Collaborative Project
Small-medium-scale focused research project (STREP)

Grant Agreement n. 224117 FP7-ICT-2007-2

WIDE

Decentralized Wireless Control of Large-Scale Systems

Starting date: 01 September 2008 Duration: 3 years

Deliverable number	**D3.3**
Title	**Algorithms for higher-level real-time optimization and their cooperation with decentralized/distributed MPC**
Work package	WP3 - Multilayer distributed control and model management (RTD)
Due date	M30
Actual submission date	28/02/2011
Lead contractor for this deliverable	HPL
Author(s)	P. Trnka `pavel.trnka@honeywell.com`
With the help of	A. Bemporad `bemporad@ing.unitn.it`, D. Barcelli `barcelli@dii.unisi.it` C. Rocchi `rocchi@ing.unitn.it` G. Ripaccioli `ripaccioli@dii.unisi.it` C.A. Pascucci `pascucci@dii.unisi.it` M. Tubino `marco.tubino@ing.unitn.it` M. Toffolon `marco.toffolon@ing.unitn.it` G. Zolezzi `guido.zolezzi@ing.unitn.it`
Revision	v1.1

Dissemination Level

→ **PU**	Public
PP	Restricted to other programme participants (including the Commission Services)
RE	Restricted to a group specified by the consortium (including the Commission Services)
CO	Confidential, only for members of the consortium (including the Commission Services)

Executive summary

This deliverable presents novel results in the field of real-time optimization. The algorithms are demonstrated on the examples of autonomous navigation of a formation of unmanned aerial vehicles under obstacle and collision avoidance constraints and optimal control of large-scale water distribution network.

Contents

1 Introduction

This deliverable targets algorithms for real-time optimization and their cooperation with decentralized/distributed MPC. The first section present results for hierarchical multi-rate control design for constrained linear systems. The second section applies hierarchical and decentralized hybrid model predictive control to autonomous navigation of a formation of unmanned aerial vehicles under obstacles and collision avoidance constraints. The third section present results for distributed real time optimization with parametric coordination together with a demonstration on large-scale optimal control of water distribution network.

Hierarchical multi-rate control design approach to linear systems subject to linear constraints on input and output variables is a control strategy where, at the lower level, a linear controller stabilizes the open-loop process without considering the constraints. A higher-level controller commands reference signals at a lower sampling frequency so as to enforce linear constraints on the variables of the process. By optimally constraining the magnitude and the rate of variation of the reference signals applied to the lower control layer, we provide quantitative criteria for selecting the ratio between the sampling rates of the upper and lower layers to preserve closed-loop stability without violating the prescribed constraints. The approach is particularly useful when the lower layer is a decentralized stabilizing controller, and the higher layer is asked to enforce global constraints on system variables without relying on a full dynamical model of the entire process. The approach was tested in detail on the hierarchical control of a linear, dynamically coupled, multi-mass-spring system.

For hybrid dynamical systems, precise methodologies for hierarchical control design are more difficult. We have only tested the hierarchical MPC design on a nontrivial application example involving hybrid dynamics, namely autonomous navigation of a formation of unmanned aerial vehicles (UAVs) under obstacle and collision avoidance constraints, including lower-level vehicle stabilization. Each vehicle, of quadcopter type, is stabilized by a local linear MPC controller around commanded desired set-points. These are generated at a slower sampling rate by a hybrid MPC controller per vehicle at the upper control layer, based on a hybrid dynamical model of the UAV and of its surrounding environment (i.e., the other UAVs and obstacles). The resulting decentralized scheme controls the formation based on a leader-follower approach. The performance of the hierarchical control scheme is assessed through simulations and comparisons with other path planning strategies, showing the ability of linear MPC to handle the strong couplings among the dynamical variables of each quadcopter under motor voltage and angle/position constraints, and the flexibility of the proposed hierarchical decentralized hybrid MPC scheme in planning the desired paths on-line.

2 Hierarchical Multi-Rate Control Design for Constrained Linear Systems

2.1 Introduction

The increasing demand for automation of large-scale systems requires engineers to develop more complex and scalable control designs, based on distributed and multi-layer architectures. Centralized control schemes are in fact often inadequate for large-scale systems composed by several interacting subsystems. This is because all measurements must be collected in one location where a possibly complex global control decision must be take, causing problems of scalability, robustness, reliability, and maintenance of large-scale models. Decentralized control has been advocated during at least the last four decades to overcome the complexity issue [27], but when global performance objectives need to be optimized under local and possibly global constraints, a (usually centralized) upper layer of coordination is required, typically running at a smaller sampling frequency.

Model predictive control (MPC) has been extensively used in the process industries for control and coordination of large-scale systems subject to constraints [28]. Traditionally, MPC is used for generating reference signals to single-loop controllers in order to optimize a global performance and enforce constraints on multiple inputs and outputs. In order to achieve this task, MPC requires a dynamical model of the entire process, used to make predictions over which to optimize the control signals. As a consequence, MPC suffers from the aforementioned scalability and model maintenance issues, exacerbated by the complexity issue of solving a large-scale optimization problem on-line.

Decentralized and hierarchical MPC schemes have been investigated recently to address the complexity issue of centralized MPC. We refer the reader to the excellent recent survey [29] and to the recent literature on decentralized MPC [30–35]. Reference governors (RG) were also proposed to mitigate the complexity of MPC by separating the stabilization problem from the constraint fulfillment problem [36–41]. In the RG approach, a (global) model of the underlying closed-loop system is exploited in a predictive manner to provide a reference signal to the lower-level controller which is as close as possible to the desired one, compatibly with the given constraints. Although providing good computational benefits, reference governors have still the drawback of needing a detailed global dynamical model of the entire underlying closed-loop system for on-line optimization.

In this work we propose a hierarchical multi-rate control approach that exploits the idea of manipulating reference signals to enforce constraints. We assume that the open-loop process is stabilized by a linear (possibly decentralized) controller with sampling time T_L without taking care of the constraints, whose reference signals are generated by a higher-level controller running at a larger sampling time $T_H = NT_L$. As in [40], the higher level controller bounds the commanded reference signals to prevent violations of the contraints. In this work, however, constraints are set also on the *variations* of the reference signals. In addition we adopt a multi-rate setting, providing quantitative relations between the maximum allowed reference variations and to ratio $N = T_H/T_L$ between the sampling times.

Multirate MPC schemes have been addressed in a variety of papers, see e.g. the early work [42], and the application papers [43, 44], where hybrid MPC control is used at the higher level to enforce complex linear and logical constraints. Two main issues arise in hierachical MPC design: the choice of a simple ("as much abstracted as possible, but not too much") prediction model of the underlying subsystem, and the choice of the sampling time T_H. Rule of thumbs suggest that the latter must be "large enough" to assume that the adopted prediction model is "enough consistent" with the true underlying closed-loop system, but "not too small" to ensure enough reactiveness of the hierarchical scheme to changes of desired references. In this work we quantify exactly what "large enough" should be, and free the designer from concerns about the choose of the prediction model of the underlying closed-loop system. In fact, safe operations are guaranteed by the resulting magnitude

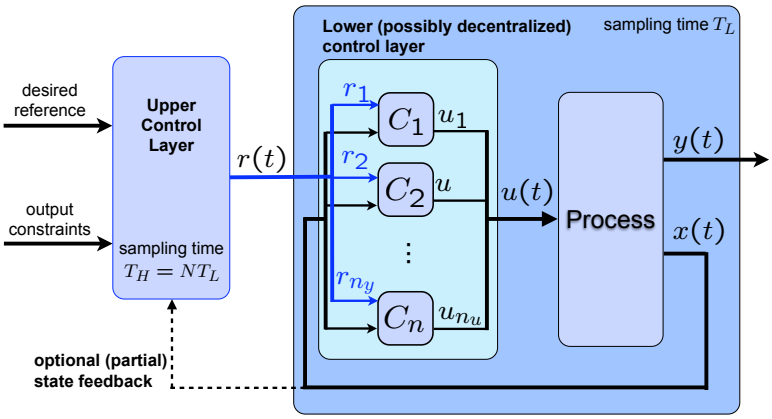


Figure 1: Hierarchical control scheme

and rate constraints on reference signals, no matter how the performance index (if any) is optimized on top by the higher-level controller.

The proposed hierarchical control architecture is described in Section 2.2. In Section 2.3 the constraints on reference signals and their dependence on the ratio between sampling times is characterized and optimized, and used in Section 2.4 to the define the general hierarchical control design. A particular design for the upper control layer based on on-line optimization is described in Section 2.5. Simulation results are reported in Section 2.6.

2.2 Problem setup

Consider the hierarchical control architecture depicted in Figure 1. The open-loop process is stabilized by a lower-level control layer running at a sampling frequency $\frac{1}{T_L}$ (possibly decentralized, as depicted in Figure 1). At the higher level a supervisor running at a lower sampling frequency $\frac{1}{T_H}$ decides the reference signals to send to the lower layer, possibly optimizing a performance criterion (such as an economic criterion), so as to make sure that a certain number of linear constraints on input and output variables are satisfied. Hierarchical control arrangements are frequent in industrial automation, because one can separate the concerns related to stabilization and disturbance rejection (taken care by the lower level controller at a high sampling frequency) and to steady-state optimization and constraint handling (taken care by the higher level, usually at a slower pace).

