected to be capable of performing to perform civilian and dangerous missions by navigating in formation within obstacle-populated environments. In particular, the state-of-the-art for unmanned vehicles concerns with three guidelines: Leader-Follower (LF) techniques, virtual structure schemes and behavioural methods. More recently, the research has pointed out its attention to stabilization and path following issues, whose integration can be obtained by considering two separate problems: 1) determine an admissible state trajectory for the (virtual) leader; 2) develop control architectures capable to keep within acceptable ranges the follower relative positions with respect to the current leader condition.

On the other hand, one of the major limitations relies on the difficulty to achieve a stable formation when the vehicle team moves in dynamically changing, unknown environments filled with obstacles. Under such circumstances, as the number of involved agents increases, the above mentioned control methods could fail: as a consequence, adequate implementations exploiting communication facilities amongst the vehicles have to be developed in order to be aware about the vehicle state condition and the actions pertaining to its teammates. Beyond this, another important challenge concerns with the active obstacle avoidance on the follower vehicles. In fact, it is important to combine formation planning and obstacle avoidance requirements on the follower path, because these vehicles must be also capable to comply with the prescribed team topology.

A novel distributed MPC scheme is here developed for autonomous multi-agents systems subject to input/state constraints, obstacle avoidance and formation requirements. Specifically, the group of agents is organized as a finite set of swarms and a leader-follower topology is imposed to these configurations with the aim to define a sort of priority among them. The key idea is as follows: the leader swarm (for the sake of simplicity consisting of a single agent) is in charge to compute the tube of trajectories towards the target

position and to detect possible obstacle occurrences within the unknown environment. Then, the follower swarm receives the hyper-ball where its father swarm lies and uses this information at each time instant as a local target. This reasoning repeats for the other swarms. Such a strategy is made viable in virtue of the following property concerning with the swarm kinematics: in a finite time each agent converges to a pre-assigned hyper-ball. Then, it is always possible to compute a reference kinematic state trajectory towards the hyper-ball of the father swarm along the platoon chain.

This thesis contents are organized in 5 chapters and 2 appendices. In Chap. 1, following a brief introduction, the robot modelling, notation and preliminaries are presented. Mobile robot kinematics and dynamics are also introduced. In Chap. 2, Modelling of multi-agent systems is presented in terms of graph theoretic methods. Model Based Predictive Control is presented in Chap. 3. Basic model predictive control schemes and Distributed model predictive control approaches are also discussed and illustrated. In Chap. 4, a Novel swarm-based distributed MPC architecture is developed for autonomous multi-agent systems subject to input/state constraints, obstacle avoidance and formation requirements. In Chap. 5, a laboratory experiment is used to evaluate the performance of the proposed DMPC-Swarm Algorithm. It is also included the conclusions and future research directions. Appendix A is devoted to fundamental definitions and descriptions and presents the notations and examples. Special care has been devoted to the selection of bibliographical references (more than 195) which are cited at the end of this dissertation in relation to the historical development of the field.

Finally, the author wish to acknowledge all those who have been helpful in the preparation of this thesis. **A special note of thanks** goes to Prof. Giuseppe Franzè for his punctual and critical advisor and supervising, as well as to Prof. Giuseppe Fedele and Antonio Bono for their contributions and comments on some sections.

Rende (CS), *Babak Rahmani*
May 2021

Contents

1

Introduction

An autonomous robot is a robot that performs behaviors or tasks with a high degree of autonomy (without external influence). Autonomous robotics is usually considered to be a subfield of artificial intelligence, robotics, and information engineering. Early versions were proposed and demonstrated by David L. Heiserman [151]. Autonomous robots are particularly desirable in fields such as spaceflight, household maintenance (such as cleaning), waste water treatment, and delivering goods and services. Some modern factory robots are "autonomous" within the strict confines of their direct environment.

It may not be that every degree of freedom exists in their surrounding environment, but the factory robot's workplace is challenging and can often contain chaotic, unpredicted variables. The exact orientation and position of the next object of work and (in the more advanced factories) even the type of object and the required task must be determined. This can vary unpredictably (at least from the robot's point of view).

1.1 Robot modelling, notation and preliminaries

One important area of robotics research is to enable the robot to cope with its environment whether this be on land, underwater, in the air, underground, or in space. Autonomous mobile robots have various applications in the field of industry, military and security environment. The problem of autonomous motion planning and control of wheeled mobile robots have attracted lot of research interest in the field of robotics. Consequently engineers working on design of mobile robots have proposed various drive mechanisms to drive such robots. However the most common way to build a mobile robot is to use two-wheel drive with differential steering and a free balancing wheel (castor). Controlling the two motors independently make such robots to have good manoeuvring and work well in indoor environment [123]. Mobile robots with such drive systems are a typical example of non-holonomic mechanisms due to the perfect rolling constraints on a wheel motion (no longitudinal or lateral

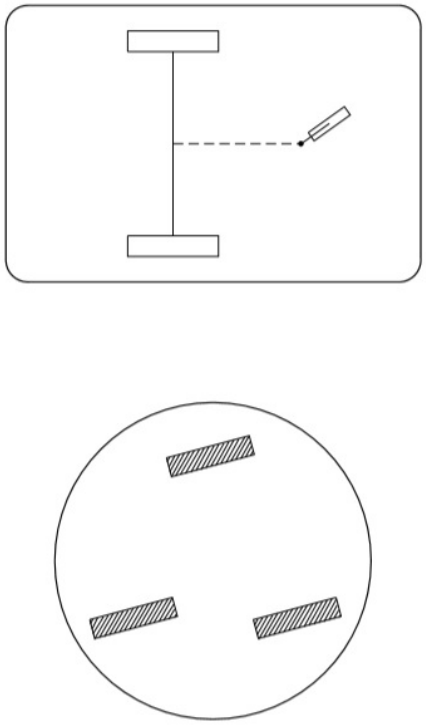

Fig. 1.1. A differential-drive mobile robot

slipping).

A differential wheeled robot is a mobile robot whose movement is based on two separately driven wheels placed on either side of the robot body [119]. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion.
In a *differential-drive* vehicle there are two fixed wheels with a common axis of rotation, and one or more caster wheels, typically smaller, whose function is to keep the robot statically balanced (Fig. 1.1) [118]. The two fixed wheels are separately controlled, in that different values of angular velocity may be arbitrarily imposed, while the caster wheel is passive. Such a robot can rotate on the spot (i.e., without moving the midpoint between the wheels), provided that the angular velocities of the two wheels are equal and opposite.

A vehicle with similar mobility is obtained using a synchro-drive kinematic

arrangement (Fig 1.1). This robot has three aligned steerable wheels which are synchronously driven by only two motors through a mechanical coupling, e.g., a chain or a transmission belt. The first motor controls the rotation of the wheels around the horizontal axis, thus providing the driving force (traction) to the vehicle. The second motor controls the rotation of the wheels around the vertical axis, hence affecting their orientation. Note that the heading of the chassis does not change during the motion. Often, a third motor is used in this type of robot to rotate independently the upper part of the chassis (a turret) with respect to the lower part. This may be useful to orient arbitrarily a directional sensor (e.g., a camera) or in any case to recover an orientation error.

In a tricycle vehicle (Fig 1.2) [121] there are two fixed wheels mounted on a rear axle and a steerable wheel in front. The fixed wheels are driven by a single motor which controls their traction,[1] while the steerable wheel is driven by another motor which changes its orientation, acting then as a steering device. Alternatively, the two rear wheels may be passive and the front wheel may provide traction as well as steering.

To balance the robot, additional wheels or casters may be added. If both the wheels are driven in the same direction and speed, the robot will go in a straight line. If both wheels are turned with equal speed in opposite directions, as is clear from the diagram shown, the robot will rotate about the central point of the axis [124]. Otherwise, depending on the speed of rotation and its direction, the center of rotation may fall anywhere on the line defined by the two contact points of the tires. While the robot is traveling in a straight line, the center of rotation is an infinite distance from the robot. Since the direction of the robot is dependent on the rate and direction of rotation of the two driven wheels, these quantities should be sensed and controlled precisely.

A differentially steered robot is similar to the differential gears used in automobiles in that both the wheels can have different rates of rotations, but unlike the differential gearing system, a differentially steered system will have both the wheels powered. Differential wheeled robots are used extensively in robotics, since their motion is easy to program and can be well controlled. Virtually all consumer robots on the market today use differential steering primarily for its low cost and simplicity.

[1] The distribution of the traction torque on the two wheels must take into account the fact that in general they move with different speeds. The mechanism which equally distributes traction is the differential.

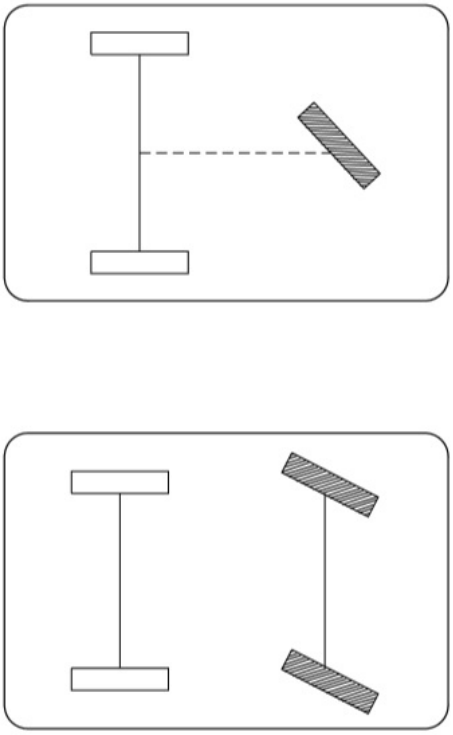

Fig. 1.2. A tricycle mobile robot and A car-like mobile robot

A mobile robot needs locomotion mechanisms that enable it to move unbounded throughout its environment. But there are a large variety of possible ways to move, and so the selection of a robot's approach to locomotion is an important aspect of mobile robot design. In the laboratory, there are research robots that can walk, jump, run, slide, skate, swim, fly, and, of course, roll. Most of these locomotion mechanisms have been inspired by their biological counterparts [124]. There is, however, one exception: the actively powered wheel is a human invention that achieves extremely high efficiency on flat ground. This mechanism is not completely foreign to biological systems. Our bipedal walking system can be approximated by a rolling polygon, with sides equal in length to the span of the step. As the step size decreases, the polygon approaches a circle or wheel. But nature did not develop a fully rotating, actively powered joint, which is the technology necessary for wheeled locomotion. Biological systems succeed in moving through a wide variety of harsh

environments [133].

Therefore it can be desirable to copy their selection of locomotion mechanisms. However, replicating nature in this regard is extremely difficult for several reasons. To begin with, mechanical complexity is easily achieved in biological systems through structural replication. Cell division, in combination with specialization, can readily produce a millipede with several hundred legs and several tens of thousands of individually sensed cilia [97]. In man-made structures, each part must be fabricated individually, and so no such economies of scale exist. Additionally, the cell is a microscopic building block that enables extreme miniaturization. With very small size and weight, insects achieve a level of robustness that we have not been able to match with human fabrication techniques. Finally, the biological energy storage system and the muscular and hydraulic activation systems used by large animals and insects achieve torque, response time, and conversion efficiencies that far exceed similarly scaled man-made systems.

Owing to these limitations, mobile robots generally locomote either using wheeled mechanisms, a well-known human technology for vehicles, or using a small number of articulated legs, the simplest of the biological approaches to locomotion. In general, legged locomotion requires higher degrees of freedom and therefore greater mechanical complexity than wheeled locomotion. Wheels, in addition to being simple, are extremely well suited to flat ground. As a biped walking robot, on flat surfaces wheeled locomotion is one to two orders of magnitude more efficient than legged locomotion. The railway is ideally engineered for wheeled locomotion because rolling friction is minimized on a hard and flat steel surface. But as the surface becomes soft, wheeled locomotion accumulates inefficiencies due to rolling friction whereas legged locomotion suffers much less because it consists only of point contacts with the ground [94].

1.1.1 Wheeled Mobile Robots

The wheel has been by far the most popular locomotion mechanism in mobile robotics and in man-made vehicles in general [108]. It can achieve very good efficiencies, and does so with a relatively simple mechanical implementation. In addition, balance is not usually a research problem in wheeled robot designs, because wheeled robots are almost always designed so that all wheels are in ground contact at all times. Thus, three wheels are sufficient to guarantee stable balance, although, as we shall see below, two-wheeled robots can

also be stable.

When more than three wheels are used, a suspension system is required to allow all wheels to maintain ground contact when the robot encounters uneven terrain. Instead of worrying about balance, wheeled robot research tends to focus on the problems of traction and stability, maneuverability, and control: can the robot wheels provide sufficient traction and stability for the robot to cover all of the desired terrain, and does the robot's wheeled configuration enable sufficient control over the velocity of the robot? As we shall see, there is a very large space of possible wheel configurations when one considers possible techniques for mobile robot locomotion [125]. We begin by discussing the wheel in detail, as there are a number of different wheel types with specific strengths and weaknesses. Then, we examine complete wheel configurations that deliver particular forms of locomotion for a mobile robot.

1.2 Robot dynamics

The main feature of mobile robots is the presence of a mobile base which allows the robot to move freely in the environment. Unlike manipulators, such robots are mostly used in service applications, where extensive, autonomous motion capabilities are required [120]. From a mechanical viewpoint, a mobile robot consists of one or more rigid bodies equipped with a locomotion system. This description includes the following two main classes of mobile robots:[2]

- *Wheeled* mobile robots typically consist of a rigid body (*base* or *chassis*) and a system of wheels which provide motion with respect to the ground. Other rigid bodies (*trailers*), also equipped with wheels, may be connected to the base by means of revolute joints.
- *Legged* mobile robots are made of multiple rigid bodies, interconnected by prismatic joints or, more often, by revolute joints. Some of these bodies form lower limbs, whose extremities (*feet*) periodically come in contact with the ground to realize locomotion. There is a large variety of mechanical structures in this class, whose design is often inspired by the study of living organisms (*biomimetic robotics*): they range from biped humanoids to hexapod robots aimed at replicating the biomechanical efficiency of insects.

There are four major wheel classes. They differ widely in their kinematics, and therefore the choice of wheel type has a large effect on the overall kinematics of the mobile robot. The standard wheel and the castor wheel have a

[2] Other types of mechanical locomotion systems are not considered here. Among these, it is worth mentioning tracked locomotion, very effective on uneven terrain, and undulatory locomotion, inspired by snake gaits, which can be achieved without specific devices.

primary axis of rotation and are thus highly directional. To move in a different direction, the wheel must be steered first along a vertical axis. The key difference between these two wheels is that the standard wheel can accomplish this steering motion with no side effects, as the center of rotation passes through the contact patch with the ground, whereas the castor wheel rotates around an offset axis, causing a force to be imparted to the robot chassis during steering. The spherical wheel is a truly omnidirectional wheel, often designed so that it may be actively powered to spin along any direction.

One mechanism for implementing this spherical design imitates the computer mouse, providing actively powered rollers that rest against the top surface of the sphere and impart rotational force. Regardless of what wheel is used, in robots designed for all-terrain environments and in robots with more than three wheels, a suspension system is normally required to maintain wheel contact with the ground.

One of the simplest approaches to suspension is to design flexibility into the wheel itself. For instance, in the case of some four-wheeled indoor robots that use castor wheels, manufacturers have applied a deformable tire of soft rubber to the wheel to create a primitive suspension. Of course, this limited solution cannot compete with a sophisticated suspension system in applications where the robot needs a more dynamic suspension for significantly non flat terrain. The choice of wheel types for a mobile robot is strongly linked to the choice of wheel arrangement, or wheel geometry.

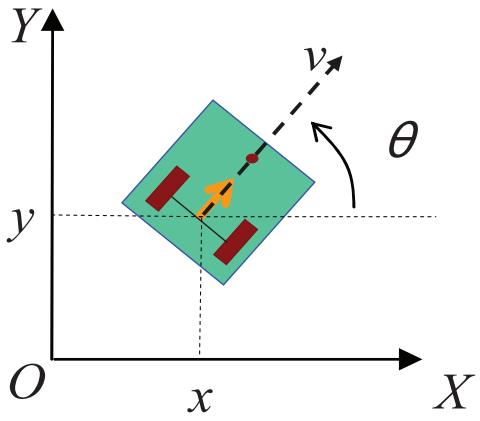

Fig. 1.3. Robot model

The mobile robot designer must consider these two issues simultaneously when designing the locomoting mechanism of a wheeled robot. Why do wheel

type and wheel geometry matter? Three fundamental characteristics of a robot are governed by these choices: maneuverability, controllability, and stability [101]. Unlike automobiles, which are largely designed for a highly standardized environment (the road network), mobile robots are designed for applications in a wide variety of situations. Automobiles all share similar wheel configurations because there is one region in the design space that maximizes maneuverability, controllability, and stability for their standard environment: the paved roadway. However, there is no single wheel configuration that maximizes these qualities for the variety of environments faced by different mobile robots.So you will see great variety in the wheel configurations of mobile robots.

In fact, few robots use the Ackerman wheel configuration of the automobile because of its poor maneuverability, with the exception of mobile robots designed for the road system. Surprisingly, the minimum number of wheels required for static stability is two. As shown above, a two-wheel differential-drive robot can achieve static stability if the center of mass is below the wheel axle. Cye is a commercial mobile robot that uses this wheel configuration [112]. However, under ordinary circumstances such a solution requires wheel diameters that are impractically large. Dynamics can also cause a two-wheeled robot to strike the floor with a third point of contact, for instance, with sufficiently high motor torques from standstill.

Conventionally, static stability requires a minimum of three wheels, with the additional caveat that the center of gravity must be contained within the triangle formed by the ground contact points of the wheels. Stability can be further improved by adding more wheels, although once the number of contact points exceeds three, the hyperstatic nature of the geometry will require some form of flexible suspension on uneven terrain. In the research community, other classes of mobile robots are popular which achieve high maneuverability, only slightly inferior to that of the omnidirectional configurations.

In such robots, motion in a particular direction may initially require a rotational motion. With a circular chassis and an axis of rotation at the center of the robot, such a robot can spin without changing its ground footprint. The most popular such robot is the two-wheel differential-drive robot where the two wheels rotate around the center point of the robot [117]. One or two additional ground contact points may be used for stability, based on the application specifics. In contrast to the above configurations, consider the Ackerman steering configuration common in automobiles. Such a vehicle typically has a turning diameter that is larger than the car. Furthermore, for such a vehicle to move sideways requires a parking maneuver consisting of repeated changes in direction forward and backward. Nevertheless, Ackerman steering geometries have been especially popular in the hobby robotics market, where a robot can be built by starting with a remote control racecar kit and adding

sensing and autonomy to the existing mechanism. In addition, the limited maneuverability of Ackerman steering has an important advantage: its directionality and steering geometry provide it with very good lateral stability in high-speed turns.

There is generally an inverse correlation between controllability and maneuverability. For example, the omnidirectional designs such as the four-castor wheel configuration require significant processing to convert desired rotational and translational velocities to individual wheel commands [118]. Furthermore, such omnidirectional designs often have greater degrees of freedom at the wheel. For instance, the Swedish wheel has a set of free rollers along the wheel perimeter. These degrees of freedom cause an accumulation of slippage, tend to reduce dead-reckoning accuracy and increase the design complexity.

Controlling an omnidirectional robot for a specific direction of travel is also more difficult and often less accurate when compared to less maneuverable designs. For example, an Ackerman steering vehicle can go straight simply by locking the steerable wheels and driving the drive wheels [103]. In a differential-drive vehicle, the two motors attached to the two wheels must be driven along exactly the same velocity profile, which can be challenging considering variations between wheels, motors, and environmental differences. With fourwheel omnidrive, such as the Uranus robot, which has four Swedish wheels, the problem is even harder because all four wheels must be driven at exactly the same speed for the robot to travel in a perfectly straight line. In summary, there is no "ideal" drive configuration that simultaneously maximizes stability, maneuverability, and controllability. Each mobile robot application places unique constraints on the robot design problem, and the designer's task is to choose the most appropriate drive configuration possible from among this space of compromises.

1.2.1 Mobile Robot Kinematics

Kinematics is the most basic study of how mechanical systems behave. In mobile robotics, we need to understand the mechanical behavior of the robot both in order to design appropriate mobile robots for tasks and to understand how to create control software for an instance of mobile robot hardware. Of course, mobile robots are not the first complex mechanical systems to require such analysis. Robot manipulators have been the subject of intensive study for more than thirty years[140]. In some ways, manipulator robots are much more complex than early mobile robots: a standard welding robot may have five or more joints, whereas early mobile robots were simple differential-drive machines. In recent years, the robotics community has achieved a fairly complete understanding of the kinematics and even the dynamics (i.e., relating to

force and mass) of robot manipulators.

The mobile robotics community poses many of the same kinematic questions as the robot manipulator community. A manipulator robot's workspace is crucial because it defines the range of possible positions that can be achieved by its end effector relative to its fixture to the environment. A mobile robot's workspace is equally important because it defines the range of possible poses that the mobile robot can achieve in its environment [128]. The robot arm's controllability defines the manner in which active engagement of motors can be used to move from pose to pose in the workspace. Similarly, a mobile robot's controllability defines possible paths and trajectories in its workspace. Robot dynamics places additional constraints on workspace and trajectory due to mass and force considerations. The mobile robot is also limited by dynamics; for instance, a high center of gravity limits the practical turning radius of a fast, car-like robot because of the danger of rolling. But the chief difference between a mobile robot and a manipulator arm also introduces a significant challenge for position estimation.

A manipulator has one end fixed to the environment. Measuring the position of an arm's end effector is simply a matter of understanding the kinematics of the robot and measuring the position of all intermediate joints [147]. The manipulator's position is thus always computable by looking at current sensor data. But a mobile robot is a self-contained automaton that can wholly move with respect to its environment. There is no direct way to measure a mobile robot's position instantaneously. Instead, one must integrate the motion of the robot over time. Add to this the inaccuracies of motion estimation due to slippage and it is clear that measuring a mobile robot's position precisely is an extremely challenging task.

The process of understanding the motions of a robot begins with the process of describing the contribution each wheel provides for motion. Each wheel has a role in enabling the whole robot to move. By the same token, each wheel also imposes constraints on the robot's motion; for example, refusing to skid laterally. In the following section, we introduce notation that allows expression of robot motion in a global reference frame as well as the robot's local reference frame [88]. Then, using this notation, we demonstrate the construction of simple forward kinematic models of motion, describing how the robot as a whole moves as a function of its geometry and individual wheel behavior. Next, we formally describe the kinematic constraints of individual wheels, and then combine these kinematic constraints to express the whole robot's kinematic constraints. With these tools, one can evaluate the paths and trajectories that define the robot's maneuverability.

1.2.2 Kinematic Models and Constraints

Deriving a model for the whole robot's motion is a bottom-up process. Each individual wheel contributes to the robot's motion and, at the same time, imposes constraints on robot motion. Wheels are tied together based on robot chassis geometry, and therefore their constraints combine to form constraints on the overall motion of the robot chassis. But the forces and constraints of each wheel must be expressed with respect to a clear and consistent reference frame. This is particularly important in mobile robotics because of its self-contained and mobile nature; a clear mapping between global and local frames of reference is required. We begin by defining these reference frames formally, then using the resulting formalism to annotate the kinematics of individual wheels and whole robots. Throughout this process we draw extensively on the notation and terminology presented in Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots.

In the following, the kinematic models of two wheeled vehicles of particular interest will be analyzed in detail. A large part of the existing mobile robots have a kinematic model that is equivalent to one of these two.

Unicycle

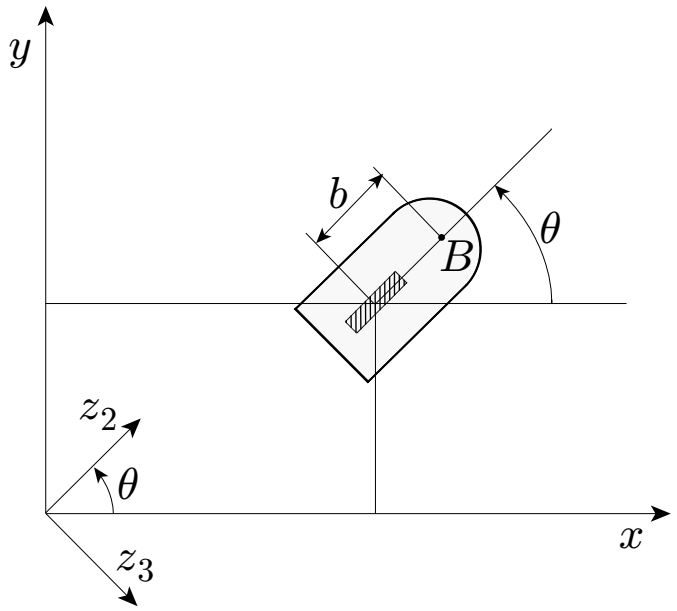

Fig. 1.4. Generalized coordinates for a unicycle

A *unicycle* is a vehicle with a single orientable wheel [143]. Its configuration is completely described by $\boldsymbol{q} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$, where (x, y) are the Cartesian coordinates of the contact point of the wheel with the ground (or equivalently, of the wheel centre) and θ is the orientation of the wheel with respect to the x axis (see Fig. 1.4).

As already seen in, the pure rolling constraint for the wheel is expressed as

$$\dot{x}\sin\theta - \dot{y}\cos\theta = \begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \dot{\boldsymbol{q}} = 0, \tag{1.1}$$

entailing that the velocity of the contact point is zero in the direction orthogonal to the sagittal axis of the vehicle. The line passing through the contact point and having such direction is therefore called *zero motion line*. Consider the matrix

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{g}_1(\boldsymbol{q}) & \boldsymbol{g}_2(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}, \tag{1.2}$$

whose columns $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$ are, for each , a basis of the null space of the matrix associated with the Pfaffian constraint. All the admissible generalized velocities at are therefore obtained as a linear combination of $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$. The kinematic model of the unicycle is then

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega, \tag{1.3}$$

where the inputs v and ω have a clear physical interpretation. In particular, v is the *driving velocity*, i.e., the modulus[3] (with sign) of the contact point velocity vector, whereas the *steering velocity* ω is the wheel angular speed around the vertical axis. The Lie bracket of the two input vector fields is

$$[\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{q}) = \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \end{bmatrix}, \tag{1.4}$$

that is always linearly independent from $\boldsymbol{g}_1(\boldsymbol{q})$, $\boldsymbol{g}_2(\boldsymbol{q})$. Therefore, the iterative procedure for building the accessibility distribution $\Delta_{\mathcal{A}}$ ends with

$$\dim \Delta_{\mathcal{A}} = \dim \Delta_2 = \dim \operatorname{span} \{\boldsymbol{g}_1, \boldsymbol{g}_2, [\boldsymbol{g}_1, \boldsymbol{g}_2]\} = 3.$$

This indicates that the unicycle is controllable with degree of nonholonomy $\kappa = 2$, and that constraint (1.1) is nonholonomic — the same conclusion reached by applying the integrability condition. A unicycle in the strict sense

[3] Note that v is given by the angular speed of the wheel around its horizontal axis multiplied by the wheel radius.

(i.e., a vehicle equipped with a single wheel) is a robot with a serious problem of balance in static conditions. However, there exist vehicles that are kinematically equivalent to a unicycle but more stable from a mechanical viewpoint. Among these, the most important are the *differential drive* and the *synchro drive* vehicles, had been already introduced.

For the differential drive mobile robot of Fig 1.3, denote by (x, y) the Cartesian coordinates of the midpoint of the segment joining the two wheel centres, and by θ the common orientation of the fixed wheels (hence, of the vehicle body). Then, the kinematic model(1.3) of the unicycle also applies to the differential drive vehicle, provided that the driving and steering velocities v and ω are expressed as a function of the actual velocity inputs, i.e., the angular speeds ω_R and ω_L of the right and left wheel, respectively. Simple arguments can be used to show that there is a one-to-one correspondence between the two sets of inputs:

$$v = \frac{r\,(\omega_R + \omega_L)}{2} \qquad \omega = \frac{r\,(\omega_R - \omega_L)}{d}, \tag{1.5}$$

where r is the radius of the wheels and d is the distance between their centres.

The equivalence with the kinematic model(3) is even more straightforward for the *synchro drive* mobile robot of Fig.2, whose control inputs are indeed the driving velocity v and the steering velocity ω, that are common to the three orientable wheels. The Cartesian coordinates (x, y) may represent in this case any point of the robot (for example, its centroid), while θ is the common orientation of the wheels. Note that, unlike a differential drive vehicle, the orientation of the body of a synchro drive vehicle never changes, unless a third actuator is added for this specific purpose.

Bicycle

Consider now a *bicycle*, i.e., a vehicle having an orientable wheel and a fixed wheel arranged as in Fig. 1.5. A possible choice for the generalized coordinates is $\boldsymbol{q} = \begin{bmatrix} x & y & \theta & \phi \end{bmatrix}^T$, where (x, y) are the Cartesian coordinates of the contact point between the rear wheel and the ground (i.e., of the rear wheel centre), θ is the orientation of the vehicle with respect to the x axis, and ϕ is the steering angle of the front wheel with respect to the vehicle.

The motion of the vehicle is subject to two pure rolling constraints, one for each wheel:

$$\dot{x}_f \sin(\theta + \phi) - \dot{y}_f \cos(\theta + \phi) = 0 \tag{1.6}$$

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0, \tag{1.7}$$

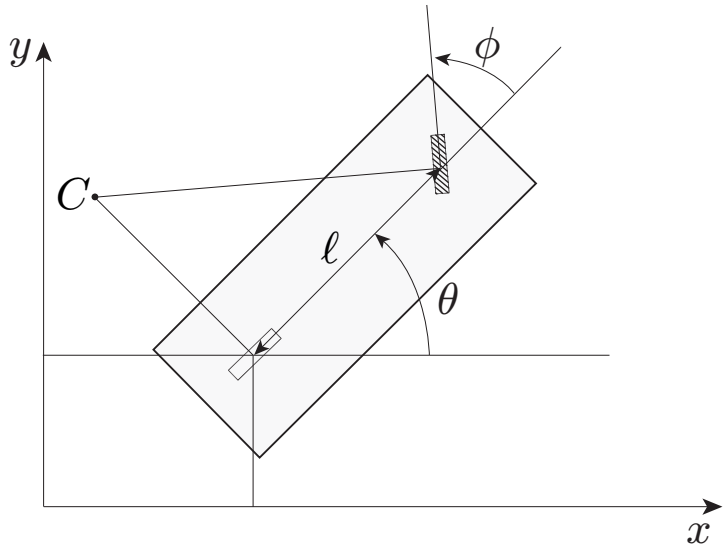

Fig. 1.5. Generalized coordinates and instantaneous centre of rotation for a bicycle

where (x_f, y_f) is the Cartesian position of the centre of the front wheel. The geometric meaning of these constraints is obvious: the velocity of the centre of the front wheel is zero in the direction orthogonal to the wheel itself, while the velocity of the centre of the rear wheel is zero in the direction orthogonal to the sagittal axis of the vehicle. The zero motion lines of the two wheels meet at a point C called *instantaneous centre of rotation* (Fig. 1.5), whose position depends only on (and changes with) the configuration of the bicycle. Each point of the vehicle body then moves instantaneously along an arc of circle with centre in C (see also Problem in [116]).

Using the rigid body constraint

$$\begin{aligned} x_f &= x + \ell \cos\theta \\ y_f &= y + \ell \sin\theta, \end{aligned}$$

where ℓ is the distance between the wheels, constraint (1.6) can be rewritten as

$$\dot{x} \sin(\theta + \phi) - \dot{y} \cos(\theta + \phi) - \ell \dot{\theta} \cos\phi = 0. \tag{1.8}$$

The matrix associated with the Pfaffian constraints (1.7), (1.8) is then

$$\boldsymbol{A}^T(\boldsymbol{q}) = \begin{bmatrix} \sin\theta & -\cos\theta & 0 & 0 \\ \sin(\theta+\phi) & -\cos(\theta+\phi) & -\ell\cos\phi & 0 \end{bmatrix} \tag{1.9}$$

with constant rank $k = 2$. The dimension of its null space is $n - k = 2$, and all the admissible velocities at may be written as a linear combination of a basis of $\mathcal{N}(AT(q))$, for example

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi/\ell \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2 \tag{1.10}$$

Since the front wheel is orientable, it is immediate to set $u_2 = \omega$, where ω is the *steering* velocity. The expression of u_1 depends instead on how the vehicle is driven.

If the bicycle has *front-wheel drive*, one has directly $u_1 = v$, where v is the *driving* velocity of the front wheel. The corresponding kinematic model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi/\ell \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{1.11}$$

Denoting by $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$ the two input vector fields, simple computations give

$$\boldsymbol{g}_3(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{q}) = \begin{bmatrix} \cos\theta\sin\phi \\ \sin\theta\sin\phi \\ -\cos\phi/\ell \\ 0 \end{bmatrix} \quad \boldsymbol{g}_4(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_3](\boldsymbol{q}) = \begin{bmatrix} -\sin\theta/\ell \\ \cos\theta/\ell \\ 0 \\ 0 \end{bmatrix},$$

both linearly independent from $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$. Hence, the iterative procedure for building the accessibility distribution $\Delta_{\mathcal{A}}$ ends with

$$\dim \Delta_{\mathcal{A}} = \dim \Delta_3 = \dim \operatorname{span}\{g_1, g_2, g_3, g_4\} = 4.$$

This means that the front-wheel drive bicycle is controllable with degree of nonholonomy $\kappa = 3$, and constraints (1.6), (1.7) are (completely) nonholonomic.

The kinematic model of a bicycle with *rear-wheel drive* can be derived by noting that in this case the first two equations must coincide with those of the unicycle model (1.3). It is then sufficient to set $u_1 = v/\cos\phi$ to obtain

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{1.12}$$

where v is the *driving* velocity of the rear wheel.[4] In this case, one has

$$g_3(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{q}) = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\ell \cos^2 \phi} \\ 0 \end{bmatrix} \quad \boldsymbol{g}_4(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_3](\boldsymbol{q}) = \begin{bmatrix} -\frac{\sin\theta}{\ell \cos^2 \phi} \\ \frac{\cos\theta}{\ell \cos^2 \phi} \\ 0 \\ 0 \end{bmatrix},$$

again linearly independent from $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$. Hence, the rear-wheel drive bicycle is also controllable with degree of nonholonomy $\kappa = 3$.

Like the unicycle, the bicycle is also unstable in static conditions. Kinematically equivalent vehicles that are mechanically balanced are the *tricycle* and the *car-like* robot, the kinematic model is given by (1.11) or by(13) depending on the wheel drive being on the front or the rear wheels. In particular, (x, y) are the Cartesian coordinates of the midpoint of the rear wheel axle, θ is the orientation of the vehicle, and ϕ is the steering angle.

1.2.3 Representing robot position

Throughout this analysis we model the robot as a rigid body on wheels, operating on a horizontal plane. The total dimensionality of this robot chassis on the plane is three, two for position in the plane and one for orientation along the vertical axis, which is orthogonal to the plane [113]. Of course, there are additional degrees of freedom and flexibility due to the wheel axles, wheel steering joints, and wheel castor joints. However by robot chassis we refer only to the rigid body of the robot, ignoring the joints and degrees of freedom internal to the robot and its wheels.

Consider autonomous vehicles whose dynamics is described by a differential drive model with two driving wheels and a single caster, as shown in Fig. A.1. A differential drive robot is a typical nonholonomic wheeled vehicle, which has two rear drive wheels and a front castor for body support. It is assumed that the motion of mobile robot cannot slip laterally so that the translational velocity is in the direction of heading, i.e. a pure rolling contact between the wheels and the ground [87]. The velocity of the two rear wheels (v_l and v_r) are used to impose the translation ($v = (v_l + v_r)/2$) and angular ($\omega = (v_r - v_l)/B$) speeds of the robot (B is the wheelbase). The robot pose is described by its position (p_x, p_y), the midpoint of the rear axis of the robot, and its orientation (θ). Then, the kinematics equation is

[4] Note that the kinematic model is no longer valid for $\phi = \pm\pi/2$, where the first vector field is not defined. This corresponds to the mechanical jam in which the front wheel is orthogonal to the sagittal axis of the vehicle.

$$\begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \\ 0 \end{bmatrix} v(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega(t) \tag{1.13}$$

where maximum linear and angular velocities are prescribed:

$$\text{ì}|v(t)| \leq V_{MAX}, \;\; |\omega(t)| \leq W_{MAX}, \; \forall t. \tag{1.14}$$

Notice that the kinematic model (1.24) is nonintegratable and, as a consequence, kinematics constraints can not be converted into geometrical requirements. Moreover since the number of control variables is less than the number of state variables, a nonholonomic constraint holds and a continuous time-invariant feedback control law cannot be used. On the other hand, since the accessibility rank condition is globally satisfied, the model plant (1.24) is controllable by means of a nonlinear or time-varying controller.