Consider the linear time-invariant (LTI) discrete-time model of the lower-level closed-loop system

$$\left\{\begin{array}{rcl} x(t+1) & = & \bar{A}x(t)+\bar{B}u(t) \\ y(t) & = & \bar{C}x(t)+\bar{D}u(t) \\ u(t) & = & Kx(t)+Er(t) \end{array}\right. \tag{1}$$

where $x(t) \in \mathbb{R}^n$, $y(t) \in \mathbb{R}^{n_y}$, $u(t) \in \mathbb{R}^{n_u}$, and $r(t) \in \mathbb{R}^{n_y}$ is the reference signal. We assume that K is an asymptotically stabilizing gain, which could either be a centralized or a decentralized one (see e.g. [45] for LMI-based synthesis of decentralized linear controllers). We also assume that a gain $E \in \mathbb{R}^{n_u \times n_y}$ exists such that the DC-gain from r to y is the identity,

$$E = ((\bar{C}+\bar{D}K)(I-\bar{A}-\bar{B}K)^{-1}\bar{B}+\bar{D})^{-1} \tag{2}$$

The closed-loop system (1) can be rewritten as

$$\left\{\begin{array}{rcl} x(t+1) & = & Ax(t)+Br(t) \\ y(t) & = & Cx(t)+Dr(t) \end{array}\right. \tag{3}$$

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

Define the following ratio

$$N = \frac{T_H}{T_L} \tag{4}$$

between the two sampling times of the control layers, where we assume $N \in \mathbb{N}$.

The goal of the higher-level controller is to command the piecewise constant vector of references $r(t)$

$$r(t) = r_k,\ t = kN, \ldots, (k+1)N-1, \quad k = 0, 1 \ldots \tag{5}$$

to the lower-level controller $u(t) = Kx(t) + Er(t)$ in a way that the output vector $y(t)$ is kept within the admissible output polytope

$$\mathscr{Y} = \{y \in \mathbb{R}^{n_y} :\ H_y y \leq K_y\} \tag{6}$$

where $H_y \in \mathbb{R}^{q \times n_y}$, $K_y \in \mathbb{R}^q$. Note that input constraints may be also embedded in (6) by augmenting the output vector so that matrix $[C\ D]$ includes the rows of $[K\ E]$.

The main goal of this contribution is to determine simultaneously the ratio N and restrictions on r_k and on the set point changes $\Delta r_k = r_k - r_{k-1}$ so that $y(t) \in \mathscr{Y}$. To this end, let the reference vector $r(t)$ be constrained within the tightened set

$$\mathscr{R} = \{r \in \mathbb{R}^{n_y} :\ H_y r \leq K_y - \Delta K_y\}, \tag{7}$$

where $\Delta K_y \in \mathbb{R}^q$, $\Delta K_y > 0$ component-wise, and assume set points $r(t)$ are changed in a way that the tracking error $e(t) \triangleq y(t) - r(t)$ is always within the set

$$\mathscr{E} = \{e \in \mathbb{R}^{n_y} :\ H_y e \leq \Delta K_y\} \tag{8}$$

Vector ΔK_y is a tuning knob of the proposed approach: the smaller the components of ΔK_y, the larger is the set $\mathscr{R}$ of admissible set points r_k, but the smaller will be the admissible reference increments Δr_k to maintain tracking errors within $\mathscr{E}$.

Let $x_r \in \mathbb{R}^n$ be the steady-state state corresponding to a reference signal $r \in \mathbb{R}^{n_y}$, $x_r = Ax_r + Br$, $x_r = G_x r$, $G_x \triangleq (I - A)^{-1}B$, and define the shift of coordinates $\Delta x = x - x_r$. Then, (3) can be rewritten as

$$\left\{ \begin{array}{rcl} \Delta x(t+1) & = & A\Delta x(t) \\ e(t) & = & C\Delta x(t) \end{array} \right. \tag{9}$$

Let $\Omega(0) \subseteq \mathbb{R}^n$ be the maximum admissible output set (MOAS) [46] for the closed-loop system (9) under the constraint $e(t) \in \mathscr{E}$

$$\Omega(0) \ = \ \{\Delta x \in \mathbb{R}^n :\ H_y C A^k \Delta x \leq \Delta K_y,\ \forall k \geq 0\} \ \triangleq \ \{x \in \mathbb{R}^n : H_0 \Delta x \leq K_0\} \tag{10}$$

where[1] $H_0 \in \mathbb{R}^{n_0 \times n}$ and $K_0 \in \mathbb{R}^{n_0}$, and define the reference-dependent invariant set

$$\Omega(r) = \{x \in \mathbb{R}^n :\ H_0(x - G_x r) \leq K_0\} \tag{11}$$

Lemma 1. *Let $x(0) \in \Omega(r)$ and $r(t) \equiv r \in \mathscr{R}$, $\forall t \geq 0$. Then $y(t) \in \mathscr{Y}$, $\forall t \geq 0$.*

Proof. $x(0) \in \Omega(r)$ implies that $y(t) - r(t) \in \mathscr{E}$, $\forall t \geq 0$. Since $r(t) \in \mathscr{R}$, it follows that $H_y y(t) = H_y e(t) + H_y r(t) \leq \Delta K_y + K_y - \Delta K_y \leq K_y$, $\forall t \geq 0$. □

The main idea is the following. Assume that the reference $r_k \in \mathscr{R}$ is issued at time $t = kN$, and that N is large enough so that $x(t + N - 1) \in \Omega(r_k)$. At time $t = (k+1)N$ consider the new reference is $r_{k+1} \in \mathscr{R}$. If $\Delta r_{k+1} = r_{k+1} - r_k$ is "small enough", then $x(t + N) \in \Omega(r_{k+1})$. The goal of the next section is to quantify the relationship between the maximum reference variation Δr_k, the ratio N between the sampling intervals T_H, T_L, and ΔK_y such that every $T_H = NT_L$ steps the state vector $x(t)$ of the plant is guaranteed to lie in an invariant set $\Omega(r_k)$.

[1] As $\Delta K_y > 0$ and A is asymptotically stable, $\Omega(0)$ is generated by a finite number of inequalities, as proved in [46]. We assume that (H_0, K_0) are a minimal hyperplane representation of $\Omega(0)$.

2.3 Computation of maximum reference rates

Assume the ratio N between the sampling times of the upper and lower layers of control is given. Consider the problem of determining the initial state $x(0) \in \Omega(r_1)$ and the minimum reference variation $\Delta r(N) = r_2 - r_1$ between two reference values $r_1, r_2 \in \mathscr{R}$ such that the state $x(N)$ is outside the invariant set $\Omega(r_2)$:

$$\begin{aligned}
\Delta r(N) = \quad & \inf_{r_1, r_2, x(0)} \quad ||r_2 - r_1||_\infty && \text{(12a)} \\
& \text{s.t.} \quad r_1, r_2 \in \mathscr{R} && \text{(12b)} \\
& \qquad x(0) \in \Omega(r_1) && \text{(12c)} \\
& \qquad x(t+1) = Ax(t) + Br_2 \quad t = 0, 1, \ldots, N-1 && \text{(12d)} \\
& \qquad x(N) \notin \Omega(r_2) && \text{(12e)}
\end{aligned}$$

Because of constraint (12e), the optimization problem (12) is nonconvex. However, it can be conveniently recast as a mixed-integer linear programming (MILP) problem by introducing an auxiliary binary vector $\delta \in \{0,1\}^{n_0}$, satisfying the following constraints

$$[\delta^i = 1] \leftrightarrow [H_0^i(x(N) - G_x r_2) \leq K_0^i] \tag{13a}$$

$$\sum_{i=0}^{n_0} \delta^i \leq n_0 - 1 \tag{13b}$$

where the superscript i denotes the ith component or row. The logical constraint (13a) can be converted to mixed-integer linear inequalities using the standard "big-M" approach

$$\begin{aligned}
H_0^i(x(N) - G_x r_2) - K_0^i \quad &\leq \quad M_+^i(1 - \delta^i) && \text{(14a)} \\
H_0^i(x(N) - G_x r_2) - K_0^i \quad &\geq \quad (M_-^i - \sigma)\delta^i + \sigma \quad i = 1, \ldots, n_0 && \text{(14b)} \\
& && \text{(14c)}
\end{aligned}$$

where $\sigma > 0$ is a small number (e.g.: the machine precision) and $M_-, M_+ \in \mathbb{R}^{n_0}$ are vectors of lower and upper bounds obtained by solving the following linear programs

$$M_-^i = \min_{\begin{bmatrix} x(0) \\ r_1 \\ r_2 \end{bmatrix}} \begin{bmatrix} H_0 A^N & 0 & H_0 R_G \end{bmatrix}^i \begin{bmatrix} x \\ r_1 \\ r_2 \end{bmatrix} - K_0^i \tag{15a}$$

$$\text{s.t.} \begin{bmatrix} H_0 & -H_0 G_x & 0 \\ 0 & H_y & 0 \\ 0 & 0 & H_y \end{bmatrix} \begin{bmatrix} x \\ r_1 \\ r_2 \end{bmatrix} \leq \begin{bmatrix} K_0 \\ K_y - \Delta K_y \\ K_y - \Delta K_y \end{bmatrix} \tag{15b}$$

where $R_G \triangleq R_N - G_x$, $R_N \triangleq \left(\sum_{i=0}^{N-1} A^i B\right)$, and vector M_+ is determined by changing min to max in (15). By introducing an additional variable $\varepsilon \geq \|r_2 - r_1\|_\infty$, we address problem (12) by solving the following