Hereafter, we refer to (1.24) by means of its compact form:

$$\dot{x}(t) = F_p(x(t), u(t)) \tag{1.15}$$

where $x(t) = [p_x(t), \; p_y(t), \; \theta(t)]$ is the plant state and $u(t) = [v(t), \; \omega(t)]^T$ the command input. A Polytopic Linear Differential Inclusion (PLDI) of (1.25) can be derived by using the arguments outlined in. First, the plant is linearized around a given nominal solution $(\hat{x}(t), \; \hat{u}(t))$ as follows

$$\dot{\tilde{x}}(t) = A(\tilde{\theta}, \; \tilde{v})\, \tilde{x}(t) + B(\tilde{\theta}, \; \tilde{v})\, \tilde{u}(t) \tag{1.16}$$

where $\tilde{x}(t) := x(t) - \hat{x}(t)$, $\tilde{u}(t) := u(t) - \hat{u}(t)$

Then, by considering a maximum displacement on the nominal instantaneous azimuth angle $\hat{\theta}(t)$

$$\tilde{\theta} \in [\hat{\theta}(t) - \overline{\theta}, \; \hat{\theta}(t) + \overline{\theta}], \tag{1.17}$$

and under the following constraint on the nominal instantaneous linear velocity $\hat{v}(t)$

$$\tilde{v} \in [\hat{v}(t) - \overline{v}, \; \hat{v}(t) + \overline{v}], \tag{1.18}$$

we have that all the solutions of (1.25) are also solutions of the following parameter varying PLDI

$$\dot{\tilde{x}}(t) \in \left(\sum_{k=1}^{4} \lambda_k(t) A_k \right) \tilde{x}(t) + \left(\sum_{k=1}^{4} \lambda_k(t) B_k \right) \tilde{u}(t) \tag{1.19}$$

where (A_k, B_k), $k = 1, \ldots, 4$, are computed by evaluating the Jacobian matrix

of (1.25) along the vertices of the constraints (1.29)-(1.30). A robust approximation of (1.31) can derived by means of the following arguments. First, a finite set of nominal pairs $(\hat{\theta}_w(t),\ \hat{v}_w(t)),\ w = 1, \ldots, P$, equally spaced within a given time interval ΔT, is selected along the nominal robot path lying in the plane $(\hat{p}_x(t),\ \hat{p}_y(t))$. Then, a PLDI (1.31) corresponding to each operating point $(\hat{\theta}_w(t),\ \hat{v}_w(t))$ is computed. Hence, by evaluating the convex hull of all obtained matrix vertices [92]

$$\Omega = Conv\left\{\{[A_{k^w},\ B_{k^w}]\}_{k=1}^{4},\ w = 1, \ldots, M\right\} \tag{1.20}$$

a robust PLDI of the following form comes out

$$\dot{\tilde{x}}(t) \in \left(\sum_{k=1}^{\Gamma} \mu_k Vert\{\Omega\}_k\right)\ [\tilde{x}^T(t), \tilde{u}^T(t)]^T \tag{1.21}$$

1.3 State-Space Model

A state-space model of the mobile robot can be obtained by using the kinetic and potential energy expressions T and U

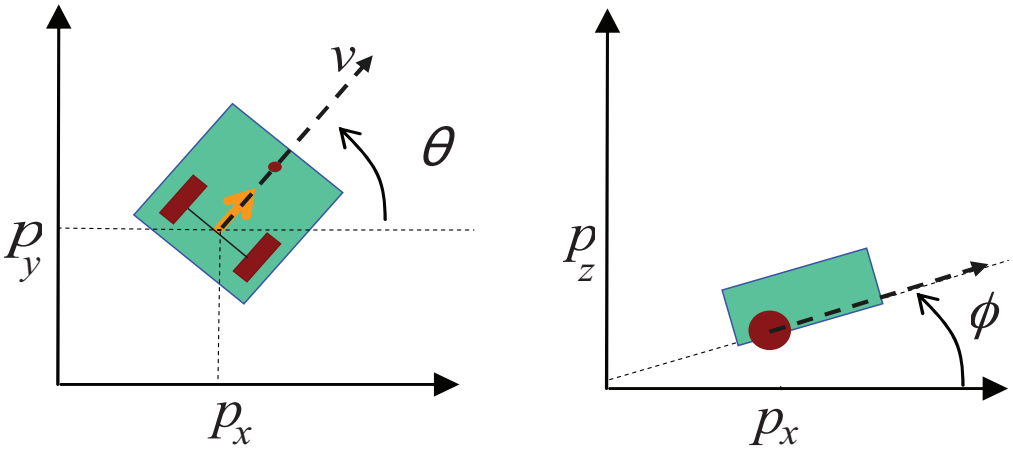

Fig. 1.6. State-Space Robot model

Table 1.1. Model parameters

Symbol	Meaning
M	chassis mass
M_L	Left wheel mass
M_R	Right wheel mass
J_L	moment of inertia of the Left wheel
J_R	moment of inertia of the Right wheel
J_θ	moment of inertia of the wheel w.r.t. the z-axis
R	wheels radius
D	distance between the wheels along the axle center
b	friction coefficient
ϕ	tilt angle of the incline plane

$$T := \frac{1}{2} M\, v^2 + J_W\, \frac{v^2}{R^2} + \frac{1}{2}\, J_\theta\, \dot{\theta}^2 + \frac{1}{2}\, \frac{J_W}{R^2}\, D^2\, \dot{\theta}^2 \tag{1.22}$$

$$U := -(M + 2\, M_W)\, g\, p_x\, \sin(\phi) - M_W\, g\, D \sin(\phi)\, |\sin(\theta)| \tag{1.23}$$

and by resorting to the classical Langrange approach:

$$\begin{cases} \dot{p}_x(t) = v(t)\, \cos(\theta(t)) \\ \dot{p}_y(t) = v(t)\, \sin(\theta(t)) \\ \dot{v}(t) = \dfrac{\frac{T_L(t)}{R} + \frac{T_R(t)}{R}}{M + \frac{J_W}{R^2}} - (M + 2\, M_W)\, g\, \sin(\phi(t)) - b\, v(t) \\ \dot{\theta}(t) = \omega(t) \\ \dot{\omega}(t) = \dfrac{\frac{D}{2}\left(\frac{T_L(t)}{R} - \frac{T_R(t)}{R}\right) + M_W\, g\, D\, \sin(\phi(t))\, \cos(\theta(t))\, \frac{\sin(\theta(t))}{|\sin(\theta(t))|}}{J_\theta + \frac{J_W}{R^2}\, D^2} \end{cases} \tag{1.24}$$

where $x(t) = [p_x(t),\ p_y(t),\ v(t),\ \theta(t),\ \omega(t)]$ is the plant state,

$u(t) = [T_L(t),\ T_R(t)]^T$ the command input, $M_W = M_L = M_R$ and $J_W = J_L = J_R$. Hereafter, we refer to (1.24) by means of its compact form:

$$\dot{x}(t) = F_p(x(t)) + B_p(x(t))u(t) \tag{1.25}$$

A Polytopic Linear Differential Inclusion (PLDI) of (1.25) can be derived by using the arguments outlined in [188]. First, the plant is linearized around a given nominal solution $(\hat{x}(t),\ \hat{u}(t))$ as follows

$$\dot{\tilde{x}}(t) = A(\tilde{\theta},\ \tilde{v})\,\tilde{x}(t) + B(\tilde{\theta})\,\tilde{u}(t) \tag{1.26}$$

where $\tilde{x}(t) := x(t) - \hat{x}(t)$, $\tilde{u}(t) := u(t) - \hat{u}(t)$ and

$$A(\tilde{\theta}\ ,\tilde{v}) = \begin{pmatrix} 0\ 0 & \cos(\tilde{\theta}) & -\tilde{v}\ \sin(\tilde{\theta}) & 0 \\ 0\ 0 & \sin(\tilde{\theta}) & \tilde{v}\ \cos(\tilde{\theta}) & 0 \\ 0\ 0 & -b & 0 & 0 \\ 0\ 0 & 0 & 0 & 1 \\ 0\ 0 & 0 & \frac{D}{2}\frac{M_W g D \sin\phi[-\sin^6(\tilde{\theta})+\sin^2(\tilde{\theta})+\sin(\tilde{\theta})-1]}{\left(J_{\tilde{\theta}}+\frac{D^2}{2}\left(\frac{J_W}{R^2}+M_W\right)\right)\ \sin(\tilde{\theta})\,|\sin(\tilde{\theta})|} & 0 \end{pmatrix} \tag{1.27}$$

$$B(\tilde{\theta}) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\frac{1}{R}}{M+\frac{J_W}{R^2}} & \frac{\frac{1}{R}}{M+\frac{J_W}{R^2}} \\ 0 & 0 \\ \frac{\frac{D}{2R}}{J_{\tilde{\theta}}+\frac{D^2}{2}\left(\frac{J_W}{R^2}+M_W\right)} & \frac{-\frac{D}{2R}}{J_{\tilde{\theta}}+\frac{D^2}{2}\left(\frac{J_W}{R^2}+M_W\right)} \end{pmatrix} \tag{1.28}$$

Then, by considering a maximum displacement on the nominal instantaneous azimuth angle $\hat{\theta}(t)$

$$\tilde{\theta} \in [\hat{\theta}(t) - \overline{\theta},\ \hat{\theta}(t) + \overline{\theta}], \tag{1.29}$$

and under the following constraint on the nominal instantaneous linear velocity $\hat{v}(t)$

$$\tilde{v} \in [\hat{v}(t) - \overline{v},\ \hat{v}(t) + \overline{v}], \tag{1.30}$$

we have that all the solutions of (1.25) are also solutions of the following parameter varying PLDI

$$\dot{\tilde{x}}(t) \in \left(\sum_{k=1}^{4} \lambda_k(t) A_k\right) \tilde{x}(t) + \left(\sum_{k=1}^{4} \lambda_k(t) B_k\right) \tilde{u}(t) \tag{1.31}$$

where (A_k, B_k), $k = 1, \ldots, 4$, are computed by evaluating the Jacobian matrix (1.27) along the vertices of the constraints (1.29)-(1.30). A robust approximation of (1.31) can derived by means of the following arguments. First, a finite set of nominal pairs $(\hat{\theta}_w(t),\, \hat{v}_w(t))$, $w = 1, \ldots, P$, equally spaced within a given time interval ΔT, is selected along the nominal robot path lying in the plane $(\hat{p}_x(t),\, \hat{p}_y(t))$. Then, a PLDI (1.31) corresponding to each operating point $(\hat{\theta}_w(t),\, \hat{v}_w(t))$ is computed. Hence, by evaluating the convex hull of all the obtained matrix vertices

$$\Omega = Conv\left\{\{[A_{k^w},\ B_{k^w}]\}_{k=1}^{4},\, w = 1, \ldots, M\right\} \tag{1.32}$$

a robust PLDI of the following form comes out

$$\dot{\tilde{x}}(t) \in \left(\sum_{k=1}^{\Gamma} \mu_k Vert\{\Omega\}_k\right)\, [\tilde{x}^T(t), \tilde{u}^T(t)]^T \tag{1.33}$$

with the input torque subject to point-wise constraints:

$$|\tilde{u}_i| < \tilde{u}_{i,max}\, [Nm], i = 1, 2. \tag{1.34}$$

Kinematics of a manipulator represents the basis of a systematic, general derivation of its *dynamics*, i.e., the equations of motion of the manipulator as a function of the forces and moments acting on it. The availability of the dynamic model is very useful for mechanical design of the structure, choice of actuators, determination of control strategies, and computer simulation of manipulator motion. This Chapter is dedicated to the study of dynamics, whereas on the following Appendix recalls some fundamentals on *Definitions and Descriptions* .

Modelling of *mobile robots* requires a preliminary analysis of the kinematic constraints imposed by the presence of wheels. Depending on the mechanical structure, such constraints can be integrable or not; this has direct consequence on a robot's mobility. The *kinematic model* of a mobile robot is essentially the description of the admissible instantaneous motions in respect of the constraints. On the other hand, the *dynamic model* accounts for the reaction forces and describes the relationship between the above motions and the generalized forces acting on the robot. These models can be expressed in a canonical form which is convenient for design of planning and control techniques.

2

Modelling of Multi-agent Systems (in Terms of Graph Theoretic Methods)

2.1 Multi-agent Systems

Network science has emerged as a powerful conceptual paradigm in science and engineering. Constructs and phenomena such as interconnected networks, random and small-world networks, and phase transition nowadays appear in a wide variety of research literature, ranging across social networks, statistical physics, sensor networks, economics, and of course multi-agent coordination and control. The reason for this unprecedented attention to network science is twofold. On the one hand, in a number of disciplines– particularly in biological and material sciences–it has become vital to gain a deeper understanding of the role that inter-elemental interactions play in the collective functionality of multilayered systems. On the other hand, technological advances have facilitated an ability to synthesize networked engineering systems–such as those found in multi-vehicle systems, sensor networks, and nanostructures that resemble, sometimes remotely, their natural counterparts in terms of their functional and operational complexity.

A basic premise in network science is that the structure and attributes of the network influence the dynamical properties exhibited at the system level. The implications and utility of adopting such a perspective for engineering networked systems, and specifically the system theoretic consequences of such a point of view, formed the impetus for much of this section.[1]

Engineered, distributed multi-agent networks, such as distributed robots and mobile sensor networks, have posed a number of challenges in terms of their system theoretic analysis and synthesis. Agents in such networks are required to operate in concert with each other in order to achieve system level objectives, while having access to limited computational resources and local communications and sensing capabilities. In this introductory chapter,

[1] One needs to add, however, that–judging by the vast apparatus of social networking, e.g.,

we first discuss a few examples of such distributed and networked systems, such as multiple aerospace vehicles, sensor networks, and nanosystems. We then proceed to outline some of the insights that a graph theoretic approach to multi-agent networks is expected to provide, before offering a preview of the chapter's content.[2]

Graph-based abstractions of networked systems contain virtually no information about what exactly is shared by the agents, through what protocol the exchange takes place, or what is subsequently done with the received information. Instead, the graph-based abstraction contains high-level descriptions of the network topology in terms of objects referred to as vertices and edges. In this chapter, we provide a brief overview of graph theory. Of particular focus will be the area of algebraic graph theory, which will provide the tools needed in later chapters for tying together inherently dynamic objects (such as multi-agent robotic systems) with combinatorial characterization of networks (graphs).

2.1.1 Boids Model

The Reynolds boids model, originally proposed in the context of computer graphics and animation, illustrates the basic premise behind a number of multi agent problems, in which a collection of mobile agents are to collectively solve a global task using local interaction rules. This model attempts to capture the way social animals and birds align themselves in swarms, schools, flocks, and herds. In the boids flocking model, each "agent," in this case a computer animated construct, is designed to react to its neighboring flockmates, following an ad hoc protocol consisting of three rules operating at different spatial scales.

These rules are *seperation* (avoid colliding with neighbors), *alignment* (align velocity with neighbors' velocities), and *cohesion* (avoid becoming isolated from neighbors). A special case of the boids model is one in which all agents move at the same constant speed and update their headings according to a nearest neighbor rule for group level alignment and cohesion. It turns out that based on such local interaction rules alone, velocity alignment and other types of flocking behaviors can be obtained. An example of the resulting behavior is shown in Figure 2.1.

2.1.2 Graph-Based Interaction Models

The interaction geometry will indeed play an important role in the analysis and synthesis of networked multiagent systems regardless of whether the infor-

[2] email, facebook, twitter, and a multitude of networked, coordinated, and harmonic behavior in nature and the arts–our fascination with multi-agent networks is more intrinsic.

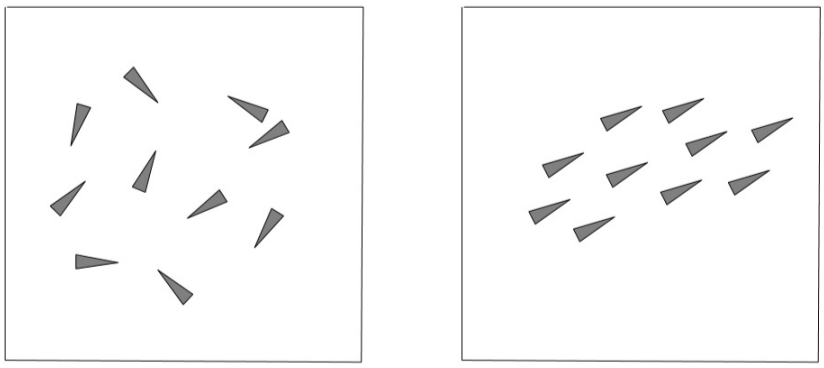

Fig. 2.1. A Reynolds boids model in action. Ten agents, each with an arbitrary initial heading (given by the orientation of the triangles) and spacing, are considered (left); after a while they are aligned, moving in the same general direction at regular interagent distances (right).

mation exchange takes place over a communication network or through active sensing, or for that matter whether it assumes a wireless, chemical, physical, or sociological character. It turns out, however, that making the interaction protocol and its geometry explicit in the system-level analysis and control synthesis is far from trivial. In this direction, it becomes judicious to treat interactions as essentially combinatorial–at least initially–to codify whether an interaction exists and to what degree.

An example of this abstraction is seen in Figure 2.2, in which the interaction geometry is defined by omnidirectional range sensors. As we will see throughout this section, such an abstraction, which cuts through the particular realization of the interaction, allows us to highlight the role of the interconnection topology, not only in the analysis of these systems but also in their synthesis.

2.2 Graphs

A *finite, undirected, simple graph*–or a graph for short–is built upon a finite set, that is, a set that has a finite number of elements. We refer to this set as the vertex set and denote it by V ; each element of V is then a *vertex* of the graph. When the vertex set V has n elements, it is represented as

$$V = \{v_1, v_2, \ldots, v_n\}\,.$$

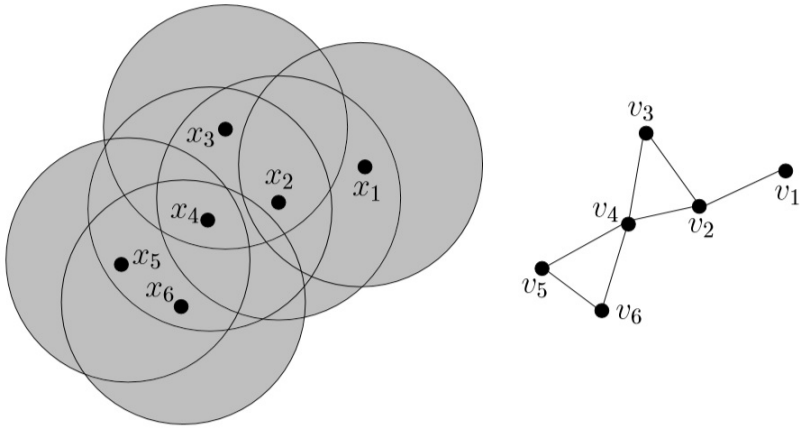

Fig. 2.2. A network of agents equipped with omnidirectional range sensors can be viewed as a graph, with nodes corresponding to the agents and edges to the interactions.

Now consider the set of 2-element subsets of V , denoted by $\left[V^2\right]$. This set consists of elements of the form $\{v_i, v_j\}$ such that i, j =1 ,2,...,n and $i \neq j$. The finite graph $\mathcal{G}$ is formally defined as the pair $\mathcal{G}$ =(V, E), where V is a finite set of vertices and E is a particular subset of $\left[V^2\right]$; we refer to E as the set of edges ofG. We occasionally refer to vertices and edges of $\mathcal{G}$ as V (G) and E(G), respectively, and simplify our notation for an edge $\{v_i, v_j\}$ by sometimes denoting it as $v_i v_j$ or even ij.

A graph is inherently a set theoretic object; however, it can conveniently be represented graphically, which justifies its name. The graphical representation of $\mathcal{G}$ consists of "dots" (the vertices v_i), and "lines" between v_i and v_j when $v_i v_j \in E$. This graphical representation leads to many definitions, insights, and observations about graphs. For example, when an edge exists between vertices v_i and v_j, we call them *adjacent*, and denote this relationship by $v_i \sim v_j$. In this case, edge $v_i v_j$ is called incident with vertices v_i and v_j. Figure 2.3 gives an example of an undirected graph, $\mathcal{G}$ =(V,E), where $V = \{v_1, v_2, \ldots, v_5\}$ and $E = \{v_1v_2, v_2v_3, v_3v_4, v_3v_5, v_2v_5, v_4v_5\}$.

Analogously, the *neighborhood* $N(i) \subseteq V$ of the vertex v_i will be understood as the set $\{v_j \in V \mid v_i v_j \in E\}$, that is, the set of all vertices that are adjacent to v_i. If $v_j \in N(i)$, it follows that $v_i \in N(j)$, since the edge set in a (undirected) graph consists of unordered vertex pairs. The notion of adjacency in the graph can be used to "move" around along the edges of the

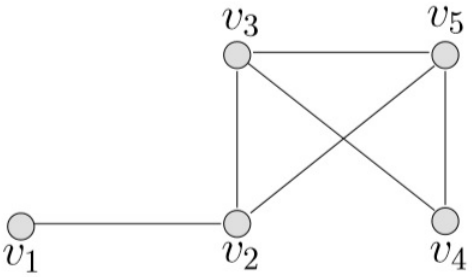

Fig. 2.3. An undirected graph on 5 vertices

graph. Thus, a path of length m in $\mathcal{G}$ is given by a sequence of distinct vertices

$$v_{i_0}, v_{i_1}, \ldots, v_{i_m}, \tag{2.1}$$

such that for $k = 0, 1, \ldots, m-1$, the vertices v_{i_k} and $v_{i_{k+1}}$ are adjacent. In this case, v_{i_0} and v_{i_m} are called the end vertices of the path; the vertices $v_{i_1}, \ldots, v_{i_m-1}$ are the inner vertices. When the vertices of the path are distinct except for its end vertices, the path is called a cycle. A graph without cycles is called a forest.

We call the graph $\mathcal{G}$ *connected* if, for every pair of vertices in $\mathrm{V}(\mathcal{G})$, there is a path that has them as its end vertices. If this is not the case, the graph is called *disconnected*. For example, the graph in Figure 2.3 is connected. We refer to a connected graph as having one connected component–a component in short. A component is thus a subset of the graph, associated with a minimal partitioning of the vertex set, such that each partition is connected. Hence, a disconnected graph has more than one component. A forest with one component is–naturally–called a tree.

The graphical representation of graphs allows us to consider graphs as *logical* constructions without the explicit identification of a vertex with an element of a vertex set $\mathcal{V}$. This is achieved by deleting the "labels" on the dots representing the vertices of the graph; in this case, the graph is called unlabeled. An unlabeled graph thus encodes the qualitative features of the incident relation between a finite set of an otherwise unidentified objects. When the vertices in an unlabeled graph are given back their identities, the graph is called labeled.

Graphs can represent relations among social entities. For example, in a party of six consisting of Anna, Becky, Carolyn, David, Eaton, and Frank, the graph shown in Figure 2.4 depicts a scenario where all males in the group are each others' friends, all females in the group are each others' friends, and Anna and David are the only cross-gender friends in the group.

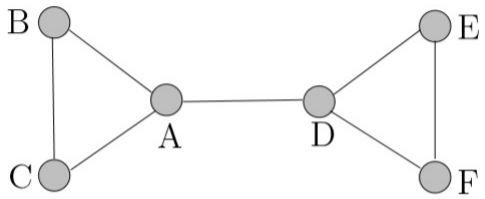

Fig. 2.4. Boys and girls

Now Consider a group of three robots coordinating their respective speeds according to the following chain of command: the rate of change of the second robot's speed is dictated by its speed difference with respect to the first one; the rate of change of the third robot's speed is adjusted analogously with respect to the second one. Finally, the first robot adjusts its speed by the taking the average of its speed differences with respect to the second and third robots.
Denoting the speed of robot i by s_i, the dynamics of the resulting system can be written as

$$\begin{aligned} \dot{s}_1(t) &= \tfrac{1}{2}\left((s_3(t) - s_1(t)) + (s_2(t) - s_1(t))\right), \\ \dot{s}_2(t) &= s_1(t) - s_2(t) \\ \dot{s}_3(t) &= s_2(t) - s_3(t) \end{aligned} \tag{2.2}$$

which assumes the form

$$\dot{s}(t) = \begin{bmatrix} -1 & \frac{1}{2} & \frac{1}{2} \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} s(t) \tag{2.3}$$

where $s(t) = [s_1(t)s_2(t)s_3(t)]^T$. We note that the matrix (2.3) corresponds to the negative of the in-degree Laplacian of the network shown in Figure 2.5;

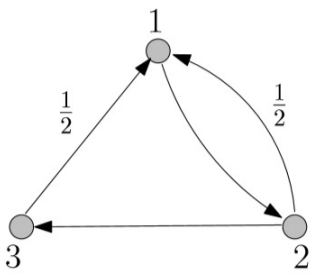

Fig. 2.5. A robotic chain of command represented by the directed graph $\mathcal{D}$

thus

$$\dot{s}(t) = -L(\mathcal{D})s(t), \tag{2.4}$$

where $\mathcal{D}$ is the underlying directed interconnection, that is, the weighted digraph of the network.

We note that in the above examples, the dynamics of each vertex in the network is "pulled" toward the states of the neighboring vertices. It is tempting then to conjecture that asymptotically, all vertices will reach some weighted average of their initial states, which also corresponds to the fixed point of their collective dynamics. As such a state of *agreement* is of great interest to us, we are obliged to formally define it.

The agreement set $\mathcal{A} \subseteq \mathbf{R}^n$ is the subspace span, that is,

$$\mathcal{A} = \{x \in \mathbf{R}^n \mid x_i = x_j, \text{ for all } i, j\}. \tag{2.5}$$

Our first goal in this chapter is to expand upon the mechanism by which the dynamics (2.4) over an undirected graph guides the vertices of the network to their agreement state, or the consensus value. We will then revisit the agreement protocol over directed networks, for example, those that can be represented as in (2.4).

2.3 Organization of Multi-Agent Systems

Complexity and highly distribution are the key characteristics of modern real world systems. The complexity of the near future and even present applications can be characterized as a combination of aspects such as great number of components taking part in the applications, knowledge and control have to be distributed, the presence of non-linear processes in the system, the fact that the system is more and more often open, its environment dynamic and the interactions unpredictable. Further, the increasing complexity, heterogeneity, and openness of modern software systems have reached a point that imposes new demands on their engineering technologies. It is expected that conventional engineering approaches will stand powerless in front of future systems increase in scale and complexity either vertically (control and information layers) or horizontally (physical distribution).

It doesn't mean that conventional engineering techniques will become obsolete and have to be thrown away. Absolutely, they only need to be integrated with new engineering styles where concepts such as, decomposition, autonomy, modularity, and adaptivity can be collectively combined in one system. MAS are considered as a promising engineering (i.e., architectural) style for developing adaptive software systems able to handle the continuous increase in their complexity as a result of their open, heterogeneous, and continuous evolution nature. They model the system as distributed autonomous agents cooperate together to achieve system goals. The ability of agents to dynamically reorganize to adapt working environment dynamic changes is a key feature provided by MAS.

It is obvious that the natural way to model a complex system is in terms of multiple autonomous components that can act and interact in flexible ways in order to achieve their objectives, and also that agents provide a suitable abstraction for modeling systems consisting of many subsystems, components and their relationships [93]. Ferber [79] described how agents, as a form of distributed artificial intelligence, are suitable for use in application domains which are widely distributed. MAS are currently considered as the most representatives among artificial systems dealing with complexity and highly distribution.

MAS (Multi-Agent system) allow the design and implementation of software systems using the same ideas and concepts that are the very founding of human societies and habits. These systems often rely on the delegation of goals and tasks among autonomous software agents, which can interact and collaborate with others to achieve common goals. In other words, an agent falls somewhere between a simple event-triggered program and one with human collaborative abilities. In contrast to initial MAS research, which concerned individual agents' aspects such as agents' architectures, agents' mental

capabilities, behaviors, etc, the current research trend of MAS is actively interested in the adaptivity, environment, openness and the dynamics of these systems. Also, there is a great attention towards the MAS technique as a way to design self-organized systems. In open environments, agents must be able to adapt towards the most appropriate organizations according to the environment conditions and their unpredictable changes.

Agent organizations are considered as an emergent area of MAS research that relies on the notion of openness and heterogeneity of MAS and imposes new demands on traditional MAS models. MAS that have the ability to dynamically reorganize (regardless of the type of reorganization, self or enforced) will be adaptive enough to survive against their dynamic and continuously changing working environments. Dynamic reorganization can take many forms, for instance, agents can dynamically change their roles, behaviors, locations, acquaintances, or the whole system organization structure can be dynamically changed.

An agent organization can also be defined as a social entity composed of a specific number of members (agents) that accomplish several distinct tasks or functions and that are structured following some specific topology and communication interrelationships in order to achieve the main aim of the organization. Thus, agent organizations assume the existence of global common goals, outside the objectives of any individual agent, and they exist independently of agents [76].

2.3.1 Motivations to MAS Organization

This section is dedicated to identify from MAS literature the suggested motivations to give increasing attention to MAS organization. Basically, a MAS is formed by the collection of autonomous agents situated in a certain environment, respond to their environment dynamic changes, interact with other agents, and persist to achieve their own goals or the global system goals. There are two viewpoints of MAS engineering, the first one is the agent-centered MAS (ACMAS) in which the focus is given to individual agents. With this viewpoint, the designer concerns the local behaviors of agents and also their interactions without concerning the global structure of the system. The global required function of the system is supposed to emerge as a result of the lower level individual agents interactions in a bottom-up way.

Picard et al. [93] stated that the agent-centered approach takes the agents as the "engine" for the system organization, and agent organizations implicitly exist as observable emergent phenomena, which states a unified bottom-up and objective global view of the pattern of cooperation between agents.

Further, Picard gives the ant colony as an example, where there is no organizational behavior and constraints are explicitly and directly defined inside the ants. The main idea is that the organization is the result of the collective emergent behavior due to how agents act their individual behaviors and interact in a common shared and dynamic environment.

The key problems of the ACMAS viewpoint are unpredictability and uncertainty. Because the whole is more than the sum of its parts, this approach can lead to undesirable emergent behaviors that may impact system performance, as a result, this approach might be not suitable to design and engineer complex multi-agent systems. The MAS applications engineered by the ACMAS approach are closed for agents that are not able to use the same type of coordination and behavior, and that all global characteristics and requirements are implemented in the individual agents and not outside them.

The second viewpoint of MAS engineering is what is called organization-centered MAS (OCMAS) in which the structure of the system is given a bigger attention through the explicit abstraction of agent organization. With that approach, the designer designs the entire organization and coordination patterns on the one hand, and the agents' local behaviors on the other hand. It is considered as a top-down approach because the organization abstraction imposes some rules or norms used by agents to coordinate their local behaviors and interactions with other agents.

2.3.2 MAS Formation

In the recent years, the formation control of multi-agent system has been a very active research area. The formation problem can be addressed, for example, by implementing consensus protocol strategies. This techniques enable a team of agents or vehicles to reach an agreement on certain sates or values of interest, in such a way that the behavior of all the agents is the same. Consensus protocols have been applied in diverse areas, for example, spacecraft formation flying, sensor networks, and cooperative surveillance, see for example [75] and the references therein. Olfati et al. [82] presented a general framework for the consensus problem of n integrator agents with fixed and switching topologies. Ren et al. [83] extend the work in [84] to the case of directed graphs, and explore the minimum requirements to reach consensus under changing topologies. You et al. [89] present the consensus condition for linear multi-agent systems over randomly switching topologies.

Robust H_∞ distributed consensus is presented in the works [95] and [91]. Lietal[94]presented two adaptive consensus protocols; one of them being relative-state and the other relative-output, respectively. Su et al. [85] studied

the problem of flocking of multi-agent systems with a virtual leader, while different cases of leader-follower in multi-agent systems are studied in [102], to name a few. The consensus problem for double-integrator dynamics is presented by in [116]. The case of systems with high-order integrators are studied in [104]. Some important applications of consensus protocol are proposed by Hung which studied the flocking control and the predator avoidance, respectively. In the aforementioned work, the authors proposed consensus protocols for different application cases such as switching topologies and leader-follower, as well as different control techniques to reach consensus.

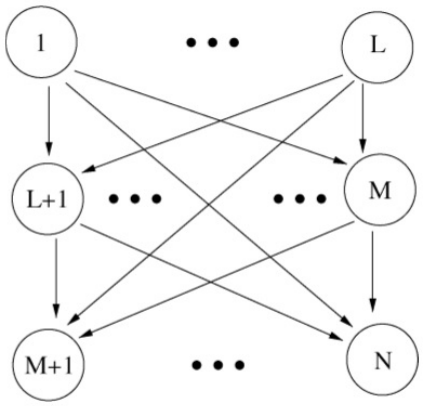

Fig. 2.6. Generic formation control graph.

(*Formation graph*). A formation graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ that describes the communication among the agents consists of aset of vertices $\mathcal{V} = \{1, \ldots, N\}$ corresponding to each of the agents in the group and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. An edge is an ordered pair $(i, j) \in \mathcal{E}$ if agent j can be directly supplied with information from agent i.

2.4 MAS Leader-Follower Configuration

For a networked leader-follower agent system, the multi-layer topology structure has been widely considered by hierar- chical control. In this kind of topology structure, agents can be divided into several layers based on information flow. It is easier for the topology to be illustrated and managed, and it is more

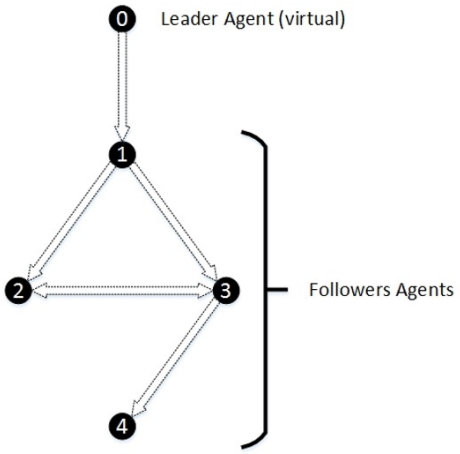

Fig. 2.7. The Generic spanning tree graph of the leader-follower consensus algorithm.

convenient to analyze stability of the whole topology layer by layer. Since the structure can be formulated by a directed graph, a necessary and sufficient condition for tracking is that the topology matrix is a spanning tree. Under this condition, coupling strength is provided to improve tracking performance of the system.

In some distributed consensus control designs, the coupling strength was usually exploited to deal with the perturbed factors such as Lipschitz-type nonlinearities, unknown but bounded in- put uncertainties, stochastic communication noises, and norm-bounded disturbances. Recently, state-dependent perturbations satisfying Lipschitz conditions have been considered in multiagent system designs. Asymptotic consensus results have been achieved under the perturbations and system uncertainties. The results can also be obtained for the stochastic communication noises when the coupling strength is effective on them.

On the other hand, serious attention should be paid to the sensitivity of controllers in systems to ensure the stability of systems with acceptable performances. Over the past two decades, large numbers of results regarding gain variations (uncertainties) of controllers or filters have been shown based on the insensitive (non-fragile) designs. In these studies, additive and multiplicative controller/filter gain variations have been widely investigated by using linear matrix inequality (LMI) technique. Under the types of norm-bounded and interval-bounded coefficient variations, many studies have been conducted for the H_∞ control and filtering of linear systems and nonlinear fuzzy system,

guaranteed cost designs in Markovian jump systems, distributed sensor networks.

However, following these previous works, we can see that although the impact of additive/multiplicative variations on performance can be determined by optimizing the L_2/H_∞ performance, the compensation problem of gain variations (uncertainties) of controllers was rarely to be investigated. In [96], a compensation filter was designed for eliminating additive filter gain variations and the relationship between performance degradation of systems and coefficient variation compensations is clearly demonstrated by LMIs. For the case of multiplicative controller gain variations, it seems compensation design methods should be further developed. Moreover, there is an important and original issue that should be determined exactly in insensitive tracking designs, that is, what is the rigorous relationship between bounds of tracking errors and sizes of variations?

Fig 2.8 is given to illustrate the multi-layer network architecture of six networked agents, including one leader agent and five follower agents. Here, we consider the following standard state-space equation of a linearized agent in this part:

$$\dot{x}_i(t) = (A + \Delta_{Ai}(t))\, x_i(t) + (B + \Delta_{Bi}(t))\, u_i(t), \tag{2.6}$$

where $i = 0, 1, 2, \ldots, N, x_0(t) \in R^n$ denotes the leader agent state and $x_i(t) \in R^n, i \in \{1, 2, \ldots, N\}$ is the i th follower agent state; $u_i(t) \in R^m$ is the control input of the i th agent. A and B are system and control input matrices respectively, and $\Delta_{Ai}(t)$ and $\Delta_{Bi}(t)$ stand for their time-varying model uncertainties of agent i . Here, we suppose that the pair $(A; B)$ is stabilizable and there is sufficient redundancy in control surfaces to compensate for the uncertainties $\Delta_{Ai}(t)$.

The measured output of the i th agent connected with the jth agent is formulated as follows:

$$y_{ij}(t) = \sigma(t) x_i(t) + d_{ij}(t) \tag{2.7}$$

where $i \in \{0, 1, \ldots, N\}, j \in \{1, 2, \ldots, N\}, j \neq i, 1 < \sigma(t) \leq \bar{\sigma}$ is a sequence of adjustable function denoting the coupling strength of networks and $\bar{\sigma}$ is the largest strength; The signal $d_{ij}(t) \in R^n$ stands for the bounded network disturbance between the ith and the jth agents. It can be described by a time-varying nonlinear function and satisfied by $\|d_{ij}(t)\| \leq \bar{d}_{ij}$, where $\bar{d}_{ij}$ is a known positive constant.

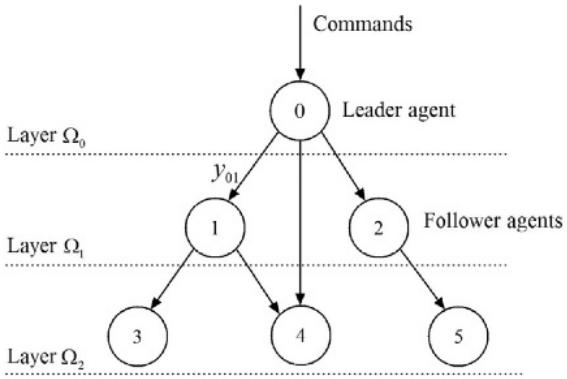

Fig. 2.8. Example of a multi-layer networked agent system with one leader agent and five follower agents.

In a practical networked agent system, such as an Unmanned Aerial Vehicle (UAV) team, signal transmission is based on high-frequency electronic theory. According to the knowledge of high-frequency electronic circuits, a signal transmission system is combined by power amplifiers, oscillators, modulators, demodulators, antennas and etc. The coupling strength $\sigma(t)$ in (7) is considered as the magnification of state $x_i(t)$. It can be realized by a power amplifier when the state is transmitted to an electrical signal by sensors. Thus, the value of $\sigma(t)$ is always larger than 1, and the value setting can be accomplished by changing the parameters in amplifiers.

We define that $\alpha(t) \in R^n$ is a given bounded command of states for the leader agent satisfying

$$\dot{\alpha}(t) = f(t), \tag{2.8}$$

where $f(t)$ is a continuous bounded signal. Then tracking controllers are considered as follows: i.e., for the leader agent

$$u_0(t) = -K_{01}\left(x_0(t) - \alpha(t)\right) + \left(I + \Delta_{K02}(t)\right) K_{02}(t), \tag{2.9}$$

and for the ith follower agent ($i = 1, 2, ..., N$)

$$\begin{aligned} u_i(t) = &- K_{i1}\left[\sum_{j=0,j\neq i}^{N} a_{ij}c_{ij}\left(y_{ji}(t) - \sigma(t)x_i(t)\right)\right] \\ &+ \left(I + \Delta_{Ki2}(t)\right)K_{i2}(t), \end{aligned} \tag{2.10}$$

where $a_{ij} \in \{0, 1\}$ represents the topological structure of networks; $c_{ij} \in (0, 1]$ is a weighting of connected networks, and it satisfies $\sum_{j=0,j\neq i}^{N} a_{ij}c_{ij} = 1; K_{i1}, i = 0, 1, \ldots, N$ is a constant control matrix, and K_{i2} is an adaptive compensation function to be designed later; $\Delta_{Ki2}(t)$ is a diagonal matrix and represents multiplicative variations of K_{i2}.

2.5 Multiple Interacting Leaders

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ be a directed graph with the set of nodes $\mathcal{V} = \{1, 2, \ldots, N\}$, the set of directed links $\mathcal{E} \subseteq \{(i, j), i, j \in \mathcal{V}\}$, and a weighted adjacency matrix $\mathcal{A} = [a_{ij}]_{N\times N}$ with elements $a_{ij} \geq 0$. The edge (i, j) in graph $\mathcal{G}$ is starting from node j and ending at node i. A path on G from node i_1 to node i_s is a sequence of ordered edges of the form $(i_{k+1}, i_k), k = 1, 2, \ldots, s - 1$.

A directed graph has or contains a directed spanning tree if there exists a node called root such that there exists a directed path from this node to every other node. The adjacency matrix $\mathcal{A} = [a_{ij}]_{N\times N}$ of a directed graph $\mathcal{G}$ is defined by $a_{ii} = 0$ for $i = 1, 2, \ldots, N$, and $a_{ij} > 0$ for $(i, j) \in \mathcal{E}$ but 0 otherwise. The Laplacian matrix $\mathcal{L} = [l_{ij}]_{N\times N}$ is defined as $l_{ij} = -a_{ij}, \quad i \neq j$, and $l_{ii} = \sum_{i=1}^{N} a_{ij}$ for $i = 1, 2, \ldots, N$. For an arbitrarily given directed graph $\mathcal{G}$, its Laplacian matrix $\mathcal{L}$ has the following property.

It is assumed that there are multiple interacting leaders in the present multi-agent system. To facilitate analysis, it is further assumed that the multi-agent systems under consideration consist of two layers, i.e., the leaders' layer and the followers' layer. Suppose that both the leaders' and the followers' layers contain N agents. Unlike most existing literature on distributed control of multi-agent systems that there is no interaction among the multiple leaders, in the present part, the evolution of each leader will be affected by its neighbors located in the leaders layer.

The above statements indicate that the leaders do not receive any information from the followers. However, the dynamics of each follower will be influenced by those of its neighbors in both leaders' and followers' layers. Specifically, the evolution of the i -th $(1 \leq i \leq N)$ leader is described as

$$\dot{x}_i(t) = Ax_i(t) + cBK\sum_{j=1}^{N} a_{ij}\left(x_j(t) - x_i(t)\right), \tag{2.11}$$

where $x_i(t) \in \mathbf{R}^m$ is the state of the i-th leader, $A \in \mathbf{R}^{m\times m}$ is the state matrix, $B \in \mathbf{R}^{m\times h}$ is the control input matrix, $K \in \mathbf{R}^{h\times m}$ is the inner linking matrix to be determined later and $c > 0$ indicates the coupling strength. It is assumed that matrix pair (A, B) is stabilizable. Furthermore, the evolution of the i -th $(1 \leq i \leq N)$ follower is given as

$$\dot{\hat{x}}_i(t) = A\hat{x}_i(t) + cBK\sum_{j=1}^{N} a_{ij}\left(\hat{x}_j(t) - \hat{x}_i(t)\right) + cp_i(t)BK\left(x_i(t) - \hat{x}_i(t)\right), \tag{2.12}$$

where $\hat{x}_i(t) \in \mathbf{R}^m$ is the state of the i-th follower, the pinning link $p_i(t) \in \{0, 1\}$ and $p_i(t) = 1$ if and only if the i-th follower can directly sense the i-th leader at time t, i.e., there exists a directed link from the i-th leader to the i-th follower at time t. It is assumed that, for each $i \in \{1, 2, \ldots, N\}$, there exists an infinite sequence of uniformly bounded non-overlapping time intervals $\left[\hat{t}_k^i, \widehat{t}_{k+1}^i\right), k \in \mathbf{N}$, with $\hat{t}_1^i = 0$, $\widehat{t}_{k+1}^i - \hat{t}_k^i \geq \tilde{\tau}_0$, and $\tilde{\tau}_0 > 0$, over which $p_i(t)$ is fixed. Here, $\tilde{\tau}_0$ is called the dwell time.

It can be verified that distributed node-to-node consensus for multi-agent systems will be achieved if and only if the zero equilibrium point of error dynamical system is globally attractive. Since N is a finite positive natural number and the topology among N followers is fixed, one may use $\mathbb{L} = \left\{\widehat{\mathcal{L}}^1, \widehat{\mathcal{L}}^2, \ldots, \widehat{\mathcal{L}}^S\right\}$ to indicate the set of all possible augmented Laplacian matrices of the considered multi-agent systems, i.e., $\widehat{\mathcal{L}}(t) \in \mathbb{L}$ for all t. Obviously, $s \leq 2^N$. By taking all the leaders as a single node, labeling as node N+1, one may then define a new graph $\tilde{\mathcal{G}}(t)$ associated with Laplacian matrix

$$\tilde{\mathcal{L}}(t) = \begin{pmatrix} \widehat{\mathcal{L}}(t) & \mathbf{p}(t) \\ \mathbf{0}_N^T & 0 \end{pmatrix} \in \mathbf{R}^{(N+1)\times(N+1)}, \tag{2.13}$$

where $\mathbf{p}(t) = (p_1(t), p_2(t), \ldots, p_N(t))^T$. This indicates that for each follower $i(1 \leq i \leq N)$ in (11), there exists at least one leader $j(1 \leq j \leq N)$ such that there exists a directed path from j to i (see Fig. 2.9 for illustration).

2.6 Other Possible MAS Formations

First Throughout this section, we consider a team of L vehicles described previously organized as the LF configuration of Fig 2.10 with a unique path connecting each follower to the leader. Moreover, for each follower the following definitions are exploited

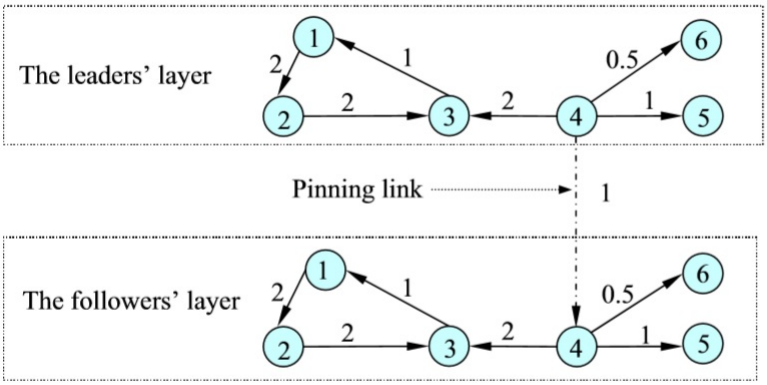

Fig. 2.9. Communication topology of multi-agent systems where for each follower, there is a directed path from leader node 4 to that follower.

- *tree level*:
$$level(k) : \{1, \ldots, L\} \to \mathbb{Z}_+ \tag{2.14}$$
which provides the position of the $k-th$ agent along the LF configuration;
- *set of neighbours*:
$$\begin{aligned} \mathcal{N}^i := \{q \in \{1, \ldots, i-1, i+1, \ldots, L\} : \\ level(q) \equiv level(i)\} \end{aligned} \tag{2.15}$$

Now Let consider the LF team with multi-model discrete-time linear descriptions for the LF team, obtained by discretizing via the forward Euler difference method:

$$x^i(t+1) = \Phi^i(\alpha(t))x^i(t) + G^i(\alpha(t))u^i(t), i = 1, \ldots, L, \tag{2.16}$$

where $t \in_+ := \{0, 1, ...\}$, $x^i(t) \in^n$ denotes the state plant and $u^i(t) \in^m$ the control input. The time-varying vector $\alpha(t) \in^p$ belongs to the unit simplex

$$\mathcal{P} := \left\{ \alpha \in \mathbf{R}^\Gamma : \sum_{k=1}^{\Gamma} \alpha_k = 1, \alpha_k \geq 0 \right\} \tag{2.17}$$

whereas the system matrices $(\Phi^i(\alpha),\, G^i(\alpha))$ lie in

$$\Omega(\mathcal{P}) := \left\{ \left(\Phi^i(\alpha), G^i(\alpha)\right) = \sum_{k=1}^{\Gamma} \alpha_k \left(\left(\Phi^i\right)^k, \left(G^i\right)^k \right), \alpha \in \mathcal{P} \right\} \tag{2.18}$$

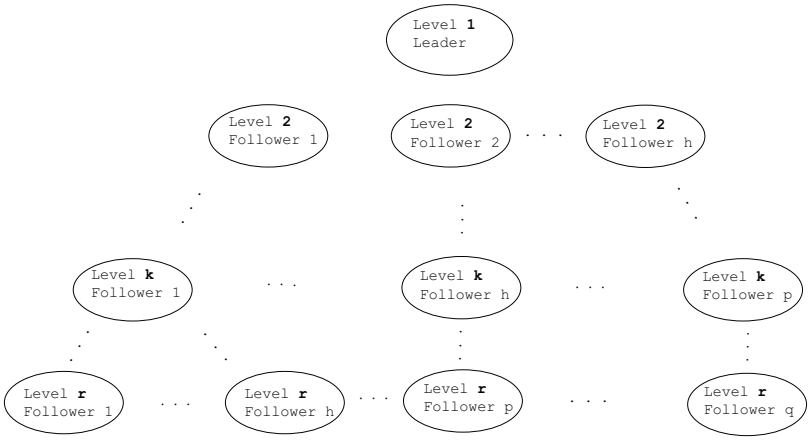

Fig. 2.10. Tree Leader-Follower topology

with $((\bar{\Phi}^i)^k, (G^i)^k)$ the polytope vertices of $\Sigma(\mathcal{P})$.

Moreover, state and input constraints are recast as follows:

$$x^i(t) \in \mathcal{X}^i, u^i(t) \in \mathcal{U}^i, \forall t \geq 0, i = 1, \ldots, L, \tag{2.19}$$

with $\mathcal{X}^i$ and $\mathcal{U}^i$ compact and convex subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively. In the sequence by resorting to MAS formation ideas, two ad-hoc distributed receding horizon control schemes can be proposed and designed for properly dealing with obstacle free and corridor occurrences. The first algorithm refers to an grid configuration and it is oriented to deal with uncertain operating conditions, while the second is formulated to comply with a standard platoon configuration exploited for narrowed passages (corridors).

A Proposed grid configuration can be initially assumed, see Fig 2.11. For each UAV the following definitions are used:

- *grid level*: $level(i) : \{1, \ldots, L\} \to \mathbb{Z}_+$, which provides the position of the $i-th$ vehicle along the grid configuration;
- *set of neighbours*:

$$\begin{aligned} \mathcal{N}^i := \{ & j \in \{1, \ldots, i-1, i+1, \ldots, L\} : \\ & level(j) \equiv level(i) \} \end{aligned} \tag{2.20}$$

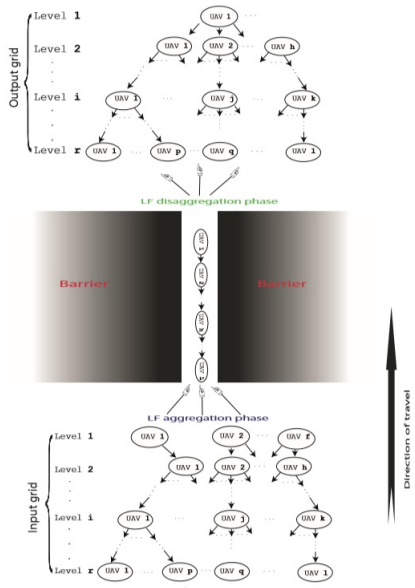

Fig. 2.11. Proposed Grid and Platoon Configuration

- *father*: the operator $pre(i) : \{1, \ldots, L\} \to \{1, \ldots, L\}$ denotes the predecessor (father) of the $i - th$ UAV;
- *set of lower level nodes*: $post(i)$, $i = 1, \ldots, L$, refers to an index subset of the UAVs belonging to the $(i + 1) - th$ level.

By resorting to the ideas proposed in above, let restrict the attention to the corridor of Fig 2.11 and denote with x_{in} and x_{fin} (x_{fin} an equilibrium) the entrance point and the exit point along the corridor, respectively.

For simplicity and without loss of generality, if we consider a platoon of r vehicles and introduce the following change of variables:

$$e_1^{\mathrm{II}} = z_1^{\mathrm{II}} - v_{d1} \quad \text{for the leading vehicle}$$

$$\left\{ \begin{array}{r} e_i^{\mathrm{I}} = z_{i-1}^{\mathrm{I}} - z_i^{\mathrm{I}} - d_{i-1} \\ e_i^{\mathrm{II}} = z_i^{\mathrm{II}} - v_{di} \end{array} \right\}, \quad i \in \{2, \ldots, r\}, \tag{2.21}$$

where $d_{i-1} \in \mathbb{R}^2$ is a constant desired Euclidean distance between the $(i-1)st$ and ith vehicles, $i \in \{2, \ldots, r\}$, and $v_{di}, v_{di} \in \mathbb{R}^2$, represents the desired speed

for the ith vehicle, $i \in \{1, 2, \ldots, r\}$. In Fig 2.12, for example, platoon 1 would include vehicles 1, 2, and 3, and platoon 2 would include vehicles 1, 4, and 5. Notice that for controlling distances between vehicles, position of the leading vehicle $\left(\text{i.e.}, \mathrm{z}_1^{\mathrm{I}}\right)$ is not needed. Since the desired Euclidean distances between vehicles are assumed to be constant, the following assumption is necessary:

$$v_{di} = v_d, \quad i \in \{1, 2, \ldots, r\}. \tag{2.22}$$

In other words, in order to achieve constant desired spacing in the formation, the desired speed for each vehicle must be the same. Then, $\dot{e}_1^{\mathrm{II}} = u_1$ for the leading vehicle

$$\left\{ \begin{matrix} \dot{e}_i^{\mathrm{I}} = e_{i-1}^{\mathrm{II}} - e_i^{\mathrm{II}} \\ \dot{e}_i^{\mathrm{II}} = u_i \end{matrix} \right\}, \quad i \in \{2, \ldots, r\}. \tag{2.23}$$

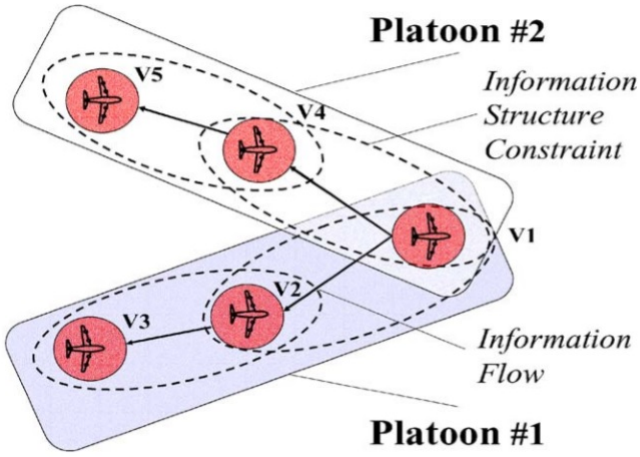

Fig. 2.12. Leader-follower type formation with five vehicles and two platoons.

In the sequel, the Leader Follower team can be reorganized as a platoon of r swarm aggregations $\{SW_j\}_{j=1}^{r}$, each one collecting the agents having the same tree level position, and the agents will be denoted as Σ_j^i, with j accounting for $level(\cdot)$, see Fig. 4.1. Moreover, the vehicle models defined in previous

chapter are discretized via the forward Euler difference method:

$$\begin{cases} x_j^i(t+1) = F_j^i(x_j^i(t)) + B_j^i(x_j^i(t))u(t), \\ i = 1, \ldots, l_j,\ j = 1, \ldots, r, \end{cases} \tag{2.24}$$

$$\begin{cases} x_j^i(t+1) = \Phi_j^i(\alpha(t))x_j^i(t) + G_j^i(\alpha(t))u_j^i(t), \\ i = 1, \ldots, l_j,\ j = 1, \ldots, r, \end{cases} \tag{2.25}$$

where $t \in \mathbb{Z}_+ := \{0, 1, ...\}$, $x_j^i(t) \in \mathbb{R}^n$ denotes the state plant and $u_j^i(t) \in \mathbb{R}^m$ the control input. The time-varying vector $\alpha(t) \in \mathbb{R}^p$ belongs to the unit simplex

$$\mathcal{P} := \left\{ \alpha \in \mathbb{R}^\Gamma : \sum_{k=1}^{\Gamma} \alpha_k = 1, \alpha_k \geq 0 \right\} \tag{2.26}$$

whereas the system matrices $(\Phi_j^i(\alpha),\ G_j^i(\alpha))$ lie in

$$\Omega(\mathcal{P}) := \left\{ \left(\Phi_j^i(\alpha), G_j^i(\alpha)\right) = \sum_{k=1}^{\Gamma} \alpha_k \left(\left(\Phi_j^i\right)^k, \left(G_j^i\right)^k \right), \alpha \in \mathcal{P} \right\} \tag{2.27}$$

with $((\Phi_j^i)^k, (G_j^i)^k)$ the polytope vertices of $\Sigma(\mathcal{P})$. Moreover, $x_j^i \in \mathcal{X}_j^i \subset \mathbb{R}^n$ and $u_j^i \in \mathcal{U}_j^i \subset \mathbb{R}^m$.

Finally and as the other possible formations, it could be considered a three-robot formation as shown in Figure 2.14, where $G_i = LF\left((R_j, R_k) \leftarrow R_i\right)$. Under the leader–follower scheme regarding more than one single leader, R_i is required to maintain desired separations $l_{i,j}^d$ and $l_{i,k}^d$ with respect to R_j and R_k respectively, and meanwhile to maintain a desired orientation deviation $\beta_{i,j}^d$ with respect to R_j . The desired posture for the follower R_i can be determined by

$$p_i^d = \begin{bmatrix} x_i^d \\ y_i^d \\ \theta_i^d \end{bmatrix} = \begin{bmatrix} \left(b_i \pm \left(b_i^2 - 4a_i c_i\right)\right)/a_i \\ \left(M_i - \Delta x_i x_i^d\right)/\Delta y_i \\ \theta_j + \beta_{i,j}^d \end{bmatrix}, \tag{2.28}$$

where $M_i = \frac{1}{2}\left[\left(x_j^2 - x_k^2\right) + \left(y_j^2 - y_k^2\right) - \left(l_{i,j}^d\right)^2 + \left(l_{i,k}^d\right)^2\right]$, $\Delta x_i = x_j - x_k$, $\Delta y_i = y_j - y_k$, and

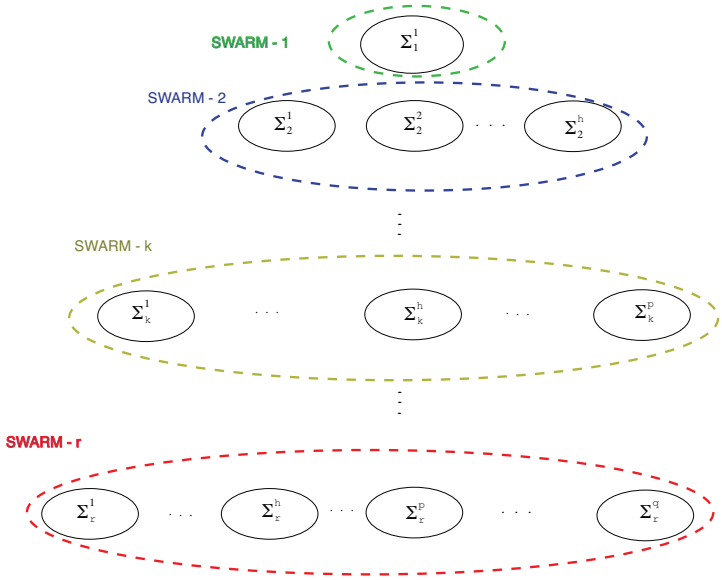

Fig. 2.13. Proposed Swarm platoon

$$\begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} = \begin{bmatrix} 2\left(\Delta x_i^2 + \Delta y_i^2\right) \\ 2\left(M_i \Delta x_i - \Delta y_i \left(\Delta x_i y_j - \Delta y_i x_j\right)\right) \\ \left(M_i - \Delta y_i y_j\right)^2 - \left(\Delta y_i x_j\right)^2 - \left(\Delta y_i l_{i,j}^d\right)^2 \end{bmatrix}. \tag{2.29}$$

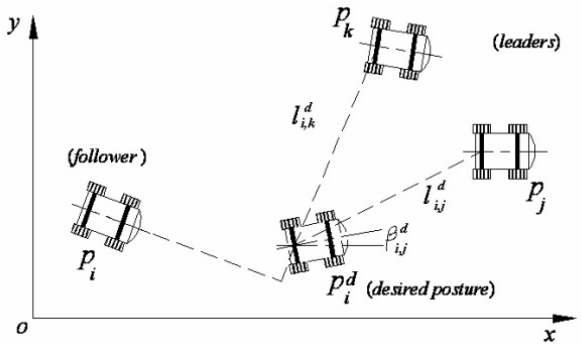

Fig. 2.14. Proposed Three-robot formation with more than one single Leader.

Then let Consider the formation depicted in Fig. 2.15. All robots are thought to use the controllers, with only the difference that vehicle 3 uses higher feedback gains compared with all the others. In view of the increased performance capabilities of robot 3, one may consider assigning robots5, 6, and 7to follow 3. However, an LFS [81] analysis reveals that such a change will, in fact,increase the magnitude of the formation errors:assume that $k_1^i = k_2^i = k$, for $i = \{1, 2, 4, 5, 6, 7\}$ and $k_1^3 = k_2^3 = k' > k$.

To move on, if we consider to study the containment control of multi-agent system with multiple leaders to extend the former results in this framework. Consider the case of multi-agent system with multiple leaders. Suppose that there are N followers and N_1 leaders,where $N_1 > 1$. The communication topology among the $N + N_1$ agents is denoted by g.

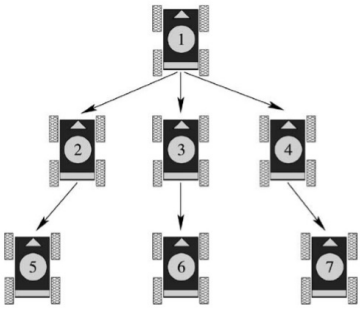

Fig. 2.15. Nearest neighbor following as a topology with multiple leaders.

An agent is called a leader if the agent has no neighbor. An agent is called a follower if the agent has at least one neighbor. Without loss of generality, it is assumed that the agents labeled by $1, ..., N$, are followers,while the agents labeled by $N + 1, ..., N + N_1$,leaders whose control inputs are set to be zero. The dynamics of the multiple leaders are given by

$$\dot{q}_i^d = p_i^d, \quad \dot{p}_i^d = 0, \tag{2.30}$$

with an initial position $q_i^d(0) = r_i$ and an initial velocity $p_i^d(0) = v_i$, for $i = N + 1, \ldots, N + N_1$. Denote by $\mathcal{L}$ the Laplacian matrix of $\mathcal{G}$. Then,$\mathcal{L}$ can be partitioned as

$$\mathcal{L} = \begin{pmatrix} \mathcal{L}_f & \mathcal{L}_I \\ 0 & 0 \end{pmatrix}, \tag{2.31}$$

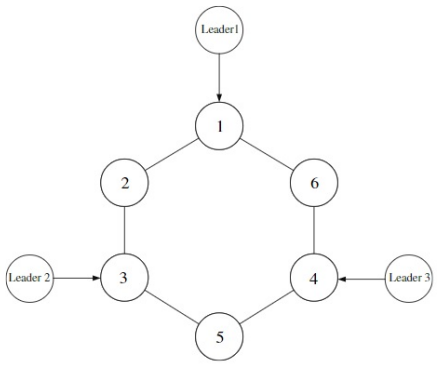

Fig. 2.16. Communication topology with multiple leaders

where $\mathcal{L}_f \in \mathbb{R}^{N\times N}$ and $\mathcal{L}_l \in \mathbb{R}^{N\times N_1}$.
LFS gains can be used to check and implement safety specifications that are related to formation errors. for instance, in the examples of this section, we could consider a formation of three robots connected in cascade via the separation-bearing controllers equation. The group is supposed to maneuver maintaining a triangular shape for which the faces must not exceed a certain distance. This will ensure that the robots move in a tight formation,in the same way as fighters, when flying in formation, have to maintain certain patterns to avoid detection by enemy radar.

The leader of the formation is to follow a reference trajectory. The time parameterization of the reference trajectory defines a desired velocity for the leader. This reference velocity can be regarded as an input to the formation, and as such, it will affect the size of the formation errors.If the magnitude of this velocity were a design parameter, then a question that arises is whether one can select an appropriate value to ensure that the formation can track the reference trajectory without violating its safety specification. One of the major considerations when dealing with large-scale interconnected systems, such as large vehicle formations, is the ability to compute the gain estimates efficiently, regardless of the size of the system. For nonlinear systems, due to their inherent complexity, LFS gain computation using is cumbersome and does not scale well. The conclusions that can be drawn in the case of large-scale vehicle formations are basically qualitative.

3

Model Based Predictive Control

3.1 Basic Model Predictive Control Philosophy

Model Predictive Control (MPC), also known as Moving Horizon Control (MHC) or Receding Horizon Control (RHC), is a popular technique for the control of slow dynamical systems, such as those encountered in chemical process control in the petrochemical, pulp and paper industries, and in gas pipeline control. At every time instant, MPC requires the on-line solution of an optimization problem to compute optimal control inputs over a fixed number of future time instants, known as the "time horizon". Although more than one control move is generally calculated, only the first one is implemented. At the next sampling time, the optimization problem is reformulated and solved with new measurements obtained from the system. The on-line optimization can be typically reduced to either a linear program or a quadratic program.

Using MPC, it is possible to handle inequality constraints on the manipulated and controlled variables in a systematic manner during the design and implementation of the controller. Moreover, several process models as well as many performance criteria of significance to the process industries can be handled using MPC. A fairly complete discussion of several design techniques based on MPC and their relative merits and demerits can be found in the review article by Garcia et al. (1989) [93].

Perhaps the principal shortcoming of existing MPC-based control techniques is their inability to *explicity* incorporate plant model uncertainty. Thus, nearly all known formulations of MPC minimize, on-line, a *nominal* objective function, using a single linear time-invariant model to predict the future plant behavior. Feedback, in the form of plant measurement at the next sampling time, is expected to account for plant model uncertainty. Needless to say, such control systems which provide "optimal" performance for a particular model may perform very poorly when implemented on a physical system which is not exactly described by the model (for example, see [94])

Similarly, the extensive amount of literature on stability analysis of MPC algorithms is by and large restricted to the nominal case, with no plant-model mismatch; the issue of the behavior of MPC algorithms in the face of uncertainty, i.e., "robustness", has been addressed to a much lesser extent. Broadly, the existing literature on robustness in MPC can be summarized as follows:

- *Analysis of robustness properties of MPC.* Garcia and Morari [133] have analyzed the robustness of unconstrained MPC in the framework of internal model control (IMC) and have developed tuning guidelines for the IMC filter to guarantee robust stability.
- *MPC with explicit uncertainty description.* The basic philosophy of MPC-based design algorithms which explicitly account for plant uncertainty [137] is the following: Modify the on-line minimization problem (minimizing some objective function subject to input and output constraints) to a min-max problem (minimizing the worst-case value of the objective function, where the worst-case is taken over the set of uncertain plants).

One of the problems with this linear programming approach is that to simplify the on-line computational complexity, one must choose simplistic, albeit unrealistic model uncertainty descriptions, for e.g., fewer FIR coefficients. Secondly, this approach cannot be extended to unstable systems. From the preceding review, we see that there has been progress in the *analysis* of robustness properties of MPC. But *robust synthesis*, i.e., the explicit incorporation of realistic plant uncertainty description in the problem formulation, has been addressed only in a restrictive framework for FIR models. There is a need for computationally inexpensive techniques for robust MPC synthesis which are suitable for on-line implementation and which allow incorporation of a broad class of model uncertainty descriptions.

3.2 Background

3.2.1 Models for uncertain systems

In this section we would present two paradigms for robust control which arise from two different modeling and identification procedures. The first is a "multi-model" paradigm, and the second is the more popular "linear system with a feedback uncertainty" robust control model. Underlying both these paradigms is a linear time-varying (LTV) system

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k), \\ y(k) &= Cx(k) \\ [A(k) \quad B(k)] &\in \Omega \end{aligned} \tag{3.1}$$

where $u(k) \in \mathcal{R}^{n_u}$ is the control input, $x(k) \in \mathcal{R}^{n_x}$ is the state of the plant and $y(k) \in \mathcal{R}^{n_y}$ is the plant output, and Ω is some prespecified set.

Polytopic or multi-model paradigm

For polytopic systems, the set Ω is the polytope

$$\Omega = \mathrm{Co}\left\{[A_1 B_1], [A_2 B_2], \ldots, [A_L B_L]\right\} \tag{3.2}$$

where Co refers to the convex hull. In other words, if $\begin{bmatrix} A & B \end{bmatrix} \in \Omega$, then for some nonnegative $\lambda_1, \lambda_2, \ldots, \lambda_L$ summing to one, we have

$$[AB] = \sum_{i=1}^{L} \lambda_i [A_i B_i].$$

When $L = 1$, we have a linear time-invariant system, which corresponds to the case when there is no plant-model mismatch.

Polytopic system models can be developed as follows. Suppose that for the (possibly nonlinear) system under consideration, we have input/output data sets at different operating points, or at different times. From each data set, we develop a number of linear models (for simplicity, we assume that the various linear models involve the same state vector). Then, it is reasonable to assume that any analysis and design methods for the polytopic system (3.1), (3.2) with vertices given by the linear models will apply to the real system.

Alternatively, suppose the Jacobian $\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial u} \end{bmatrix}$ of a nonlinear discrete time-varying system $x(k+1) = f(x(k), u(k), k)$ is known to lie in the polytope ω. Then it can be shown that every trajectory (x, u) of the original nonlinear system is also a trajectory of (3.1) for some LTV system in ω. Thus, the original nonlinear system can be approximated (possibly conservatively) by a polytopic uncertain linear time-varying system. Similarly, it can be shown that bounds on impulse response coefficients of SISO FIR plants can be translated to a polytopic uncertainty description on the state-space matrices. Thus, this polytopic uncertainty description is suitable for several problems of engineering significance.

Structured feedback uncertainty

A second, more common paradigm for robust control consists of a linear time-invariant system with uncertainties or perturbations appearing in the feedback loop (see Figure 3.1):

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + B_p p(k), \\ y(k) &= Cx(k), \\ q(k) &= C_q x(k) + D_{qu} u(k), \\ p(k) &= (\Delta q)(k). \end{aligned} \tag{3.3}$$

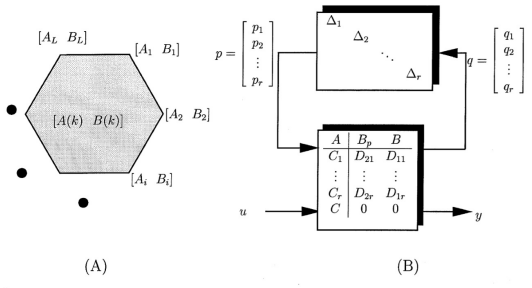

Fig. 3.1. (A) Graphical representation of polytopic uncertainty; (B) Structured uncertainty.

The operator Δ is block diagonal:

$$\Delta = \begin{bmatrix} \Delta_1 & & & \\ & \Delta_2 & & \\ & & \ddots & \\ & & & \Delta_r \end{bmatrix} \tag{3.4}$$

with $\Delta_i : \mathcal{R}^{n_i} \longrightarrow \mathcal{R}^{n_i}$. Δ can represent either a memoryless time-varying matrix with $\|\Delta_i(k)\|_2 \equiv \bar{\sigma}\left(\Delta_i(k)\right) \leq 1, \quad i = 1, 2, \ldots, r, \quad k \geq 0$; or a convolution operator (for e.g., a stable linear time invariant (LTI) dynamical system) with the operator norm induced by the truncated ℓ_2 -norm less than 1, i.e.,

$$\sum_{j=0}^{k} p_i(j)^T p_i(j) \leq \sum_{j=0}^{k} q_i(j)^T q_i(j), i = 1, \ldots, r, \quad \forall k \geq 0 \tag{3.5}$$

Each Δ_i is assumed to be either a *repeated scalar* block or a full block and models a number of factors, such as nonlinearities, dynamics or parameters, that are unknown, unmodeled or neglected. A number of control systems with uncertainties can be recast in this framework [85]. For ease of reference, we will refer to such systems as systems with structured uncertainty. Note that in this case, the uncertainty set Ω is defined by (3.3) and (3.4).

When Δ_i is a stable LTI dynamical system, the quadratic sum constraint (3.5) is equivalent to the following frequency domain specification on the z -transform $\hat{\Delta}_i(z)$

$$\left\|\hat{\Delta}_i\right\|_{\mathcal{H}_\infty} \equiv \sup_{\theta \in [0, 2\pi)} \bar{\sigma}\left(\hat{\Delta}_i\left(e^{j\theta}\right)\right) \leq 1$$

Thus, the structured uncertainty description is allowed to contain both LTI and LTV blocks, with frequency domain and time-domain constraints respectively. For the LTV case, it is easy to show through routine algebraic manipulations that system (3.3) corresponds to system (3.1) with

$$\Omega = \left\{ \left[A + B_p \Delta C_q \; B + B_p \Delta D_{qu} \right] : \Delta \text{ satisfies(1.4) with } \bar{\sigma}\left(\Delta_i\right) \leq 1 \right\} \quad (3.6)$$

The case where $\Delta \equiv 0, p(k) \equiv 0, k \geq 0$, corresponds to the nominal system, i.e., no plant-model mismatch.

The issue of whether to model a system as a polytopic system or a system with structured uncertainty depends on a number of factors, such as the underlying physical model of the system, available model identification and validation techniques etc. For example, nonlinear systems can be modeled either as polytopic systems or as systems with structured perturbations. We will not concern ourselves with such issues here; instead we will assume that one of the two models discussed thus far is available.

3.3 Basic Model Predictive Control Schemes

Model Predictive Control is an open-loop control design procedure where at each sampling time k, plant measurements are obtained and a model of the process is used to predict future outputs of the system. Using these predictions, m control moves $u(k+i \mid k), i = 0, 1, \ldots, m-1$ are computed by minimizing a *nominal* cost $J_p(k)$ over a prediction horizon p as follows:

$$\min_{u(k+i|k), i=0,1,\ldots,m-1} J_p(k), \quad (3.7)$$

subject to constraints on the control input $u(k+i \mid k), i = 0, 1, \ldots, m-1$ and possibly also on the state $x(k+i \mid k)$ and the output $y(k+i \mid k), i = 0, 1, \ldots, p$. Here

- $x(k+i \mid k), y(k+i \mid k)$: state and output respectively, at time $k+i$, predicted based on the measurements at time k; $x(k \mid k)$ and $y(k \mid k)$ refer respectively to the state and output measured at time k.
- $u(k+i \mid k)$: control move at time $k+i$, computed by the optimization problem (3.7) at time k; $u(k \mid k)$ is the control move to be implemented at time k.
- p : *output* or *prediction* horizon
- m : input or *control* horizon

It is assumed that there is no control action after time $k+m-1$, i.e., $u(k+i \mid k) = 0, i \geq m$. In the receding horizon framework, only the first computed control move $u(k \mid k)$ is implemented. At the next sampling time, the optimization (3.7) is resolved with new measurements from the plant. Thus, both the control horizon m and the prediction horizon p move or *recede* ahead by one step as time moves ahead by one step. This is the reason why MPC is also sometimes referred to as Receding Horizon Control (RHC) or Moving Horizon Control (MHC). The purpose of taking new measurements at each time step is to compensate for unmeasured disturbances and model

inaccuracy both of which cause the system output to be different from the one predicted by the model. We assume that exact measurement of the state of the system is available at each sampling time k, i.e.,

$$x(k \mid k) = x(k). \tag{3.8}$$

Several choices of the objective function $J_p(k)$ in the optimization (3.7) have been reported [116,119] and compared. In this section, we could consider the following quadratic objective:

$$J_p(k) = \sum_{i=0}^{p} \left(x(k+i \mid k)^T Q_1 x(k+i \mid k) + u(k+i \mid k)^T R u(k+i \mid k) \right),$$

where $Q_1 > 0$ and $R > 0$ are symmetric weighting matrices. In particular, we will consider the case $p = \infty$ which is referred to as the infinite horizon MPC (IH-MPC). Finite horizon control laws have been known to have poor nominal stability properties [112]. Nominal stability of finite horizon MPC requires imposition of a terminal state constraint $(x(k+i \mid k) = 0, i = m)$ and/or use of the contraction mapping principle [105] to tune Q_1, R, m and p for stability. But the terminal state constraint is somewhat artificial since only the first control move is implemented. Thus, in the closed loop, the states actually approach zero only asymptotically. Also, the computation of the contraction condition at all possible combinations of active constraints at the optimum of the on-line optimization can be extremely time consuming, and as such, this issue remains unaddressed. On the other hand, infinite horizon control laws have been shown to guarantee nominal stability. We therefore believe that rather than using the above methods to "tune" the parameters for stability, it is preferable to adopt the infinite horizon approach to guarantee at least nominal stability.

3.3.1 Control Architectures within MPC

Model predictive control (MPC) is widely adopted in industry as an effective approach to deal with large multivariable constrained control problems. The main idea of MPC is to choose control actions by repeatedly solving an online constrained optimization problem, which aims at minimizing a performance index over a finite prediction horizon based on predictions obtained by a system model. In general, an MPC design is composed of three components:

1) A model of the system. This model is used to predict the future evolution of the system in open-loop and the efficiency of the calculated control actions of an MPC depends highly on the accuracy of the model.
2) A performance index over a finite horizon. This index will be minimized subject to constraints imposed by the system model, restrictions on control inputs and system state and other considerations at each sampling time to obtain a trajectory of future control inputs.

3) A receding horizon scheme. This scheme introduces feedback into the control law to compensate for disturbances and modeling errors.

Typically, MPC is studied from a centralized control point of view in which all the manipulated inputs of a control system are optimized with respect to an objective function in a single optimization problem. Figure 3.2 is a schematic of a centralized MPC architecture for a system comprised of two coupled subsystems. Consider the control of the system and assume that the state measurements of the system are available at synchronous sampling time instants $\{t_{k\geq 0}\}$, a standard MPC is formulated as follows:

$$\min_{u_1,\ldots,u_m \in S(\Delta)} J\left(t_k\right) \tag{3.9}$$

$$\text{s.t. } \dot{\tilde{x}}(t) = f(\tilde{x}) + \sum_{i=1}^{m} g_i(\tilde{x}) u_i(t) \tag{3.10}$$

$$u_i(t) \in U_i, i = 1, \ldots, m \tag{3.11}$$

$$\tilde{x}\left(t_k\right) = x\left(t_k\right) \tag{3.12}$$

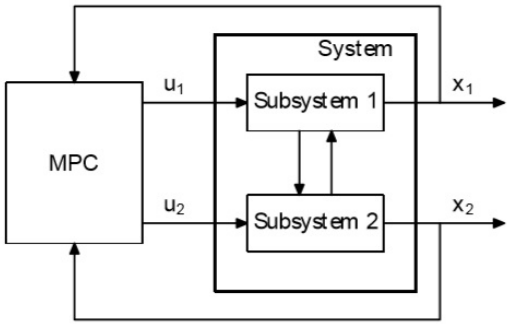

Fig. 3.2. Centralized MPC architecture.

with

$$J\left(t_k\right) = \sum_{i=1}^{m} \int_{t_k}^{t_{k+N}} \left[\|\tilde{x}_i(\tau)\|^2_{Q_{ci}} + \|u_i(\tau)\|^2_{R_{ci}} \right] d\tau$$

where $S(\Delta)$ is the family of piece-wise constant functions with sampling period Δ, N is the prediction horizon, Q_{ci} and R_{ci} are strictly positive definite symmetric weighting matrices, and $\tilde{x}_i, i = 1, \ldots, m$, are the predicted trajectories of the nominal subsystem i with initial state $x_i(t_k), i = 1, ..., m$, at time t_k. The objective of the MPC of Eq.3.9 is to achieve stabilization of the nominal system of Eq.3.9 at the origin, i.e., $(x, u) = (0, 0)$.

The optimal solution to the MPC optimization problem defined by Eq. 3.9 is denoted as $u_i^*(t \mid t_k), i = 1, \ldots, m$, and is defined for $t \in [t_k, t_{k+N})$. The first step values of $u_i^*(t \mid t_k), i = 1, \ldots, m$, are applied to the closed-loop system for $t \in [t_k, t_{k+1})$. At the next sampling time t_{k+1}, when new measurements of the system states $x_i(t_{k+1}), i = 1, \ldots, m$, are available, the control evaluation and implementation procedure is repeated. The manipulated inputs of the system under the control of the MPC of Eq. 3.9 are defined as follows:

$$u_i(t) = u_i^*(t \mid t_k), \forall t \in [t_k, t_{k+1}), i = 1, \ldots, m \tag{3.13}$$

which is the standard receding horizon scheme.

In the MPC formulation of Eq. 3.9, the constraint of Eq. 3.10 defines a performance index or cost index that should be minimized. In addition to penalties on the state and control actions, the index may also include penalties on other considerations; for example, the rate of change of the inputs. The constraint of Eq. 3.10 is the nominal model, that is, the uncertainties are supposed to be zero in the model which is used in the MPC to predict the future evolution of the process. The constraint takes into account the constraints on the control inputs, and the constraint of Eq. 3.10 provides the initial state for the MPC which is a measurement of the actual system state. Note that in the above MPC formulation, state constraints are not considered but can be readily taken into account.

Entirely a control technique that, in particular, benefits from these latest developments is model predictive control (MPC). MPC is an optimization-based control technique that uses 1) a mathematical model of a system to predict the system's behavior over a given horizon, 2) an objective function that represents what system behavior is desirable, 3) a mathematical formalization of operational constraints that have to be satisfied, 4) measurements of the state of the system at each time step, and 5) any information regarding upcoming disturbances that may be available.

Figure 3.3 illustrates the various control architectures within which MPC can be encountered. The classical architecture for MPC is a centralized control architecture. Conventionally, MPC is considered in a centralized control setting, in which the controller (sometimes also referred to as a control agent) can use measurements of the complete system and has access to a model that

describes the dynamics of the complete system. At a particular time step, MPC employs an optimization procedure to solve an optimization problem based on these components to determine a sequence of actions that are expected to steer the system in the right direction. Only the first component of this sequence is applied to the system. At the next time step, the optimization problem is solved again in a receding horizon fashion, taking into account the latest available measurement of the system's state and up-to-date information regarding disturbances.

MPC benefits greatly from advances in communication technology. As sensor information becomes more easily available, the MPC controller can take more information into account regarding a system's state. Moreover, MPC benefits greatly from advances in computational resources. Solving the MPC optimization problem at each time step takes time. As computational resources become more powerful, it takes less time to solve an MPC optimization problem. This makes it possible to improve the accuracy of the system models used for prediction or consider models of larger-scale systems.

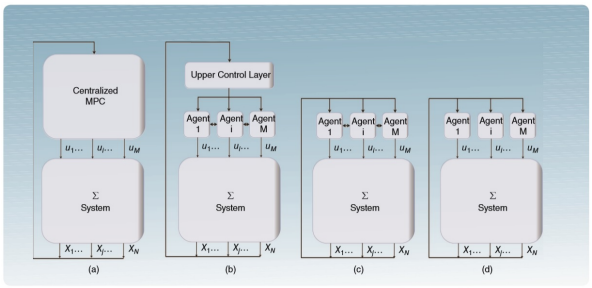

Fig. 3.3. Control architectures. (a) Centralized model-predictive control. (b) Hierarchical distributed MPC (c) Distributed MPC (d) Decentralized MPC

MPC has been popular in practical applications since its very early days. Its ability to handle complex phenomena, such as actuator constraints and multiple control objectives, in an explicit manner, while being able to take into account possible forecasts regarding disturbances and time delays in system dynamics is particularly important in refining, solar plants, or aerospace, to name a few of the many fields in which it is applied. Despite the strengths of centralized MPC, there are several issues that can prohibit the successful implementation of MPC in a centralized control setting:

1. Even though computational power has increased dramatically, the MPC optimization problem for a particular system may still not be solved fast enough, due to complexity of the relevant system dynamics. The control actions need to be computed within the control system's sampling pe-

riod, the length of which is limited by the dynamics of the system to be controlled. If the system is too complex, solving the MPC optimization problem will take too much time.
2. The structure of the system could be flexible, and it could therefore be impossible to have a constant model structure. Moreover, the uncertainty in the system structure also hinders the implementation of classical centralized or hierarchical control schemes. In this situation, it is hard to construct the centralized model of the system required for centralized MPC in the first place and maintain its validity over time in real time in the second place. This issue is particularly a concern if the number of subsystems is large and the events of connecting and disconnecting appear in a high frequency.
3. The control system infrastructure may be implemented in a way in which technical constraints regarding the transmission of information arise.
4. Besides the technical limitations, there may be constraints on the information flows. For example, consider systems that spread over large geographical areas or are owned by several entities, with each responsible for the proper functioning of a part of the system such as water systems, which may be partitioned in regions controlled by different water boards for political reasons.

It is well known that the MPC of Eq. 3.9 is not necessarily stabilizing. To achieve closed-loop stability, different approaches have been proposed in the literature. One class of approaches is to use infinite prediction horizons or welldesigned terminal penalty terms; Another class of approaches is to impose stability constraints in the MPC optimization problem. There are also efforts focusing on getting explicit stabilizing MPC laws using offline computations. However, the implicit nature of MPC control law makes it very difficult to explicitly characterize, a priori, the admissible initial conditions starting from where the MPC is guaranteed to be feasible and stabilizing. In practice, the initial conditions are usually chosen in an *ad hoc* fashion and tested through extensive closed-loop simulations.

3.3.2 Decentralized Model Predictive Control

While there are some important reviews on decentralized control (e.g., [132], [139], [140]), in this section we focus on results pertaining to decentralized MPC. The key feature of a decentralized control framework is that there is no communication between the different local controllers. A schematic of a decentralized MPC architecture with two subsystems is shown in Fig. 3.4. It is well known that strong interactions between different subsystems may prevent one from achieving stability and desired performance with decentralized control.

In general, in order to achieve closed-loop stability as well as performance in the development of decentralized MPC algorithms, the interconnections

between different subsystems are assumed to be weak and are considered as disturbances which can be compensated through feedback so they are not involved in the controller formulation explicitly. Consider the control of the system of Eq. 3.10 and assume that the state measurements of the system of Eq. 3.10 are available at synchronous sampling time instants $\{t_{k\geq 0}\}$, a typical decentralized MPC is formulated as follows:

$$\min_{u_i \in S(\Delta)} J_i\left(t_k\right) \tag{3.14}$$

$$\text{s.t. } \dot{\tilde{x}}_i(t) = f_i\left(\tilde{x}_{i-}(t)\right) + g_{si}\left(\tilde{x}_{i-}\right) u_i(t) \tag{3.15}$$

$$u_i(t) \in U_i \tag{3.16}$$

$$\tilde{x}_i\left(t_k\right) = x_i\left(t_k\right) \tag{3.17}$$

with

$$J_i\left(t_k\right) = \int_{t_k}^{t_{k+N}} \left[\left\|\tilde{x}_i(\tau)\right\|_{Q_{ci}}^2 + \left\|u_i(\tau)\right\|_{R_{ci}}^2\right] d\tau$$

where $x_{i-} = \left[0 \cdots x_i \cdots 0\right]^T$, J_i is the cost function used in each individual local controller based on its local subsystem states and control inputs.

In [189], a decentralized MPC algorithm for nonlinear discrete time systems subject to decaying disturbances was presented. No information is exchanged between the local controllers and the stability of the closed-loop system relies on the inclusion of a contractive constraint in the formulation of each of the decentralized MPCs. In the design of the decentralized MPC, the effects of interconnections between different subsystems are considered as perturbation terms whose magnitude depend on the norm of the system states. In [124], the stability of a decentralized MPC is analyzed from an input-to-state stability (ISS) point of view. In [133], a decentralized MPC algorithm was developed for large-scale linear processes subject to input constraints. In this work, the global model of the process is approximated by several (possibly overlapping) smaller subsystem models which are used for local predictions and the degree of decoupling among the subsystem models is a tunable parameter in the design.In [114], possible date packet dropouts in the communication between the distributed controllers were considered in the context of linear systems and their influence on the closed-loop system stability was analyzed.

To develop coordinated decentralized control systems, the dynamic interaction between different units should be considered in the design of the control systems. This problem of identifying dynamic interactions between units was studied in [155].

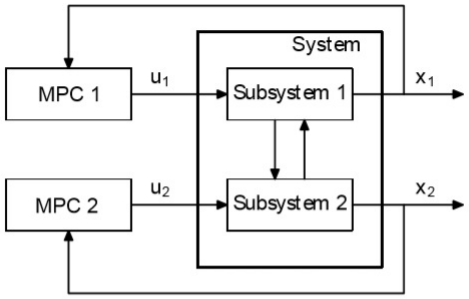

Fig. 3.4. Decentralized MPC architecture.

Within process control, another important work on the subject of decentralized control includes the development of a quasi-decentralized control framework for multi-unit plants that achieves the desired closed-loop objectives with minimal cross communication between the plant units under state feedback control [146]. In this work, the idea is to incorporate in the local control system of each unit a set of dynamic models that provide an approximation of the interactions between the different subsystems when local subsystem states are not exchanged between different subsystems and to update the state of each model using states information exchanged when communication is reestablished.