MILP

$$\Delta r(N) = \min_{[x'\ r_1'\ r_2'\ \delta'\ \varepsilon]'} \quad \varepsilon \tag{16a}$$

$$\text{s.t.} \quad \varepsilon \geq \pm(r_2^j - r_1^j) \ , \quad j = 1,\dots,n_y \tag{16b}$$

$$H_y r_1 \leq K_y - \Delta K_y \tag{16c}$$

$$H_y r_2 \leq K_y - \Delta K_y \tag{16d}$$

$$H_0(x - G_x r_1) \leq K_0 \tag{16e}$$

$$H_0^i(A^N x + R_G r_2) - K_0^i \leq M_+^i(1-\delta^i) \tag{16f}$$

$$-H_0^i(A^N x + R_G r_2) + K_0^i \leq -(M_-^i - \sigma)\delta^i - \sigma \tag{16g}$$

$$\sum_{i=0}^{n_o} \delta^i \leq n_o - 1 \tag{16h}$$

$$\delta^i = \{0,1\},\ i = 1,\dots,n_0 \tag{16i}$$

The quantity $\Delta r(N)$ in (16) is the smallest change of reference vector (expressed in infinity norm) that can be applied to the closed-loop system (3) such that, starting from an invariant set $\Omega(r_k)$, the state vector lands outside a new invariant set $\Omega(r_{k+1})$ after N steps. Or, in other words, for all reference changes $\|r_k - r_{k-1}\|_\infty \leq \Delta r(N) - \sigma$, $\forall \sigma > 0$, the closed-loop system (3) is such that, starting from an invariant set $\Omega(r_k)$, the state vector always arrives into a new invariant set $\Omega(r_{k+1})$ after N steps. Note that, because of the constraint $r_1, r_2 \in \mathscr{R}$, problem (16) may become infeasible for large N, that is any feasible perturbation of the set-point keeps the state within the invariant set $\Omega(r_2)$ after N steps. The following lemma shows a monotonicity property of $\Delta r(N)$ with respect to N, for those values $N \in \mathbb{N}$ for which $\Delta r(N)$ is defined.

Lemma 2. *Let $\Delta r(N)$ be defined by the optimization problem* (16)*. Then for any $N_1, N_2 \in N$, $N_1 < N_2$, such that $\Delta r(N_1)$, $\Delta r(N_2)$ are defined it holds that*

$$\Delta r(N_1) \leq \Delta r(N_2) \tag{17}$$

Proof. We first prove by contradiction that $\Delta r(N) \leq \Delta r(N+1)$, $\forall N \in \mathbb{N}$ such that $\Delta r(N+1)$ is defined. Assume that $N \in \mathbb{N}$ exists such that $\Delta r(N+1) < \Delta r(N)$. This implies that there exists a state x and two references $r_1, r_2 \in \mathscr{R}$ such that $\Delta r(N+1) \leq \|r_1 - r_2\|_\infty < \Delta r(N)$, $x \in \Omega(r_1)$, $A^{N+1}x + \sum_{i=1}^{N} A^i B r_2 \notin \Omega(r_2)$. Then, also $A^N x + \sum_{i=1}^{N-1} A^i B r_2 \notin \Omega(r_2)$, otherwise, by invariance of $\Omega(r_2)$, also $A^{N+1}x + \sum_{i=1}^{N} A^i B r_2$ would belong to $\Omega(r_2)$. Hence, the optimality of $\Delta r(N)$ is violated, a contradiction. The monotonicity condition (17) for generic N_1, N_2 easily follows. □

2.4 Hierarchical controller

Assume that N has been fixed and that the upper control layer commands set-points r_k under the constraints

$$\|r_k - r_{k-1}\|_\infty \leq \Delta r(N) - \sigma,\ \forall k = 0,1,\dots \tag{18a}$$

$$r_k \in \mathscr{R},\ \forall k = -1,0,1,\dots \tag{18b}$$

feeding the lower control layer as in (5).

Theorem 1. *Let K be a lower-level feedback gain such that $\bar{A} + \bar{B}K$ is a strictly Schur matrix, and assume that matrix E in* (2) *exists. Assume a vector $r_{-1} \in \mathscr{R}$ exists such that the initial state $x(0) \in \Omega(r_{-1})$. Let the upper-level controller change the set-points r_k according to the constraints* (18)*, in which $\Delta r(N)$ is the solution of* (16) *and $\sigma > 0$ is arbitrary small. Then the linear system $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ satisfies the constraints $y(t) \in \mathscr{Y}$ for all $t \geq 0$. If in addition $\lim_{t\to\infty} r(t) = r \in \mathscr{R}$ then $\lim_{t\to\infty} y(t) = r$.*

Proof. Because of (18), $x(kN) \in \Omega(r_k)$, $\forall k \geq 0$. By Lemma 1, it follows that $y(t) \in \mathscr{Y}$, $\forall t = kN, \ldots, k(N+1)-1$, $\forall k = 0, 1, \ldots$, that is $y(t) \in \mathscr{Y}$, $\forall t \geq 0$. To prove convergence of $y(t)$ to r when $\lim_{t\to\infty} r(t) = r$, similarly to (34) define $\Delta x(t) = x(t) - G_x r$ and rewrite (3) as

$$\begin{cases} \Delta x(t+1) &= A\Delta x(t) + B(r(t) - r) \\ e(t) &= C\Delta x(t) + D(r(t) - r) \end{cases} \tag{19}$$

As (19) is an asymptotically stable linear system it is also input-to-state stable [47], and hence it immediately follows that $\lim_{t\to\infty} \Delta x(t) = 0$, which in turn implies that $\lim_{t\to\infty} y(t) - r = 0$. □

Theorem 1 shows that *any* upper-level reference generation strategy satisfying constraints (18) guarantees the fulfillment of output constraints and asymptotic convergence to constant set-points. The MILP (16) provides the supremum $\Delta r(N)$ of the reference variations $\|r_k - r_{k-1}\|_\infty$ that the higher-level controller can apply for a given ratio $N = T_H/T_L$ between sampling times. It is worth to investigate the relation between $\Delta r(N)$ and N further. In fact, the design of the higher control layer could be addressed from a different point of view: given a desired Δr, determine the minimum N such that $\Delta r < \Delta_r(N)$. In practical applications N is restricted to a range $[N_{min},\ N_{max}]$ of values: the upper layer is executed at a slower pace than the lower layer ($N_{\min}$ not too small), but at the same time the upper layer should be reactive enough to adjust set-points ($N_{\max}$ not too large). Hence, it is worth to solve the MILP (16) only within the restricted range $N \in [N_{\min}, N_{\max}]$ to characterize $\Delta r(N)$ that, by Lemma 2, we know increases with N. In particular, it is of interest the ratio

$$R(N) = \frac{\Delta r(N)}{N} \tag{20}$$

which characterizes the maximum speed of change of the reference signal. In fact, the larger N the larger is the supremum of the variations Δr that the supervisor can issue, but the less frequently such variation happens, that is every NT_L sampling times. Another issue related to tuning of the upper control layer is the choice ΔK_y: from one hand a larger ΔK_y tightens the range of admissible references $\mathscr{R}$, but on the other hand it enlarges the size of the invariant set $\Omega(r)$, and therefore augments the achievable $\Delta r(N)$. There is therefore a tradeoff the designer must choose between constraints on reference signals ($\mathscr{R}$) and constraints on reference speed ($R(N)$).

Because of the need of enforcing constraints (18) in the upper control layer, in the next section we propose a model predictive control (MPC) design strategy for such a layer, although any other constraint-handling strategy could be employed, such as static optimization or a rule-based selection.

2.5 MPC design of upper control layer

We introduce an upper-layer MPC strategy, denoted as HiMPC, for generating the reference signal r under constraints (18).

2.5.1 Prediction model

We consider an under-sampled and possibly reduced-order model of the lower-level closed-loop model (3), evolving with sampling time $T_H = NT_L$

$$\begin{cases} x^H_{k+1} &= A_H x^H_k + B_H r_k \\ y_k &= C_H x^H_k + D_H r_k \end{cases} \tag{21}$$

where $y_k = y(kN)$, $x^H_k = Zx(kN)$, and Z is a matrix mapping the original state $x(kN)$ into the new state x^H_k (in case the order of the system is not reduced $Z = I$). Model (21) can be easily obtained

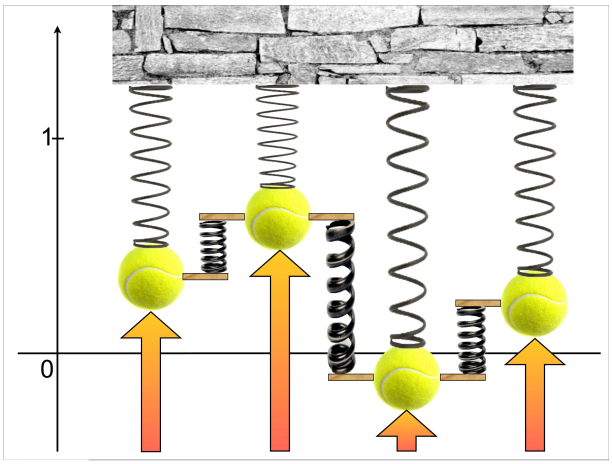


Figure 2: Mass-spring system considered in the simulation example

by resampling system (3) using standard discretization methods. As a consequence, fast-enough modal responses become negligible, which implies that the HiMPC algorithm can exploit only an incomplete information about the underlying closed-loop dynamics. This is a very convenient feature when HiMPC is applied to supervise a decentralized control layer, where maintaining a global detailed dynamical model of the entire lower-level closed-loop process may be a hard task. In the extreme case in which all dynamics are neglected, model (21) becomes the following static model + one-step delay

$$\begin{cases} x_{k+1}^H &= r_k \\ y_k &= x_k^H \end{cases} \tag{22}$$

where $x_k^H = r_{k-1} \in \mathbb{R}^{n_y}$ is a state buffering the reference signal for one step T_H. Model (21) is particularly appropriate for large values of N. Note that in case model (22) is used, no feedback from the states x of the process is required by the upper control layer.