In general,the overall closed-loop performance under a decentralized control system is limited because of the limitation on the available information and the lack of communication between different controllers [116]. This leads us to the design of model predictive control architectures in which different MPCs coordinate their actions through communication to exchange subsystem state and control action information.

3.4 Distributed Model Predictive Control

In distributed MPC, there is not a single MPC controller but instead there are multiple MPC controllers, each for a particular system; see Figure 3.3. Typically, there is dynamical interaction among the systems that the individual controllers consider. Each of the controllers adopts the MPC strategy as outlined above for controlling its system but now not only considering dynamics, constraints, objectives, and disturbances of the subsystem under consideration but also considers the interactions among the systems. Each local controller

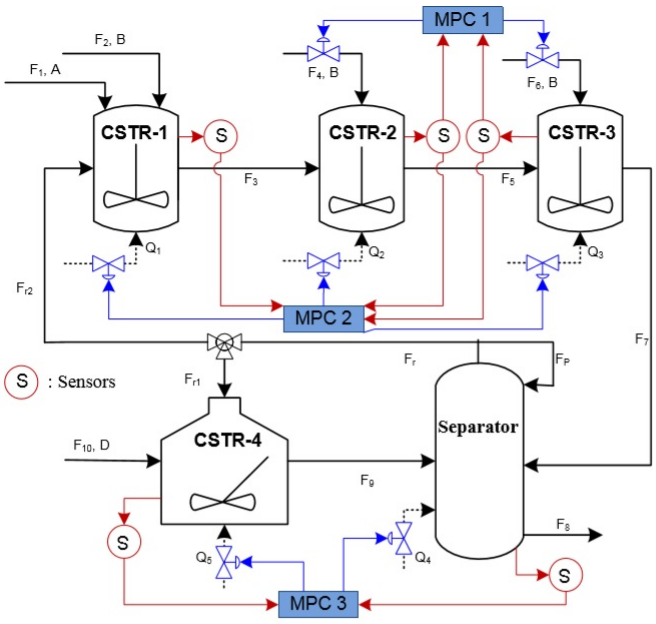

Fig. 3.5. Decentralized MPC configuration for the example chemical process

solves an MPC problem based on local information and may hereby share information with the other controllers to improve the overall performance.

When moving from a centralized MPC to a distributed MPC setting, several key concepts become relevant. In a distributed MPC setting, a system or subsystem refers to the entity being controlled by the controller. The overall system is the combination of all systems or subsystems under control merged into one large-scale system. The notions of global versus local distinguish between the overall system and the system or subsystem under control by a particular controller. Hence, frequently appearing concepts are local objectives, local dynamics, local constraints, and local disturbances. The terms *interconnecting* and *shared* are often used in combination with the terms *variables* and *constraints* to denote explicitly those components that represent the connections or couplings between different systems.

In a distributed setting, a particular controller has neighbors, or neighboring controllers. The neighbors are those controllers that control systems that are coupled or influence the system under control by this particular controller.

Communication takes places among the controllers, which can exchange information, for example, regarding local states, local objectives, and/or local constraints. Information can then be taken into account by the controllers to implement a coordination or negotiation process. The controllers can be structured in control layers or levels, leading to a hierarchical control structure. Here, typically at higher levels, controllers consider slower time scales and larger systems in a more abstract way, whereas at lower levels, controllers consider faster time scales and smaller systems in a more detailed way.

The potential of distributed MPC lies in the unique combination of the strengths of MPC (namely, feedback with feed-forward control in a receding horizon fashion, multiobjective optimization, and explicitly handling of constraints) with the negotiation and coordination possibilities provided by communication. The role played by communication in this context is essential. For example, the distinction between decentralized and distributed MPC lies in whether or not the controllers actively communicate with one another to determine actions (see Figure 3.3). In a decentralized control architecture, there is no direct communication between controllers; controllers take into account the influence of neighboring systems only by responding to the dynamics of the systems they are controlling. This constraint typically limits the performance that the decentralized MPC control scheme can achieve. In fact, in a decentralized MPC scheme, controllers may be opposing one another's actions, even though they may not have the intention to do so. In a distributed MPC setting, such a situation can be prevented because the controllers can communicate about what actions they are going to take when and obtain agreement on an optimal timing.

3.4.1 Categorizing Distributed MPC Schemes

Although many approaches for distributed MPC have been developed, there is no coherent and easily accessible overview of the components of these approaches. Such an overview would facilitate making the approaches known to a wider community and help students, researchers, and practitioners in choosing the approach most suitable for their particular application. To obtain a coherent picture of the available distributed MPC techniques, a survey of major techniques currently available has been carried out. The goal was to obtain for each approach a description, including:

1) the rationale and background of the approach considered
2) the assumptions made on the system dynamics, control objectives, and constraints
3) a step-by-step description of the computations and equations involved
4) the possible availability of theoretical results
5) the possible availability of real or simulated applications.

To achieve this goal, experts were asked to answer five sets of questions about their particular distributed MPC approach. The five sets of questions,

given in Table 3.6, covered background, boundary conditions, a step-by-step description of the approach, theoretical results, and application results. This survey resulted in detailed descriptions of 35 approaches to distributed MPC. Below, the main characteristics of these techniques are presented as well as a summarizing overview of the distribution of the techniques over the various categories. The details of the investigation can be found in [104].

TABLE 1 An overview of the questions part of the inquiry on distributed model-predictive control approaches.

Background
- What is the rationale behind the approach?
- From where did it originate?
- What makes this approach interesting?

Boundary Conditions
- What kind of system partition and type of dynamical model is assumed?
- What kind of control problem is being solved for this kind of system?
- What kind of communication architecture is assumed available?

Step-by-Step Description
- What initialization is required?
- What equation and optimization is used when?
- When does what communication take place with which controller?
- When do controllers agree on/decide to take an action?

Availability of Theoretical Results
- What theoretical properties have been investigated?
- How does the approach relate to other existing approaches?
- Where have results of this been published?

Availability of Actual Applications
- What are the systems in which your approach has been tested?
- Where have results been published?

Fig. 3.6. TABLE 1 An overview of the questions part of the inquiry on distributed model-predictive control approaches.

To achieve better closed-loop control performance, some level of communication may be established between the different controllers, which leads to distributed model predictive control (DMPC). With respect to available results in this direction, several DMPC methods have been proposed as well as some important review articles [129],[134] have been written which primarily focus the review of the various DMPC schemes at a conceptual level. With respect to the DMPC algorithms available in the literature, a classification can be made according to the topology of the communication network, the

different communication protocols used by local controllers, and the cost function considered in the local controller optimization problem. In the following, we will classify the different algorithms based on the cost function used in the local controller optimization problem as used in [129]. Specifically, we will refer to the distributed algorithms in which each local controller optimizes a local cost function as *non-cooperative* DMPC algorithms, and refer to the distributed algorithms in which each local controller optimizes a global cost function as cooperative DMPC algorithms.

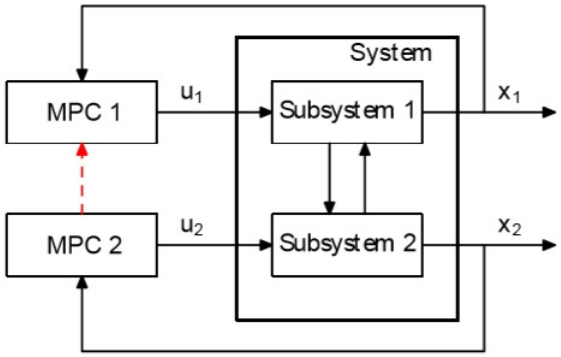

Fig. 3.7. Sequential DMPC architecture.

In a DMPC algorithm was proposed for a class of decoupled systems with coupled constraints. This class of systems captures an important class of practical problems, including, for example, maneuvering a group of vehicles from one point to another while maintaining relative formation and/or avoiding collisions. In the distributed controllers are evaluated in sequence which means that controller $i + 1$ is evaluated after controller i has been evaluated or vice versa. A sequential DMPC architecture with two local controllers is shown in Fig. 3.7 An extension of this work proposes the use of the robust design method described in [98] for DMPC.

3.4.2 Comparing Distributed MPC Approaches

Many categorizations could be made and many different features examined. The goal of the categorization in this article is to attain a trade-off between the complexity of highlighting all the particularities that characterize a particular

distributed MPC scheme and the simplicity of an overview of the distributed MPC field, which is necessary for the newcomers to the field. The particular categorization chosen here considers the perspective of a potential distributed MPC user: Starting from a system, which distributed MPC schemes could be appropriate? With this approach, the overview will be most beneficial. As a result, and based on the analysis of the distributed MPC schemes, three groups of features for categorizing distributed MPC approaches are proposed: 1) process features, 2) control architecture features, and 3) theoretical property features, as illustrated in Figure 3.8 The proposed categorization is by no means exhaustive but still considers six possible subcategories related with the process and its associated control problem, which is likely the starting point for those looking for a distributed MPC solution for a particular problem; the main architectural differences regarding the schemes are captured in seven subcategories. Finally, results regarding four common theoretical properties are also analyzed so that readers know if the demonstration of these properties has been formally addressed. In all, there are following possible features presented in the proposed categorization.

1) *Process features*: features related to the specifications of the physical system to be controlled

- System type: the way in which the system considered has been constructed: either starting from a group of autonomous systems and then introducing communication to obtain coordination or from a monolithic system decomposed into subsystems that are coordinated taking into account limitations in communication/processing power.
- Process type: the way in which the dynamics of the behavior of the system are considered to be best captured: linear, nonlinear, or hybrid.
- Type of model: the way in which the system model is described mathematically: transfer function or state space.
- Type of control: the control goal: regulation (that is, to keep the system stable at or bring the system to a particular given state), tracking (that is, to have the system follow a given reference), or economic (where the optimization is focused on an economic cost function).
- Coupling source: what makes the overall system nonseparable: inputs, outputs, states, objectives, or constraints.

2) *Control architecture features*: features related to the essence of the control scheme

- Architecture: the way in which the coordination between local controllers is structured: decentralized, distributed, or hierarchical (see Figure 3.3). In general, the controllers can be categorized depending on how many of them participate in the solution of the control problem and the relative importance between them. The control system is considered decentralized when there are local controllers controlling local subsystems of the plant

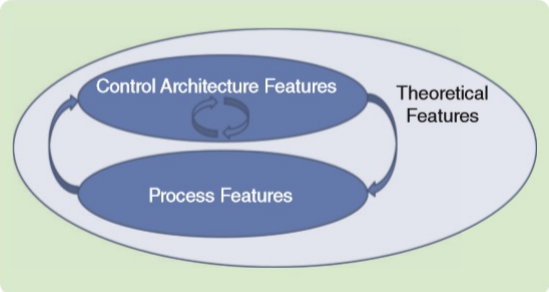

Fig. 3.8. An overview of groups of features considered.

with no required communication among controllers. When the local controllers communicate to find a cooperative solution for the overall control problem, the control system is considered distributed. If there are multiple control layers being coordinated to control the process, the control system is considered hierarchical. In this case, higher layers manage the global objectives of the process and provide reference signals for the lower layers, which control the plant directly.

- Controller knowledge: the type of information that a controller has: strictly local or partially global.
- Computation type: the way in which joint actions are calculated, in an iterative or noniterative fashion.
- Controller's attitude: the way in which an agent takes into account other agents' objectives: noncooperative or cooperative. In general, attitude is related to collaboration between subsystems. A controller is considered to have a noncooperative attitude if it behaves selfishly, that is, it only seeks the maximization of its own objective function. On the other hand, the controller's attitude is considered cooperative when it minimizes not only its cost but the system-wide cost. Hence, the controller may make sacrifices in terms of its own welfare to help the overall system attain a better global situation.
- Communication: the way in which the agents transmit and receive information: serial or parallel. In particular, under the serial communication paradigm, only one controller is considered to communicate at the same time, in contrast to parallel communication, in which case several controllers are considered to communicate at the same time.
- Timing: whether or not there is a strict schedule in the communication process that determines when controllers can communicate: synchronous or asynchronous.
- Optimization variables: the nature of the variables in the optimization problem: real or integer valued.

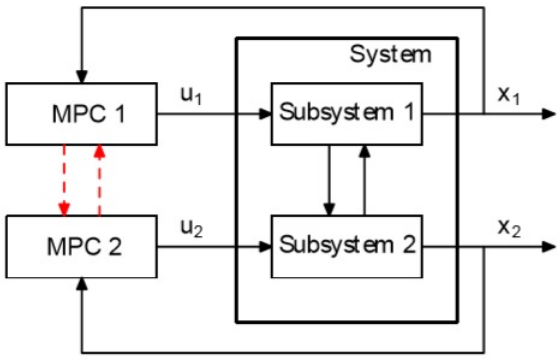

Fig. 3.9. Parallel DMPC architecture.

3) *Theoretical features*: features related to the availability of results that provide a certain formal guarantee regarding the approach's performance:

- Optimality: whether or not the scheme provides the same result as the corresponding centralized optimization problem.
- Suboptimality bounds: whether or not the scheme provides a measurement of the distance with respect to the optimum of the corresponding centralized optimization problem.
- Stability: whether or not the scheme guarantees a nondivergent evolution of the state and the output of the system.
- Robustness: whether or not the scheme is able to reject external unknown disturbances.

In the majority of the algorithms in the category of noncooperative DMPC, the distributed controllers are evaluated in parallel i.e., at the same time. The controllers may be only evaluated once (non-iterative) or iterate (iterative) to achieve a solution at a sampling time. A parallel DMPC architecture with two local controllers is shown in Fig. 3.9. Many parallel DMPC algorithms in the literature belong to the non-iterative category. In a DMPC algorithm proposed for a class of discrete-time linear systems. In this concept, a stability constraint is included in the problem formulation and the stability can be verified a posteriori with an analysis of the resulting closed-loop system. In DMPC for systems with dynamically decoupled subsystems (a class of systems of relevance in the context of multiagents systems) where the cost function and constraints couple the dynamical behavior of the system. The coupling in the system is described using a graph in which each subsystem is a node. It is assumed that each subsystem can exchange information with its neigh-

bors (a subset of other subsystems). Based on the results of [114], a DMPC framework was constructed for control and coordination of autonomous vehicle teams[115].

3.4.3 Cooperative and Noncooperative DMPC algorithm

The key feature of cooperative DMPC is that in each of the local controllers, the same global cost function is optimized. In recent years, many efforts have been made to develop cooperative DMPC for linear and nonlinear systems. In cooperative DMPC each controller takes into account the effects of its inputs on the entire plant through the use of a centralized cost function. At each iteration, each controller optimizes its own set of inputs assuming that the rest of the inputs of its neighbors are fixed to the last agreed value.Subsequently,the controllers share the resulting optimal trajectories and a final optimal trajectory is computed at each sampling time as a weighted sum of the most recent optimal trajectories with the optimal trajectories computed at the last sampling time.

The cooperative DMPCs use the following implementation strategy:

1. At k, all the controllers receive the full state measurement $x(k)$ from the sensors.
2. At iteration $c(c \geq 1)$:
   2.1.Each controller evaluates its own future input trajectory based on $x(k)$ and the latest received input trajectories of all the other controllers (when $c = 1$, initial input guesses obtained from the shifted latest optimal input trajectories are used).
   2.2.The controllers exchange their future input trajectories. Based on all the input trajectories, each controller calculates the current decided set of inputs trajectories u^c.
3. If a termination condition is satisfied, each controller sends its entire future input trajectory to its actuators; if the termination condition is not satisfied, go to Step 2($c \leftarrow c + 1$).
4. When a new measurement is received, go to Step 1 ($k \leftarrow k + 1$).

At each iteration, each controller solves the following optimization problem:

$$\min_{u_i(k),\ldots,u_i(k+N-1)} J(k) \tag{3.18}$$

subject to

$$x(k+1) = Ax(k) + Bu(k) + w(k)$$

with $w = 0$ and, for $j = 0, \ldots, N-1$,

$$u_i(k+j) \in U_i, j \geq 0 \tag{3.19}$$

$$u_l(k+j) = u_l(k+j)^{c-1}, \forall l \neq i \tag{3.20}$$

$$x(k+j) \in X, j > 0 \tag{3.21}$$

$$x(k+N) \in X_f \tag{3.22}$$

with

$$J(k) = \sum_i J_i(k) \tag{3.23}$$

and

$$J_i(k) = \sum_{j=0}^{N-1} \left[\|x_i(k+j)\|^2_{Q_i} + \|u_i(k+j)\|^2_{R_i} \right] + \|x(k+N)\|^2_{P_i} \tag{3.24}$$

Note that each controller must have knowledge of the full system dynamics and of the overall objective function. After the controllers share the optimal solutions $u_i(k+j)^*$, the optimal trajectory at iteration c, $u_i(k+j)^c$, is obtained from a convex combination between the last optimal solution and the current optimal solution of the MPC problem of each controller, that is,

$$u_i(k+j)^c = \alpha_i u_i(k+j)^{c-1} + (1-\alpha_i)\, u_i(k+j)^* \tag{3.25}$$

where α_i are the weighting factors for each agent. This distributed optimization is of the Gauss-Jacobi type.

In an iterative cooperative DMPC algorithm which was designed for linear systems. It was proven that through multiple communications between distributed controllers and using system-wide control objective functions,stability of the closed-loop system can be guaranteed for linear systems, and the closed-loop performance converges to the corresponding centralized control system as the iteration number increases. A design method to choose the stability constraints and the cost function is given that guarantees feasibility (given an initial feasible guess), convergence and optimality (if the constraints of the inputs are not coupled) of the resulting distributed optimization algorithm. In addition, the stability properties of the resulting closed-loop system, output feedback implementations and coupled constraints are also studied.

As an example of a noncooperative DMPC algorithm for discrete-time systems described by

$$x_i(k+1) = A_{ii}x_i(k) + \sum_{i \neq j} A_{ij}x_j(k) + B_i u_i(k) + w_i(k) \tag{3.26}$$

we now synthetically describe the method recently proposed in relying on the "tube-based" approach developed in [128] for the design of robust MPC. The rationale is that each subsystem i transmits to its neighbors its planned state reference trajectory $\tilde{x}_i(k+j), j = 1, \ldots, N$, over the prediction horizon and "guarantees" that, for all $j \geq 0$, its actual trajectory lies in a neighborhood of $\tilde{x}_i$, i.e. $x_i(k+j) \in \tilde{x}_i(k+j) \oplus \mathcal{E}_i$, where $\mathcal{E}_i$ is a compact set including the origin. In this way, the above equation can be written as

$$x_i(k+1) = A_{ii}x_i(k) + B_i u_i(k) + \sum_j A_{ij}\tilde{x}_j(k) + w_i(k) \tag{3.27}$$

where $w_i(k) = \sum_j A_{ij}\left(x_j(k) - \tilde{x}_j(k)\right) \in W_i$ is a bounded disturbance, $W_i = \oplus_j A_{ij}\mathcal{E}_i$ and the term $\sum_j A_{ij}\tilde{x}_j(k)$ can be interpreted as an input, known in advance over the prediction horizon. Note that in this case, we assume that the only disturbance of each model is due to the mismatch between the planned and real state trajectories.

3.4.4 Sequential and Iterative DMPC

In a sequential DMPC architecture shown in Figure 3.10 for fully coupled nonlinear systems was developed based on the assumption that the full system state feedback is available to all the distributed controllers at each sampling time. In the proposed sequential DMPC, for each set of the control inputs u_i, a Lyapunov-based MPC (LMPC), denoted LMPC i, is designed. The distributed LMPCs use the following implementation strategy:

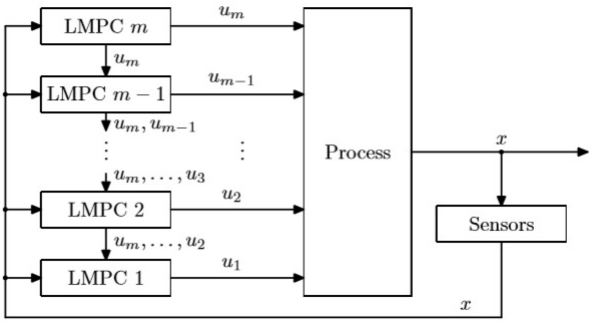

Fig. 3.10. Sequential DMPC architecture using LMPC.

1) At t_k, all the LMPCs receive the state measurement $x\left(t_k\right)$ from the sensors.

2) For $j = m$ to 1
   2.1. LMPC j receives the entire future input trajectories of $u_i, i = m, \ldots, j+1$, from LMPC $j+1$ and evaluates the future input trajectory of u_j based on $x(t_k)$ and the received future input trajectories.
   2.2. LMPC j sends the first step input value of u_j to its actuators and the entire future input trajectories of u_i, $i = m, ..., j$, to LMPC $j1$.
3) When a new measurement is received ($k \leftarrow k+1$), go to Step 1.

In this architecture, each LMPC only sends its future input trajectory and the future input trajectories it received to the next LMPC (i.e., LMPC j sends input trajectories to LMPC $j1$). This implies that LMPC $j, j = m, ..., 2$, does not have any information about the values that u_i, $i = j1, ..., 1$ will take when the optimization problems of the LMPCs are designed. In order to make a decision, LMPC $j, j = m, ..., 2$ must assume trajectories for u_i, $i = j1, ..., 1$, along the prediction horizon. To this end, an explicit nonlinear control law $h(x)$ which can stabilize the closed-loop system asymptotically is used. In order to inherit the stability properties of the controller $h(x)$, a Lyapunov function based constraint is incorporated in each LMPC to guarantee a given minimum contribution to the decrease rate of the Lyapunov function $V(x)$. Specifically, the design of LMPC $j, j = 1, ..., m$, is based on the following optimization problem:

$$\min_{u_j \in S(\Delta)} J(t_k) \tag{3.28}$$

$$\text{s.t. } \dot{\tilde{x}}(t) = f(\tilde{x}(t)) + \sum_{i=1}^{m} g_i(\tilde{x}(t)) u_i \tag{3.29}$$

$$u_i(t) = h_i(\tilde{x}(t_{k+l})), i = 1, \ldots, j-1 \forall t \in [t_{k+l}, t_{k+l+1}), l = 0, \ldots, N-1 \tag{3.30}$$

$$u_i(t) = u^*_{s,i}(t \mid t_k), i = j+1, \ldots, m \tag{3.31}$$

$$u_j(t) \in U_j \tag{3.32}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3.33}$$

Ina Lyapunov-based iterative DMPC algorithm which is shown in Figure 3.11 was proposed for coupled nonlinear systems. The implementation strategy of this iterative DMPC is as follows:

1. At t_k, all the LMPCs receive the state measurement $x(t_k)$ from the sensors and then evaluate their future input trajectories in an iterative fashion with initial input guesses generated by $h(\cdot)$.

2. At iteration $c(c \geq 1)$:
   2.1. Each LMPC evaluates its own future input trajectory based on $x(t_k)$ and the latest received input trajectories of all the other LMPCs (when $c = 1$, initial input guesses generated by $h(\cdot)$ are used).
   2.2. The controllers exchange their future input trajectories. Based on all the input trajectories, each controller calculates and stores the value of the cost function.
3. If a termination condition is satisfied, each controller sends its entire future input trajectory corresponding to the smallest value of the cost function to its actuators; if the termination condition is not satisfied, go to Step $2(c \leftarrow c + 1)$.
4. When a new measurement is received, go to Step $1(k \leftarrow k + 1)$.

Note that at the initial iteration, all the LMPCs use $h(x)$ to estimate the input trajectories of all the other controllers. Note also that the number of iterations c can be variable and it does not affect the closed-loop stability of the DMPC architecture presented in this subsection. For the iterations in this DMPC architecture, there are different choices of the termination condition. For example, the number of iterations c may be restricted to be smaller than a maximum iteration number $c_{\max}$ (i.e., $c \leq c_{\max}$) and/or the iterations may be terminated when the difference of the performance or the solution between two consecutive iterations is smaller than a threshold value and/or the iterations maybe terminated when a maximum computational time is reached.

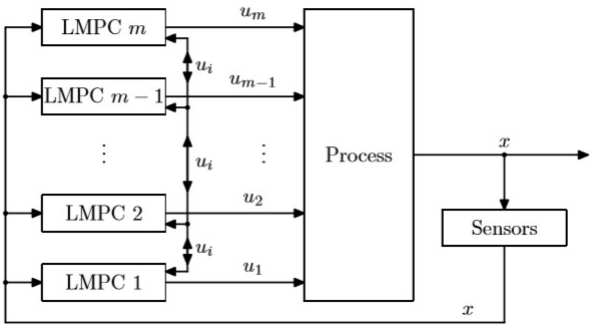

Fig. 3.11. Iterative DMPC architecture using LMPC.

In general, there is no guaranteed convergence of the optimal cost or solution of an iterated DMPC to the optimal cost or solution of a centralized MPC for general nonlinear constrained systems because of the non-convexity of the MPC optimization problems. However, with the above implementation

strategy of the iterative DMPC presented in this section, it is guaranteed that the optimal cost of the distributed optimization is upper bounded by the cost of the Lyapunov-based controller $h()$ at each sampling time.

We review next a line of work on DMPC algorithms which adopt an iterative approach for constrained linear systems coupled through the inputs. Figure 3.12 shows a scheme of this class of controllers. Note that there is one agent for each subsystem and that the number of controlled inputs may differ from the number of subsystems. In this class of controllers, the controllers (agents, in general) do not have any knowledge of the dynamics of any of its neighbors, but can communicate freely among them in order to reach an agreement. The proposed strategy is based on negotiation between the agents. At each sampling time, following a given protocol, agents make proposals to improve an initial feasible solution on behalf of their local cost function, state and model. These proposals are accepted if the global cost improves the cost corresponding to the current solution.

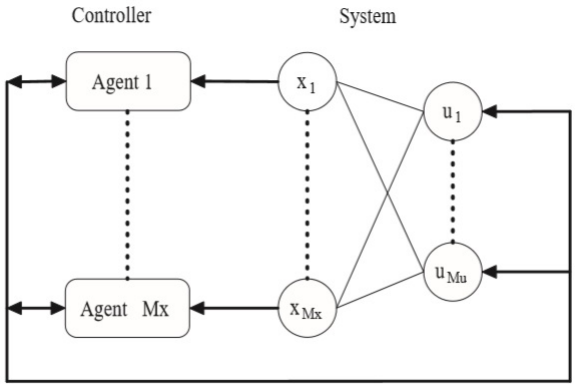

Fig. 3.12. DMPC based on agent negotiation.

3.5 Decompositions for DMPC

An important and unresolved in its generality issue in DMPC is how to decompose the total number of control actuators into small subsets, each one of them being controlled by a different MPC controller. There have been several ideas for how to do this decomposition based on plant layout considerations as well as via time-scale considerations. Below, we review some of these de-

compositions.

Partitioning and decomposition of a process into several subsystems is an important topic. The recent works describes the design of a network-based DMPC system using multirate sampling for large-scale nonlinear systems composed of several coupled subsystems. A schematic of the plant decomposition and of the control system is shown in Fig. 3.13. Furthermore, the assumption is made that there is a distributed controller associated with each subsystem and the distributed controllers are connected through a shared communication network. At a sampling time in which slowly and fast sampled states are available, the distributed controllers coordinate their actions and predict future input trajectories which, if applied until the next instant that both slowly and fast sampled states are available, guarantee closed-loop stability. At a sampling time in which only fast sampled states are available, each distributed controller tries to further optimize the input trajectories calculated at the last instant in which the controllers communicated, within a constrained set of values to improve the closed-loop performance with the help of the available fast sampled states of its subsystem.

In the process industry, the control structure is usually organized in a number of different layers. At the bottom level, standard PI-PID regulators are used for control of the actuators, while at a higher layer MPC is usually applied for set-point tracking of the main control variables. Finally, at the top of the hierarchy, optimization is used for plant wide control with the scope of providing efficient, cost-effective, reliable, and smooth operation of the entire plant. Recent results on the design of two-level control systems designed with MPC and allowing for reconfiguration of the control structure have also been reported in[142]. As an additional remark, it is worth mentioning that a recent stream of research is devoted to the so-called economic MPC, with the aim to directly use feedback control for optimizing economic performance, rather than simply stabilizing the plant and maintaining steady operation.

In addition to the development of the composite control system of Fig. 3.13, the singular perturbation framework can be also used to develop composite control systems where an MPC controller is used tin the fast time scale. In this case, a convenient way from a control problem formulation point of view is to design a fast MPC that uses feedback of the deviation variable y in which case u_f is only active in the boundary layer (fast motion of the fast dynamics) and becomes nearly zero in the slow timescale. The resulting control architecture in this case is shown in Figure 3.14 where there is no need for communication between the fast MPC and the slow MPC; in this sense, this control structure can be classified as decentralized.

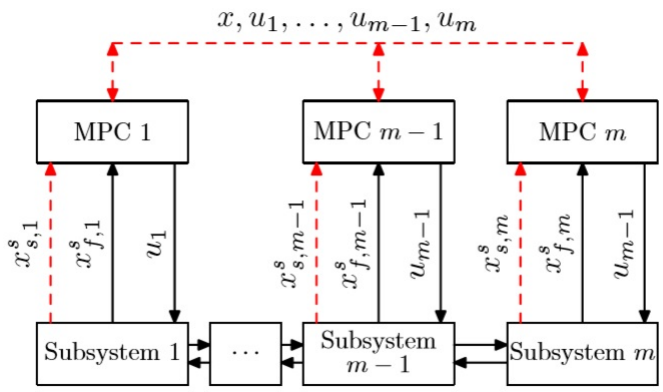

Fig. 3.13. Multirate DMPC architecture.

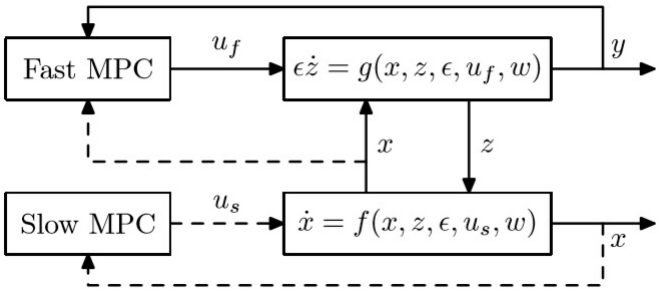

Fig. 3.14. A schematic of a composite control system using MPC in both the fast and slow time-scales.

3.6 Future Research Directions of Distributed MPC

Despite the fact that the field has seen strong research activity over the last decade, new schemes still appear in the literature now and then. In addition, enhancements of previous schemes are also common, such as guarantees of new theoretical properties. Likely directions for future research are outlined below. A topic that deserves more attention is the development of flexible distributed MPC architectures able to modify the control network topology and the communication burden depending on the circumstances. The rationale behind these control schemes is to foster cooperation whenever the system performance is poor and to reduce cooperation when it is not necessary. That is, communication is only allowed whenever it significantly improves the

system performance. The controllers are then grouped into time-varying coalitions that work cooperatively. Likewise, another related field where research is needed is that of plug-and-play systems, that is, control schemes capable of seamlessly handling controllers that enter or leave the system, perhaps due to maintenance or because the system size dynamically changes.

Most of the available DMPC schemes rely on the assumption of availability of measurements of the complete state vector. However, it is possible that a distributed controller in a large-scale control system may not have access to all the measurements or that measurements of all the process states are not available. In this case, in the design of the distributed controllers, we need to take into account that different distributed controllers may have access to measurements of different parts of the process states, so that methods for DMPC with partial state measurements are required. Future research in this direction should take advantage of the available distributed state estimation schemes reviewed and should look at how best the combination of a DMPC algorithm with centralized/distributed state estimators can be addressed. One approach is to design a different state observer for each controller (i.e., distributed state estimation), while an alternative approach is to design a centralized observer that sends the estimated state to all the distributed controllers. In this context, the integration of the state estimation schemes with the DMPC algorithms so that desired levels of stability, performance and robustness are attained in the closed-loop system should be rigorously studied.

Most industrial process control applications are based on hierarchical control schemes in which first the operation point of the plant is determined based on economic, safety and environmental considerations (usually using steady state models), and then process control systems are used to drive the plant to this steady state (usually using dynamic models). It is clear that, for large scale systems, DMPC may be an appropriate path to tackle the resulting economic optimization problem. Furthermore, DMPC stands to benefit from distributed optimization schemes that are tailored to handle DMPC optimization problems in an optimal fashion accounting for control-loop decomposition, plant variable interaction patterns and controller communication strategies. Research in this direction may start from the extension to the distributed case of well-known techniques for centralized MPC, such as the multiple shooting method.

Hybrid systems constitute an important class of mathematical models that explicitly account for the intricate coupling between continuous dynamics and discrete events. While there has been extensive work over the last fifteen years on analysis and control of hybrid systems and the references therein), distributed MPC of hybrid systems is a research topic that has received no attention. In the context of chemical process control and operations, due to

changes in raw materials, energy sources, product specifications and market demands, and abrupt actuator and sensor faults, it is possible to describe process behavior with classes of switched nonlinear systems that involve differential equation models whose right-hand-side is indexed with respect to different modes of operation. From a controller design standpoint, in order to achieve closed-loop stability, discrete mode transition situations should be carefully accounted for in the control problem formulation and solution. In order to achieve mode transitions in an optimal setting and accommodate input/state constraints, distributed model predictive control (MPC) framework can be employed, particularly in cases where the computational complexity of a centralized MPC may significantly increase as the number of operational modes, control inputs and states increases.

Monitoring and reconfiguration of DMPC is an important research topic. DMPC systems offer a vast set of possibilities for reconfiguration in the event of sensor and actuator faults to maintain the desired closed-loop performance. In addition to a monitoring method, appropriate DMPC reconfiguration (fault-tolerant control) strategies were designed to handle the actuator faults and maintain the closed-loop system state within a desired operating region. There is certainly a lot more to be done in the context of DMPC monitoring and fault-tolerance. Furthermore, in addition to its importance in the context of DMPC fault-tolerance, reconfigurability of DMPC could provide flexibility to the control system and could be explored in the context of other areas as follows. During steady-state operation, it is not necessary to continuously transmit among the distributed estimation/control agents. In fact, in the case where one system does not receive any new information (and can be sure that no transmission faults have occurred), it can be assumed that the other agents basically maintain their previous state. This reduction of the information transmitted can be particularly significant in sensor networks with local power supply in order to have significant energy savings, which could guarantee a longer "life" of the sensors and/or of the actuators.

In addition, moving from industrial plants to very large-scale systems, such as transportation or distribution net works, or to the so-called "System-of-Systems" (i.e., very large-scale infrastructures of interacting subsystems, which are by themselves composed of large-scale and complex systems; with operational and managerial independence, it is clear that the problem of reconfiguration of the control system is fundamental to cope with changing requirements. For these systems, also the problem of partitioning and clustering is a very important one (recent work can be found in [118]). In general, there is a substantial lack of methodologies for appropriate temporal and spatial partitions and for the development of consistent multi-level, multi-scale models for DMPC design.

4

A Novel swarm-based distributed MPC architecture

Starting from the premilinary ideas outlined in previous chapters, a novel distributed MPC scheme will be developed for autonomous multi-agents systems subject to input/state constraints, obstacle avoidance and formation requirements. Specifically, the group of agents is organized as a finite set of swarms and a leader-follower topology is imposed to these configurations with the aim to define a sort of priority among them. The key idea is as follows: the leader swarm (for the sake of simplicity consisting of a single agent) is in charge to compute the tube of trajectories towards the target position and to detect possible obstacle occurrences within the unknown environment. Then, the follower swarm receives the hyper-ball where its father swarm lies and uses this information at each time instant as a local target. This reasoning repeats for the other swarms. Such a strategy is made viable in virtue of the following property concerning with the swarm kinematics: in a finite time each agent converges to a pre-assigned hyper-ball. Then, it is always possible to compute a reference kinematic state trajectory towards the hyper-ball of the father swarm along the platoon chain. As a matter of fact, this *modus operandi* has the main merit to avoid the use of perception modules on the follower agents because the computations are executed by the leader agent. Moreover, feasibility and asymptotic closed-loop stability are formally proved. In the following Section the constrained path planning problem under collision avoidance requirements will be stated.

4.1 Problem formulation

We consider a team of L vehicles described by

$$\dot{\tilde{x}}(t) \in \left(\sum_{k=1}^{\Gamma} \mu_k \, \mathrm{Vert}\{\Omega\}_k \right) \left[\tilde{x}^T(t), \tilde{u}^T(t) \right]^T \tag{4.1}$$

with the input torque subject to point-wise constraints:

$$|\tilde{u}_i| < \tilde{u}_{i,\max}[Nm], i = 1, 2$$

organized as the Leader-Follower configuration of Fig. 2.10 with a unique path connecting each follower to the leader. Moreover, for each follower the following definitions are exploited

- *tree level*:
$$level(k) : \{1, \ldots, L\} \to \mathcal{Z}_+ \tag{4.2}$$
which provides the position of the $k-th$ agent along the LF configuration;
- *set of neighbours*:
$$\mathcal{N}^i := \{q \in \{1, \ldots, i-1, i+1, \ldots, L\} : level(q) \equiv level(i)\} \tag{4.3}$$

Let us consider the described Leader-Follower team with multi-model discrete-time linear descriptions for the LF team, obtained by discretizing (4.1)) via the forward Euler difference method:

$$x^i(t+1) = \Phi^i(\alpha(t))x^i(t) + G^i(\alpha(t))u^i(t), i = 1, \ldots, L, \tag{4.4}$$

where $t \in \mathcal{Z}_+ := \{0, 1, ...\}$, $x^i(t) \in \mathcal{R}^n$ denotes the state plant and $u^i(t) \in \mathbb{R}^m$ the control input. The time-varying vector $\alpha(t) \in \mathbb{R}^p$ belongs to the unit simplex

$$\mathcal{P} := \left\{ \alpha \in \mathbb{R}^\Gamma : \sum_{k=1}^{\Gamma} \alpha_k = 1, \alpha_k \geq 0 \right\} \tag{4.5}$$

whereas the system matrices $(\Phi^i(\alpha),\ G^i(\alpha))$ lie in

$$\Omega(\mathcal{P}) := \left\{ \left(\Phi^i(\alpha), G^i(\alpha)\right) = \sum_{k=1}^{\Gamma} \alpha_k \left(\left(\Phi^i\right)^k, \left(G^i\right)^k \right), \alpha \in \mathcal{P} \right\} \tag{4.6}$$

with $((\Phi^i)^k, (G^i)^k)$ the polytope vertices of $\Sigma(\mathcal{P})$.

Moreover, state and input constraints are recast as follows:

$$x^i(t) \in \mathcal{X}^i, u^i(t) \in \mathcal{U}^i, \forall t \geq 0, i = 1, \ldots, L, \tag{4.7}$$

with $\mathcal{X}^i$ and $\mathcal{U}^i$ compact and convex subsets of $\mathcal{R}^n$ and $\mathcal{R}^m$, respectively.

By assuming that the Leader-Follower team moves within unknown environments where obstacles may obstruct the nominal path, at each time instant $t \in \mathcal{Z}_+$ time-varying obstacle scenarios $\mathcal{O}^t$ and nonconvex obstacle-free regions

$$\mathcal{O}^t_{free} := \{\zeta \in \mathbb{R}^2 : \zeta \in f(\zeta, t)\}, \tag{4.8}$$

with $f(\zeta, t)$ having the same structure of $f(\zeta)$, come out. Moreover, state and input constraints are recast as follows:

$$x^i(t) \in \mathbb{X}^i,\ u^i(t) \in \mathbb{U}^i,\ \forall t \geq 0,\ i = 1, \ldots, L, \tag{4.9}$$

with $\mathcal{X}^i$ and $\mathcal{U}^i$ compact and convex subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively.

Starting from these premises, Then, the problem to solve can be stated as follows.

Leader-Follower Obstacle Avoidance Regulation (LF-OA-R) Problem:

Given a LF configuration of the discussed agents , determine a distributed state-feedback control policy

$$\begin{array}{l} u^1(t) = g(x^1(t)), \\ u^i(t) = g(x^i(t), \{x^k(t)\}),\ k \in \mathcal{N}^i,\ i = 2, \ldots, L, \end{array} \tag{4.10}$$

compatible with (4.9) such that, starting from admissible initial conditions $x^i(0)$, $i = 1, \ldots, L$, and despite any admissible obstacle scenario $\mathcal{O}^t$, $\forall t \geq 0$, the LF team is driven towards prescribed target positions x_f^i, $i = 1, \ldots, L$, such that

$$\max_{i=1,\ldots,L} \|x^i(\infty) - x_f^i\|_2$$

is minimized and

$$x_f = [x_f^{1^T}, x_f^{2^T}, \ldots, x_f^{L^T}]^T$$

regardless of any admissible obstacle scenario $\mathcal{O}^t$, $\forall t \geq 0$.

The problem will be addressed by developing a control framework based on two key ingredients:

1) a swarm characterization of the basic LF topology;
2) the formalization of an *ad-hoc* distributed MPC scheme.

The idea we would develop consists in exploiting relevant properties of the kinematic swarm evolution in order to infer information usable in a receding horizon fashion by underlying MPC controllers. Specifically, the leader is in charge to inspect the surrounding environment and to compute a *minimum-time* trajectory, while the reaming elements of the team are hypothesized to be *blind*, i.e. no sensors are mounted or exploited during the on-line operations.

To this end, an important aspect to be formally taken into account concerns with the time-varying nature of the robot multi-model description (4.1). In fact, as underlined in previous Section its validity is confined to the state trajectory tube computed under the hypotheses that the time interval Δt is exactly known and no obstacles occur. As a result of the latter, such a plant description should be on-line updated in order to comply with possibly time-varying obstacle scenarios. According to this reasoning, the next two sections will be devoted to formalize the vehicle team modelling.

4.2 The continuous-time swarm kinematics solution

Let us consider a swarm composed of n robots with a fixed network topology described by an undirected graph $\mathcal{G} = \{V, E\}$ modeling the local interaction among robots. G is encoded by the Laplacian matrix $\mathcal{L}(\mathcal{G}) = \Delta(\mathcal{G}) - A(\mathcal{G})$, where $\Delta(\mathcal{G})$ is the diagonal degree matrix and $A(\mathcal{G})$ is the adjacency matrix. To simplify the notation we will refer to it as L dropping its dependence on G unless strictly required. Furthermore, the second smallest eigenvalue λ_2, namely the algebraic connectivity, provides an information regarding the connectedness of the graph. The pose of the i-th robot in a m-dimensional Euclidean space is denoted by $x_i \in \mathbb{R}^m$. The following interaction dynamics is considered for each robot i:

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i} g\left(x_i - x_j\right), \quad i = 1, \ldots, n \tag{4.11}$$

where $g(\cdot)$ is the interaction function representing the function of attraction and repulsion between neighboring robots and $\mathcal{N}_i = \{j \in V : (i, j) \in E\}$ is the neighborhood of the agent i.

In the sequel, the LF team is reorganized as a platoon of r swarm aggregations $\{SW_j\}_{j=1}^{r}$, each one collecting the agents having the same tree level position, and the agents will be denoted as Σ_j^i, with j accounting for $level(\cdot)$. For the sake of clarity, the state space description is without loss of generality accordingly modified:

$$\Sigma_j^i : \begin{cases} x_j^i(t+1) = A_j^i x_j^i(t) + B_j^i u_j^i(t) \\ i = 1, \ldots, l_j, j = 1, \ldots, r \end{cases} \tag{4.12}$$

with $x_j^i \in \mathcal{X}_j^i \subset \mathbb{R}^n$ and $u_j^i \in \mathcal{U}_j^i \subset \mathbb{R}^m$.

Let $SW_j := \{\Sigma_j^i\}_{i=1}^{l_j}$ be a swarm aggregation of l_j agents, then the continuous-time kinematic evolution of each agent Σ_j^i be governed by

$$\dot{z}_j^i(t) = -\left(z_j^i(t) - \mathcal{G}_j\right) + \frac{m_j}{l_j} \sum_{k=1}^{l_j} \frac{z_j^i(t) - z_j^k(t)}{||z_j^i(t) - z_j^k(t)|| + \epsilon}, i = 1, \ldots, l_j, \tag{4.13}$$

with $\epsilon > 0$, $m_j > 0$ and $\mathcal{G}_j$ a constant reference to denoting the goal of the swarm centroid $c_j(t) := \frac{1}{l_j} \sum_{k=1}^{l_j} z_j^k(t)$.

Lemma 4.1. *Let $z_j^i(t_0)$, $i = 1, \ldots, l_j$, be any given set of initial conditions. Then, the swarm centroid $c_j(t)$ asymptotically converges to $\mathcal{G}_j$. whatever is the swarm initial conditions $\left\{z_i^{(s)}(t_0)\right\}$, $i = 1, ..., n_s$.*

Notice that the dynamical behaviour of the swarm centroid is governed by the following differential equation and the swarm centroid is defined as

$$c_j(t) = \frac{1}{l_j}\sum_{i=1}^{l_j} z_j^i(t).$$

Therefore, its evolution is governed by

$$\dot{c}_j(t) = -\frac{1}{l_j}\sum_{i=1}^{l_j}(z_j^i(t) - \mathcal{G}_j) + \frac{m_j}{l_j^2}\sum_{i=1}^{l_j}\sum_{k=1}^{l_j}\frac{(z_j^i(t) - z_j^k(t))}{||z_j^i(t) - z_j^k(t)|| + \varepsilon}$$

whose last component on the r.h.s. straightforwardly vanishes. Then, one has that because of the symmetry null, due to the symmetry in the summations, then

$$\dot{c}_j(t) = -c_j(t) + \mathcal{G}_j$$

and, as a consequence, the centroid $c_j(t)$ exponentially settles down to $\mathcal{G}_j$.

Theorem 4.2. *Let $\{\Sigma_j^i\}_{i=1}^{l_j}$ be a generic agent aggregation and then let $z_j^i(t_0)$, $i = 1, \ldots, l_j$, be any given set of initial conditions. Then, each agent Σ_j^i converges in finite time $\bar{t}_j^i$. All the agents in the swarm converge into the following hyper-ball*

$$\mathcal{B}_j(G_j, m_j) = \left\{z \in \mathbb{R}^2 : ||z - G_j|| \leq m_j\right\}, \tag{4.14}$$

viceversa $z_j^i(t) \in \mathcal{B}_j(G_j, m_j), \forall t \geq \bar{t}_j^i$, with in finite time. In particular each agent i-th is in $\mathcal{B}^{(s)}$ for $t \geq t_{min}^i$ where

$$\bar{t}_j^i(t_0) := -\frac{l_j}{2}\log\left(\frac{m_j^2}{||z_j^i(t_0) - G_j||}\right) \tag{4.15}$$

First, it is shown that the swarm agents will enter into the following hyper-ball

$$\tilde{\mathcal{B}} = \left\{z \in \mathbb{R}^2 : ||z - \mathcal{G}_j|| \leq m_j(l_j - 1)l_j\right\}$$

Let $g_j^i(t) := z_j^i(t) - \mathcal{G}_j$ and $V_j^i(t) := \frac{1}{2}g_j^i(t)^T g_j^i(t)$ be a candidate Lyapunov function associated to the i-th agent dynamics belonging to the $j-th$ swarm. Note that $V(t)$ takes into account the square distance of the agent form the goal $\mathcal{G}$. Since

$$\begin{aligned}\dot{V}_j^i(t) =& g_j^i(t)^T \dot{g}_j^i(t) \\ =& g_j^i(t)^T \left(-g_j^i(t) + \frac{m_j}{l_j} \sum_{k=1}^{l_j} \frac{g_j^i(t) - g_j^k(t)}{||g_j^i(t) - g_j^k(t)|| + \varepsilon} \right) \\ =& -g_j^i(t)^T g_j^i(t) - \frac{m_j}{n} \sum_{k=1}^{l_j} \frac{g_j^i(t)^T (g_j^k(t) - g_j^i(t))}{||g_j^i(t) - g_j^k(t)|| + \varepsilon} \\ \leq& -||g_j^i(t)||^2 + \frac{m_j}{l_j} ||g_j^i(t)|| \sum_{k=1}^{l_j} \frac{||g_j^k(t) - g_j^i(t)||}{||g_j^i(t) - g_j^k(t)|| + \varepsilon}\end{aligned}$$

and

$$\frac{||g_j^k(t) - g_j^i(t)||}{||g_j^i(t) - g_j^k(t)|| + \varepsilon} \leq 1$$

it follows that

$$\dot{V}_j^i(t) \leq -||g_j^i(t)||^2 + \frac{m_j(l_j - 1)}{l_j} ||g_j^i(t)||$$

Therefore, if $||g_j^i(t)|| > m_j$ the $\dot{V}_j^i(t)$ is negative definite and this ensures that the agent dynamics belongs to the hyper-ball $\tilde{\mathcal{B}}$. Moreover, in virtue of the fact that $m_j > \frac{m_j(l_j-1)}{l_j}$, the whole swarm will lie in $\mathcal{B}$.

Notice also that Again, considering the previous defined Lyapunov function. Its derivative can be also bounded as

$$\begin{aligned}\dot{V}_j^i(t) &\leq -||g_j^i(t)||^2 + m_j ||g_j^i(t)|| \frac{l_j - 1}{l_j} \\ &= -||g_j^i(t)||^2 + m_j \frac{||g_j^i(t)||^2}{||g_j^i(t)||} \frac{l_j - 1}{l_j}\end{aligned}$$

and if the i-th swarm agent is outside $\mathcal{B}$, one has that $||g_j^i(t)|| \geq m_j$ and $\frac{1}{||g_j^i(t)||} \leq \frac{1}{m_j}$. As a consequence

$$\begin{aligned}\dot{V}_j^i(t) &\leq -||g_j^i(t)||^2 + \frac{||g_j^i(t)||^2}{m_j} m_j \frac{l_j - 1}{l_j} \\ &\leq -||g_j^i(t)||^2 + \frac{l_j - 1}{l_j} ||g_j^i(t)||^2 = -\frac{||g_j^i(t)||^2}{l_j}\end{aligned}$$

Finally, since $\dot{V}_j^i(t) \leq -\frac{2}{n} V_j^i(t)$ one obtains

$$V_j^i(t) \leq V_j^i(0) e^{-\frac{2}{n}t}$$

which shows that the i-th agent will lie to $\mathcal{B}$ when

$$m_j^2 \geq 2V(0)e^{-\frac{2}{l_j}t} \rightarrow t \geq -\frac{l_j}{2}\log\left(\frac{m_j^2}{||z_j^i(t)-\mathcal{G}_j||}\right)$$

Proposition 4.3. *The $j-th$ swarm asymptotically converges to the following static configuration*

$$\lim_{t\to\infty} \dot{z}_j^i(t) = 0_2, \forall i = 1,\ldots,l_j \tag{4.16}$$

Let us define

$$Q_j(t) := \sum_{i=1}^{l_j} Q_j^i(t) + \frac{1}{2m_j}\sum_{i=1}^{l_j} g_j^i(t)^T g_j^i(t)$$

with

$$\begin{aligned} Q_j^i(t) := \tfrac{1}{2m_j} g_j^i(t)^T g_j^i(t) - \tfrac{1}{l_j}\sum_{k=1}^{l_j}\big(||g_j^i(t)-g_j^k(t)|| \\ -\varepsilon\log\left(\varepsilon + ||g_j^i(t)-g_j^k(t)||\right)\big) \end{aligned}$$

By exploiting the dynamical evolution (4.13), it follows that

$$\dot{Q}_j(t) = -\frac{2}{m_j}\sum_{i=1}^{l_j} \dot{g}_j^i(t)^T \dot{g}_j^i(t)$$

asymptotically converges to the following invariant set in virtue of the LaSalle's theorem and using the LaSalle's theorem, the swarm is ensured to converge to the invariant set defined as

$$\mathcal{E} = \left\{ g_j^i(t) \mid \dot{Q}_j(t) = 0 \right\} = \left\{ g_j^i(t) \mid \dot{g}_j^i = 0 \right\}$$

stating that the swarm agents converge to their equilibrium points.

4.3 The time-varying swarm platoon modelling

In the sequel, the LF team is reorganized as a platoon of r swarm aggregations $\{SW_j\}_{j=1}^r$, each one collecting the agents having the same tree level position, and the agents will be denoted as Σ_j^i, with j accounting for $level(\cdot)$, see Fig. 4.1. Moreover, the vehicle models and 4.1 are discretized via the forward Euler difference method:

$$\begin{cases} x_j^i(t+1) = F_j^i(x_j^i(t)) + B_j^i(x_j^i(t))u(t), \\ i = 1,\ldots,l_j,\ j = 1,\ldots,r, \end{cases} \tag{4.17}$$

For the sake of clarity, the state space description (4.4) is without loss of generality accordingly modified:

$$\begin{cases} x_j^i(t+1) = \Phi_j^i(\alpha(t))x_j^i(t) + G_j^i(\alpha(t))u_j^i(t), \\ i = 1,\dots,l_j,\ j = 1,\dots,r, \end{cases} \tag{4.18}$$

where $t \in \mathbb{Z}_+ := \{0,1,...\}$, $x_j^i(t) \in \mathbb{R}^n$ denotes the state plant and $u_j^i(t) \in^m$ the control input. The time-varying vector $\alpha(t) \in^p$ belongs to the unit simplex

$$\mathcal{P} := \left\{ \alpha \in \mathbb{R}^{\Gamma} : \sum_{k=1}^{\Gamma} \alpha_k = 1,\ \alpha_k \geq 0 \right\} \tag{4.19}$$

whereas the system matrices $(\Phi_j^i(\alpha),\ G_j^i(\alpha))$ lie in

$$\Omega(\mathcal{P}) := \left\{ (\Phi_j^i(\alpha), G_j^i(\alpha)) = \sum_{k=1}^{\Gamma} \alpha_k ((\Phi_j^i)^k, (G_j^i)^k), \alpha \in \mathcal{P} \right\} \tag{4.20}$$

with $((\Phi_j^i)^k, (G_j^i)^k)$ the polytope vertices of $\Sigma(\mathcal{P})$. Moreover, $x_j^i \in \mathcal{X}_j^i \subset \mathbb{R}^n$ and $u_j^i \in \mathcal{U}_j^i \subset \mathbb{R}^m$.

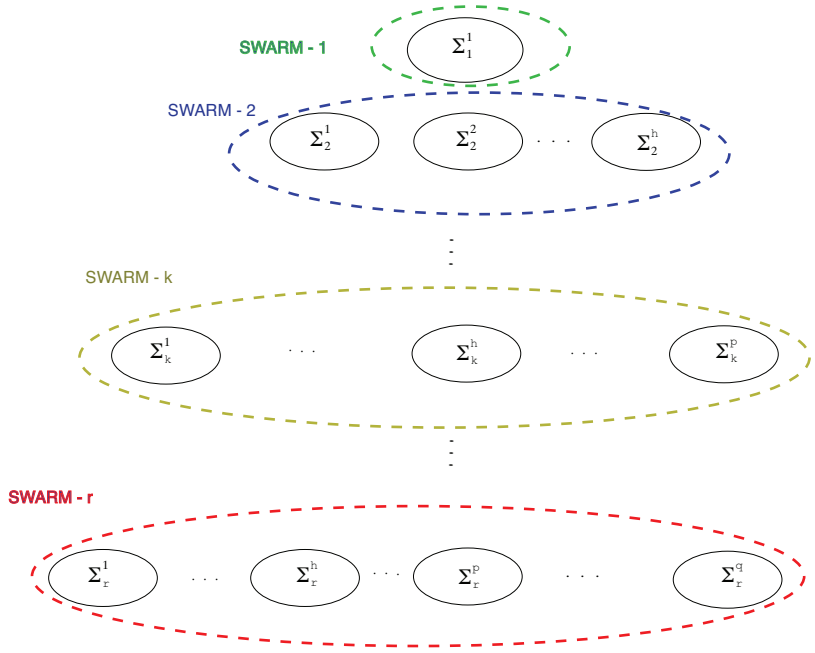

Fig. 4.1. Swarm platoon model

Communication issues

It is important to underline that platoon configuration of Fig. 4.1 can be characterized by considering that two any elements along the chain, namely SW_k and SW_{k+1} are connected each other in virtue of the fact that at each time instant the SW_k sends to SW_{k+1} a data packet containing its hyper-ball at $t-1$, i.e. the centroid position $c_k(t-1)$ and the radius m_k.

As pointed out in the problem formulation section, it is well known that by construction the model plants (4.18) remain valid only within pre-computed compact and convex state regions. In view of this, a model description updating policy is here proposed by exploiting the following arguments:

- the leader Σ_1^1 moves within the vision radius R_v of its sensor module: at each time instant starting and final state conditions are directly available;
- the follower swarm SW_k receives the pair $(c_{k-1}(t-1), m_{k-1})$: each agent Σ_k^i knows the time interval $\bar{t}_k^i$ required to converge to the hyper-ball $\mathcal{B}_j(c_{k-1}(t-1), m_{k-1})$ starting from its current position (see Theorem 1).

To better describe the *modus operandi* of the updating phase, we shall refer to a single vehicle along the swarm platoon chain as depicted in Fig. 4.2. There, we refer to the projection of the vehicle dynamics on the $2-D$ environment (x, y), while the time axis is shown in order to characterize the time trend of the sequence of events. With the symbols $+$, $*$ and $\circ$ we denote the current state measurement, the corresponding position within the next hyper-ball along the vehicle path and the centroid, respectively. Moreover, the following operator is defined.

Definition 4.4. *Let $\mathcal{B}_{in}(c_{in}, m_{in}) \subset^n$ and $\mathcal{B}_{fin}(c_{fin}, m_{fin}) \subset^n$ be two hyper-balls. Given any point $x_{in} \in \mathcal{B}_{in}(c_{in}, m_{in})$, the mapping operator $\mathcal{I} : \mathcal{B}_{in}(c_{in}, m_{in}) \rightarrow \mathcal{B}_{fin}(c_{fin}, m_{fin})$ returns the mirror point of x_{in}, i.e. $x_{fin} \in \mathcal{B}_{fin}(c_{fin}, m_{fin})$.*

At the initial time instant t, the nominal solution $(\hat{x}(t), \hat{u}(t))$ is computed by considering the state measurement $x_k^i(t)$ as the initial condition and the mirror point $\xi_k^i \in \mathcal{B}_j(c_{k-1}(t-1), m_{k-1})$ as the target, see the dashed line in Fig. 4.2. Moreover, the results of Theorem 1 are exploited: starting from the current condition $z_k^i(t) \equiv x_k^i(t)$, it is ensured that the kinematics of Σ_k^i will be driven within the hyper-ball $\mathcal{B}_j(c_{k-1}(t-1), m_{k-1})$ in at most $\bar{t}_k^i(t)$.

Then, the robust PLDI (green line) is computed by using $\bar{t}_k^i(t)$ in place of Δt and implementing the numerical procedure of Section 1.1. At $t + \bar{t}_k^i(t)$, the vehicle will enter into $\mathcal{B}_j(c_{k-1}(t-1), m_{k-1})$ not necessarily in a point coincident with the local target, i.e. $x_k^i(t + \bar{t}_k^i(t)) \neq \xi_k^i$, and from now on the green embedding is no longer usable to approach the **LF-OA-R** objective x_f^i. Therefore, a new robust PLDI (red line) is determined by using the initial condition, $x_k^i(t + \bar{t}_k^i(t))$, the local target $\mathcal{I}(x_k^i(t + \bar{t}_k^i(t)))$, the hyper-ball

$\mathcal{B}_j(c_{k-1}(t+\bar{t}_k^i(t)-1), m_{k-1})$ and interval time $\bar{t}_k^i(t+\bar{t}_k^i(t))$. The same reasoning exactly applies for all future events, e.g. the sketched blue embedding in Fig. 4.2. is iterated, see the red and blue embeddings in Fig. 4.2.

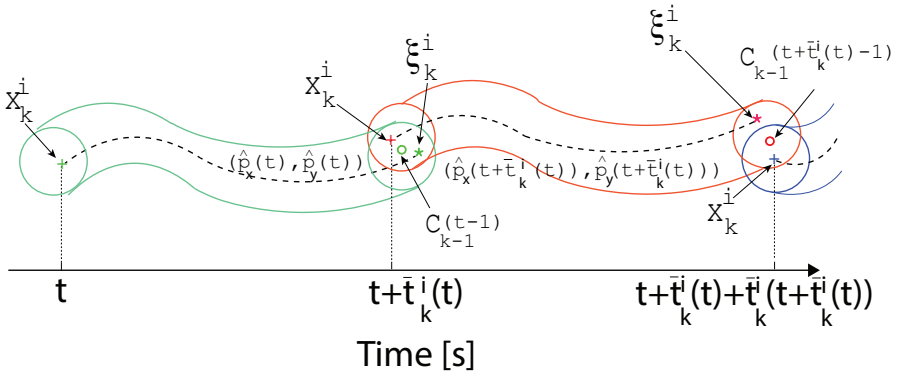

Fig. 4.2. Time varying robust PLDI: agent Σ_k^i

As the whole swarm is concerned, the above arguments remain still valid except that the switching between two PLDIs occurs when all the swarm vehicles enter into the prescribed hyper-ball, e.g. $\mathcal{B}_j(c_{k-1}(t-1), m_{k-1})$. As a consequence, in order to be guaranteed on the latter, the time interval Δt must be chosen as follows:

$$\bar{t}_k(t) := \max_{i=1,\ldots,l_k} \bar{t}_k^i(t) \tag{4.21}$$

Finally, a different analysis can be done for the leader vehicle. By taking into account that this vehicle always operates within the hyper-ball $D(x_1^1(t), R_v)$, the current state$x_1^1(t)$ and the local target $\xi_1^1(t)$ jointly belongs to $D(x_1^1(t), R_v)$, the model

$$\Sigma_1^1 : x_1^1(t+1) = F_1^1(x_1^1(t)) + B_1^1(x_1^1(t))u_1^1(t) \tag{4.22}$$

must be exploited according to the following reasoning. Let $\Xi(t) \subseteq D(x_1^1(t), R_v)$ be the positively invariant (PI) region associated to (4.22) compatible with the prescribed constraints (4.9).

In this case,the model switching will result from simple set-containment arguments, see Fig. 4.3. During the on-line operations, the admissible state trajectory tube is defined via overlapped sets which are designed such that the selected point (target) $\xi_1^1(t) \in D(x_1^1(t), R_v)$ is the ellipsoid center while

$x_1^1(t) \in \Xi(t-1) \cup \Xi(t)$, see Fig. 4.3.

Notice that the PI regions $\Xi(t), \forall t \geq 0$, are computed by resorting to well established semi-algebraic technicalities: first the nonlinear description (4.22) is recast in a polynomial form by means the so-called recasting procedure provided in [190], then Sum-of-Squares (SOS) based optimizations are performed see e.g. [191] and references therein.

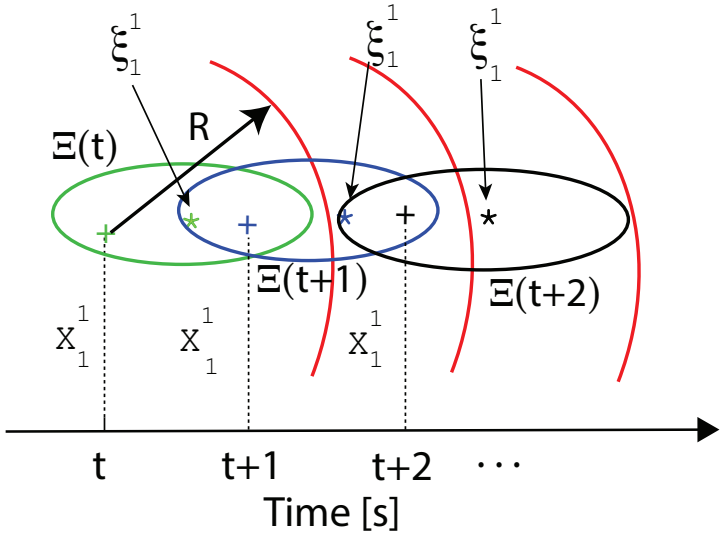

Fig. 4.3. State trajectory tube: leader Σ_1^1

4.4 The swarm-based distributed MPC architecture

In the sequel, a distributed model predictive control strategy will be developed for a the swarm-platoon configuration of Fig. 4.1.

First, the following premises are made:

- there exists at least an admissible path complying with the **LF-OA-R** problem prescriptions;
- communication facilities of **Remark 1** are allowed.

The control architecture depicted in Fig. 4.4 for a generic $j-th$ swarm is here proposed. Essentially, it consists of three units: a **Path Planner**, a **MPC controllers bank** each one tuned w.r.t. the set of swarm agents Σ_j^i, $i = 1, \ldots, l_j$, and an **Update Logic**.

The basic idea can be summarized as follows:

1. at each time instant t, the $j-th$ swarm receives from its father along the LF chain the hyper-ball at $t-1$ of the $(j-1)-th$ swarm and the state measurements of its neighbour, namely $x_j^i(t-1)$, $i=1,\ldots,l_j$;
2. the **Path Planner** unit generates the kinematic state trajectories $z_j^i(\cdot)$ of the agents Σ_j^i, $i=1,\ldots,l_j$;
3. the **MPC controllers bank** computes the admissible control actions u_j^i, $i=1,\ldots,l_j$, in a distributed receding horizon fashion;
4. the **Update Logic** unit in charge to adequately modify the swarm plant models Σ_j^i, $i=1,\ldots,l_j$ according to the prescriptions of the previous Section.

To develop such an abstract procedure, it is mandatory to formally characterize structure and properties of these two elements. The next subsections will be devoted to this aim.

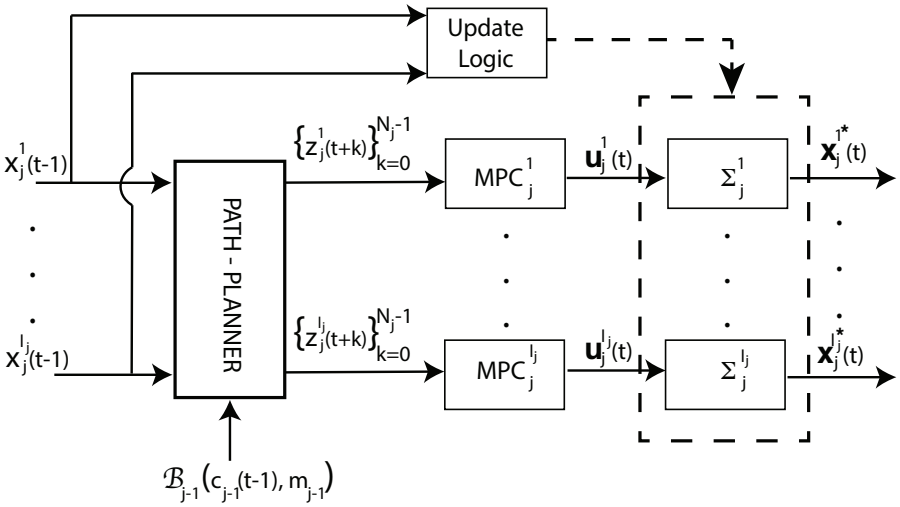

Fig. 4.4. Distributed control architecture for the $j-th$ swarm

4.4.1 Distributed MPC controllers

For any agent $i=1,\ldots,l_j$, let the $k-th$ predicted state and predicted control within the horizon at time t be denoted by $x_j^i(k|t)$ and $u_j^i(k|t)$, respectively. Let $\mathbf{x}_j^i(t) := \{x_j^i(k|t)\}_{k=0}^{N_j}$, $i=1,\ldots,l_j$, $j=1,\ldots,r$ and $\mathbf{u}_j^i(t) := \{u_j^i(k|t\}_{k=0}^{N_j}$, $i=1,\ldots,l_j$, $j=1,\ldots,r$, be the predicted states and predicted controls within the control horizon, respectively. Moreover, over any prediction interval $[t+k,\ t+k+N_j]$ and for any $i-th$ agent we further define:

- the optimal state trajectory: $\mathbf{x}_j^{i^*}(t) := \{x_j^{i^*}(k|t\}_{k=0}^{N_j}, i = 1,\ldots,l_j, j = 1,\ldots,r$;
- the assumed state trajectory: $\hat{\mathbf{x}}_j^i(t) := \{\hat{x}_j^i(k|t\}_{k=0}^{N_j}, i = 1,\ldots,l_j, j = 1,\ldots,r$.

Notice that the sequence $\hat{\mathbf{x}}_j^i$ is transmitted to all the neighbors $k \in \mathcal{N}^i$ which in turn hypothesizes that the $i-th$ agent will implement during the update prediction time interval $[t+k+1,\ t+k+N_j+1]$.
In the proposed distributed scheme, local MPC units have to be obtained by also taking care of the main **LF-OA-R** goal, i.e converging to the target x_f. Then, it is necessary to deal with the fact that the LF formation moves towards x_f : the latter prescribes that the terminal condition is time-varying and the related constraint must be accordingly modified. In particular, the aim is of developing a control strategy capable to jointly reduce as much as possible data communications amongst agents belonging to different swarms and to improve energy savings on the use of on-board sensors. To this end, the actions pertaining to the leader and follower swarms will be addressed by resorting to different arguments.

Leader swarm

As the leader swarm SW_1 is concerned, in order to formally define the optimal control problem underlying the MPC strategy, the following ingredients are required:

- Control strategy:

$$u_1^1(\cdot|t) = K_1(x_1^1(k+t|t)),\ \ k \geq 0 \tag{4.23}$$

  with $K_1(x_1^1(\cdot)) \in \mathbb{R}[x]$ a polynomial stabilizing and admissible state feedback law. With $\mathbb{R}[x]$ we denote the ring of multivariate scalar polynomials $\varphi \in \mathbb{R}[x]$ in the unknown $x \in^n$;
- the LQ quadratic performance index:

$$J_\infty(t) := \sum_{k=0}^{\infty}\left[\|x_1^1(t+k|t) - x_f^1\|_{R_{x_1^1}}^2 + \|u_1^1(t+k|t)\|_{R_{u_1^1}}^2\right] \tag{4.24}$$

  where $R_{x_1^1} > 0$ and $R_{u_1^1} \geq 0$ are symmetric state and input weighting matrices.
- Terminal constraint:

$$x_1^1(t|t) \in \Xi_1 \subset \mathbb{R}^n \tag{4.25}$$

  with the pair $(\Xi_1, K_1(x_1^1)$ computed such that Ξ_1 is a positively invariant (PI) region for the state evolutions of the closed-loop system, see e.g. [192].

An important technical issue concerns with the updating of the terminal PI region Ξ_1 : in fact, the goal consists to move towards the target x_f^1 and, as a consequence, the terminal condition becomes time-varying and the related constraint must be accordingly modified. By noticing that the plant model (4.4) is linear and the unknown environment is planar, At each time instant t the new set $\Xi_1(t)$ is obtained according to the following set-membership requirements:

$$x_1^1(t|t) \in \Xi_1(t-1) \cup \Xi_1(t) \tag{4.26}$$

where $\Xi_1(t)$ is computed such that and

$$\xi_1^1(t-1) \in \Xi_1(t-1) \cap \Xi_1(t) \tag{4.27}$$

$$\Xi_1(t) \subset^1 \cap \left(\mathcal{O}^t \setminus \Xi_1(t-2)\right) \tag{4.28}$$

$$\Xi^1(t) \subseteq \mathcal{B}(x_1^1(t), R_v) \tag{4.29}$$

with $\xi_1^1(t-1)$ denoting the equilibrium point selected at the time instant $t-1$.

Notice that (4.28) is imposed for ensuring that the sequence of the PI regions is pairwise overlapped. The resulting non-convex constraint can be convexified by means of the arguments presented in [193]. Then, the optimization problem for the leader agent, hereafter denoted as $\mathcal{P}_L(t)$, is:
DMPC-$\mathcal{P}_L(t)$:

$$\min_{K_1(x_1^1), \Xi_1} \; J_\infty(t) \tag{4.30}$$

subject to

$$x_1^1(t+k+1|t) = F_1^1(x_1^1(t+k|t)) + B_1^1(x_1^1(t+k|t))u_1^1(t+k|t) \tag{4.31}$$

$$F_1^1(\Xi_1) + B_1^1(\Xi_1)K_1(\Xi_1) \subset \Xi_1 \tag{4.32}$$

$$K_1(x_1^1(t+k|t)) \in_1^1,\, k \geq 0 \tag{4.33}$$

$$x_1^1(t+k|t) \in_1^1,\, k \geq 0 \tag{4.34}$$

$$\xi_1^1(t-1) \in \Xi_1(t-1) \cap \Xi_1 \tag{4.35}$$

$$\Xi_1(t) \subset^1 \cap \left(\mathcal{O}^t \setminus \Xi_1(t-2)\right) \tag{4.36}$$

We have also Notice that the non-convex constraints (4.26) and (4.35) could lead to computational intractability. This difficult can be straightforwardly overcome by using inner convex approximations computed as shown e.g. in [188].

Follower swarms

In this case, the idea is to exploit the properties of the kinematics model (4.13) which ensure that any swarm agent converges in finite time to the hyper-ball (4.14) in the finite time (4.15) by avoiding collisions with the other agents belonging to the same swarm configuration. Then, the key ingredients of the MPC controller design are below summarized:

- Input sequence parametrizations:

$$u_j^i(\cdot|t) = \begin{cases} u_j^i(k+t|t), & k = 0, \ldots, N_j - 1 \\ K_j(x_j^i(k+t|t)), & k \geq N_j \end{cases} \tag{4.37}$$

  with $K_j(x) \in [x]$ polynomial stabilizing and admissible state feedback laws;
- Cost-to-go functions:

$$\begin{aligned} &J_j^i(x_j^i(t|t), x_f^i, \mathbf{u}_j^i(t)) := \max_{\alpha(\cdot)\in\mathcal{P}} \\ &\sum_{k=t}^{t+N_j-1} \left[\|x_j^i(t+k|t) - z_j^i(t+k)\|^2_{R_{x_j^i}} + \|u_j^i(t+k|t)\|^2_{R_{u_j^i}} \right] + \\ &\|x_j^i(t+N_j|t) - x_f^i\|^2_{P_j} \end{aligned} \tag{4.38}$$

  where N_j is the prediction horizon, $R_{x_j^i} > 0$ and $R_{u_j^i} \geq 0$ symmetric state and input weighting matrices and $P_j \geq 0$;
- Terminal constraint:

$$x_j^i(t+N_j|t) \in \Xi_j \subset \mathbb{R}^n \tag{4.39}$$

  a set-membership constraint is imposed in terms of a PI region $\Xi_i \subset^n$, computed such that by assuming that starting from the initial condition $x_i(t+N_i|t) \in \Xi_i$ one has that $(A^i + B^i K_i)^k x_i(t+N_i|t) \in \Xi_i, \forall k$; where Ξ_j is a PI region such that $(F_j^i(x_j^i(t+N_j|t)) + B_j^i(x_j^i(t+N_j|t))K_j(x_j^i(t+N_j|t)))^{t+N_j+k} \in \Xi_j,\ \forall k \geq 0$.

In order to properly formalize the MPC design within the swarm framework of Section 3, the following set-containments have to be preliminarily satisfied:

Requirement 1 - compute the hyper-balls $\mathcal{B}_j(c_j(t), m_j) \subseteq \Xi_j$, $j = 2, \ldots, r$, such that

$$\begin{array}{l} \mathcal{B}_r(c_r(t), m_r) \subseteq \mathcal{B}_{r-1}(c_{r-1}(t), m_{r-1}) \subseteq \cdots \subseteq \\ \mathcal{B}_2(c_2(t), m_2) \subseteq \Xi_1(\cdot) \end{array} \tag{4.40}$$

Requirement 2 - Let $c_j((t-1)+k)$, $k = 1, \ldots, N_j-$, be the sequence of centroids of SW_j exploited at $t-1$. Then, it is required that $k-$th step ahead state prediction of the followers $\Sigma_j^i, i = 1, \ldots, l_j, j = 2, \ldots, r$, are jailed as follows:

$$\begin{array}{l} x_j^i(t+k|t) \subseteq \mathcal{B}_j(c_j((t-1)+k), m_j), \\ k = 1, \ldots, N_j - 1, i = 1, \ldots, l_j,\ j = 2, \ldots, r. \end{array} \tag{4.41}$$

Notice that the set-containment conditions (4.40)-(4.41) are instrumental to guarantee that all the swarm followers are capable to avoid collisions without resorting to on-board sensors. Since only the leader is in charge to detect the obstacles by using the available sensors, the PI region Ξ_1 is computed by encapsulating it inside the ball of radius R_v (the vision radius).

In fact, the leader detects the obstacles by means of its sensors (vision radius), so that the PI region Ξ_1 is computed by encapsulating it within the ball of radius R. Then, imposing (4.40) means that the first follower swarm will be driven after N_2 steps into Ξ_1 with the equilibrium $\bar{x}_1^1$ acting as the centroid of SW_2, while as the successive swarms along the platoon chain are concerned the same property holds true with $c_{j-1}(t)$ in place of $\bar{x}_1^1$. Conversely, (4.41) characterizes a sort of transient phase, i.e. the follower swarm configurations SW_j, $j = 2, \ldots, r$, are preserved inside the corresponding translated balls $\mathcal{B}_j(\cdot, m_j)$, $j = 2, \ldots, r$, along the $k-$th step ahead state predictions. Finally for feasibility reasons that will be clarified in the next section, the terminal condition for each vehicle of the swarm SW_j complies with the following set-containment

$$\begin{array}{l} x_j^i(t+N_j|t) \in \mathcal{B}_j(c_{j-1}(t_j^{curr}), m_j) \cap \mathcal{B}_j(c_{j-1}(t_j^{prec}), m_j), \\ \forall i = 1, \ldots, l_j, \end{array} \tag{4.42}$$

where t_j^{curr} and t_j^{prec} must be understood as the time instants related to the most recent data transmitted by the father SW_{j-1}.

In view of these developments, on the followers side two actions are prescribed. Whenever new data (hyper-balls sent by father swarms) are available, the first vehicle Σ_j^1 updates the PI region Ξ_j of the swarm SW_j as follows:

$$\min_{K_j, \Xi_j} \sum_{k=0}^{\infty} \left[\|x_j^1(t+k|t)\|^2_{R_{x_j^1}} + \|u_j^1(t+k|t)\|^2_{R_{u_j^1}} \right] \tag{4.43}$$

subject to

$$F_j^i(\Xi_j) + B_j^i(\Xi_j)K_j(\Xi_j) \subset \Xi_j \subset {}_j^i \cap \left(\mathcal{O}^t \setminus \Xi_j(t_j^{prec})\right), \quad i = 1, \ldots, l_j \tag{4.44}$$

$$K_j(\Xi_j) \subseteq \bigcap_{i=1}^{l_j} {}_j^i, \tag{4.45}$$

$$\xi_j(t_j^{curr}) \in \Xi_j(t_j^{curr}) \cap \Xi_j \tag{4.46}$$

$$\mathcal{B}_j(c_{j-1}(t_j^{curr}), m_j) \subseteq \Xi_j \tag{4.47}$$

Then, the above developments allow to formalize the optimization problem for the $i-th$ follower, hereafter denoted as $\mathcal{P}_F^{i,j}(t)$:

DMPC-$\mathcal{P}_F^{i,j}(t)$:

$$\min_{\mathbf{u^i(t)}} J_j^i(x(t|t), x_f^i, \mathbf{u}^i(t)) \tag{4.48}$$

$$x_j^i(t+k+1|t) = \Phi_j^i(\alpha)x_j^i(t+k|t) + G_j^i(\alpha)u_j^i(t+k|t), \quad \forall \alpha \in \mathcal{P} \tag{4.49}$$

$$u_j^i(t+k|t) \in {}_j^i,\ k = 0, 1, \ldots, N_j - 1, \tag{4.50}$$

$$x_j^i(t+k|t) \in {}_j^i \cap \mathcal{B}_j(c_j((t-1)+k), m_j), \quad k = 0, 1, \ldots, N_j - 1, \forall \alpha \in \mathcal{P} \tag{4.51}$$

$$x_j^i(t+N_j|t) \in \mathcal{B}_j(c_{j-1}(t_j^{curr}), m_j) \cap \mathcal{B}_j(c_{j-1}(t_j^{prec}), m_j), \quad \forall \alpha \in \mathcal{P} \tag{4.52}$$

$$\beta_{min} \leq \|x_j^i(t+k|t) - \hat{x}_j^q(t+k|t)\| \leq \beta_{max}, \quad k = 0, 1, \ldots, N_j,\ \forall q \in \mathcal{N}^i, \forall \alpha \in \mathcal{P}, \tag{4.53}$$

where

- the assumed sequence $\hat{x}_j^q(\cdot)$ has the following structure:

$$\hat{\mathbf{x}}_j^q(t) := \begin{cases} x_j^{q^*}(t-1+k|t-1), & k = 1, \ldots, N_j - 1, \\ F_j^q(x_j^{q^*}(t-1+N_j|t-1)) + & \\ G_j^q K_j(x_j^{q^*}(t-1+N_j|t-1))^{t-1+k}, & k = N_j; \end{cases} \tag{4.54}$$

- constraints (4.53) are in charge to jointly comply with (4.41) and to avoid collisions with neighbour agents. Specifically, the two positive scalars $\beta_{max}\, \beta_{min} \in \mathbb{R}^+$ are guaranteed bounds on the agents' dynamics and the l.h.s. of (4.53) can be convexified as outlined in **Remark 2** by again exploiting the results reported in [193].

Notice that the non-convex constraints on the l.h.s. of (4.53) could lead to computational intractability. Nonetheless, this difficulty can be overcome by means of the convexification results proposed in [193].

4.4.2 Path Planner

In the present context, this unit has the key role to provide an adequate *reference* state trajectory to each follower agent capable to ensure that in a finite number of time steps the i-th agent of the j-th swarm is steered to the safe hyper-ball corresponding to the higher $(j-1)$-th swarm along the LF chain. By resorting to the architecture of Fig. 4.4, the path planner exploits the available information at the previous times instant $t-1$: the hyper-ball $\mathcal{B}(c_{j-1}(t^1_{prec}), m_{j-1})$ and the state measurements of the l_j agents of the j-th swarm. Then, by resorting to the developments of Section 4.3, the following procedure results.

Path planning

Input: $x^i_j(t-1)$, $i = 1, \ldots, l_j$; $\mathcal{B}(c_{j-1}(t^{curr}_j), m_{j-1})$;

Output: $z^i_j(t+k)$, $k = 0, \ldots, N_j - 1$;

Initialization: $\mathbf{P^k} \equiv \mathbf{P}$ *[1]* **Set** $\mathcal{G}_j \leftarrow c_{j-1}(t^{curr}_j)$ *and* $m_j \leftarrow m_{j-1}$ *in (4.13);*

Extract *the initial conditions* $z^i_j(t)$, $i = 1, \ldots, l_j$, *from* $x^i_j(t-1)$, $i = 1, \ldots, l_j$;

Set $\mathcal{G}_j \leftarrow c_{j-1}(t-1$ *and* $m_j \leftarrow m_{j-1}$ *in (kine1);*

Split $\bar{t}^i_j$ *(see 4.15) in* N_j *equally distributed time steps;*

Compute *the state kinematic predictions* $z^i_j(t+k)$, $k = 0, \ldots, N_j - 1$ *via (4.13).*

Updating Logic procedure

The module operates by following the prescriptions of Section 4.3 Specifically, the following procedure comes out:

Embedding

Input: t_{act}; $\bar{t}_j(t)$; h; $\mathbf{x}^i_j(t-1)$, $i = 1, \ldots, l_j$; $\mathcal{B}(c_{j-1}(t-1), m_{j-1})$;

Output: $(\Phi_j^i,\, G_j^i)$, $i = 1, \ldots, l_j$;

Initialization: $\mathbf{P^k} \equiv \mathbf{P}$

[1] **Compute** *nominal solutions of (1.25) connecting the current state measurements* $x_j^i(t)$ *and the mirror targets* $\xi_j^i(t) = \mathcal{I}\left(\mathcal{B}(c_{j-1}(t-1), m_{j-1})\right)$, *i.e.* $(\hat{x}_j^i(\cdot), \hat{u}_j^i(\cdot))$, $i = 1, \ldots, l_j$;

Along the nominal robot paths $(\hat{x}_j^i(\cdot), \hat{u}_j^i(\cdot))$, **Extract** *sequences of operating points* $\{(\hat{\theta}_j^{i_w}(t),\, \hat{v}_j^{i_w}(t))\}_{w=1}^{h}$, $i = 1, \ldots, l_j$, *equally spaced within the time interval* $[t_{act},\, t_{act} + \bar{t}_j(t)]$;

For each pair $(\hat{\theta}_j^{i_w}(t),\, \hat{v}_j^{i_w}(t))$, **Determine** *a family of PLDIs (1.31) according to (1.29)-(1.30)*

Determine *the linearised model (1.26) of (1.25) around the nominal solution* $(\hat{x}_j^i(t), \hat{u}_j^i(t))$, $i = 1, \ldots, l_j$;

Compute *the convex hull (4.1);*

Discretize *the continuous-time model (4.1)-(4.2) via the forward Euler difference method.*

4.5 A developed distributed MPC algorithm

In this section, the **LF-OA-R** problem will be addressed by developing a distributed MPC (**DMPC**) algorithm which takes advantage of the above developments. First, the following assumptions are exploited: The algorithm is globally initialized by means of the following off-line computations:

- an admissible sequence of pairs $(K_j(x), \Xi_j)$, $j = 1, \ldots, r$, complying with (4.39);
- a sequence of hyper-balls $\mathcal{B}_j(c_j(0), m_j)$, $j = 1, \ldots, r$, satisfying the requirement (4.40);
- the switching time instants $\bar{t}_j(0)$, $j = 2, \ldots, r$, by means of (4.21),
- the swarm horizon lengths obtained by splitting $\bar{t}_j(0)$ in N_j, $j = 2, \ldots, r$, equally distributed time steps.