2.5.2 Cost function and constraints

The upper-layer MPC controller must embed constraints (18) on the generated references, to ensure stability and constraint satisfaction. It may also embed additional constraints on the reference signals, such as mixed logical/linear constraints (see e.g. [43]).

The MPC controller can optimize virtually any cost function of r_k, Δr_k, and x_k^H, that may be dictated for instance by economic objectives.

Note also that if H_y is block-diagonal (for example, $\mathscr{Y}$ is a box), then $\mathscr{R}$ is also block diagonal, and if performance objectives and possibly other additional constraints are also block diagonal, so that HiMPC based on model (22) can be implemented in a decentralized way.

Finally, note that when HiMPC is based on model (22), a simple static optimization with respect to r_k can be setup, that possibly leads to a small-scale linear or quadratic programming problem. In this case, multiparametric programming algorithms can be exploited to convert HiMPC into a piecewise affine control law [48].

2.6 Simulation example

2.6.1 Problem description

We test the performance of the proposed HiMPC approach on the multi-mass-spring system depicted in Figure 2. Although the example is academic and relatively low-dimensional, the concepts illustrated in the example are immediately scalable to larger systems.

The process is composed by four masses moving vertically, each one connected by a spring to a fixed ceiling, subject to damping due to viscous friction with the environment, and connected to its neighbor mass by another spring. The values of the parameters of the system are reported in Table 1. An input force u [Nm] can be applied to each mass by the lower-level controller. The output of the system is the vector y collecting the vertical positions $y^1, \dots, y^4$ of the masses.

Table 1: Plant parameters

Physical characteristic	symbol	value
mass	M	5 [kg]
viscous friction	β	0.1 [kg/s]
vertical elastic coefficient	K_v	1 [N/m]
lateral elastic coefficient	K_l	0.1 [N/m]

The dynamics of the discrete-time model of the system obtained by exact discretization with sampling time $T_L = .25$ s are described the following matrices

$$\bar{A} = \begin{bmatrix} 1 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.055 & 0.995 & 0.005 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.25 & 0 & 0 & 0 & 0 \\ 0.005 & 0 & -0.06 & 0.995 & 0.005 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.25 & 0 & 0 \\ 0 & 0 & 0.005 & 0 & -0.06 & 0.995 & 0.005 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.25 \\ 0 & 0 & 0 & 0 & 0.005 & 0 & -0.055 & 0.995 \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.05 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix} \quad \bar{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \bar{D} = 0$$

The lower level regulator was designed as a centralized LQR with unit weights on all inputs and outputs, modified by zeroing all extra block-diagonal terms to obtain the decentralized linear gain

$$K = \begin{bmatrix} -0.3102 & -2.1343 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.2842 & -2.0997 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.2842 & -2.0997 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.3102 & -2. \end{bmatrix}.$$

It is immediate to verify that the closed-loop matrix $\bar{A} + \bar{B}K$ is strictly Schur and that matrix E in (2) is well posed. The HiMPC controller is designed to enforce the output constraint $y(t) \in \mathscr{Y}$, where $\mathscr{Y} = \{y \in \mathbb{R}^4 : \ -0.3 \leq y^i \leq 1, y^2 \leq y^1 + 0.3, \ i = 1, \dots, 4\}$, or $H_y = \begin{bmatrix} I \\ -I \\ -1 \ 1 \ 0 \ 0 \end{bmatrix}'$, $K_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \end{bmatrix}'$, corresponding to constraining mass positions between -0.3 and 1 m, and by preventing mass #1 to go below mass #2 by more than 0.3 m.

HiMPC adopts a linear MPC formulation based on model (21) or, in alternative, model (22), using the linear MPC setup of the Hybrid Toolbox [49]. The prediction horizon is 2, the control horizon 1, unit weights are used on reference increments and on mass position errors, i.e., on the deviations of y_k from a user-defined reference position signal $p(t)$. The constraints on control signals $r_k \in \mathscr{R}$ and on

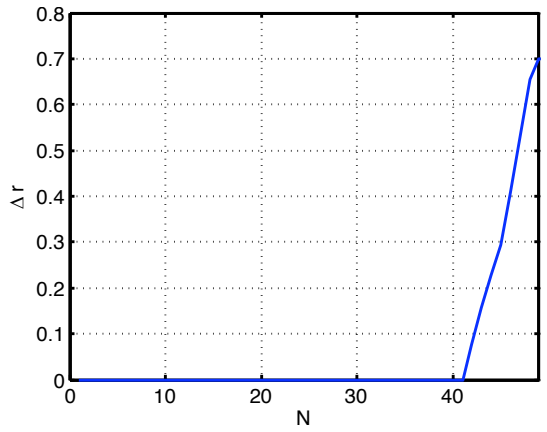


Figure 3: $\Delta r(N)$, obtained by solving (16) for $\Delta_0 = 0.3$

their increments $|r_k^i - r_{k-1}^i| \leq \Delta r(N) - \sigma$, $i = 1,2,3,4$ are enforced (σ=machine precision). The quantity

$$\Delta K_y = \Delta_0 \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0.4 \end{bmatrix}'$$

is chosen to restrict the tracking error, where Δ_0 is a scaling factor. The relation between N, Δ_0, and Δr is reported in Table 2. A "*" in the table denotes that (12) has no solution, which means that constraints on Δr become redundant with respect to the condition $r \in \mathscr{R}$. The scaling factor $\Delta_0 = 0.3$ will be used in the following simulations, as it provides a good compromise between the size of the invariant sets $\Omega(r)$ and the size of the admissible reference set $\mathscr{R} = \{r \in \mathbb{R}^4 : \ 0 \leq r^i \leq 0.7, \ i = 0, \ldots, 4\}$.

Table 2: Δr for different values of N and scalings factors Δ_0

$N \backslash \Delta_0$	0.1	0.3	0.4	0.55
18	0.01	0	0	0
21	0.04	0	0	0.20
42	0.41	0.07	0	*
44	0.41	0.22	0.14	*
45	0.42	0.33	0.36	*
46	0.44	0.38	0.50	*
48	0.46	0.66	*	*
49	0.48	0.70	*	*
54	1.00	*	*	*
57	1.10	*	*	*

To give an example of the complexity of the MILP optimization problem (16), this is solved in 141.2 s for $N = 49$ and $\Delta_0 = 0.3$ on an iMac Intel Core 2 Duo 2.8GHz running Matlab R2009b under Windows XP. The MPT Toolbox [50] was used for invariant set computations, YALMIP [51] and Cplex 9.0 [52] to formulate and solve, respectively, the MILP (16). Figures 3 and 4 show the resulting $\Delta r(N)$ and the ratio $R(N) = \Delta r(N)/N$ as a function of N, respectively. The function depicted in Figure 3 is non-decreasing, in accordance with Lemma 2. Figure 4 shows the maximum reference rate that can be generated by the upper layer of the proposed hierarchical control scheme, which in this particular case is also increasing with N. By inspecting Figures 3 and 4, a good value of N is 49, where both $R(N)$ and $\Delta r(N)$ are maximized. The resulting reference variations are $\Delta r(N) = 0.7$.

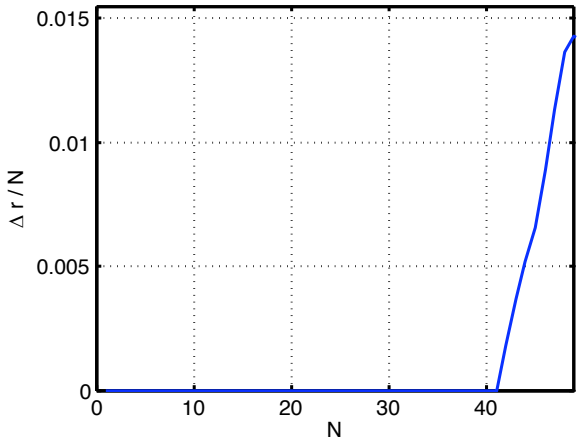


Figure 4: $R(N) = \frac{\Delta r(N)}{N}$ for $\Delta_0 = 0.3$

2.6.2 Simulation results

Denote by HiMPC^{49} the upper-layer MPC controller running every $T_H = 49 \cdot T_L$ seconds, based on prediction model (21), obtained by resampling (3) with sampling time T_H. Denote by HiMPC$^{49}_{DC}$ the alternative controller based on model (22). To demonstrate the effectiveness of the proposed hierarchical approach, we compare it to other two controllers: "HiNone", where the upper layer is simply bypassed and constraints are ignored by feeding $p(t)$ to the lower level, and "HiQP" that selects r_k according to the following quadratic program (QP)

$$\min_{r_k} \quad \|r_k - r_{k-1}\|_2^2 + \|r_k - p(t)\|_2^2 \tag{23a}$$

$$\text{s.t.} \quad r^2 - r^1 \leq 0.3 \tag{23b}$$

$$-0.3 \leq r_k^i \leq 1, \; i = 1,2,3,4 \tag{23c}$$

which completely ignores the underlying closed-loop dynamics, and therefore gives no guarantee that constraints on $y(t)$ are enforced. We assume that the user-defined reference signal $p(t)$ only varies at common multiples of the sampling times, $p(t) = p_k$, $\forall t \in [kN, \ldots, (k+1)N-1]$, $k = 0, 1 \ldots$.