Moreover, the convexification of the obstacle-free region $\mathcal{O}_{free}^t$ is obtained by means of the **CP** algorithm fully detailed in pg. 7 of [194]: notice that it provides a polygon **S** by using a sequence of detected obstacle points $\mathbf{P} := \{p_1, \ldots, p_s\}$.

DMPC-Swarm-Algorithm - Agent $i-th$

Input: β_{min}, β_{max}, γ, Ξ^-, Ξ^+, $x_j^i(0)$, x_f^i, update:=false;

Output: $u_j^{i^*}(t/t)$;

Initialization: N_j, $\mathcal{N}^i, t_j^{curr}, t_j^{prec}$

If $j=1$ then Swarm leader

If $\mathcal{B}(x_1^1(t),R)\cap ({}^n/\mathcal{O}^t)\neq\emptyset$ then an obstacle is detected

Extract a set of points $\mathbf{P}=\{p_1,\ldots,p_s\}$;

Activate the **CP** algorithm and **Built** the polygon **S**;

Solve the optimization (4.30)-(4.36) subject to the additional constraint (4.8) with **S** in place of $f(x,t)$;

else

Solve (4.30)-(4.36);

end if

If $x_1^1(t)\in\Xi_1(t-1)\cap\Xi_1(t)$

Transmit $\mathcal{B}_1(c_1(t-1),m_1)\subseteq\Xi_1(t)$ to SW_2;

end if

else Swarm followers

If $t>\bar{t}_j^{prec}+\bar{t}_j^{succ}$ then

Update the model Σ_j^i via the **Embedding** procedure with $t_{act}\leftarrow t+\bar{t}_j^{prec}$ and $\bar{t}_j(t)\leftarrow\bar{t}_j^{succ}$;

$\bar{t}_j^{prec}\leftarrow\bar{t}_j^{succ}$;

Compute $\bar{t}_j^i(t)$ by (4.15) with $t_0\leftarrow\bar{t}_j^{prec}+\bar{t}_j^{succ}$ and $\mathcal{G}_j\leftarrow c_{j-1}(t-1)$;

Send $\bar{t}_j^i(t)$ to all the neighbours $k \in \mathcal{N}^i$;

Compute $\bar{t}_j^{succ}$ by (4.21);

end if

If $i = 1$ then

If $\mathcal{B}_{j-1}(c_{j-1}(t-1), m_{j-1})$ has been **received**

then **Update** $t_j^{curr} \leftarrow t-1$ and $t_j^{prec} \leftarrow t_{curr}^j$;

Solve the optimization (4.43)-(4.47);

Transmit (Ξ_j, K_j) to the neighbours $\mathcal{N}_j^1$; end if

end if

Generate the kinematic state trajectory $z_j^i(t+k|t)$, $k = 1, \ldots, N_j - 1$, via the **Path Planning** procedure;

Solve the optimization (4.48)-(4.53)

Compute $\hat{\mathbf{x}}_j^i(t)$ according to (4.54);

Receive $\hat{\mathbf{x}}_j^k(t)$ from all neighbours $k \in \mathcal{N}^i$;

If $j < r$ then

if $x_j^i(t) \in \mathcal{B}_j(c_j(t_j^{curr}), m_j) \cap \mathcal{B}_j(c_j(t_j^{prec}), m_j)$
then

Transmit $\mathcal{B}_j(c_j(t), m_j)$ to SW_{j+1};

end if
End if

end if
Apply $u_j^{i^*}(t|t)$;

$t \leftarrow t+1$ and goto **Step 1**.

The main properties of the **DMPC-Swarm** Algorithm are summarized in the next proposition and also recursive feasibility and asymptotic stability of the **DMPC** Algorithm are formally stated in the following proposition.

Proposition 4.5. *Let the initial $x_{in} = x(0)$ and the target x_f conditions be given. Then, the* **DMPC-Swarm** *Algorithm always satisfies the prescribed constraints and ensures that the closed-loop state trajectories are asymptotically stable.*

Under the hypothesis that feasible input sequences are off-line available both for $\mathcal{P}_L$ and $\mathcal{P}_F^{i,j}$ optimization problems, the following arguments are exploited for feasibility purposes. By first considering the current **leader** of the LF configuration, the feasibility arises from the fact that if an optimal solution there exists at the time instant t, namely $u_1^{1^*}(t|t) = K_1(x_1^1(t))$, then at the next time instant $t+1$ the positively invariance property of $\Xi_1(t)$ ensures that the state trajectory at least will remain confined within $\Xi^1(t)$ and, in virtue of (4.26)-(4.29), the transition to the new controller is asymptotically guaranteed.

On the other hand, when follower agents belonging to the generic $j-$th swarm are concerned, the admissibility of the strategy comes out in virtue of the fact that the path planner provides a feasible state trajectory driving the swarm to the hyper-ball $\mathcal{B}_r(c_{j-1}(t-1), m_j) \subseteq \Xi_j$. There, constraints satisfaction and obstacle avoidance requirements are guaranteed in virtue of conditions (4.40)-(4.41), while in the worst case the state trajectory of each $i-$th follower belonging to SW_j will jailed into Ξ_j., and the feasibility arguments exactly trace the lines exploited for the leader vehicle. In fact, during the the leader disconnection phase the feasibility of each $\mathcal{P}_F^i$ is ensured thanks to **Proposition** 1; The same arguments allow to straightforwardly show the asymptotic stability property.

As the leader is concerned, the feasibility straightforwardly comes out in virtue of two considerations: 1) the control horizon length is $N_1 = 0$; 2) the PI sequence $\Xi_1(\cdot)$ is overlapped by construction, see constraints (4.27), and the equilibrium $\xi_1^1(t-1)$ belongs to the set intersection of the overlapped positively invariant regions. As a consequence, the existence of a solution of the underlying RHC optimization only depends on the invariance condition (4.32) which in turn is always verified if a solution there exists at $t = 0$. On the other hand, it is ensured that at least the state trajectory will remain confined within $\Xi_1(t)$. Moreover, notice that the leader vehicle asymptotically moves towards the target x_f^1 thanks to the same arguments.

Conversely, the admissibility of the generic swarm SW_j exploits the following arguments: a) the PI sequence $\Xi_1(\cdot)$ is pairwise overlapped (4.28); b) according to the control architecture of Fig. 4.4, at each time instant the SW_j receives an admissible state trajectory path; c) the satisfaction of the transient phase

conditions (4.41).

First, the swarm SW_j receives the delayed state trajectory tube defined by the PI sequence $\Xi_1(\cdot)$ starting from the time instant t^j_{curr}, i.e $\Xi_j(t^j_{curr})$. Since the path planner provides a feasible state trajectory driving the swarm to the hyper-ball $\mathcal{B}_r(c_j(t^j_{curr}), m_j) \subseteq \Xi_1(t^j_{curr})$, the arguments a) and c) ensure that constraint and collision avoidance requirements with respect to the father swarm SW_{j-1} are always satisfied (see the constraint (4.51) of the optimization $\mathcal{P}^{i,j}_F$), while in the worst case the state trajectory of each $i-$th follower belonging to SW_j will be jailed into $\Xi_j(t^j_{curr})$, i.e the constraint (4.52) of $\mathcal{P}^{i,j}_F$. In virtue of the same arguments, the asymptotic stability property straightforwardly follows.

5

Laboratory Experiment and Results

In this chapter, a laboratory experiment is used to evaluate the performance of the proposed **DMPC-Swarm** Algorithm. All the computations have been carried out by using a setup implemented within the *MATLAB R2018b* environment and the Multi-Parametric toolbox 3.0 (MPT3) over a desktop computer equipped with an Intel Core $i7$ processor. The Multi-Parametric Toolbox (MPT) is a software tool for Matlab that aims at solving parametric optimization problems that arise in constrained optimal control. In particular, as the name of the toolbox suggests, its primal objective is to provide computationally efficient means for design and application of explicit model predictive control (MPC). Since the initial release in 2004 there has been a significant progress in the development of the toolbox and the scope of the toolbox has widened to deal also with problems arising in computational geometry.

5.1 Multi-Parametric Toolbox 3.0

The Multi-Parametric Toolbox is a collection of algorithms for modeling, control, analysis, and deployment of constrained optimal controllers developed under Matlab. It features a powerful geometric library that extends the application of the toolbox beyond optimal control to various problems arising in computational geometry. The new version 3.0 is a complete rewrite of the original toolbox with a more flexible structure that offers faster integration of new algorithms. The numerical side of the toolbox has been improved by adding interfaces to state of the art solvers and by incorporation of a new parametric solver that relies on solving linear-complementarity problems. The toolbox provides algorithms for design and implementation of real-time model predictive controllers that have been extensively tested.

On the market there exist toolboxes that offer operations involved purely in computational geometry, i.e. GEOMETRY toolbox , CGLAB, and Ellipsoidal Toolbox. Other toolboxes beside geometrical tools offer also algorithms

for computing and implementation of control routines e.g. the Hybrid toolbox, MOBYDIC toolbox, RACT toolbox, PnPMPC toolbox, and RoMulOC. MPT is also one of the tools that combines computational geometry with control routines. Many of these toolboxes including MPT rely on YALMIP which provides a high level language for modeling and formulating optimization problems.

The content of MPT can be divided into four modules:

- modeling of dynamical systems,
- MPC-based control synthesis,
- closed-loop analysis,
- deployment of MPC controllers to hardware.

Each part represents one stage in design and implementation of explicit MPC. The modeling module of MPT allows to describe discrete-time systems with either linear or hybrid dynamics. The latter can be directly imported from the HYSDEL environment. The control module allows to formulate and solve constrained optimal control problems for both linear and hybrid systems. For a detailed overview of employed mathematical formulations the reader is referred to [107]. The analysis module provides methods for investigation of closed-loop behavior and performance. Moreover, it also features methods to reduce complexity of explicit MPC feedbacks. The deployment part allows to export control routines to the ANSIC language, which can be subsequently downloaded to a target hardware implementation platform.

Compared to the previous release, the 3.0 version of MPT significantly improves capabilities of all four aforementioned modules. The main advances can be summarized as follows:

- Completely new installation procedure using a software manager.
- New optimization engines based on linear-complementarity problem solvers.
- Extended support for computational geometry.
- New flexible user interface based on object-oriented programming.
- Modular structure for easier integration of new algorithms.
- Extended support for real-time control.
- Improved numerical reliability based on extensive testing.
- Detailed documentation including examples and demos.

The new version of MPT is distributed in a modular structure that is operated by a Toolbox Manager available at www.tbxmanager.com. Toolbox Manager provides means for automatic installation, uninstallation and updates of Matlab toolboxes. The manager can be installed as per instructions on its web page. MPT 3.0 is composed of several modules that are required to achieve the full functionality. The base package is referred to as *mpt* and the related documentation as *mptdoc* which can be installed by issuing at the Matlab prompt. The other modules can be installed by pointing to the names

of the submodules.After installation of the submodules, the user can start using the software directly. If any module has been updated, the new versions can be obtained and installed with the help of the Toolbox Manager thus provides a very simple approach to keep updated with any future releases of MPT, including its submodules.

The MPT 3.0 comes with an extended structure that is based on submodules and object-oriented programming. The main motivation for this change was to achieve easier maintainability of the toolbox and to provide flexible structure for possible future enhancements. For instance, in the previous version of MPT there was a single object encompassing multiple algorithms. In the version 3.0, several new objects have been introduced that follow a hierarchy derived from object-oriented programming approach. Using this hierarchy it is possible to introduce new objects and methods to the existing framework with a minimal effort. The new class can be added by creating a new folder and by subclassing an existing object. The new object inherits properties and methods of the superclass and can be used to associate specific methods for tackling a particular problem.

Majority of the optimization problems involved in the computational geometry can be expressed as linear (LP) or quadratic problems (QP). To solve these problems effectively, MPT requires additional solvers that can be installed easily as submodules using the Toolbox Manager. Version 3.0 of MPT comes with new solvers that tackle both of these problems effectively. The new optimization engines are based on solvers for a linear complementarity problem (LCP) that represents a superclass for LP and QP. The advantage of representing and solving the optimization problems as LCPs is that a single solver covers all three scenarios and there is no need for multiple solvers that could potentially return different results. There are two new solvers implemented in MPT 3.0: LCP solver and parametric LCP solver. Both of these solvers will be reviewed next including their properties and implementation details.

1) *LCP Solver:* The linear-complementarity problem represents the class of optimization problems given as

$$\text{find } w, z$$

$$\begin{aligned} &\text{s.t.:} \\ &w - Mz = q \qquad (5.1) \\ &w^T z = 0 \end{aligned}$$

$$w, z \geq 0 \qquad (5.2)$$

where the problem data is given by a sufficient matrix $M \in \mathbb{R}^{n \times n}$ and vector $q \in \mathbb{R}^n$. The unknown variables are z and w that are coupled by the

linear complementarity constraints. LCP problems are well studied in the literatures and several efficient methods for solving such problems have been proposed. One of the most successful approaches to solve LCP is by employing the lexicographic Lemke's algorithm. This active set algorithm features a symbolic perturbation technique that ensures unique pivot step selection at each iteration, which prevents the method from internal cycling.

MPT 3.0 provides a C-code implementation of the lexicographic Lemke's algorithm, enriched by various techniques and methods to improve speed and numerical robustness of the method. In particular, the LU recursive factorization based on rank-one updates [126] has been incorporated to reduce computational time at each iteration. The LCP solver automatically performs scaling of the input data in case the problem is not well-conditioned. In addition, the LCP solver executes re-factorization of the basis if the lexicographic perturbation did not properly identify the unique pivot. The package is linked to BLAS and LAPACK numerical routines that provide state-of-the art algorithms for implementation of linear algebra. With all these features implemented, the LCP solver should provide a numerically reliable engine for resolving also difficult degenerate cases that may easily arise in formulations of MPC problems. The LCP solver is seamlessly integrated in MPT, but can also be installed separately via the Toolbox manager.

2) *Parametric LCP Solver:* The parametric LCP (PLCP) solver aims at solving the following class of problems

$$\text{find } w, z$$

$$\begin{aligned} \text{s.t.: } & w - Mz = q + Q\theta \\ & w^T z = 0 \end{aligned} \tag{5.3}$$

$$\begin{aligned} & w, z \geq 0 \\ & \theta \in \Theta \end{aligned} \tag{5.4}$$

which differs from (5.1) by the addition of the parametric term $Q\theta$ in (5.3) with $Q \in \mathbb{R}^{n \times d}$. Here, $\theta \in \mathbb{R}^d$ represents a free parameter, which is assumed to be bounded by (5.4), where $\Theta \subset \mathbb{R}^d$ is a polytope. The problem data are furthermore given by a sufficient matrix $M \in \mathbb{R}^{n \times n}$ and the vector $q \in \mathbb{R}^n$.

There are few advantages of solving PLP/PQP as PLCP. Firstly, a single method is used to tackle all three classes of problems that prevents from encoding inconsistencies that may be eventually caused by different algorithms and different tolerance settings. This was one of the problems in the

previous version where there were multiple versions for multi-parametric LP/QP solvers. Secondly, the PLCP approach is numerically robust and superior in efficiency to other methods. Furthermore, the PLCP approach can handle PLP/PQP problems where the parameters appear linearly in the cost function and in the right hand side of constraints and therefore is applicable to solve wider classes of practical problems.

3) *Interfaces to External Solvers:* Besides the new LCP solvers, MPT 3.0 provides interfaces to external state-of-the-art solvers. Supported solvers include, but are not limited to, CDD, GLPK, CLP, QPOASES, QPSPLINE, SeDuMi, GUROBI, and CPLEX. With the exception of the latter two, all other solvers are provided under an open-source license and can easily be installed using the Toolbox manager.

   It is worth noting that MPT 3.0 relies heavily on CDD solver for performing many tasks related to computational geometry. In particular, facet and vertex enumeration for convex polyhedra and polytopes, as well as elimination of redundant constraints, are delegated to CDD. For more information, the interested reader is referred to [118]. In addition, MPT 3.0 also requires a freely-available Fourier solver for computing projections of polyhedra and polytopes.

MPT 3.0 allows to formulate and solve model predictive control problems for discrete-time linear and hybrid prediction models. The control synthesis is split into two parts. First, the user specifies the prediction model either as a linear time invariant system, as a piecewise affine system, or as a Mixed Logical Dynamical (MLD) system. Subsequently, the model, along with constraints and specifications of the objective function, are passed to the control module which converts them into a suitable mathematical description of the optimal control problem.

1) *Modeling of Dynamical Systems:* MPC synthesis for linear systems in MPT 3.0 assumes that the prediction model takes the form

$$x(t+\Delta) = Ax(t) + Bu(t) + f, \tag{5.5}$$

$$y(t) = Cx(t) + Du(t) + g, \tag{5.6}$$

   where $x(t)$ is the state vector at time instant $t, x(t+)$ is the successor state at time $t+$ with denoting the sampling time, $u(t)$ is the vector of control inputs, and $y(t)$ denotes the vector of outputs. Such systems are represented in MPT 3.0 as instances of the LTI System class.

2) *Control Interface:* The basic type of an optimal control problem assumed in MPT 3.0 is formulated the following form:

$$\min \sum_{k=0}^{N-1} \left(\|Q_x x_k\|_p + \|Q_u u_k\|_p \right) \tag{5.7}$$

$$\text{s.t. } x_{k+1} = f\left(x_k, u_k\right), \tag{5.8}$$

$$\underline{u} \leq u_k \leq \bar{u}, \tag{5.9}$$

$$\underline{x} \leq x_k \leq \bar{x}, \tag{5.10}$$

where x_k and u_k denote, respectively, prediction of states and inputs at the k-th step of the prediction horizon N, $f(,)$ is the prediction equation, $\underline{x}, \bar{x}$ are lower/upper limits on the states, and $\underline{u}, \bar{u}$ represent limits of the control authority. If $p \in \{1, \infty\}$ in (10a), then $\|\cdot\|_{\{1,\infty\}}\|$ denotes the standard vector 1- or ∞-norm. If p = 2, then $\|Q_x x_k\|_2 = x_k^T Q_x x_k$ is assumed.

The numerical reliability of the toolbox has been tested on a large set of problems including randomly generated cases, MPC problems designed from a library of linear models [115], and numerous benchmark examples. At the time of alpha release the test set contained 1439 problems from which 206 were for interfaced solvers, 995 for the polyhedral library, and 238 for remaining functions in the control interface. These number are not final because the test problems are continuously added in the development process. During the testing period it has been shown that MPT 3.0 provides superior performance to the previous version and numerous problematic cases have been tackled by introducing new algorithms.

5.2 Elisa-3 robot introduction

The Elisa-3 [197] is a small circular robot with the following dimensions: diameter 50 mm, height 30 mm and weight 39 g. The two wheels are connected to a DC motor with a 25 : 1 reduction gear. The diameter of the wheels is 9 mm, while the distance amongst them 40.8 mm, and the other detailed specifications are reported in the following table.

Moreover, it is equipped with a wireless communication RF 2.4GHz, based on Nordic Semiconductor nRF24L01, in charge to perform send/receive data tasks from the robot to the personal computer and *viceversa*. The latter is achieved by means of the following devices: a radio base-station connected to a personal computer via a USB and a radio chip mounted on the robot.

Detailed specifications

Feature	Technical information
Size, weight	50 mm diameter, 30 mm height, 39 g
Battery, autonomy	LiIPo rechargeable battery (2 x 130 mAh, 3.7 V). About 3 hours autonomy. Recharging time about 1h e 30.
Processor	Atmel ATmega2560 @ 8MHz (~ 8 MIPS); 8 bit microcontroller
Memory	RAM: 8 KB; Flash: 256 KB; EEPROM: 4 KB
Motors	2 DC motors with a 25:1 reduction gear; speed controlled with backEMF
Magnetic wheels	Adesion force of about 1 N (100 g) depending on surface material and painting Wheels diamater = 9 mm Distance between wheels = 40.8 mm
Speed	Max: 60 cm/s
Mechanical structure	PCB, motors holder, top white plastic to diffuse light
IR sensors	8 infra-red sensors measuring ambient light and proximity of objects up to 6 cm; each sensor is 45° away from each other 4 ground sensors detecting the end of the viable surface (placed on the front-side of the robot)
IR emitters	3 IR emitters (2 on front-side, 1 on back-side of the robot)
Accelerometer	3D accelerometer along the X, Y and Z axis
LEDs	1 RGB LED in the center of the robot; 8 green LEDs around the robot
Switch / selector	16 position rotating switch
Communication	Standard Serial Port (up to 38kbps) Wireless: RF 2.4 GHz; the throughput depends on number of robot: eg. 250Hz for 4 robots, 10Hz for 100 robots; up to 10 m
Remote Control	Infra-red receiver for standard remote control commands
Expansion bus	Optional connectors: 2 x UART, I2C, 2 x PWM, battery, ground, analog and digital voltage
Programming	C/C++ programming with the AVR-GCC compiler (WinAVR for Windows). Free compiler and IDE (AVR Studio / Arduino)

They have one wheel on either side and a DC motor connected to each wheel with a 25 : 1 reduction gear. The robots have a $40.8mm$ distance between both wheels and the wheels themselves have a diameter of $9mm$. Elisa-3 is an evolution of the Elisa robot based on a different microcontroller and including a comprehensive set of sensors:

- Atmel 2560 microcontroller (Arduino compatible)
- central RGB led
- 8 green leds around the robot
- IRs emitters
- 8 IR proximity sensors (Vishay Semiconductors Reflective Optical Sensor)
- 4 ground sensors (Fairchild Semiconductor Minature Reflective Object Sensor)
- 3-axis accelerometer (Freescale MMA7455L)
- RF radio for communication (Nordic Semiconductor nRF24L01+)
- micro USB connector for programming, debugging and charging
- IR receiver

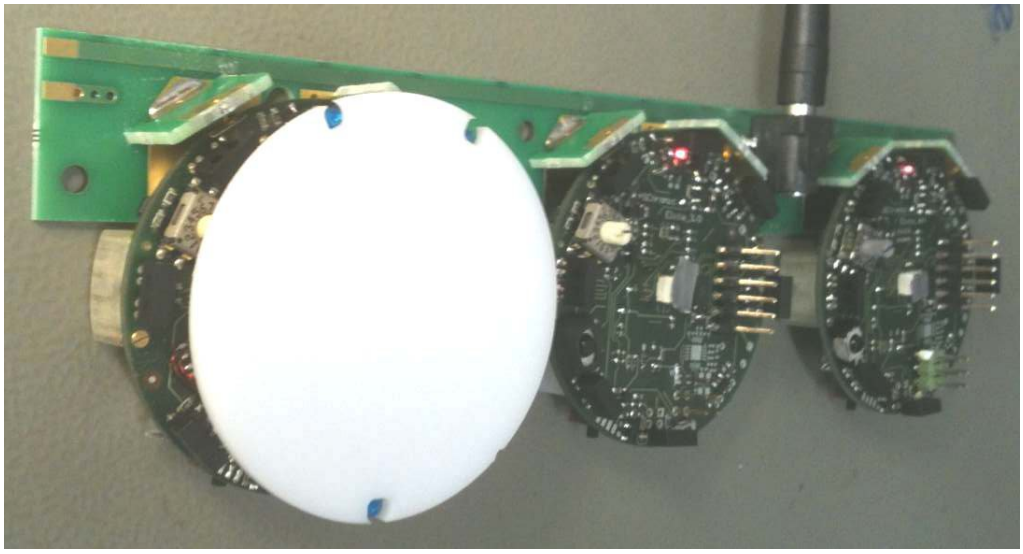

Fig. 5.1. Elisa3 and the charger.

- 2 DC motors
- top light diffuser
- selector

The robot is able to self charge using the charger station, as shown in the previous figure. The following figure illustrates the position of the various sensors:

- the top light diffuser and robot are designed to lock together, but the diffuser isn't fixed and can thus be removed as desired; the top light diffuser, as the name suggests, helps the light coming from the RGB led to be smoothly spread out, moreover the strip attached around the diffuser let the robot be better detected from others robots. Once the top light diffuser is removed, pay attention not to look at the RGB led directly. In order to remove the top light diffuser simply pull up it, then to place it back on top of the robot remember to align the 3 holes in the diffuser with the 3 IRs emitters and push down carefully until the diffuser is stable; pay attention to not apply too much force on the IRs emitters otherwise they can bend and stop working.

- when the top light diffuser is fit on top of the robot, then in order to change the selector position you can use the tweezers; the selector is located near the front-left IR emitter, as shown in Fig. 5.3.

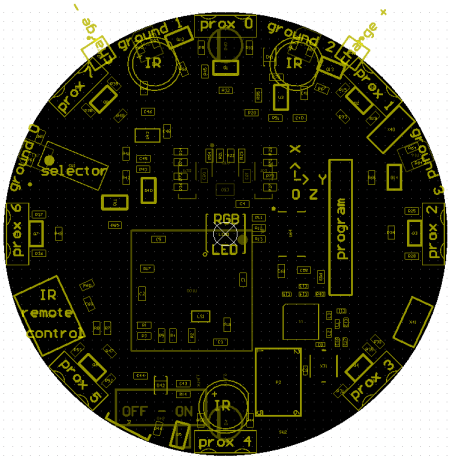

Fig. 5.2. Position of the various sensors.

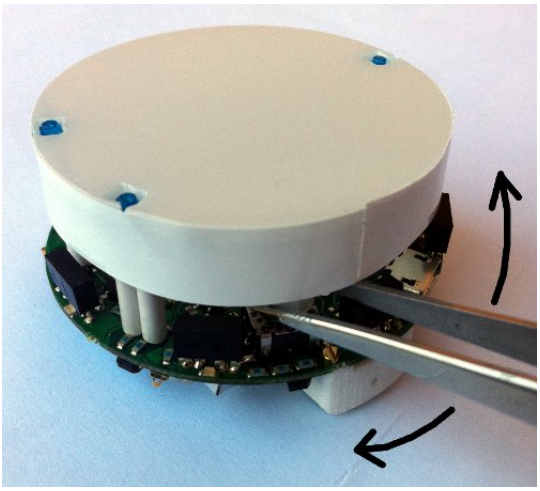

Fig. 5.3. Location of the selector.

- if you encounter problems with the radio communication (e.g. lot of packet loss) then you can try moving the antenna that is a wire near the robot label. Place the antenna as high as possible, near the plastic top light diffuser; try placing it in the borders in order to avoid seeing a black line on the top light diffuser when the RGB led is turned on.

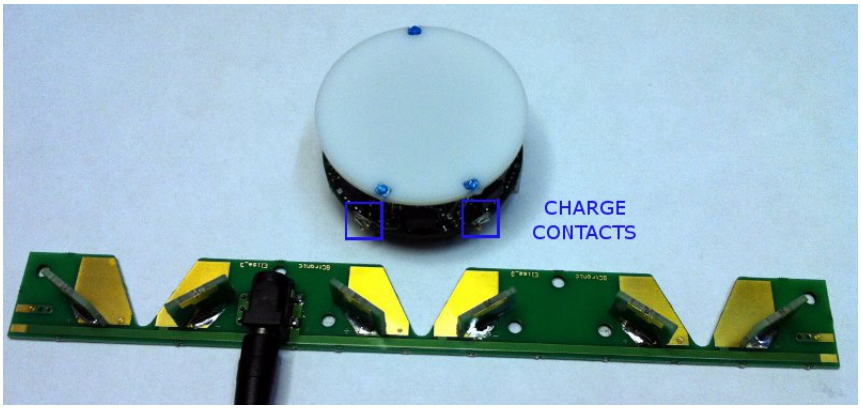

Fig. 5.4. The robot charger station.

The Elisa-3 can be piloted in the charger station in order to be automatically self charged; there is no need to unplug the battery for charging. The following figure shows the robot approaching the charger station; a led indicates that the robot is in charge.

The microcontroller is informed when the robot is in charge and this information is also transferred to the PC in the flags byte; this let the user be able to pilote the robot to the charger station and be informed when it is actually in charge. More information about the radio protocol can be found in the section Communication. Moreover the robot is also charged when the micro USB cable is connected to a computer; pay attention that if the USB cable is connected to a hub, this one need to be power supplied. The following video link shows the Elisa-3 piloted through the radio to the charging station using the monitor application: https://youtu.be/kjliXlQcgzw

From February 2013 onwards the Elisa-3 is equipped with a new top light diffuser designed to fit perfectly in the 3 IRs emitters of the robot. The diffuser is made of plastic (3d printed), it is more robust and it simplifies the removal and insertion. The following figures show the main components offered by the Elisa-3 robot and where they are physically placed.

The robot is equipped with two batteries for a duration of about 3 hours at normal usage (motors run continuously, IRs and RGB leds turned on). The radio base-station is connected to the PC through USB and transfers data to and from the robot wirelessly. In the same way the radio chip (nRF24L01+)

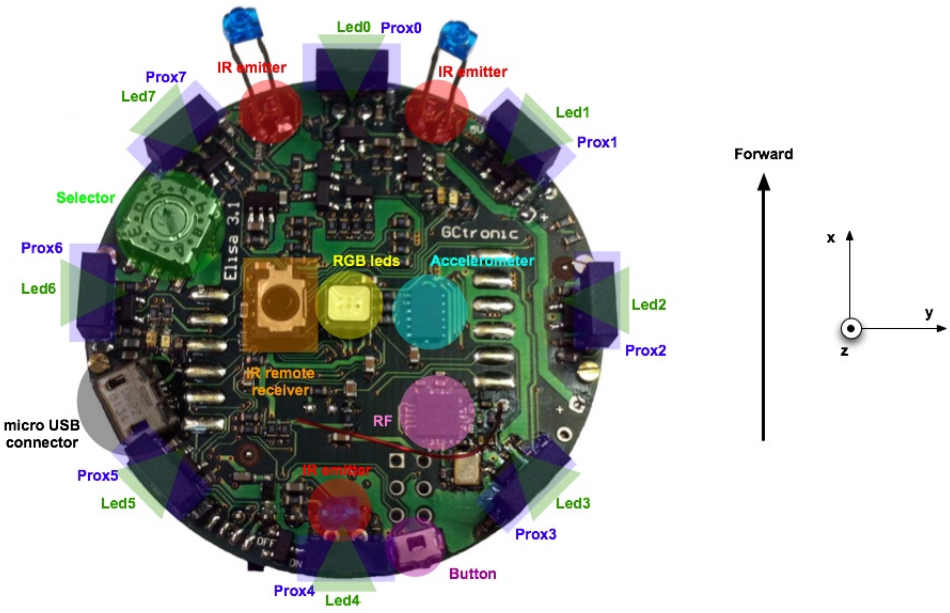

Fig. 5.5. The hardware physically placements.

mounted on the robot communicates through SPI with the microcontroller and transfers data to and from the PC wirelessly. The robot is identified by an address that is stored in the last two bytes of the microcontroller internal EEPROM; the robot firmware setup the radio module reading the address from the EEPROM. This address corresponds to the robot id written on the label placed under the robot and should not be changed.

The 13 bytes payload packet format is shown below (the number in the parenthesis expresses the bytes):

- Command: 0x27 = change robot state; 0x28 = goto base-station bootloader (this byte is not sent to the robot)
- Red, Blue, Green leds: values from 0 (OFF) to 100 (ON max power)
- IR + flags: first two bits are dedicated to the IRs
- third bit is reserved for enabling/disabling IR remote control (0 =>disabled, 1 =>enabled)
- fourth bit is used for sleep (1 => go to sleep for 1 minute)

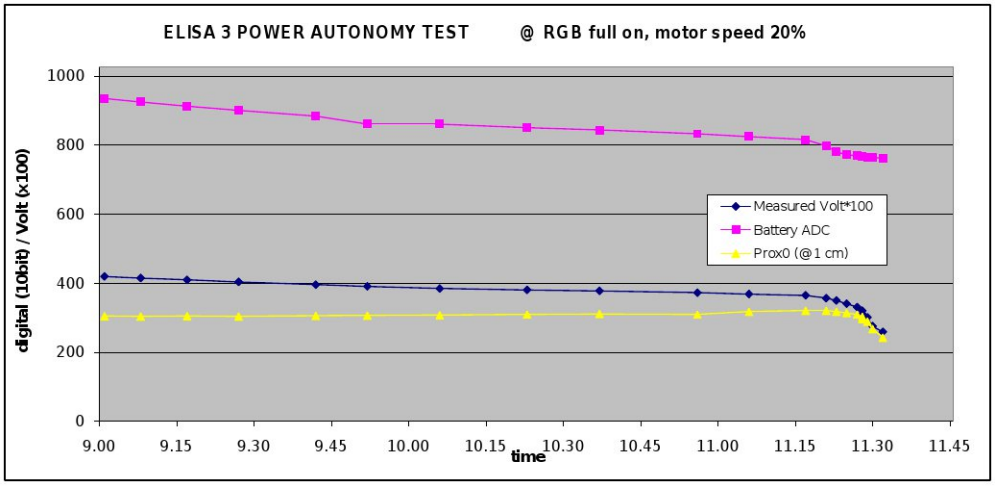

Fig. 5.6. The Elisa-3 power autonomy test.

- fifth bit is used to calibrate all sensors (proximity, ground, accelerometer) and reset odometry
- sixth bit is reserved (used by radio station)
- Right, Left motors: speed expressed in 1/5 of mm/s (i.e. a value of 10 means 50 mm/s); MSBit indicate direction: 1=forward, 0=backward; values from 0 to 127
- Small green leds: each bit define whether the corresponding led is turned on (1) or off (0); e.g. if bit0=1 then led0=on
- Remaining bytes free to be used

The communication between the pc and the base-station is controlled by the master (computer) that continuously polls the slave (base-station); the polling is done once every millisecond and this is a restriction on the maximum communication throughput. To overcome this limitation we implemented an optimized protocol in which the packet sent to the base-station contains commands for four robots simultaneously; the base-station then separate the data and send them to the correct robot address. The same is applied in reception, that is the base-station is responsible of receiving the ack payloads of 4 robots (64 bytes in total) and send them to the computer. This procedure let us have a throughput 4 times faster.

5.3 Operating arena and experimental knobs

Six autonomous robots of the Elisa-3 type are used for experiment purposes. Robots move within an arena of $0.8\,m \times 0.6\,m$ where their positions are detected via a Trust spotlight PRO RGB Camera, connected via a USB to the personal computer, and localized on the top of the arena, see Fig. 5.7.

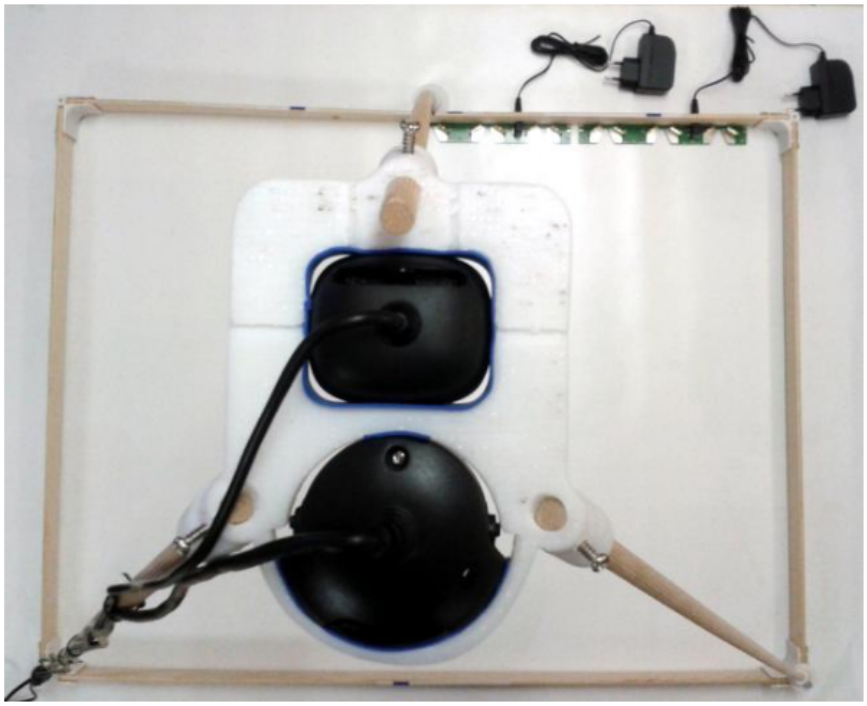

Fig. 5.7. Operating arena.

An *ad-hoc* video processing module (VPM) is in charge to recognize the robot position by detecting the center of the corresponding circular shape. For the sake of accuracy, the robot orientation within the arena is obtained as follows: by using a small circular black tag glued to the robot, the VPM first identifies the overall circle brighter than its external environment, see Fig. 5.8, then robot and black tag center positions are determined. Such information are then used to estimate position and orientation of each involved vehicle.

The operating scenario is depicted in Fig. 5.8. Three polyhedral obstacles

are considered $\mathcal{O} = \{Ob^1, Ob^2, Ob^3\}$. By choosing the left-down corner of the arena as the origin of the reference frame and according to (A.12), one has

$$H^1 = H^2 = H^3 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}^T$$

$g^1 = [0, -0.37, 0.31, -0.37]^T$, $g^2 = [0.23, -0.47, 0, -0.07]^T$ and $g^3 = [0.63, -0.80, 0.23, -0.45]^T$ $[m]$.

$$\begin{aligned} g^1 &= [0, -0.37, 0.31, -0.37]^T \quad [m], \\ g^2 &= [0.23, -0.47, 0, -0.07]^T \quad [m], \\ g^3 &= [0.63, -0.80, 0.23, -0.45]^T \ [m]. \end{aligned}$$

The team of six vehicles is organized according to the LF structure of Section 4. Models (1.25) and (4.1) are discretized with a sampling time $T_s = 0.4\,[s]$. Three swarms are defined: the vehicle belonging to SW_1 acts as the leader, while the followers SW_2 and SW_3 consist of three and two agents, respectively.

In Table 5.1, initial robot positions and orientations are reported, while the target for each vehicle is set to the same position, i.e. $x_f^i = x_f = [0.13,\ 0.13]^T\,[m], i = 1, \ldots, 6$. Moreover, the input constraints $|v| \leq 0.02\,[m/s]$ and $|\omega| \leq 0.5\,[rad/s]$, are prescribed for all six vehicles.

Table 5.1. Robots initial positions and orientations.

	Σ_1^1	Σ_2^1	Σ_2^2	Σ_2^3	Σ_3^1	Σ_3^2
$p_x(0)$ [m]	0.302	0.223	0.220	0.148	0.074	0.072
$p_y(0)$ [m]	0.518	0.579	0.458	0.519	0.571	0.472
$\theta(0)$ [rad]	0.245	0.139	-0.146	0.073	0.192	0.154

The leader vision radius is $R_v = 0.45\,[m]$ and its external perception is obtained by reconstructing the environment via the VPM capabilities. Finally, the following knobs are chosen: $\epsilon = 0.002\,[m]$, $\beta_{min} = 0.06\,[m]$, $\beta_{max} = 0.15\,[m]$. The control horizon lengths $N_2 = 13$ and $N_3 = 10$ are

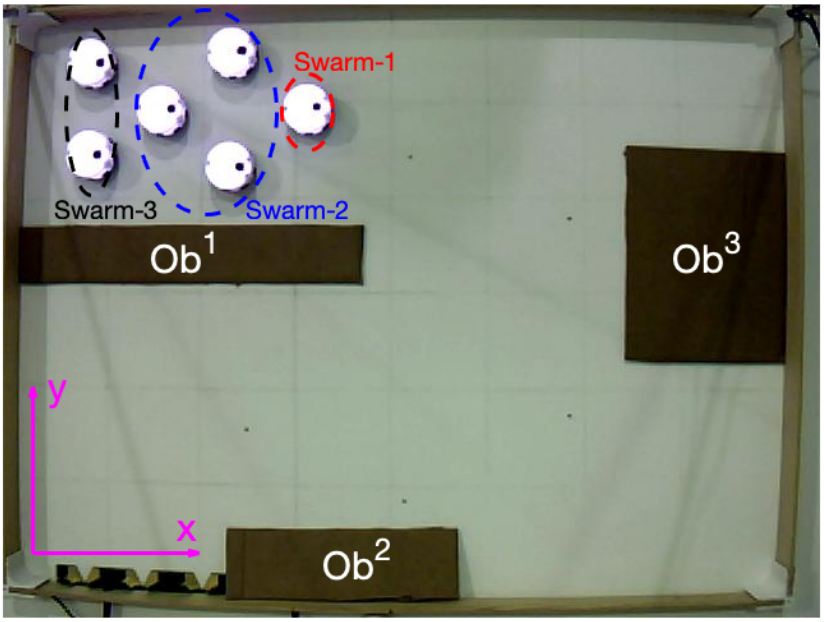

Fig. 5.8. Robots initial positions.

achieved by uniformly splitting the convergence kinematics time intervals $\bar{t}_2(0) = 5.02\,[s]$ and $\bar{t}_3(0) = 3.68\,[s]$.