Figures 5 and 6 show the closed-loop trajectories of the positions of masses #1 and #2 and of the commanded references, respectively, from zero initial condition $x(0)$. The trajectories obtained by using HiMPC^{49} and the trajectories of masses #3 and #4 are not reported in the figures. The reasons are that the HiMPC controller based on the resampled model (21) behaves very similarly to the one based on the static+delay model (22), due to the fact that the sampling time $T_H = NT_l = 12.25$ s is long enough to neglect the closed-loop dynamics; moreover, despite the coupling due to springs, masses #3 and #4 track a constant reference very tightly, even during setpoint variations on the other masses #1, #2.

The unfiltered reference $p^1 = 0.7$ applied by HiNone during the first sample instant $[0, T_H]$ makes mass #1 violate the upper limit $y^1 \leq 1$ between time $t \approx 3$ s to $t \approx 6$ s. At time $t = 49$ s a transition from $p^1 = 0.7$ to $p^1 = 0.1$ is requested, while p^2 is decreased to 0.6. The user is demanding a steady-state infeasible configuration of the masses, since $p^2 - p^1 \geq 0.3$. Note in Figure 7 that HiNone tracks the references violating the constraint. Since HiQP does not tighten enough the constraints by ΔK_y, at time $t \approx 50$ s a violation occurs of the constraint $y^2 - y^1 \leq 0.3$ by 0.03. At time $t = 98$ s the setpoints are set again to a feasible configuration $p^1 = 0.4$ and $p^2 = 0.7$. Comparing HiNone, HiQP, HiMPC, it is apparent that HiMPC is the most cautious, as evidenced by the commanded reference signals depicted in Figure 6, but it is also the only one that is able to enforce all constraints correctly.

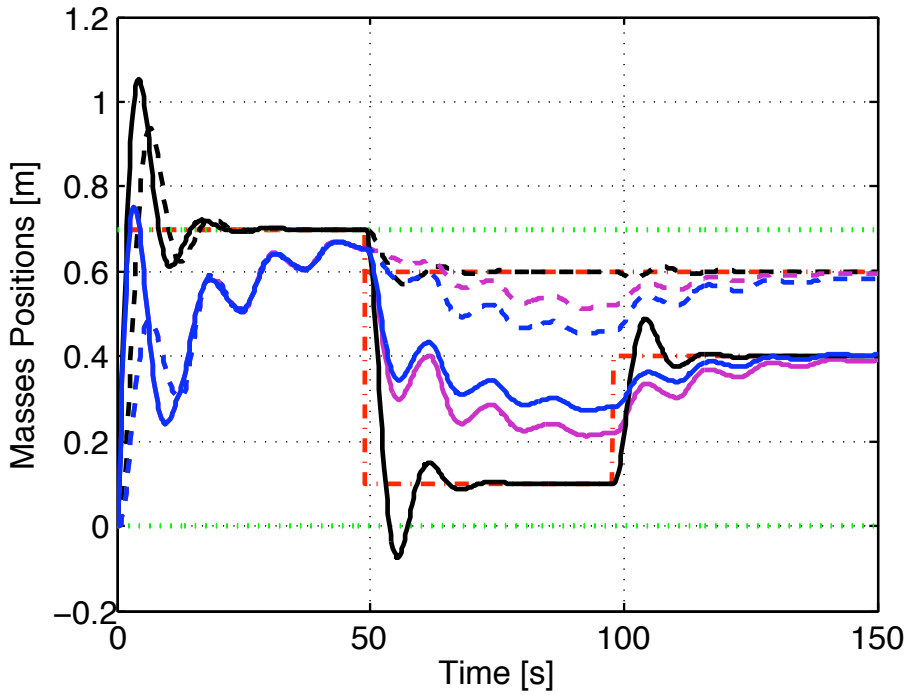


Figure 5: Closed-loop trajectories showing the position of mass #1 (continuous line) and #2 (dashed line): HiMPC (blue lines), HiNone (black lines), HiQP (purple lines). User defined reference $p(t)$ (dash dotted red lines) and reference constraints (dotted green line) are also shown

3 Decentralized Hierarchical Multi-Rate Control of Constrained Linear Systems

3.1 Introduction

The recent massive innovations in automation and communication technologies make control of large-scale systems (LSS) a viable technology. The complexity and spatial distribution of many sensors and actuators and their interconnections over a communication network require however novel control design approaches: most of the existing advanced control theory is based on the assumption that the decision-making process is *centralized*, which is impractical in the case of LSS; more classical methodologies based on *decentralized control*, developed starting already in the seventies (see the book of [56]), do not fully exploit the computation potentials of modern control systems. A widely used technique that exploits numerical optimization is model predictive control (MPC); current industrial MPC practice is, however, to rely on global prediction models and centralized real-time optimization, although decentralized MPC schemes were proposed recently (see e.g. [33, 57, 58, 62] and the recent survey [59]). While decentralization avoids the need of maintaining a model of the entire process and of solving a possibly large-scale optimization problem in a central control station at each sampling time, performance is typically degraded with respect to centralized schemes (the more is degraded the more local dynamics and objectives are coupled) and global constraints are often hard to impose without time-consuming iterative decision processes. Hierarchical control is a good compromise: lower-level local controllers take care of stabilization tasks based on simplified local dynamical models, and are orchestrated by a upper-level control layer that maximizes global performance and enforces global constraints [29]. A similar concept was also adopted in the reference governor (RG) literature, where the complexity of MPC was mitigated by separating the stabilization problem from the constraint fulfillment problem [36, 37, 39–41].

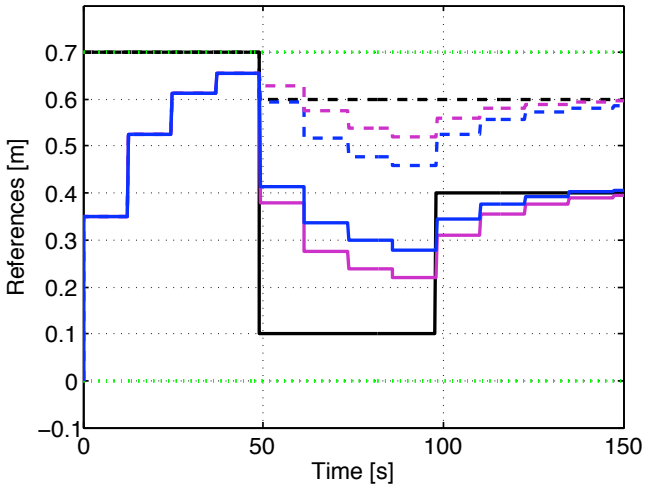


Figure 6: References generated by HiMPC (blue lines), HiNone (black lines), HiQP (purple lines) for mass #1 (continuous line) and mass #2 (dashed line)

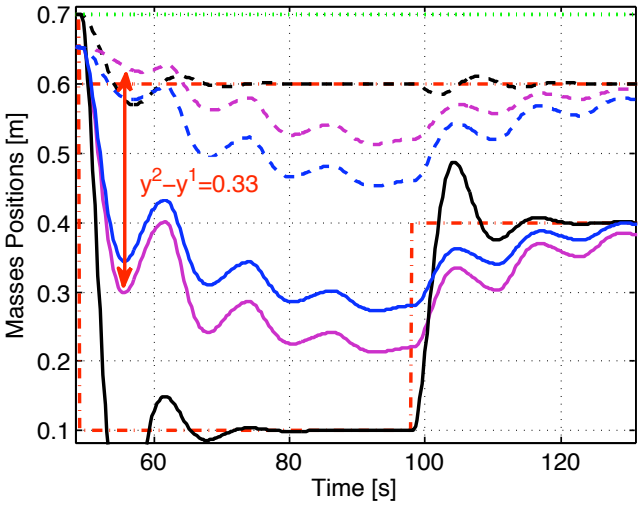


Figure 7: Zoom of Figure 5 showing the violation of the constraint $y_2 - y_1 \leq 0.3$: position of mass #1 (continuous line) and #2 (dashed line), HiMPC (blue lines), HiNone (black lines), HiQP (purple lines)

A recent approach to hierarchical control was proposed in [60] to guarantee constraint fulfillment, in which a centralized upper-level controller, running at a lower sampling frequency and based on a global more abstract model (e.g., the DC gain) of the system, restricts the set of admissible reference signals to the lower layer of decentralized linear controllers, therefore guaranteeing the fulfilment of constraints. The approach provides quantitative criteria to bound the maximum allowed reference variations and to choose the ratio N between the sampling intervals of the upper and lower level so that fulfillment of input and state constraint is guaranteed.

This contribution extends such an idea further, by decentralizing also the upper-level control layer (see Figure 8). We assume that the plant is stabilized by a set of m lower-level controllers, all running with sampling time T^L, receiving feedback only from local states. Each local upper-level controller can run at an independent sampling rate T_i^H, $i = 1,\ldots,m$, generating the reference signal r_i, $i = 1,\ldots,m$ to the corresponding lower-level controller. The absence of a centralized upper-level layer avoids centralized computations and guarantees full scalability.