5.4 Results

The experimental results are collected in Figs. 5.8-5.15. Starting from the initial conditions shown in Fig. 5.8, the state trajectories pertaining to the swarm agents are reported in Figs. 5.9-5.11, respectively. First it is important to underline that all the prescribed constraints are satisfied, see Figs. 5.13-5.15. Moreover as testified in Figs. 5.9 and 5.13, the leader is capable to exactly accomplish its own task: in fact the target x_f is reached at $t = 69.6\,[s]$, while the other robots asymptotically approach to the best admissible positions compatible with x_f by minimizing the distance criterion (A.18), see Figs.

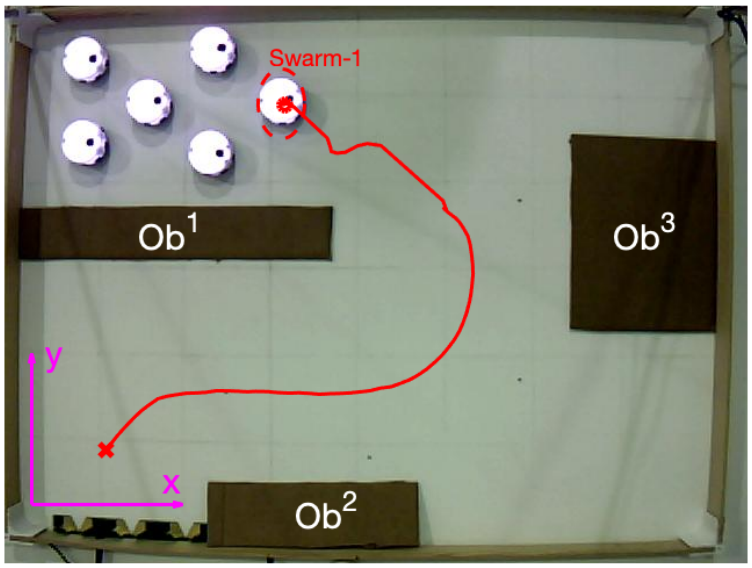

Fig. 5.9. SW_1 : state trajectories. Initial position ∘ and target ×.

5.10-5.11. According to Table 5.2 one derives that the maximum distance is $\|x_3^2(\infty) - x_f\|_2 = 0.476\,[m]$.

Table 5.2. Robot final positions

[m]	Σ_1^1	Σ_2^1	Σ_2^2	Σ_2^3	Σ_3^1	Σ_3^2
$(p_x)(\infty)$	0.137	0.387	0.417	0.326	0.428	0.540
$(p_y)(\infty)$	0.137	0.131	0.201	0.244	0.415	0.371

To appreciate the *modus operandi* of the proposed **DMPC-Swarm-Algorithm**, a detailed analysis on the state trajectory evolutions is hereafter summarized. Initially, the leader is the only robot moving within the arena: from $t = 0$ until $t = 24.4\,[s]$, see Fig. 5.9. Since the leader is the only vehicle

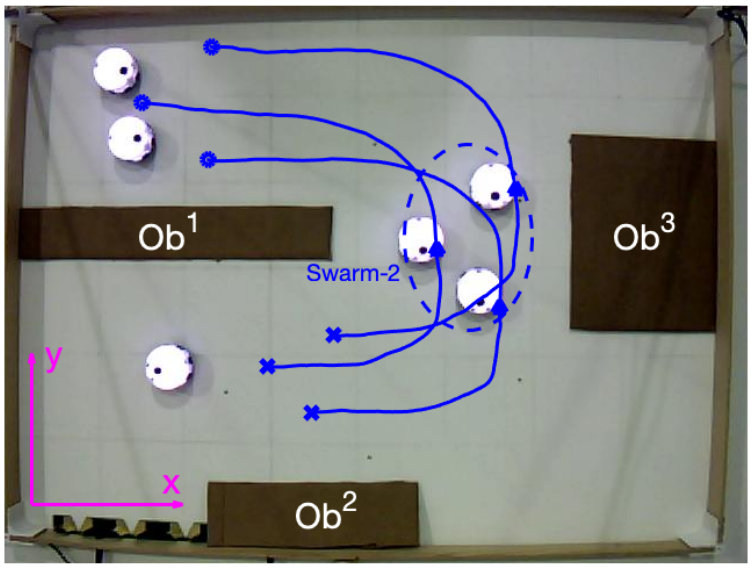

Fig. 5.10. SW_2 : state trajectories. Initial positions $\circ$, current positions $\triangle$ and targets $\times$.

equipped with a vision module, the main reason behind this phenomenon is that the overlapped condition in **Step 8** is satisfied at $t = 24.0\,[s]$: then, the swarm SW_2 receives the first admissible PI region $\Xi_1(24.0)$ at $t = 24.4\,[s]$ and, as a consequence, the command $u_1^{2^*}(24.4)$ is computed by solving the **DMPC**-$\mathcal{P}_F^{2,1}(24.4)$ optimization. The same reasoning applies to the swarm SW_3.

This behaviour can be better understood by taking a look to Fig. 5.12. There, a sub-sequence of the overlapped PI regions computed by Σ_1^1 is outlined. In particular, notice that the swarm SW_2 can transmit the green PI region $\Xi_2(24.4)$ to SW_3 (see **Step 29**) only when at $t = 58.8\,[s]$ the swarm SW_2 completely lies within the red PI region $\Xi_2(48.8)$.
While Σ_1^1 travels through the labyrinth defined by the obstacle scenario $\mathcal{O}$, see the continuous red line in Fig. 5.9, the followers SW_2 and SW_3 proceed in a "blind" fashion by exploiting the information received by the father along the platoon. As expected, each vehicle moves in a safe state trajectory tube thanks to the conditions imposed in (4.40)-(4.41), see the blue and green continuous lines in Figs. 5.10-5.11 respectively.
Finally, a complete video of the experiment is available at the following web link:

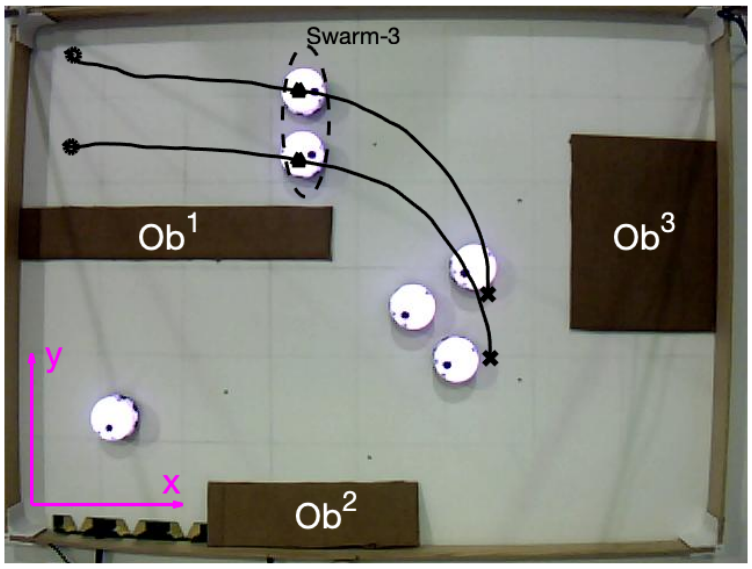

Fig. 5.11. SW_3 : state trajectories. Initial positions $\circ$, current positions $\triangle$ and targets $\times$.

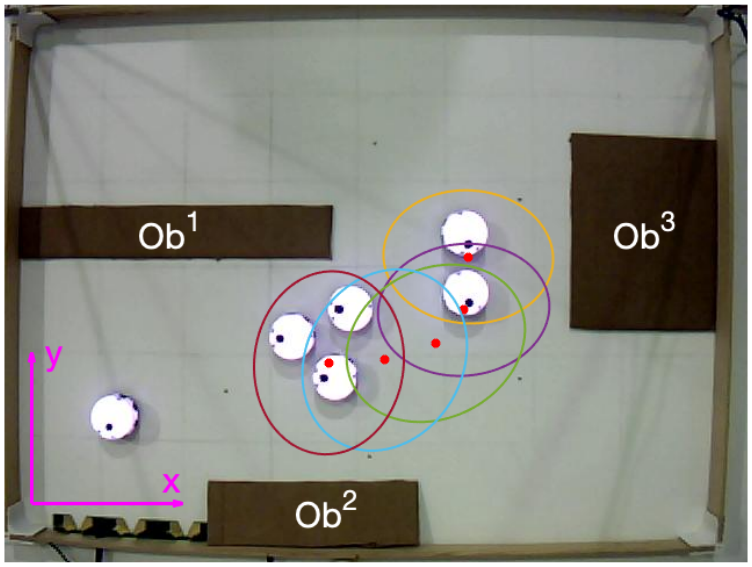

Fig. 5.12. Positively invariant regions.

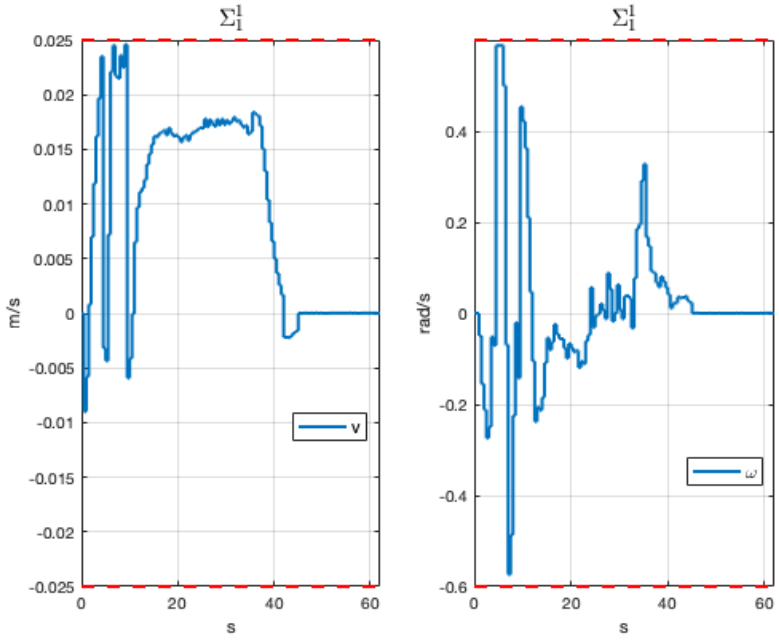

Fig. 5.13. SW_1 : command inputs. The dashed lines represent the boundaries of the prescribed constraint.

`https://www.youtube.com/watch?v=iI2G_sFUli4`.

5.5 Conclusions

Trough this thesis, a novel distributed model predictive control architecture is proposed for the coordination and control of multi-vehicle formations moving within uncertain environments. The key aim consists in mitigating the high memory consumption, resulting from local computations and the exploitation of a growing number of sensors as the involved agents increase, when path planning and obstacle avoidance requirements are concerned.

To this end, multi-agent swarm modelling and leader-follower configurations are jointly exploited within an *ad hoc* model predictive control framework to ameliorate energy savings that are essential in long-range missions. The distributed receding horizon control scheme is developed for teams of autonomous agents customized as swarms within platoon configurations.

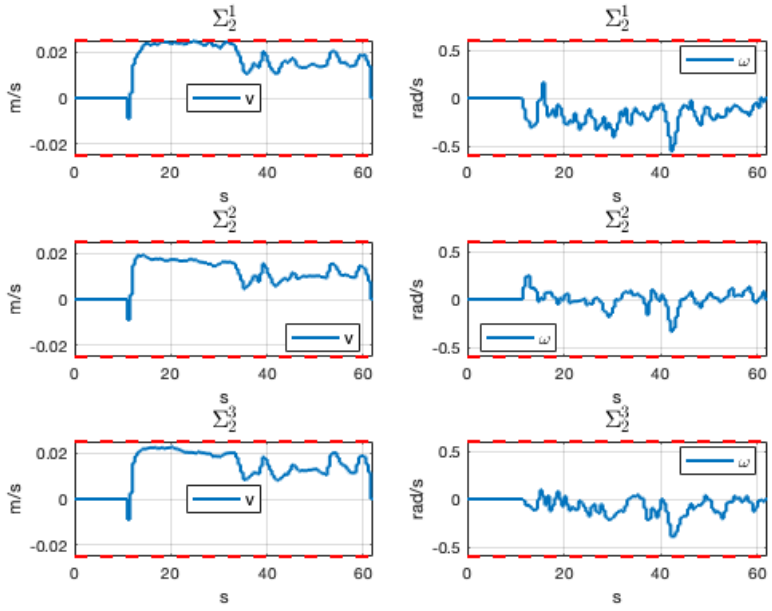

Fig. 5.14. SW_2 : command inputs. The dashed lines represent the boundaries of the prescribed constraint.

The main motivation behind the proposed control framework can be summarized as follows. Coordination and collision avoidance specs for multi-agent systems prescribe the use of high memory requirements for local computations and the exploitation of a growing number of sensors as the involved agents increase. The natural consequence is that usually short-range missions are allowed. In order to mitigate such a drawback, two key ingredients are here exploited: 1) the swarm formation modelling that allows to consider in some sense (it will be later clarified) several agents acting as a singleton; 2) an *ad hoc* model predictive scheme capable to adequately exploit swarm kinematics properties to ameliorate energy consumption savings. Finally, laboratory experiments on a group of six autonomous robots are instrumental to testify the effectiveness and the peculiarities of the proposed control strategy.

Trough the previous and presented chapters, a novel distributed MPC scheme has been developed for dealing with obstacle avoidance and path planning control problems for multi-agent systems. Differently from similar literature approaches, the proposed control architecture has been devoted to reduce as much as possible the energy consumption on each single agent. This objective has been pursued by exploiting an interesting property of the swarm kinematics: each agent belonging to a given swarm converges in a finite time

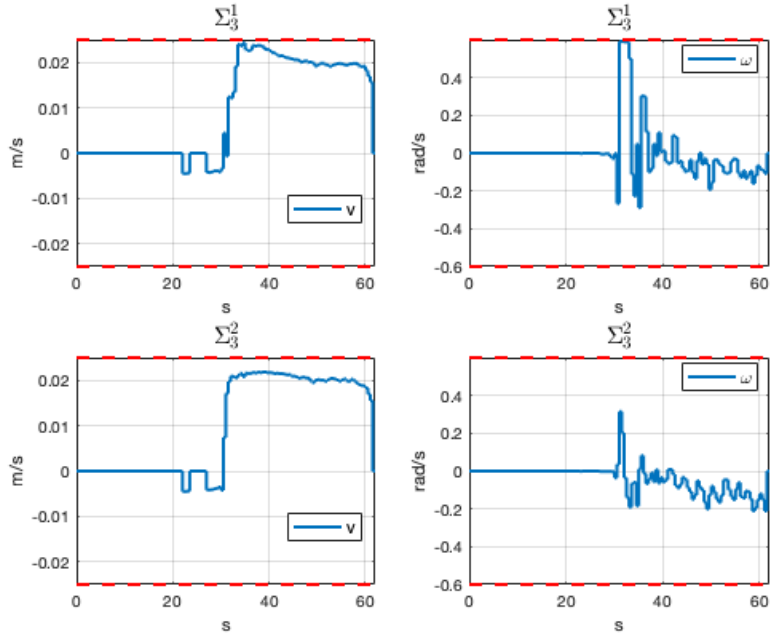

Fig. 5.15. SW_3 : command inputs. The dashed lines represent the boundaries of the prescribed constraint.

to a pre-assigned and suitably defined hyper-ball. Along these lines, the resulting control algorithm has been built up by jointly taking advantage from the swarm modelling and the receding horizon control features. A laboratory experiment on a team of Elisa-3 robots, moving within a planar environments subject to obstacle occurrences, has been carried out with the aim to show effectiveness and benefits of the proposed approach.

5.6 Future research directions

Despite the fact that the field has seen strong research activity over the last decade, new schemes still appear in the literature now and then. In addition, enhancements of previous schemes are also common, such as guarantees of new theoretical properties. Likely directions for future research are outlined below. From a theoretical perspective, Table 3 provide valuable information about what features need more attention. There are unexplored research directions for hybrid process systems. It is also interesting that only one of the 35 schemes is designed for transfer function models, given the number of pa-

TABLE 3 An overview of distributed model-predictive control (MPC) control architecture commonalities.

Distributed MPC Commonalities: Control Architecture				≈ 100% / ≈ 50% / ≈ 0%
Architecture	Decentralized [34], [57]	Distributed [28], [29], [31]–[33] [35]–[39], [41], [42] [44]–[53], [55], [56] [58], [60]–[62]	Hierarchical [30], [43], [46], [50] [54], [57], [59]	
Controller Knowledge	Strictly Local [28]–[38], [48]–[56]	Partially Global [39]–[47], [49], [57]–[62]		
Computation Type	Iterative [32]–[38], [43]–[50], [53]–[56], [60]–[62]	Noniterative [28]–[31], [39]–[42], [51], [52], [57]–[59]		
Controller Attitude	Cooperative [28], [29], [31]–[38], [43]–[46] [48]–[51], [53], [56], [58]–[62]	Noncooperative [30], [39]–[42], [47], [52], [55], [57]		
Communication	Serial [31], [38], [41], [53], [57], [59]	Parallel [28]–[30], [32]–[40], [42]–[58], [60]–[62]		
Timing	Synchronous [28]–[38], [40]–[53], [55]–[62]	Asynchronous [39], [47], [49], [54]		
Optimization Variables	Real [28]–[62]	Integer [41], [43], [49], [54]		

pers that describe predictive controllers in this context [63]–[65].

A topic that deserves more attention is the development of flexible distributed MPC architectures able to modify the control network topology and the communication burden depending on the circumstances [66]–[69]. The rationale behind these control schemes is to foster cooperation whenever the system performance is poor and to reduce cooperation when it is not necessary. That is, communication is only allowed whenever it significantly improves the system performance. The controllers are then grouped into time-varying coalitions that work cooperatively. Likewise, another related field where research is needed is that of plug-and-play [12] systems, that is, control schemes capable of seamlessly handling controllers that enter or leave the system, perhaps due to maintenance or because the system size dynamically changes.

From a practical point of view, comparative assessments of the many distributed MPC schemes available are still lacking. It is not clear what schemes are better with respect to basic features such as performance or communication and computational requirements. There are few direct comparisons between several different distributed MPC techniques on a common benchmark

[4], [70], and many schemes remain unassessed. For these comparisons to take place, it is necessary to have publicly available benchmarks. Currently, there are few benchmarks available. Having easy access to properly defined and publicly available benchmark systems will facilitate researchers in comparing the performance of their particular distributed MPC approaches with the performance of other distributed MPC approaches. To this end, a compilation of benchmarks has been made upon request at http://distributedmpc.net/. The assessment in [70] is performed on some of these benchmarks so researchers can compare the performance of their schemes with the results of other schemes reported in [70]. Finally, the application of distributed MPC to other fields must be explored. Distributed MPC has traditionally been linked strongly to distributed optimization. However, the actual application of distributed MPC schemes requires links to several other major research areas.

Besides the engineering systems to which distributed MPC and other more conventional approaches have been applied in the past, environmental systems are becoming of interest. Examples include water, ecological, and biosphere and atmospheric systems. Increasingly, sensors are being installed to continuously monitor environmental systems, leading to more information being collected. What should be done with this information is an open question. From a distributed MPC perspective, this information can be used to model the behavior and dynamics of these systems. The next step is then to use these models to predict what could happen with these systems in the near future.

Appendices

A

Definitions and Descriptions

The main goal of this appendix is to provide a brush-up of *Defenitions and Descriptions*.

A.1 Convex and non-convex Hull

There exists an incredible variety of point sets and polygons. Among them, some have certain properties that make them "nicer" than others in some respect. For instance, look at the two polygons shown below[149].

(a) A convex polygon.

(b) A non-convex polygon.

Fig. A.1. Examples of polygons

As it is hard to argue about aesthetics, let us take a more algorithmic stance. When designing algorithms, the polygon shown on the left appears

much easier to deal with than the visually and geometrically more complex polygon shown on the right. One particular property that makes the left polygon nice is that one can walk between any two vertices along a straight line without ever leaving the polygon [127]. In fact, this statement holds true not only for vertices but for any two points within the polygon. A polygon or, more generally, a set with this property is called convex.

$$A \text{ set } \mathrm{P} \subseteq \mathbb{R}^{\mathrm{d}} \text{ is convex if } \overline{\mathrm{pq}} \subseteq \mathrm{P}, \text{ for any } \mathrm{p}, \mathrm{q} \in \mathrm{P} \ . \tag{A.1}$$

The polygon shown in Figure A.1.b is not convex because there are some pairs of points for which the connecting line segment is not completely contained within the polygon. An immediate consequence of the definition is the following:

For any family $(\mathrm{P_i})_{\mathrm{i}\in\mathrm{I}}$ of convex sets, the intersection $\cap_{\mathrm{i}\in 1}\,\mathrm{P_i}$ is convex .

Indeed there are many problems that are comparatively easy to solve for convex sets but very hard in general. We will encounter some particular instances of this phenomenon later in the course. However, not all polygons are convex and a discrete set of points is never convex, unless it consists of at most one point only. In such a case it is useful to make a given set P convex, that is, approximate P with or, rather, encompass P within a convex setH $\supseteq$ P. Ideally, H differs from P as little as possible, that is, we want H to be a smallest convex set enclosing P.

A.2 Convex Polyhedral sets

A convex polyhedral cone K(G) in R^n is a set defined by the relation

$$K(G) = \left\{x \in R^n : \left(\exists z \in R_+^m : Gx = z\right)\right\} \tag{A.2}$$

where

$$G \in R^{m\times n} \tag{A.3}$$

If m = n and rank G = n, then the cone is said to be simplicial. One distinguishing idea which dominates many issues in optimization theory is convexity. An important reason is the fact that when a convex function is minimized over a convex set every locally optimal solution is global. Also, firstorder necessary conditions turn out to be sufficient. A variety of other properties conducive to computation and interpretation of solutions ride on convexity as well. In fact the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity.

Although we know by now what is the convex hull of point set, it is not

yet clear how to construct it algorithmically. As a first step, we have to find a suitable representation for convex hulls. In this section we focus on the problem in R^2 , where the convex hull of a finite point set forms a convex polygon. A convex polygon is easy to represent, for instance, as a sequence of its vertices in counterclockwise orientation. In higher dimensions finding a suitable representation for convex polytopes is a much more delicate task.
Input: $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2, n \in \mathbb{N}$

Output: Sequence $(q_1, \ldots, q_h), 1 \leqslant h \leqslant n$, of the vertices of conv (P) (ordered counterclockwise).

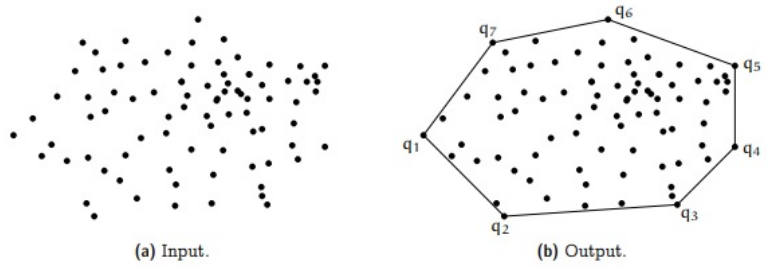

Fig. A.2. Convex Hull of a set of points in R^2 .

Another possible algorithmic formulation of the problem is to ignore the structure of the convex hull and just consider it as a point set. A generalization of the concept of positive or thant stability is that of invariance of polyhedral sets. The problem of checking whether a given polyhedron is invariant as well as the associated state-feedback synthesis problem have been considered by many authors, see e.g.,[112] and references therein.

A.3 Obstacle Scenario

The idea for approximating the obstacle shape is similar to the convexconcave procedure (also known as sequential convex programming) for solving optimization problems [Yuille and Rangarajan, 2003], where the concave part of the constraint is approximated using a linearization about the current solution.

Let Ob^j be an object with a polyhedral convex structure described as the intersection of h_j half-spaces:

$$Ob^j : \left[H_1^j, \ldots, H_{h_j}^j \right]^T \zeta \leq \left[g_1^j, \ldots, g_{h_j}^j \right]^T \tag{A.4}$$

with $\zeta \in^2$ accounting for the planar space. An obstacle scenario $\mathcal{O}$ is defined as

$$\mathcal{O} := \{Ob^1, \ldots, Ob^{n_o}\} \tag{A.5}$$

where n_o denotes the number of involved objects. □

A.3.1 Differential Inclusions

A differential inclusion (DI) is described by:

$$\dot{x} \in F(x(t), t), \quad x(0) = x_0 \tag{A.6}$$

where F is a set-valued function on $\mathbf{R}^n \times \mathbf{R}_+$. Any $x : \mathbf{R}_+ \to \mathbf{R}^n$ that satisfies (33) is called a solution or trajectory of the DI. Of course, there can be many solutions of the DI. Our goal is to establish that various properties are satisfied by all solutions of a given DI. For example, we might show that every trajectory of a given DI converges to zero as $t \to \infty$.

By a standard result called the Relaxation Theorem, we may as well assume F(x,t) is a convex set for every x and t. The DI given by

$$\dot{x} \in \mathbf{Co}F(x(t), t), \quad x(0) = x_0 \tag{A.7}$$

is called the relaxed version of the DI. Since $\mathrm{Co}\, F(x(t), t) \supseteq F(x(t), t)$ every trajectory of the DI (33) is also a trajectory of relaxed DI (34). Very roughly speaking, the Relaxation Theorem states that for many purposes the converse is true. (See the References for precise statements [142].) As a specific and simple example, it can be shown that for every DI we encounter in this book, the reachable or attainable sets of the DI and its relaxed version coincide, i.e., for every $T \geq 0$ $\{x(T) \mid x \text{ satisfies (33)}\} = \{x(T) \mid x \text{ satisfies (A.7)}\}$

In fact we will not need the Relaxation Theorem, or rather, we will always get it "for free"—every result we establish in the next two chapters extends immediately to the relaxed version of the problem. The reason is that when a quadratic Lyapunov function is used to establish some property for the DI, then the same Lyapunov function establishes the property for the relaxed DI.

A.4 Linear Differential Inclusions

A linear differential inclusion (LDI) is given by

$$\dot{x} \in \Omega x, \quad x(0) = x_0 \tag{A.8}$$

where Ω is a subset of $\mathbf{R}^{n\times n}$. We can interpret the LDI as describing a family of linear time-varying systems. Every trajectory of the LDI satisfies

$$\dot{x} = A(t)x, \quad x(0) = x_0 \tag{A.9}$$

for some $A : \mathbf{R}_+ \to \Omega$ Conversely, for any $A : \mathbf{R}_+ \to \Omega$ the solution of (36) is a trajectory of the LDI (A.8). In the language of control theory, the LDI might be described as an "uncertain time-varying linear system," with the set Ω describing the "uncertainty" in the matrix A(t).

We will encounter a generalization of the LDI described above to linear systems with inputs and outputs. We will consider a system described by

$$\begin{array}{l} \dot{x} = A(t)x + B_u(t)u + B_w(t)w, \quad x(0) = x_0 \\ z = C_z(t)x + D_{zu}(t)u + D_{zw}(t)w \end{array} \tag{A.10}$$

where $x : \mathbf{R}_+ \to \mathbf{R}^n, u : \mathbf{R}_+ \to \mathbf{R}^{n_u}, w : \mathbf{R}_+ \to \mathbf{R}^{n_w}, z : \mathbf{R}_+ \to \mathbf{R}^{n_z}$ x is referred to as the state, u is the control input, w is the exogenous input and z is the output. The matrices in (A.10) satisfy

$$\begin{bmatrix} A(t) & B_u(t) & B_w(t) \\ C_z(t) & D_{zu}(t) & D_{zw}(t) \end{bmatrix} \in \Omega \tag{A.11}$$

for all $t \geq 0$, where $\Omega \subseteq \mathbf{R}^{(n+n_z)\times(n+n_u+n_w)}$. We will be more specific about the form of $t \in \Omega$ shortly.

The differential inclusion theory represents the nonlinear system by a linear differential inclusion (LDI) model and the original nonlinear system is the son system of the linear differential inclusion system (LDIS). Though it introduces some conservativeness in the system model, the linear property applies a new method for the nonlinear filter design, which can be much easier than designing the filter for the nonlinear system directly. In this paper, a new

nonlinear filter is proposed based on the LDI theory. The nonlinear system is represented by the uncertain polytopic linear differential inclusion (PLDI) model with the existent condition for describing the general nonlinear system via a LDI model given in [152], on the basis of which the novel nonlinear filter is designed.

In some applications we can have one or more of the integers n_u, n_w, and n_z equal to zero, which means that the corresponding variable is not used. For example, the LDI $\dot{x} \in \Omega x$ results when $n_u = n_w = n_z = 0$. In order not to introduce another term to describe the set of all solutions of (A.10) and (A.11), we will call them a system described by LDIs or simply, an LDI.

A.5 Polytopic LDIs

When Ω is a singleton, the LDI reduces to the linear time-invariant (LTI) system

$$\begin{array}{c} \dot{x} = Ax + B_u u + B_w w, \quad x(0) = x_0 \\ z = C_z x + D_{zu} u + D_{zw} w \end{array} \tag{A.12}$$

where

$$\Omega = \left\{ \begin{bmatrix} A & B_u & B_w \\ C_z & D_{zu} & D_{zw} \end{bmatrix} \right\} \tag{A.13}$$

Although most of the results are well-known for LTI systems, some are new; we will discuss these in detail when we encounter them.

When Ω is a polytope, we will call the LDI a polytopic LDI or PLDI. Most of our results require that Ω be described by a list of its vertices, i.e., in the form

$$\mathrm{Co} \left\{ \begin{bmatrix} A_1 & B_{u,1} & B_{w,1} \\ C_{z,1} & D_{zu,1} & D_{zw,1} \end{bmatrix}, \ldots, \begin{bmatrix} A_L & B_{u,L} & B_{w,L} \\ C_{z,L} & D_{zu,L} & D_{zw,L} \end{bmatrix} \right\} \tag{A.14}$$

where the matrices are given.

If instead Ω is described by a set of l linear inequalities, then the number

of vertices, i.e., L, will generally increase very rapidly (exponentially) with l. Therefore results for PLDIs that require the description (41) are of limited interest for problems in which Ω is described in terms of linear inequalities.

Example 1:

Here is a PLDI example:

$$\dot{x} = A(t)x, \qquad A(t) \in \mathrm{Co}\left\{A_1, A_2\right\},$$

$$A_1 = \begin{bmatrix} -100 & 0 \\ 0 & -1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 8 & -9 \\ 120 & -18 \end{bmatrix}.$$

From

$$Q_0 \geq 0, \ldots, Q_L \geq 0, \quad Q_0 = \sum_{i=1}^{L} \left(Q_i A_i^T + A_i Q_i\right) \tag{A.15}$$

this PLDI is not quadratically stable if there exist $Q_0 \geq 0, Q_1 \geq 0$ and $Q_2 \geq 0$, not all zero, such that

$$Q_0 = A_1 Q_1 + Q_1 A_1^T + A_2 Q_2 + Q_2 A_2^T$$

It can be verified that the matrices

$$Q_0 = \begin{bmatrix} 5.2 & 2 \\ 2 & 24 \end{bmatrix}, Q_1 = \begin{bmatrix} 0.1 & 3 \\ 3 & 90 \end{bmatrix}, Q_2 = \begin{bmatrix} 2.7 & 1 \\ 1 & 1 \end{bmatrix}$$

satisfy these duality conditions.

However, the piecewise quadratic Lyapunov function

$$V(x) = \max\left\{x^T P_1 x, x^T P_2 x\right\} P_1 = \begin{bmatrix} 14 & -1 \\ -1 & 1 \end{bmatrix}, P_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{A.16}$$

proves that the PLDI is stable. To show this, we use the S-procedure. A necessary and sufficient condition for the Lyapunov function V defined in (43) to prove the stability of the PLDI is the existence of four nonnegative numbers $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ such that

$$\begin{array}{l} A_1^T P_1 + P_1 A_1 - \lambda_1 \left(P_2 - P_1\right) < 0 \\ A_2^T P_1 + P_1 A_2 - \lambda_2 \left(P_2 - P_1\right) < 0 \\ A_1^T P_2 + P_2 A_1 + \lambda_3 \left(P_2 - P_1\right) < 0, A_2^T P_2 + P_2 A_2 + \lambda_4 \left(P_2 - P_1\right) < 0 \end{array} \tag{A.17}$$

It can be verified that $\lambda_1 = 50, \lambda_2 = 0, \lambda_3 = 1, \lambda_4 = 100$ are such numbers. Finally, an LDI is stable if and only if there is a convex Lyapunov function that proves it [139].

A.6 Obstacle-free Region

We can define $\mathcal{B}(c, r)$ be the hyper-ball of center c and radius $r \in \mathbb{R}^+$. With $0_p \in \mathbb{R}^p$ we denote the vector of zero entries.

Let $\mathcal{B}_{in}(c_{in}, m_{in}) \subset^n$ and $\mathcal{B}_{fin}(c_{fin}, m_{fin}) \subset^n$ be two hyper-balls. Given any point $x_{in} \in \mathcal{B}_{in}(c_{in}, m_{in})$, the mapping operator $\mathcal{I} : \mathcal{B}_{in}(c_{in}, m_{in}) \rightarrow \mathcal{B}_{fin}(c_{fin}, m_{fin})$ returns the mirror point of x_{in}, i.e. $x_{fin} \in \mathcal{B}_{fin}(c_{fin}, m_{fin})$. of x_{in}.

There are some cases in which the form of the obstacle is so complex that the previous procedure would become computationally unfeasible. This is the case of obstacle avoidance in robotics.

Assume that a manipulator has to operate in a constrained environment as in Fig A.3 left. Assume that the angles q1 and q2 are the free coordinates of the robot which has to move in a constrained environment. Generally speaking, even though the allowable physical space is simple, the corresponding allowable region in the coordinate space might be very hard to describe. Typically it is non-convex as the white set in Fig.A.3 right.

Let $\mathcal{O}$ be an obstacle scenario. Then the non-convex obstacle-free region pertaining to $\mathcal{O}$ is identified as follows

$$\mathcal{O}_{free} := \{\zeta \in^2 \colon \zeta \in f(\zeta)\}, \tag{A.18}$$

where

$$f(\zeta) := \bigcap_{j=1}^{n_o} f^j(\zeta)$$

and

$$f^j(\zeta) := \left\{ \zeta \in^2 \colon \bigcup_{k=1}^{l_j} (H_k^j)^T \zeta > g_k^j \right\} \qquad \square$$

$f : \mathbb{R}^n \rightarrow \mathbb{R}^{n_f}$

represents the support function characterizing the admissibility state space region and n^f the number of component-wise inequalities.

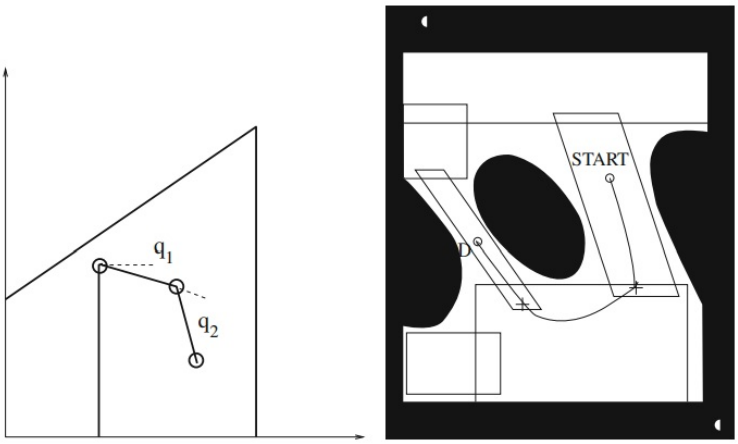

Fig. A.3. Robot in a constrained environment

One possible solution is to fill the admissible region with a family of simple overlapping sets, typically convex and compact (see, e.g., Fig A.3 right). This set-covering technique has been suggested in [143] and the idea is described in [167]. It is based on constructing regions with crossing points between regions and on equipping the system with a hierarchical control with

- *a high-level* global controller, which decides a path of connected sets in which the first includes the starting point and the last the destination point (D in the figure). The high level control makes use of a connection graph.
- a low-level local controller, active in each convex set, which tracks the reference (if the reference is inside the current set) or tracks a "crossing" point to another set of the sequence which is closer to the final set.

A.7 Positive invariance

The idea of positive invariance can be easily understood by referring to a simple autonomous system in state space form:

$$\dot{x}(t) = f(x(t)) \tag{A.19}$$

It is assumed that the above system of equations is defined in a proper open set

$$\mathcal{O} \subseteq \mathbb{R}^n \tag{A.20}$$

and that there exists a globally defined solution (i.e., for all $t \geq 0$) for every initial condition $x(0) \in \mathcal{O}$. Although the concept has been already considered, positive invariance is formally defined as follows.

The set $s \subseteq 0$ is said to be positively invariant w.r.t. (A.19) if every solution of (A.19) with initial condition $x(0) \in \mathcal{S}$ is globally defined and such that $x(t) \in \mathcal{S}$ for $t > 0$.

The above definition is all that is needed when the problem is well-posed, say there is a unique solution corresponding to each given initial condition $x(0) \in \mathcal{S}$. For the sake of generality, it is worth saying that if pathological situations have to be taken into account (say if one wants to consider the case in which the differential equation may have multiple solutions for the same initial condition) then the following weak version of the concept comes into play:

A.8 Robustly Positively Invariant Sets

The set $S \subseteq \mathcal{X}$ is said to be robustly positively invariant if, for all $x(0) \in \mathcal{S}$ and any $w(t) \in \mathcal{W}$, [1] the condition $x(t) \in \mathcal{S}$ holds for all $t \geq 0$.

To deal with synthesis problems a further definition, that of robust controlled invariance, has to be introduced. It is worth recalling that, for the control Lyapunov functions discussed in [146], synthesis requires the specification of the class of adopted controllers $\mathcal{C}$(output-feedback, state feedback, . . .). In a similar fashion, being the considered sets $\mathcal{S}$ defined in the plant state space, only static controllers (possibly of a suitably augmented plant) will be considered, since no additive dynamics can be admitted.

[1] Formally $w : \mathbb{R}^+ \to \mathcal{W}$.

The set $S \subseteq \mathcal{X}$ is said to be robust controlled positively invariant if there exists a control in the class $\mathcal{C}$ (assuring the existence and uniqueness of the solution for the closed-loop system) such that, for all $x(0) \in \mathcal{S}$ and $w(t) \in \mathcal{W}$, the condition $x(t) \in \mathcal{S}$ holds for all $t \geq 0$.

Note that this definition requires the existence of a control such that the problem is well-posed. For instance, assume $\mathrm{x}^{\cdot} = \mathrm{u}$ and $\mathcal{S}$ the positive real axis. Then any continuous control function $\boldsymbol{u} = \Phi(x)$ such that $\Phi(0) \geq 0$ would be in principle suitable since, for $x(0) \geq 0$, x(t) remains positive for $t > 0$. However $u = 1 + x^2$ is not acceptable, because the resulting equation has finite escape time.

The positive invariance notions just introduced are quite useful and several applications will be shown later. In the next section, a fundamental result which characterizes the invariance of a closed set will be presented.

Example 3:

Consider the system

$$\dot{x} = -\operatorname{sgn}[x] + w, \quad |w| \leq 1/2$$

The solution of this system maybe defined byabsorbing the system in a differential inclusions (an exhaustive book on differential inclusion is [138], see [139] for details). Intuitivelyit can be arguedthat, as long as $x(t) \neq 0$, the solution of this system is

$$x(t) = x(0) - \operatorname{sgn}(x(0))t + \int_0^t w(\sigma)d\sigma$$

which has the property $|x(t)| \leq \max\{0, |x(0)| - 1/2t\}$, and therefore converges in finite time to 0 for all the specified $w()$. Once 0 is reached the solution remains null, thus the set 0is necessarily robustly positively invariant. This claim can be easily proved in view of the fact that any interval of the form $[-\epsilon, \epsilon]$ is positively invariant(note that no discontinuity of f appears on the extrema).

A.9 The Sum of Squares Decomposition

We will now give a brief introduction to sum of squares (SOS) polynomials and show how the existence of an SOS decomposition can be verified using

semidefinite programming [139]. A more detailed description can be found in [144] and the references therein. We also present briefly an extension of the S-procedure [157] that is used in the main text.

Definition A.1. *For $x \in \mathbb{R}^n$, a multivariate polynomial p(x) is an SOS if there exist some polynomials $f_i(x), i = 1 \dots M$ such that $p(x) = \sum_{i=1}^{M} f_i^2(x)$.*

An equivalent characterization of SOS polynomials is given in the following proposition.

Proposition

A polynomial p(x) of degree 2d is an SOS if and only if there exists a positive semidefinite matrix Q and a vector of monomials Z(x) containing all monomials in x of degree $\leq d$ such that $p = Z(x)^T Q Z(x)$.

The proof of this proposition is based on the eigenvalue decomposition and can be found in [113]. In general, the monomials in Z(x) are not algebraically independent. Expanding $Z(x)^T Q Z(x)$ and equating the coefficients of the resulting monomials to the ones in p(x), we obtain a set of affine relations in the elements of Q. Since $p(x)$ being SOS is equivalent to $Q \geq 0$, the problem of finding a Q which proves that p(x) is an SOS can be cast as a semidefinite program (SDP). This was observed by Parrilo in [111].

Notation3 :

Note that p(x) being an SOS implies that $p(x) \geq 0$ for all $x \in \mathbb{R}^n$. However, the converse is not always true. Not all non-negative polynomials can be written as SOS, apart from three special cases: (i) when n = 2, (ii) when deg(p) = 2, and (iii) when n = 3 and deg(p) = 4. See [103] for more details.

Nevertheless, checking non-negativity of p(x) is an NP-hard problem when the degree of p(x) is at least 4 [138], whereas as argued in the previous paragraph, checking whether p(x) can be written as an SOS is computationally tractable — it can be formulated as an SDP, which has worst-case polynomial time complexity. We will not entail in a discussion on how conservative the relaxation is, but there are several results suggesting that this is not too conservative [114].

Notation4:

Note that as the degree of p(x) and/or its number of variables is increased, the computational complexity for testing whether p(x) is an SOS increases.

Nonetheless, the complexity overload is still a polynomial function of these parameters.