The basic idea to enforce constraints is the following: After extending the hierarchical multi-rate

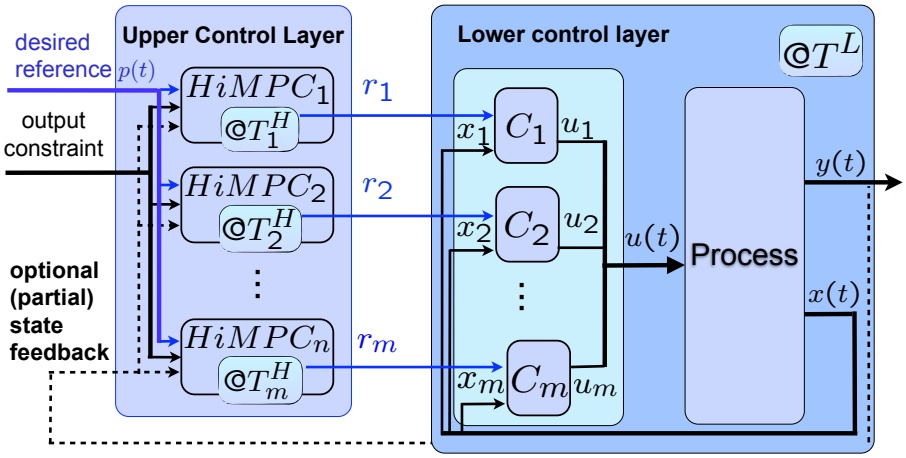


Figure 8: Decentralized hierarchical control scheme

approach of [60] to handle robust constraint fulfillment in the presence of additive state uncertainty, we model the dynamical coupling among subsystems as local disturbances (as done for instance in [57]), and exploit the limits imposed on unmodeled states to bound the sets containing such local disturbances.

Multirate MPC approaches was proposed in a variety of papers, see e.g. the early work of [42], and the application papers [43, 44] in which hybrid MPC control is used at the upper level to enforce complex linear and logical constraints. Two main issues arise in hierachical MPC design: the choice of a simple abstract model to design the upper-level controller (or possibly no model), and the choice of its sampling time T^H. Rule of thumbs suggest that the latter must be "large enough" to assume that the adopted prediction model is "enough consistent" with the true underlying closed-loop system, but "not too small" to ensure enough reactiveness of the hierarchical scheme to changes of desired references. In this work we quantify exactly what "large enough" should be in a decentralized setting. Safe operations are guaranteed by constraints on magnitude and rate of the reference signals parameterized by the ratio between the sampling intervals of the two layers, no matter how the performance index (if any) is optimized on top by the upper-level controller.

The proposed hierarchical control architecture is described in Section 3.2. In Section 3.3 the constraints on reference signals and their dependence on the ratio between sampling intervals is characterized and optimized, and used in Section 3.4 to define the overall hierarchical control scheme. A particular design for the upper control layer based on on-line optimization is described in Section 3.5. Simulation results are reported in Section 3.6.

3.2 Problem setup

Consider the decentralized hierarchical control architecture depicted in Figure 8, where the open-loop process

$$x(t+1) = \bar{A}x(t) + \bar{B}u(t) \tag{24a}$$
$$y(t) = \bar{C}x(t) + \bar{D}u(t) \tag{24b}$$

is in closed loop with the lower-level control layer

$$u(t) = Fx(t) + Er(t) \tag{24c}$$

where $x(t) \in \mathbb{R}^{n_x}$, $y(t) \in \mathbb{R}^{n_r}$, $u(t) \in \mathbb{R}^{n_u}$, and $r(t) \in \mathbb{R}^{n_r}$ is the reference signal. We assume that the gain F is asymptotically stabilizing and running at a sampling frequency $f^L = \frac{1}{T^L}$. Denote by $\mathscr{I}_x^i$, $\mathscr{I}_u^i$, $\mathscr{I}_r^i$

the sets of state, input, and output indices, respectively, corresponding to the i-th subsystem, and by n_x^i, n_u^i, n_r^i their cardinalities, $i = 1,\dots,m$. Note that in general such subsets of indices may overlap, in case different subsystems share common states, inputs, or outputs. We assume that F and E have the decentralized structure

$$F_{h,j} = 0,\ \forall h \in \mathscr{I}_u^i \text{ and } j \notin \mathscr{I}_x^i, \forall i = 1,\dots,m \tag{25}$$

$$E_{h,j} = 0,\ \forall h \in \mathscr{I}_u^i \text{ and } j \notin \mathscr{I}_r^i, \forall i = 1,\dots,m \tag{26}$$

The gains F and E can be computed by several methods, see e.g. [45, 56, 61].

To enforce constraints on state and inputs, at the upper level a set of m supervisors, running at lower sampling frequencies $f_i^H = \frac{1}{T_i^H}$, $i = 1,\dots,m$, manipulates the components $j \in \mathscr{I}_r^i$ of the desired reference vector $p \in \mathbb{R}^{n_r}$, producing a corresponding reference vector $r_i \in \mathbb{R}^{n_r^i}$. The resulting reference vector $r \in \mathbb{R}^{n_r}$ is passed to the lower level. The selection of r from p may be driven by numerical procedures optimizing performance or economic criteria.

The closed-loop system (24a), (24c) can be rewritten as

$$x(t+1) = Ax(t) + Br(t) \tag{27}$$

where $A = \bar{A} + \bar{B}F$, $B = \bar{B}E$. Despite the fact that only local state feedback is used, the global model (27) may not be block diagonal, because of possible dynamical coupling through matrices $\bar{A}$, $\bar{B}$. Hence, system (27) can be written as the collection of m dynamical systems

$$\Sigma_i :\ x_i(t+1) = A_i x_i(t) + B_i r_i(t) + d_i(t) \tag{28}$$

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

Assumption 1. *Matrix A_i has spectral radius within the unit circle.*

Assumption 1 states that the nominal local closed-loop system $(d_i(t) = 0))$ is asymptotically stable, and is a condition that should be imposed while synthesizing F.

Given a matrix M, let $M_{I,J}$ be the submatrix of matrix M obtained by collecting the row indices in I and the column indices in J, and by M_I the submatrix collecting all the rows indexed by I. A similar notation is used for subvectors v_I of a given vector v. Then we set $A_i = A_{\mathscr{I}_x^i, \mathscr{I}_x^i}$, $B_i = B_{\mathscr{I}_x^i, \mathscr{I}_r^i}$ in (28).

The neglected dynamics are modeled as follows. Let $\mathscr{J}_x^i = \{1,\dots,n_x\} \setminus \mathscr{I}_x^i$, $\mathscr{J}_r^i = \{1,\dots,n_r\} \setminus \mathscr{I}_r^i$, and set $\tilde{x}_i = x_{\mathscr{J}_x^i}$, $\tilde{r}_i = r_{\mathscr{J}_r^i}$, $\tilde{A}_i = A_{\mathscr{I}_x^i, \mathscr{J}_x^i}$, $\tilde{B}_i = B_{\mathscr{I}_x^i, \mathscr{J}_r^i}$, where $(\tilde{A}_i, \tilde{B}_i)$ capture the influence of unmodeled states $\tilde{x}_i$ and reference signals $\tilde{r}_i$ on Σ_i.

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

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

$$r_i(t) = r_i^k,\ t = kN_i,\dots,(k+1)N_i - 1,\ \ k = 0,1\dots \tag{29}$$

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

$$\mathscr{X}_i = \{ \left[\begin{smallmatrix} x_i \\ r_i \end{smallmatrix}\right] \in \mathbb{R}^{n_x^i + n_r^i} :\ H_x^i x_i + H_r^i r_i \le K^i \} \tag{30}$$

where $H_x^i \in \mathbb{R}^{q_i \times n_x^i}$, $H_r^i \in \mathbb{R}^{q_i \times n_r^i}$, $K^i \in \mathbb{R}^{q_i}$. Note that (30) covers the case of input, state, and output constraints, and constraints on the local tracking error $y_i - r_i$.

Let $A_i^0 \in \mathbb{R}^{n_x^i \times n_x}$ be the matrix obtained by collecting from A the rows indexed by $\mathscr{I}_x^i$ and then zeroing the columns indexed by $\mathscr{I}_x^i$, and similarly $B_i^0 \in \mathbb{R}^{n_x^i \times n_r}$ is the matrix obtained by collecting from B the

rows indexed by $\mathscr{I}_x^i$ and then zeroing the columns indexed by $\mathscr{I}_r^i$. Hence, $A_{\mathscr{I}_x^i}x = A_i x_i + B_i r_i + A_i^0 x + B_i^0 r$. Moreover, let $\mathscr{X} = \{\left[\begin{smallmatrix} x \\ r \end{smallmatrix}\right] : \; H_x x + H_r r \leq K\}$, where H_x, H_r, K define the set of states and references such that their subvectors x_j, r_j belong to $\mathscr{X}_j$, $\forall j = 1, \ldots, m$, which is a bounded set as $\mathscr{X}_j$ are polytopes. Under the assumption that $\left[\begin{smallmatrix} x_i(t) \\ r_i(t) \end{smallmatrix}\right] \in \mathscr{X}_i$ holds for all $t \geq 0$, $\forall i = 1, \ldots, m$, and therefore that $\left[\begin{smallmatrix} x(t) \\ r(t) \end{smallmatrix}\right] \in \mathscr{X}$, $\forall t \geq 0$, the local "disturbance" $d_i(t)$ modeling the effect of the other subsystems on Σ_i, belongs to the bounded polyhedral set (=polytope)

$$\begin{array}{rl} \mathscr{D}_i = & \{d_i \in \mathbb{R}^{n_x^i} : \exists x \in \mathbb{R}^{n_x}, \; r \in \mathbb{R}^{n_r} \text{ such that} \\ & d_i = A_i^0 x + B_i^0 r, \; [\begin{smallmatrix} x \\ r \end{smallmatrix}] \in \mathscr{X}\} \end{array} \tag{31}$$

We aim at determining the ratio N_i and restrictions on the reference values r_i^k and their variations $\Delta r_i^k = r_i^k - r_i^{k-1}$ generated by the upper-level controllers simultaneously so that $\left[\begin{smallmatrix} x_i(t) \\ r_i(t) \end{smallmatrix}\right] \in \mathscr{X}_i$, $\forall i = 1, \ldots, m$, for any $d_i \in \mathscr{D}_i$.

Let the reference vector $r_i(t)$ be constrained within the tightened set

$$\mathscr{R}_i = \{r_i \in \mathbb{R}^{n_r^i} : \; (H_x^i G_i + H_r^i) r_i \leq K^i - \Delta K^i\} \tag{32}$$

where $G_i = (I - A_i)^{-1} B_i$ is the reference-to-state DC gain of (28). Vector $\Delta K^i \in \mathbb{R}^{q_i}$ is selected to have positive components. We assume that set-points $r_i(t)$ are changed in a way that the tracking error $\Delta x_i(t) \triangleq x_i(t) - G_i r_i(t)$ is kept within the set

$$\mathscr{E}_i = \{\Delta x_i \in \mathbb{R}^{n_x^i} : \; H_x^i \Delta x_i \leq \Delta K^i\} \tag{33}$$

Vector ΔK^i is a tuning knob of the proposed approach: the smaller the components of ΔK^i are, the larger is the set $\mathscr{R}_i$ of admissible set points r_k, but the smaller is the admissible reference increments Δr_i^k to maintain tracking errors $\Delta x_i(t)$ within $\mathscr{E}_i$, $\forall i = 1, \ldots, m$.

As $x_i^r = G_i r_i = A_i x_i^r + B_i r_i$, in the new coordinates $\Delta x_i = x_i - x_i^r$ Equation (28) becomes

$$\Delta x_i(t+1) = A_i \Delta x_i(t) + d_i(t) \tag{34}$$

Let $\Omega_i(0) \subseteq \mathbb{R}^{n_x^i}$ be the maximum output admissible robustly invariant set (MOARS, [63]) for system (34) under the constraint $\Delta x_i(t) \in \mathscr{E}_i$

$$\begin{array}{rcl} \Omega_i(0) & = & \{\Delta x_i \in \mathbb{R}^{n_x^i} : H_x^i (A_i^k \Delta x_i + \sum_{j=0}^{k-1} (A_i^k)^j d_i(k-1-j)) \\ & & \leq \Delta K^i, \; d_i(k-1-j) \in \mathscr{D}_i, \; \forall k \geq 0\} \end{array} \tag{35}$$

Let (H_0^i, K_0^i) be a minimal hyperplane representation of $\Omega_i(0)$, $\Omega_i(0) = \{\Delta x_i \in \mathbb{R}^{n_x^i} : \; H_0^i \Delta x_i \leq K_0^i\}$, and let n_0^i be the number of such hyperplanes, that under Assumption 1 exists and is finite for each $i = 1, \ldots, m$, see [63]. Then

$$\Omega_i(r_i) = \{x_i \in \mathbb{R}^{n_x^i} : \; x_i - G_i r_i \in \Omega_i(0)\} \tag{36}$$

The following lemma extends Lemma 1 in [60] to cover the case of polytopic uncertainty.

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

Proof: By (10), $x_i(0) \in \Omega_i(r_i)$ implies that $H_x^i \Delta x(t) = H_x^i x(t) - H_x^i G_i r_i \leq \Delta K^i$, $\forall d_i(j) \in \mathscr{D}_i$, $j < t$, and $\forall t \geq 0$. By (32) it follows that $H_x^i x(t) - H_x^i G_i r_i \leq \Delta K^i \leq K_i - H_x^i G_i r_i - H_r^i r_i$ which in turns implies $H_x^i x(t) + H_r^i r_i \leq K_i$, $\forall t \geq 0$. □

The main idea of this work is the following. Assume that the reference $r_i^k \in \mathscr{R}_i$ is issued at the sampling instant $t = kN_i$ and that N_i is large enough so that $x_i(t+N_i-1) \in \Omega_i(r_i^k)$, for all $i = 1,\ldots,m$. At time $t = (k+1)N_i$ consider the new reference is $r_i^{k+1} \in \mathscr{R}_i$. If $\Delta r_i^{k+1} = r_i^{k+1} - r_i^k$ is "small enough", then $x_i(t+N_i) \in \Omega_i(r_i^{k+1})$. The goal of the next section is to quantify the relationship among the maximum reference variation Δr_i^k, the ratio N_i between the sampling intervals T_i^H, T^L, and ΔK^i such that every $T_i^H = N_i T^L$ time units each state subvector $x_i(t)$ is guaranteed to lie in the corresponding invariant set $\Omega_i(r_i^k)$, which in turn implies that $x(t)$, $r(t)$ satisfy the constraint $\begin{bmatrix} x(t) \\ r(t) \end{bmatrix} \in \mathscr{X}$.

3.3 Reference rate constraints

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

$$\Delta r_i(N_i) = \inf_{r_i^1, r_i^2, x_i(0), D_{N_i}} \; ||r_i^2 - r_i^1||_\infty \tag{37a}$$

$$\text{s.t.} \quad r_i^1, r_i^2 \in \mathscr{R}_i \tag{37b}$$

$$x_i(0) \in \Omega_i(r_i^1) \tag{37c}$$

$$x_i(t+1) = A_i x_i(t) + B_i r_i^2 + d_i(t) \tag{37d}$$

$$d_i(t) \in \mathscr{D}_i,\ t = 0,1,\ldots,N_i-1 \tag{37e}$$

$$x(N_i) \notin \Omega_i(r_i^2) \tag{37f}$$

Because of constraint (37f), problem (37) is nonconvex. However, it can be conveniently recast as a mixed-integer linear programming (MILP) problem by introducing an auxiliary binary vector $\delta \in \{0,1\}^{n_0}$, satisfying the following constraints

$$[\delta^h = 1] \leftrightarrow [(H_0^i)_h (x_i(N_i) - G_i r_i^2) \leq (K_0^i)_h] \tag{38a}$$

$$\sum_{h=0}^{n_0^i} \delta_h \leq n_0^i - 1 \tag{38b}$$

where the subscript h denotes the h-th component or row. The logical constraint (38a) can be converted to mixed-integer linear inequalities using the standard "big-M" approach (see e.g. [64]). By introducing an additional variable $\varepsilon \geq \|r_i^2 - r_i^1\|_\infty$, we address problem (37) by solving the following

MILP

$$\Delta r_i(N_i) = \min_{\left[x_i\ r_i^{1'}\ r_i^{2'}\ D_{N_i}\ \delta'\ \varepsilon\right]'} \quad \varepsilon \tag{39a}$$

$$\text{s.t. } \varepsilon \geq \pm((r_i^2)_j - (r_i^1)_j) \ , \ j = 1,\dots,n_r^i \tag{39b}$$

$$(H_x^i G_i + H_r^i) r_i^1 \leq K^i - \Delta K^i \tag{39c}$$

$$(H_x^i G_i + H_r^i) r_i^2 \leq K^i - \Delta K^i \tag{39d}$$

$$H_0^i (x_i - G_i r_i^1) \leq K_0^i \tag{39e}$$

$$(H_0^i)_h \left(A_i^{N_i} x_i + \sum_{j=0}^{N_i-1} A^j d_i(N_i-1-j) + R_G^i r_i^2 - G_i r_i^2 \right) - (K_0^i)_h \leq (M_+^i)_h (1-\delta^h) \tag{39f}$$

$$-(H_0^i)_h \left(A_i^{N_i} x_i + \sum_{j=0}^{N_i-1} A^j d_i(N_i-1-j) + R_G^i r_i^2 - G_i r_i^2 \right) + (K_0^i)_h \leq -((M_-^i)_h - \sigma)\delta^h - \sigma \tag{39g}$$

$$d_i(t) \in \mathscr{D}_i,\ t = 0,1,\dots,N_i-1 \tag{39h}$$

$$\sum_{w=0}^{n_o^i} \delta^w \leq n_o^i - 1 \tag{39i}$$

$$\delta^h = \{0,1\},\ h = 1,\dots,n_0^i \tag{39j}$$

where $R_G^i = \left(\sum_{h=0}^{N_i-1} A_i^h B_i \right) - G_i$. The quantity $\Delta r_i(N_i)$ in (39) is the smallest change of reference components in $\mathscr{I}_r^i$ (expressed in infinity norm) that can be applied to the closed-loop system (27) such that, if x_i starts from an invariant set $\Omega_i(r_i^k)$, the i-th state vector lands outside the new invariant set $\Omega_i(r_i^{k+1})$ after N_i steps. Or, in other words, if for all $i = 1,\dots,m$ the change of reference signals are bounded by $\|r_i^k - r_i^{k-1}\|_\infty \leq \Delta r_i(N_i) - \sigma$ and each subvector x_i of the closed-loop system (27) starts from the invariant set $\Omega_i(r_i^k)$, than x_i arrives in its corresponding new invariant set $\Omega_i(r_i^{k+1})$ after N_i steps, for all arbitrarily small $\sigma > 0$.

Note that, because of the constraint $r_i^1, r_i^2 \in \mathscr{R}_i$, problem (39) may become infeasible for large N_i, that is no $r_i^2 \neq r_1$ exists that keeps the state within the invariant set $\Omega(r_i^2)$ after N_i steps.