There is a close connection between sums of squares and robust control theory through Positivstellensatz, a central theorem in Real algebraic geometry [94]. This theorem allows us to formulate a hierarchy of polynomialtime computable stronger conditions [110] for the S-procedure type of analysis [127, 113]. To see how we will be using this result say we want to use the S-procedure to check that the set:

$$\{p(x) \geq 0 \text{ when } p_i(x) \geq 0 \text{ for } i = 1, \ldots, n\}$$

is non-empty. Instead of finding positive constant multipliers (the standard S-procedure), we search for SOS multipliers $h_i(x)$ so that

$$p(x) - \sum_i h_i(x) p_i(x) \text{ is a SOS.} \tag{A.21}$$

Since $h_i(x) \geq 0$ and condition (48) is satisfied, for any x such that $p_i(x) \geq 0$ we automatically have $p(x) \geq 0$, so sufficiency follows. This condition is at least as powerful as the standard S-procedure, and many times it is strictly better; it is a special instance of positivstellensatz. By putting an upper bound on the degree of h_i we can get a nested hierarchy of polynomial-time checkable conditions.

Besides this, what is more interesting is the case in which the monomials in the polynomial $p(x)$ have unknown coefficients, and we want to search for some values of those coefficients such that $p(x)$ is a sum of squares (and hence nonnegative). Since the unknown coefficients of $p(x)$ are related to the entries of Q via affine constraints, it is evident that the search for the coefficients that make $p(x)$ an SOS can also be formulated as an SDP (these coefficients are themselves decision variables). This observation is crucial in the construction of Lyapunov functions and other S-procedure type multipliers.

Example 4:

Consider the whirling pendulum [92] shown in Figure A.4. It is a pendulum of length l_p whose suspension end is attached to a rigid arm of length l_a, with a mass m_b attached to its free end. The arm rotates with angular velocity $\dot{\theta}_a$. The pendulum can oscillate with angular velocity $\dot{\theta}_p$ in a plane normal to the arm, making an angle θ_p with the vertical in the instantaneous plane of motion. We will ignore frictional effects and assume that all links are

slender so that their moment of inertia can be neglected.

Using $x_1 = \theta_p$ and $x_2 = \dot{\theta}_p$ as state variables, we obtain the following state equations for the system:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \dot{\theta}_a^2 \sin x_1 \cos x_1 - \frac{g}{l_p} \sin x_1 \end{aligned} \tag{A.22}$$

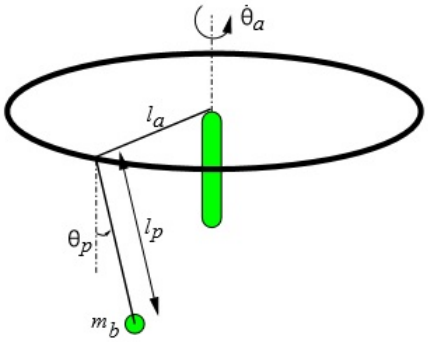

Fig. A.4. The whirling pendulum

The number and stability properties of equilibria in this system depend on the value of $\dot{\theta}_a$. When the condition

$$\dot{\theta}_a^2 < g/l_p \tag{A.23}$$

is satisfied, the only equilibria in the system are $(x1, x2)$ satisfying $\sin x_1 = 0$, $x_2 = 0$.

One equilibrium corresponds to $x_1 = 0$, i.e., the pendulum is hanging vertically downward (stable), and the other equilibrium corresponds to $x_1 = \pi$, i.e., the vertically upward position (unstable). As $\dot{\theta}_a^2$ a is increased beyond g/l_p, a supercritical pitchfork bifurcation of equilibria occurs [116]. The $(x1, x2) = (0, 0)$ equilibrium becomes unstable, and two other equilibria appear. These equilibria correspond to $\cos x_1 = \frac{g}{l_p \dot{\theta}_a^2}, x_2 = 0$.

We will now prove the stability of the equilibrium point at the origin for $\dot{\theta}_a$ satisfying (50), by constructing a Lyapunov function. Obviously the energy of this mechanical system can be used as a Lyapunov function, but since our purpose is to show that a Lyapunov function can be found using the SOS decomposition, we will assume that our knowledge is limited to the state equations describing the system and that we know nothing about the underlying energy.

Since the vector field (49) is not polynomial, a transformation to a polynomial vector field must be performed before we are able to construct a Lyapunov function using the SOS decomposition. For this purpose, introduce $u_1 = \sin x_1$ and $u_2 = \cos x_1$ to get:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \dot{\theta}_a^2 u_1 u_2 - \tfrac{g}{l_p} u_1 \end{aligned} \tag{A.24}$$

$$\begin{aligned} \dot{u}_1 &= x_2 u_2 \\ \dot{u}_2 &= -x_2 u_1 \end{aligned} \tag{A.25}$$

In addition, we have the algebraic constraint

$$u_1^2 + u_2^2 - 1 = 0. \tag{A.26}$$

The whirling pendulum system will now be described by Equations (A.24)–(A.25). Notice that all the functions here are polynomial, so that Proposition 4 can be used to prove stability.

We will perform the analysis with the parameters of the system set at some fixed values. Assume that all the parameters except g are equal to 1, and g itself is equal to 10, for which condition (50) is satisfied. For a mechanical system like this, we expect that some trigonometric terms will be needed in the Lyapunov function. Thus we will try to find a Lyapunov function of the following form:

$$\begin{aligned} V &= a_1 x_2^2 + a_2 u_1^2 + a_3 u_2^2 + a_4 u_2 + a_5 \\ &= a_1 x_2^2 + a_2 \sin^2 x_1 + a_3 \cos^2 x_1 + a_4 \cos x_1 + a_5 \end{aligned} \tag{A.27}$$

where the a_i's are the unknown coefficients. These coefficients must satisfy

$$a_3 + a_4 + a_5 = 0 \tag{A.28}$$

for V to be equal to zero at $(x1, x2) = (0, 0)$. To guarantee that V is positive

definite, we search for V_s that satisfy

$$V - \epsilon_1 (1 - u_2) - \epsilon_2 x_2^2 \geq 0 \tag{A.29}$$

where ϵ_1 and ϵ_2 are positive constants (we set $\epsilon_1 \geq 0.1, \epsilon_2 \geq 0.1$). Positive definiteness holds as

$$\epsilon_1 (1 - u_2) + \epsilon_2 x_2^2 = \epsilon_1 (1 - \cos x_1) + \epsilon_2 x_2^2$$

is a positive definite function in the $(x1, x2)$–space (assuming all x_1 that differ by 2π are in the same equivalence class). An example of Lyapunov function for this whirling pendulum system, found using the sum of squares procedure, is given by

$$V = 0.33445 x_2^2 + 1.4615 u_1^2 + 1.7959 u_2^2 - 6.689 u_2 + 4.8931$$

Useful Results:

This Part contains the basic definitions and results concerning invariant sets in control and it is the core of the Chapter. Indeed, the invariance concept is at the basis of many control schemes that will be considered later. Such a concept naturally arises when dealing with Lyapunov functions, as we have seen, since any Lyapunov function has positively invariant sublevel sets. However, the invariance concept does not require the introduction of the notion of Lyapunov functions and indeed there exist invariant sets that are not obviously related to any Lyapunov function.

The reason for the introduction of the (rarely used throughout the chapter) weak invariance concept is basically that of establishing a link with the abundant mathematical work in this area (see, among the recent literature [135, 141]). Indeed, from an engineering point of view, the existence of at least a solution in S is not that stunning, since nothing is said about all other possible solutions to the given set of equations.

To avoid having the reader dropping the chapter we guarantee that in the 99.999% of the dynamic systems which will be considered well-posedness will be assumed, namely the existence of a unique solution for any $x(0) \in \mathcal{S}$, a case in which the weak definition collapses to the standard one. The latter can be simply restated as $x(t_1) \in \mathcal{S} \Rightarrow x(t) \in \mathcal{S}$ for $t \geq t_1$. It has to be pointed out

that the role of the word "positive" is referred to the fact that the property regards the future. If $x(t_1) \in \mathcal{S}$ implies $x(t) \in \mathcal{S}$, for all t, this property is known as invariance and S is said to be an invariant set. Invariance is a too special concept to be considered. Therefore in the chapter we will always refer to positive invariance (although we will sometimes write "invariance" for brevity).

References

1. N. R. Sandell, P. Varaiya, M. Athans, and M. G. Safonov, *"Survey of decentralized control methods for large scale systems,"* IEEE Trans. Autom. Control, vol. 23, no. 2, pp. 108–128, 1978.
2. E. Camponogara and B. B. de Oliveira, *"Distributed optimization for model predictive control of linear-dynamic networks,"* IEEE Trans. Syst. Man, Cybern. A, vol. 39, no. 6, pp. 1331–1338, 2009.
3. R. R. Negenborn, B. De Schutter, and J. Hellendoorn, *"Multi-agent model predictive control for transportation networks: Serial versus parallel schemes,"* Eng. Applicat. Artif. Intell., vol. 21, no. 3, pp. 353–366, 2008.
4. R. M. Hermans, A. Jokic, M. Lazar, A. Alessio, P. P. J. van den Bosch, I. A. Hiskens, and A. Bemporad, *"Assessment of non-centralised model predictive control techniques for electrical power networks,"* Int. J. Control, vol. 85, no. 8, pp. 1162–1177, 2012.
5. E. F. Camacho and C. Bordons, *Model Predictive Control,* vol. XXII, 2nd ed. London, England: Springer-Verlag, 2004.
6. J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design. Madison,* WI: Nob Hill Publishing, 2009.
7. J. Richalet, A. Rault, J. L. Testud, and J. Papon, *"Model predictive heuristic control: Applications to industrial processes,"* Automatica, vol. 14, no. 5, pp. 413–428, 1978.
8. E. F. Camacho and M. Berenguel, *"Robust adaptive model predictive control of a solar plant with bounded uncertainties,"* Int. J. Adapt. Control Signal Process., vol. 11, no. 4, pp. 311–325, 1997.
9. Ø. Hegrenæs, J. T. Gravdahl, and P. Tøndel, *"Spacecraft attitude control using explicit model predictive control," Automatica,* vol. 41, no. 12, pp. 2107–2114, 2005.
10. S. J. Qin and T. A. Badgwell, *"A survey of industrial model predictive control technology," Control Eng. Pract.,* vol. 11, no. 7, pp. 733–764, July 2003.
11. J. R. D. Frejo and E. F. Camacho, *"Global versus local MPC algorithms in freeway traffic control with ramp metering and variable speed limits," IEEE Trans. Intell. Transport. Syst.,* vol. 13, no. 4, pp. 1556–1565, 2012.

12. S. Riverso, M. Farina, and G. Ferrari-Trecate, *"Plug-and-play decentralized model predictive control for linear systems,"* IEEE Trans. Autom. Control, vol. 58, no. 10, pp. 2608–2614, Oct. 2013.
13. J. A. Momoh, *"Smart grid design for efficient and flexible power networks operation and control,"* in Proc. IEEE Power Systems Conf. Expo., 2009, pp. 1–8.
14. F. Xia, Y. C. Tian, Y. Li, and Y. Sung, *"Wireless sensor/actuator network design for mobile control applications," Sensors,* vol. 7, no. 10, pp. 2157–2173, 2007.
15. R. R. Negenborn, P. J. Van Overloop, T. Keviczky, and B. De Schutter, *"Distributed model predictive control of irrigation canals," Netw. Heterogeneus Media,* vol. 4, no. 2, pp. 359–380, 2009.
16. J. M. Maestre, D. M. de la Peña, and E. F. Camacho, *"Distributed MPC: A supply chain case study,"* in Proc. Conf. Decision Control, 2009, pp. 7099–7104.
17. J. L. Nabais, R. R. Negenborn, R. C. Benitez, and M. A. Botto, *"Setting cooperative relations among terminals at seaports using a multi-agent system,"* in Proc. 16th Int IEEE Conf. Intelligent Transportation Systems, The Hague, The Netherlands, Oct. 2013., pp. 1731–1736.
18. H. E. Fawal, D. Georges, and G. Bornard, *"Optimal control of complex irrigation systems via decomposition-coordination and the use of augmented lagrangian,"* in Proc. IEEE Int. Conf. Systems, Man, Cybernetics, 1998, vol. 4, pp. 3874–3879.
19. E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, *"Distributed model predictive control," IEEE Control Syst. Mag.,* vol. 22, no. 1, pp. 44–52, 2002.
20. R. Scattolini, *"Architectures for distributed and hierarchical model predictive control—A review," J. Process Control,* vol. 19, no. 5, pp. 723–731, 2009.
21. P. D. Christofides, R. Scattolini, D. M. de la Peña, and J. Liu, *"Distributed model predictive control: A tutorial review and future research directions," Comput. Chem. Eng.,* vol. 51, pp. 21–41, Apr. 2013.
22. J. M. Maestre and R. R. Negenborn, Eds. *Distributed Model Predictive Control Made Easy* (Series Intelligent Systems, Control and Automation: Science and Engineering). vol. 69, New York: Springer, 2013.
23. R. R. Negenborn and H. Hellendoorn, *"Intelligence in transportation infrastructures via model-based predictive control," in Intelligent Infrastructures,* R. R. Negenborn, Z. Lukszo, and H. Hellendoorn, Eds. Dordrecht, The Netherlands: Springer, 2010, pp. 3–24.
24. W. B. Dunbar and S. Desa, *"Distributed MPC for dynamic supply chain management," in Proc. Int. Workshop Assessment Future Directions NMPC,* Freudenstadt-Lauterbad, Germany, 2005, pp. 26–30.
25. J. L. Nabais, R. R. Negenborn, R. C. Benitez, and M. A. Botto, *"A multiagent MPC scheme for vertically integrated manufacturing supply chains,"* in Proc. 6th Int. Conf. Management Control Production Logistics, Fortaleza, Brazil, Sept. 2013, pp. 59–64.
26. J. M. Maestre, D. M. de la Peña, E. F. Camacho, and T. Alamo, *"Distributed model predictive control based on agent negotiation,"* J. Process Control, vol. 21, no. 5, pp. 685–697, 2011.
27. R. R. Negenborn and J. M. Maestre, *"On 35 approaches for distributed MPC made easy," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. The Netherlands: Springer, 2014, pp. 1–37.

28. F. Valencia, J. D. López, J. A. Patiño, and J. J. Espinosa, *"Bargaining game based distributed MPC," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 41–56.
29. P. A. Trodden and A. G. Richards, *"Cooperative tube-based distributed MPC for linear uncertain systems coupled via constraints,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 57–72.
30. R. Martí, D. Sarabia, and C. de Prada, *"Price-driven coordination for distributed NMPC using a feedback control law,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 73–88.
31. M. A. Müller and F. Allgöwer, *"Distributed MPC for consensus and synchronization," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 89–100.
32. R. Bourdais, J. Buisson, D. Dumur, H. Guéguen, and P.-D. Moros ¸an, *"Distributed MPC under coupled constraints based on Dantzig-Wolfe decomposition,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 101–114.
33. F. Farokhi, I. Shames, and K. H. Johansson, *"Distributed MPC via dual decomposition and alternative direction method of multipliers," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 115–131.
34. F. Tedesco, D. M. Raimondo, and A. Casavola, *"A distributed reference management scheme in presence of non-convex constraints: An MPC based approach,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 243–257.
35. A. Casavola, E. Garone, and F. Tedesco, *"The distributed command governor approach in a nutshell,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 259–274.
36. I. Prodan, F. Stoican, S. Olaru, C. Stoica, and S.-I. Niculescu, *"Mixedinteger programming techniques in distributed MPC problems,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 275–291.
37. A. Grancharova and T. A. Johansen, *"Distributed MPC of interconnected nonlinear systems by dynamic dual decomposition," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 293–308.
38. P. Giselsson and A. Rantzer, *"Generalized accelerated gradient methods for distributed MPC based on dual decomposition," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 309–325.
39. A. Kozma, C. Savorgnan, and M. Diehl, *"Distributed multiple shooting for large scale nonlinear systems," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 327–340.

40. G. Betti, M. Farina, and R. Scattolini, *"Distributed MPC: A noncooperative approach based on robustness concepts,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 421–435.
41. R. R. Negenborn, *"Decompositions of augmented lagrange formulations for serial and parallel distributed MPC,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 437–450.
42. A. Zafra-Cabeza and J. M. Maestre, *"A hierarchical distributed MPC approach: A practical implementation,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 451–464.
43. J. L. Nabais, R. R. Negenborn, R. B. Carmona-Benitez, L. F. Mendonca, and M. A. Botto, *"Hierarchical MPC for multiple commodity transportation networks," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 535–552.
44. J. Liu, D. M. de la Peña, and P. D. Christofides, *"Lyapunov-based distributed MPC schemes: Sequential and iterative approaches,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 479–494.
45. J. M. Lemos and J. M. Igreja, *"D-SIORHC, distributed MPC with stability constraints based on a game approach,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 133–146.
46. M. Y. Lamoudi, M. Alamir, and P. Béguery, *"A distributed-in-time NMPC-based coordination mechanism for resource sharing problems,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 147–162.
47. I. Necoara, *"Rate analysis of inexact dual fast gradient method for distributed MPC," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 163–178.
48. B. Biegel, J. Stoustrup, and P. Andersen, *"Distributed MPC via dual decomposition," in Distributed Model Predictive Control Made Easy* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 179–192.
49. E. Camponogara, *"Distributed optimization for MPC of linear dynamic networks," in Distributed Model Predictive Control Made Easy,* J. M. Maestre August 2014 IEEE CONTROL SYSTEMS MAGAZINE 97 and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 193–208.
50. Y. Hu and N. H. El-Farra, *"Adaptive quasi-decentralized MPC of networked process systems," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 209–224.
51. C. Ocampo-Martinez, V. Puig, J. M. Grosso, and S. Montes-de-Oca, *"Multilayer decentralized MPC of large-scale networked systems," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 495–515.
52. B. Morcego, V. Javalera, V. Puig, and R. Vito, *"Distributed MPC using reinforcement learning based negotiation: Application to large scale systems," in*

Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 517–533.
53. J. L. Nabais, R. R. Negenborn, R. B. Carmona-Benitez, L. F. Mendonca, and M. A. Botto, *"Hierarchical MPC for multiple commodity transportation networks," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 535–552.
54. G. Pannocchia, S. J. Wright, and J. B. Rawlings, *"On the use of suboptimal solvers for efficient cooperative distributed linear MPC,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 553–568.
55. A. Ferramosca, D. Limon, and A. H. González, *"Cooperative distributed MPC integrating a steady sate target optimizer," in Distributed Model Predictive Control Made Easy,* J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 569–584.
56. A. Ferramosca, *"Cooperative mpc with guaranteed exponential stability,"* in Distributed Model Predictive Control Made Easy, J. M. Maestre and R. R. Negenborn, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 585–600.
57. D. W. Clarke, C. Mohtadi, and P. S. Tuffs, *"Generalized predictive control—Part I. The basic algorithm," Automatica,* vol. 23, no. 2, pp. 137–148, 1987.
58. D. W. Clarke, *"Application of generalized predictive control to industrial processes,"* IEEE Control Syst. Mag., vol. 8, no. 2, pp. 49–55, 1988.
59. R. Kennel, A. Linder, and M. Linke, *"Generalized predictive control (GPC)-ready for use in drive applications?" in Proc. IEEE 32nd Annu.* Power Electronics Specialists Conf., 2001, vol. 4, pp. 1839–1844.
60. F. Fele, J. M. Maestre, S. M. Hashemy, D. M. de la Peña, and E. F. Camacho, *"Coalitional model predictive control of an irrigation canal,"* J. Process Control, vol. 24, no. 4, pp. 314–325, 2014.
61. J. M. Maestre, D. M. de la Peña, A. J. Losada, E. Algaba, and E. F. Camacho, *"A coalitional control scheme with applications to cooperative game theory,"* Optim. Contr. Applicat. Methods, doi: 10.1002/oca.2090.
62. M. Jilg and O. Stursberg, *"Optimized distributed control and topology design for hierarchically interconnected systems,"* in Proc. 2013 European Control Conf., Zurich, Switzerland, July 2013, pp. 4340–4346.
63. P. Trodden and A. G. Richards, *"Adaptive cooperation in robust distributed model predictive control," in Proc. 24th IEEE Int. Symp. Intelligent Control,* St. Petersburg, Russia, July 2009, pp. 896–901.
64. I. Alvarado, D. Limon, D. M. de la Peña, J. M. Maestre, F. Valencia, H. Scheu, R. R. Negenborn, M. A. Ridao, B. De Schutter, J. Espinosa, and W. Marquardt, *"A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark,"* J. Process Control, vol. 21, no. 5, pp. 800–815, 2011.
65. D. Sarewitz, R. A. Pielke, and R. Byerly, *Prediction: Science, Decision Making, and the Future of Nature.* Washington, DC: Island Press, 2000.
66. R. Harding, *Environmental Decision-Making: The Roles of Scientists, Engineers, and the Public.* Sydney, Australia: Federation Press, 1998.
67. INFSO D.4, INFSO G.2, and EPOSS, "Internet of Things in 2020—A roadmap for the future," European Commission, Information Society and Media, Brussels Tech. Rep., 2008.
68. R. Fuller, Introduction to Neuro-Fuzzy Systems. New York: Springer, 2000.

69. M. Broy, "Engineering cyber-physical systems: Challenges and foundations," in Complex Systems Design Management, M. Aiguier, Y. Caseau, D. Krob, and A. Rauzy, Eds. Berlin, Heidelberg: Springer, 2013, pp. 1–13.
70. M. W. Maier, *"Architecting principles for system of systems," Syst. Eng.,* vol. 1, no. 4, pp. 267–284, 1998.
71. G. Weiss, *Multiagent Systems.* Cambridge, MA: MIT Press, 2013.
72. M. Guizani, Wireless Communications Systems and Networks. New York: Springer, 2004.
73. De Vito, D., Picasso, B., Scattolini, R. (2010). On the design of reconfigurable two-layer hierarchical control systems with MPC. In Proceedings of the AmericanControl Conference Baltimore, Maryland, (pp. 4707–4712).
74. Davison, E. J., Chang, T. N. (1990). Decentralized stabilization and pole assignmentfor general proper systems. IEEE Transactions on Automatic Control, 35, 652–664.
75. Christofides, P. D., Liu, J., Mu noz de la Pe na, D. (2011). Networked and distributedpredictive control: Methods and nonlinear process network applications. Advancesin industrial control series. London, England: Springer-Verlag.
76. Ne˘si c, D., Teel, A. R. (2004). Input-to-state stability of networked control systems.Automatica, 40, 2121–2128.
77. Ne˘si c, D., Teel, A., Kokotovic, P. (1999). Sufficient conditions for stabilization ofsampled-data nonlinear systems via discrete time approximations. Systems andControl Letters, 38, 259–270.
78. Necoara, I., Nedelcu, V., Dumitrache, I. (2011). Parallel and distributed optimizationmethods for estimation and control in networks. Journal of Process Control, 21,756–766.
79. Negenborn R.R. (2007). Multi-agent model predictive control with applications topower networks (PhD thesis). Delft University of Technology.
80. Negenborn, R. R., De Schutter, B., Hellendoorn, J. (2008). Multi-agent model pre-dictive control for transportation networks: Serial versus parallel schemes.Engineering Applications of Artificial Intelligence, 21, 353–366.
81. Negenborn, R. R., Leirens, S., De Schutter, B., Hellendoorn, J. (2009). Supervisorynonlinear MPC for emergency voltage control using pattern search. Control Engi-neering Practice, 7, 841–848.
82. Negenborn, R. R., Van Overloop, P. J., Keviczky, T., De Schutter, B. (2009). Distributedmodel predictive control of irrigation canals. Networks and Heterogeneous Media,4, 359–380.
83. Neumann, P. (2007). Communication in industrial automation: What is going on?Control Engineering Practice, 15, 1332–1347.
84. Ocampo-Martinez, C., Bovo, S., Puig, V. (2011). Partitioning approach oriented tothe decentralised predictive control of large-scale systems. Journal of ProcessControl, 21, 775–786.
85. Ogunnaike, B. A., Ray, W. H. (1994). Process dynamics, modeling, and control. NewYork: Oxford University Press.
86. Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks. Proceedingsof the 46th IEEE Conference on Decision and Control, 5492–5498.
87. Omell, B.P., Chmielewski, D.J. IGCC power plant dispatch using infinite-horizoneconomic model predictive control. Industrial Engineering Chemistry Research,submitted.

88. Perk, S., Teymour, F., Cinar, A. (2010). Statistical monitoring of complex chem-ical processes using agent-based systems. Industrial Engineering ChemistryResearch, 49, 5080–5093.
89. B. W. Bequette. *Process Dynamics. Modeling, Analysis and Simulation.* Prentice Hall, 1998.
90. J. Bochnak, M. Coste, and M.-F. *Roy Real Algebraic Geometry.* Springer-Verlag, Berlin, 1998.
91. S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* Society for Industrial and Applied Mathematics (SIAM), 1994.
92. K. Furuta, M. Yamakita, and S. Kobayashi. Swing-up control of inverted pendulum using pseudo-state feedback. *Journal of Systems and Control Engineering,* 206:263–269, 1992.
93. W. Hahn. *Stability of Motion.* Springer-Verlag, New York, 1967.
94. J. Marsden and T. Ratiu. *Introduction to Mechanics and Symmetry.*SpringerVerlag, NY, second edition, 1999.
95. J. D. Murray. *Mathematical Biology.* Springer-Verlag, second edition, 1993.
96. K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming,* 39:117–129, 1987.
97. A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conf. on Decision and Control,* 2002.
98. P. A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization.* PhD thesis, Caltech, Pasadena, CA, 2000. Available at http://www.control.ethz.ch/ parrilo/pubs/index.html.
99. P. A. Parrilo and B. Sturmfels. Minimizing polynomial functions. In *Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science,* March, 1998.
100. S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *Proceedings of the 41st IEEE Conf. on Decision and Control,*2002. Available at http://www.cds.caltech.edu/sostools and http://www.aut.ee.ethz.ch/ parrilo/sostools.
101. B. Reznick. Some concrete aspects of Hilbert's 17th problem. In *Contemporary Mathematics,* volume 253, pages 251–272. American Mathematical Society, 2000.
102. M. A. Savageau and E. O. Voit. Recasting nonlinear differential equations as S-systems: a canonical nonlinear form. *Mathematical Biosciences,* 87(1):83–115, 1987.
103. J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software,* 11–12:625–653, 1999. Available at http://fewcal.kub.nl/sturm/software/sedumi.html.
104. L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review,* 38(1):49–95, 1996.
105. V. A. Yakubovic. S-procedure in nonlinear control theory. *Vestnik Leningrad University,* 4(1):73–93, 1977. English translation.
106. V. Zubov. *Methods of A.M. Lyapunov and Their Application.* P. Noordhoff Ltd, Groningen, The Netherlands, 1964.
107. J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms,* Springer-Verlag, New York, 1997.

108. H. Asada, J.-J.E. Slotine, *Robot Analysis and Control*, Wiley, New York, 1986.
109. C. Canudas de Wit, B. Siciliano, G. Bastin, (Eds.), *Theory of Robot Control*, Springer-Verlag, London, 1996.
110. J.J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed., Pearson Prentice Hall, Upper Saddle River, NJ, 2004.
111. A.J. Critchlow, *Introduction to Robotics*, Macmillan, New York, 1985.
112. J.F. Engelberger, *Robotics in Service*, MIT Press, Cambridge, MA, 1989.
113. K.S. Fu, R.C. Gonzalez, C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, New York, 1987.
114. W. Khalil, E. Dombre, *Modeling, Identification and Control of Robots*, Hermes Penton Ltd, London, 2002.
115. A.J. Koivo, *Fundamentals for Control of Robotic Manipulators*, Wiley, New York, 1989.
116. F.L. Lewis, C.T. Abdallah, D.M. Dawson, *Control of Robot Manipulators*, Macmillan, New York, 1993.
117. P.J. McKerrow, *Introduction to Robotics*, Addison-Wesley, Sydney, Australia, 1991.
118. R.M. Murray, Z. Li, S.S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Boca Raton, FL, 1994.
119. S.B. Niku, *Introduction to Robotics: Analysis, Systems, Applications*, Prentice-Hall, Upper Saddle River, NJ, 2001.
120. R.P. Paul, *Robot Manipulators: Mathematics, Programming, and Control* MIT Press, Cambridge, MA, 1981.
121. L. Sciavicco, B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed., Springer, London, UK, 2000.
122. M.W. Spong, S. Hutchinson, M. Vidyasagar, *Robot Modeling and Control*, Wiley, New York, 2006.
123. M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino, "Closed loop steering of unicycle-like vehicles via Lyapunov techniques," *IEEE Robotics and Automation Magazine,* vol. 2, no. 1, pp. 27–35, 1995.
124. G. Bastin, G. Campion, B. D'Andr´ea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots, " *IEEE Transactions on Robotics and Automation,* vol. 12, pp. 47–62, 1996.
125. J. Borenstein, H.R. Everett, L. Feng, *Navigating Mobile Robots: Systems and Techniques,* A K Peters, Wellesley, MA, 1996.
126. M. Vukobratović, *Introduction to Robotics*, Springer-Verlag, Berlin, Germany, 1989.
127. C. Canudas de Wit, H. Khennouf, C. Samson, O.J. Sørdalen, "Nonlinear control design for mobile robots," *Recent Trends in Mobile Robots,* Y.F. Zheng, (Ed.), pp. 121–156, World Scientific Publisher, Singapore, 1993.
128. A. De Luca, G. Oriolo, C. Samson, "Feedback control of a nonholonomic carlike robot," in *Robot Motion Planning and Control,* J.-P. Laumond, (Ed.), Springer-Verlag, Berlin, Germany, 1998.
129. M. Fliess, J. L´evine, P. Martin, P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International Journal of Control,* vol. 61, pp. 1327–1361, 1995.
130. J.L. Jones, A.M. Flynn, *Mobile Robots: Inspiration to Implementation,* AK Peters, Wellesley, MA, 1993.
131. R.M. Murray, Z. Li, S.S. Sastry, *A Mathematical Introduction to Robotic Manipulation,* CRC Press, Boca Raton, CA. 1994.

132. J.I. Neimark, F.A. Fufaev, *Dynamics of Nonholonomic Systems,* American Mathematical Society, Providence, RI, 1972.
133. Rachid, A. (1991). Positively invariant polyhedral sets for uncertain discrete time systems. *Control Theory and Advanced Technology,* 7(1), 191–200.
134. Rakovi´c, S., Kouramas, K. (2007). The minimal robust positively invariant set for linear discrete-time systems: Approximation methods and control applications. In *Proceedings of the CDC06,* San Diego, California.
135. Rakovi´c, S., Kerrigan, E., Kouramas, K., Mayne, D. (2005). Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control,* 50(3), 406–410.
136. R. Siegwart, I.R. Nourbakhsh, *Introduction to Autonomous Mobile Robots,* MIT Press, Cambridge, MA, 2004.
137. David Avis, Comments on alower bound for convexhull determination. *Inform. Process. Lett.,* 11, 3, (1980), 126, URL http://dx.doi.org/10.1016/0020-0190(80) 90125-8.
138. Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.,* pp. 80–86, 1983, URL http://dx.doi.org/10. 1145/800061.808735.
139. Timothy M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Comput. Geom.,* 16, 4, (1996), 361–368, URL http://dx.doi. org/10.1007/BF02712873.
140. Ronald L. Graham, An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.,* 1, 4, (1972), 132–133, URL http://dx.doi.org/ 10.1016/0020-0190(72)90045-2.
141. Ray A. Jarvis, On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.,* 2, 1, (1973), 18–21, URL http://dx.doi.org/ 10.1016/0020-0190(73)90020-3.
142. K.K. Oh, M.C. Park, and H.S. Ahn, *A survey of multi-agent formation control, Automatica,* vol. 53, issue 3, pp. 424-440, March 2015.
143. R. Olfati-Saber, J. Fax and R. Murray, *Consensus and cooperation in networked multi-agent systems,* Proceedings of the IEEE, vol. 95, no. 1, pp. 215-233, 2007.
144. W. Ren, R. Beard and E. Atkins, *Information consensus in multivehicle cooperative control,* IEEE Control Systems Magazine, vol. 27, no. 2, pp. 71-82, 2007.
145. R. Olfati-Saber, *Flocking for multi-agent dynamic systems: algorithms and theory,* IEEE Transaction Automatic Control, vol. 51, no. 3, pp. 401-420, 2006.
146. R. Olfati-Saber and R. Murray, *Consensus problems in networks of agents with switching topology and time-delays,* IEEE Transactions on Automatic Control, vol. 49, no. 9, pp. 1520-1533, 2004.
147. W. Ren, and R. Beard, *Consensus seeking in multiagent systems under dynamically changing interaction topologies,* IEEE Transactions on Automatic Control, vol. 50, no. 5, pp. 655-661, 2005.
148. A. Jadbabaie, J. Lin, and A. Morse, *Coordination of groups pf mobile autonomous agents using nearest neighbor rules,* IEEE Transaction Automatic Control, vol. 48, no. 6, pp. 988-1001, 2003.
149. K. You, Z. Li, and L. Xie, *Consensus condition for linear multi-agent systems over randomly switching topologies,* Automatica, vol. 49, no. 10, pp. 3125-3132, 2013.

150. Chen J, Sun D, Yang J, Chen H. *Leader-Follower Formation Control of Multiple Non-holonomic Mobile Robots Incorporating a Receding-horizon Scheme.,* The International Journal of Robotics Research. 2010;29(6):727-747.
151. M. Mesbahi and M. Egerstedt, "Graph theoretic methods in multiagent networks", *Princeton University Press*, 2010.
152. K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control", *Automatica*, Vol. 53, pp. 424-440, 2015.
153. J.P. Desai, J.P. Ostrowski and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots", *IEEE Trans. Robot. Autom.*, Vol. 17, No. 6, pp. 905-908, 2001.
154. H. G. Tanner, G. J. Pappas and V. Kumar, "Leader-to-formation stability", *IEEE Trans. Autom. Robot*, Vol. 20, No. 3, pp. 443-455, 2004.
155. M.A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures", *Auton. Robots*, Vol. 4, No. 4, pp. 387-403, 1997.
156. R.W. Beard, J. Lawton, F.Y. Hadaegh, "A coordination architecture for spacecraft formation control", *IEEE Trans. Contr. Sys. Tech.*, Vol. 9, No. 6, pp. 777-790, 2001.
157. R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory", *IEEE Trans. Aut. Contr.*, Vol. 51, No. 3, pp.401-420, 2006.
158. K. Hengster-Movrić, S. Bogdan and I. Draganjac, "Multi-agent formation control based on bell-shaped potential functions", *J. Intell. Robot. Syst.*, Vol. 58, No. 2, pp. 165-189, 2010.
159. W. Dong, "Robust formation control of multiple wheeled mobile robots", *J. Intell. Robot. Syst.*, Vol. 62, No. 3, pp. 547-565, 2011.
160. F. Xiao, L. Wang, J. Chen, Y. Gao, " Finite-time formation control for multi-agent systems", *Automatica*, Vol. 45, No. 11, pp. 2605-2611, 2009.
161. J. Ghommam, H. Mehrjerdi, M. Saad, F. Mnif, "Formation path following control of unicycle-type mobile robots", *Robot. Auton. Syst.*, Vol. 58, No. 5, pp. 727-736, 2010.
162. S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer and M. W. Spong, "Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments", *The International Journal of Robotics Research*, Vol. 27, No. 1, pp. 107-126, 2008.
163. S. Liu and D. Sun, "Minimizing energy consumption of wheeled mobile robots via optimal motion planning", *IEEE/ASME Transactions on Mechatronics*, Vol. 19, No. 2, pp. 401-411, 2014.
164. M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications", *IEEE Rob. & Aut. Mag.*, Vol. 19, No. 1, pp. 24-39, 2012.
165. A. Viguria, I. Maza and A. Ollero, "Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions", *Adv. Rob.*, Vol. 24, No. 1-2, pp. 1-23, 2010.
166. M. Lega, C. Ferrara, G. Persechino and P. Bishop, "Remote sensing in environmental police investigations: aerial platforms and an innovative application of thermography to detect several illegal activities", *Env. Monit. and Ass.*, Vol. 186, No. 12, pp. 8291-8301, 2014.
167. W. Re and R. W. Beard, "Distributed consensus in multi-vehicle cooperative control", *London: Springer London*, 2008.
168. V. V. Vantsevich and M. V. Blundell, "Advanced Autonomous Vehicle Design for Severe Environments", *IOS Press*, 2015.

169. P. D. Christofides, R. Scattolini, D. M. de la Peña and J. Liue, "Distributed model predictive control: A tutorial review and future research directions", *Computers and Chemical Engineering*, Vol. 51, pp. 21–41, 2013.
170. R. R. Negenborn and J. M. Maestre , "Distributed model predictive control: An overview and roadmap of future research opportunities", *IEEE Control Systems Magazine*, Vol. 34, No. 4, pp. 87-97, 2014.
171. W. B. Dunbar and D. S. Caveney, "Distributed receding horizon control of vehicle platoons: Stability and string stability", *IEEE Trans. Aut. Contr.*, Vol. 57, No. 3, pp. 620-633, 2012.
172. P. Wang and B. Ding, "Distributed RHC for tracking and formation of nonholonomic multi-vehicle systems", *IEEE Trans. Aut. Contr.*, Vol. 59, No. 6, pp. 1439-1453, 2014.
173. H. Li, Y. Shi and W. Yan, "Distributed receding horizon control of constrained nonlinear vehicle formations with guaranteed γ-gain stability", *Automatica*, Vol. 68, pp. 148-154, 2016.
174. R. Van Parys and G. Pipeleers, "Distributed MPC for multi-vehicle systems moving in formation", *Rob. and Aut. Sys.*, Vol. 97, pp. 144-152, 2017.
175. G. Franzè, W. Lucia and F. Tedesco, "A distributed model predictive control scheme for leader-follower multi-agent systems", *Int. J. of Contr.*, Vol. 91, No. 2, pp. 369-382, 2018.
176. J. Maestre, M. Ridao, A. Kozma, C. Savorgnan, M. Diehl, M. Doan, A. Sadowska, T. Keviczky, B. De Schutter, H. Scheu, *et al.*, "A comparison of distributed MPC schemes on a hydro-power plant benchmark", *Optim. Contr. Appl. Methods*, Vol. 36, No. 3, pp. 306-332, 2015.
177. H. Fukushima, K. Kon and F. Matsuno, "Model predictive formation control using branch-and-bound compatible with collision avoidance problems", *IEEE Trans. on Rob.*, Vol. 29, No. 5, pp. 1308-1317, 2013.
178. T. Nguyen, H. M. La, T. D. Le, and M. Jafari, "Formation control and obstacle avoidance of multiple rectangular agents with limited communication ranges", *IEEE Trans. on Contr. of Net. Sys.*, Vol. 4, No. 4, pp. 680-691, 2016.
179. M. Farina, A. Perizzato and R. Scattolini, "Application of distributed predictive control to motion and coordination problems for unicycle autonomous robots", *Robot. Auton. Syst.*, Vol. 72, pp. 248-260, 2015.
180. G. Franzè and W. Lucia, "Multi-vehicle formation control in uncertain environments", *56th IEEE CDC*, Melbourne, Australia, 2017.
181. G. Franzè, L. D'Alfonso and G. Fedele, "Distributed model predictive control for constrained multi-agent systems: a swarm aggregation approach", *ACC 2018*, Milwaukee, WI, USA, 2018.
182. V. Gazi and K. M. Passino, "Swarm stability and optimizations", *Springer Science & Business Media*, 2011.
183. V. Gazi and K. M. Passino, "A class of attractions/repulsion functions for stable swarm aggregations", *Int. J. of Contr.*, Vol. 77, No. 18 , pp. 1567-1579, 2004.
184. R. Fierro and F. L. Lewis, "Control of nonholonomic mobile robot: Backstepping kinematics into dynamics", *IEEE CDC*, New Orleans, LA, 1995, pp. 3805-3810, 1995.
185. J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft", *IEEE Trans. on Contr. Sys. Tech.*, Vol. 20, pp. 604–620, 2011.

186. G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation", *IEEE Trans. on Contr. Sys. Tech.*, Vol. 10, No. 6, pp. 835-852, 2002.
187. Bruce Torby, "Energy Methods. Advanced Dynamics for Engineers", *HRW Series in Mechanical Engineering. USA: CBS College Publishing*, 1984.
188. S. Boyd, L. El Ghaoui, E. Feron and V. Balakrishnan, "Linear Matrix Inequalities in System and Control Theory", *SIAM Studies in Applied Mathematics*, 15, SIAM, London.
189. G. Fedele and L. D'Alfonso, " A coordinates mixing matrix-based model for swarm formation", *Int. J. of Contr.*, *DOI: 10.1080/00207179.2019.1613561*, 2019.
190. A. Papachristodoulou and S. Prajna, "Analysis of Non-polynomial Systems Using the Sum of Squares Decomposition", *Positive Polynomials in Control*, LNCIS 312/2005, Springer, pp. 23–43, 2005.
191. G. Franzè, A. Casavola, D. Famularo and E. Garone, "An off-line MPC strategy for nonlinear systems based on SOS programming", *Nonlinear Model Predictive Control*, pp. 491-499, 2009.
192. F. Blanchini and S. Miani,"Set-Theoretic Methods in Control", *Birkäuser*, Boston, 2008.
193. A. Casavola, D. Famularo and G. Franzè, "Robust fault detection of uncertain linear systems via quasi-LMIs", *Automatica*, Vol. 44, No. 1, pp. 289-295, 2008.
194. W. Lucia, G. Franzè and M. Sznaier, "A Hybrid Command Governor Scheme for Rotary Wings Unmanned Aerial Vehicles", *IEEE Trans. on Contr. Sys. Tech.*, DOI: 10.1109/*TCST*.2018.2880936, 2018.
195. M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0", *ECC*, pp. 502–510, 2013.
196. Y. Kuwata, T. Schouwenaars, A. Richards and J. How, "Robust Constrained Receding Horizon Control for Trajectory Planning", *AIAA Guid., Nav.and Contr. Conf. and Exh.*, San Francisco, California, 2005.
197. Elisa-3, http://www.gctronic.com/doc/index.php/Elisa-3