The following lemma shows a monotonicity property of $\Delta r_i(N_i)$ as a function of N_i, for $N_i \in \mathbb{N}$ such that $\Delta r_i(N_i)$ is defined.

Lemma 4. *Let $\Delta r_i(N_i)$ be defined by the optimization problem* (39)*. Then for any $N_i^1, N_i^2 \in \mathbb{N}$, $N_i^1 < N_i^2$, such that $\Delta r_i(N_i^1)$, $\Delta r_i(N_i^2)$ are defined it holds that*

$$\Delta r_i(N_i^1) \leq \Delta r_i(N_i^2) \tag{40}$$

Proof: We first prove by contradiction that $\Delta r_i(N_i) \leq \Delta r_i(N_i+1)$, $\forall N_i \in \mathbb{N}$ such that $\Delta r_i(N_i+1)$ is defined. Assume that $N_i \in \mathbb{N}$ exists such that $\Delta r_i(N_i+1) < \Delta r_i(N_i)$. This implies that there exists a state x_i, a disturbance sequence D_{Ni}, and two references $r_i^1, r_i^2 \in \mathscr{R}_i$ such that $\Delta r_i(N_i+1) \leq \|r_i^1 - r_i^2\|_\infty < \Delta r_i(N_i)$, $x_i \in \Omega_i(r_i^1)$, $A_i^{N_i+1} x_i + \sum_{h=0}^{N_i} A_i^h (d_i(N_i-h) + B_i r_i^2) \notin \Omega_i(r_i^2)$. Then, also $A_i^{N_i} x_i + \sum_{h=0}^{N_i-1} A_i^h (d_i(N_i-1-h) + B_i r_i^2) \notin \Omega_i(r_i^2)$, otherwise, by invariance of $\Omega_i(r_i^2)$, also $A_i^{N_i+1} x_i + \sum_{h=0}^{N_i} A_i^h (d_i(N_i-h) + B_i r_i^2)$ would belong to $\Omega_i(r_i^2)$. Hence, the optimality of $\Delta r_i(N_i)$ is violated, a contradiction. The monotonicity condition (40) for generic N_i^1, N_i^2 easily follows. □

3.4 Hierarchical controller

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

$$\|r_i^k - r_i^{k-1}\|_\infty \leq \Delta r_i(N_i) - \sigma, \ \forall k = 0,1,\dots \tag{41a}$$

$$r_i^k \in \mathscr{R}_i, \ \forall k = -1,0,1,\dots \tag{41b}$$

for some small $\sigma > 0$, to the lower control layer as in (29).

Theorem 2. *Assume that K is a decentralized asymptotically stabilizing linear gain, that Assumption 1 holds, and that a set of vectors $r_i^{-1} \in \mathscr{R}_i$ exists such that the initial states $x_i(0) \in \Omega_i(r_i^{-1})$, for $i = 1,\dots,m$. Let all the upper-level controllers change the set-points r_i^k according to the constraints* (41)*, in which $\Delta r_i(N_i)$ is the solution of* (39) *and $\sigma > 0$ is an arbitrary small number. Then the linear system* (1) *satisfies the constraints $\begin{bmatrix} x(t) \\ r(t) \end{bmatrix} \in \mathscr{X}$, for all $t \geq 0$. If in addition $\lim_{t\to\infty} r(t) = r \in \mathscr{R}$ then $\lim_{t\to\infty} x(t) = Gr$, $G = (I-A)^{-1}B$.*

Proof: Because of (41), $x_i(kN_i) \in \Omega_i(r_i^k)$, $\forall k \geq 0$. By Lemma 3, it follows that $\begin{bmatrix} x_i(t) \\ r_i(t) \end{bmatrix} \in \mathscr{X}_i$, $\forall t = kN_i,\dots,k(N_i+1)-1$, $\forall k = 0,1,\dots$ and $i = 1,\dots,m$, that is $\begin{bmatrix} x(t) \\ r(t) \end{bmatrix} \in \mathscr{X}$, $\forall t \geq 0$. As $\lim_{t\to\infty} r(t) = r$, similarly to (34) define $\Delta x(t) = x(t) - Gr$ and rewrite (27)as

$$\Delta x(t+1) = A\Delta x(t) + B(r(t) - r) \tag{42}$$

As (42) is an asymptotically stable linear system it is also input-to-state stable [47], and hence it immediately follows that $\lim_{t\to\infty} \Delta x(t) = 0$, which in turn implies that $\lim_{t\to\infty} x(t) = Gr$. □

Theorem 2 shows that *any* decentralized upper-level reference generation strategy satisfying constraints (41) guarantees the fulfillment of state+reference constraints and asymptotic convergence to constant set-points. The MILP (39) provides the supremum $\Delta r_i(N_i)$ of the reference variations $\|r_i^k - r_i^{k-1}\|_\infty$ for each subsystem i that the i-th upper-level controller can apply for a given ratio $N_i = T_i^H/T^L$ between consecutive sampling times. We stress that for each subsystem the ratio N_i is determined independently on the other hierarchical arrangements.

It is worth to investigate the relations between $\Delta r_i(N_i)$ and N_i further for each subsystem $i = 1,\dots,m$. In fact, the design of the i-th upper control layer could be addressed from a different point of view: given a desired Δr_i, determine the minimum N_i such that $\Delta r_i < \Delta r_i(N_i)$. In practical applications N_i is restricted to a range $[N_i^{min},\ N_i^{max}]$ of values: the upper layer is executed at a slower pace than the lower layer ($N_i^{\min}$ not too small), but at the same time the upper layer should be reactive enough to adjust set-points ($N_i^{\max}$ not too large). Hence, it is worth to solve the MILP (39) only within the restricted range $N_i \in [N_i^{\min}, N_i^{\max}]$ to characterize $\Delta r_i(N_i)$ that, by Lemma 4, we know increases with N_i. In particular, it is of interest the ratio $R_i(N_i) = \frac{\Delta r_i(N_i)}{N_i}$ which characterizes the maximum speed of change of the reference signal. In fact, for each subsystem, the larger N_i the larger is the supremum of the variations Δr_i that the local supervisor can issue, but the less frequently such variation happens, that is every $N_i T^L$ sampling times. Another issue related to tuning of the upper control layer is the choice of ΔK^i for each i: from one hand a larger ΔK^i tightens the range of admissible references $\mathscr{R}_i$, but on the other hand it enlarges the size of the invariant set $\Omega_i(r_i)$, and therefore augments the achievable $\Delta r_i(N_i)$. There is therefore a tradeoff: the designer must choose between constraints on reference signals ($\mathscr{R}_i$) and constraints on reference speed ($R_i(N_i)$).

Because of the need of enforcing constraints (41) in the upper control layer, in the next section we propose a decentralized model predictive control (MPC) design strategy for such a layer, although any other constraint-handling strategy could be employed, such as static optimization or a rule-based selection.

3.5 Decentralized MPC design of upper control layer

This section introduces a simple decentralized MPC strategy for the upper layer of control in the hierarchy, denoted as DHiMPC, for generating each reference signal r_i while enforcing constraints (41).

3.5.1 Prediction model

We consider an under-sampled and possibly reduced-order model of each nominal lower-level closed-loop model (28) ($d_i(t) = 0$), evolving with sampling time $T_i^H = N_i T^L$

$$x_i^H(k+1) = A_i^H x_i^H(k) + B_i^H r_i(k) \tag{43}$$

where $x_i^H(k) = Z_i x_i(kN_I)$, and Z_i is a matrix mapping each original state $x_i(kN_i)$ into the new state $x_i^H(k)$ (in case the order of the system is not reduced $Z_i = I$). Model (43) can be easily obtained by resampling system (28) for $d_i(t) = 0$ using standard discretization methods. Hence, fast-enough modal responses become negligible, which implies that the HiMPC algorithm can exploit only an incomplete information on the underlying local closed-loop dynamics. However, each HiMPC controller is independent from the others, which allows one to tune the upper-layer sampling rates individually. Therefore, reference changes can be commanded at different time instants, which guarantees maximum flexibility in large-scale systems that have different time constants. Note that, contrarily to [60] where a complex *centralized* upper-level supervisor based on a global (yet possibly reduced-order) model is used, in this work we propose a *decentralized* design and decentralized implementation that allows treating each subsystem independently.

3.5.2 Cost function and constraints

Each upper-level MPC controller must embed constraints (41) on the generated references, to ensure stability and constraint satisfaction of state-dependent constraints. Moreover, it is possible to embed additional constraints on the reference signals, such as mixed logical/linear constraints (see e.g. [43]) on local reference signals.

The i-th MPC controller can potentially optimize any cost function of $r_i(k)$, $\Delta r_i(k)$, and $x_i^H(k)$, that may be dictated for instance by economic objectives. Note that a global cost function cannot be directly addressed by means of DHiMPC, nevertheless various consensus [65] and distributed optimization [66] approaches can be applied.

3.6 Simulation example

3.6.1 Problem description

We test the effectiveness of the proposed approach on a multi-mass-spring system similar to the one described in [60], where a centralized hierarchical approach was used. The process is composed by four mass-spring-damper systems moving on the vertical axis, as described in Figure 9. Contrarily to [60], neighboring systems i and j are connected by dampers with coefficient $\mu_{i,j} = 0.005$ [kg/s], $\forall i, j$, instead of springs, which makes condition (26) satisfied. The remaining parameters are as in [60]. The system is described by a 8th order linear dynamics whose states are mass positions z and velocities v, $x = [z_1\ v_1\ z_2\ v_2\ z_3\ v_3\ z_4\ v_4]'$, whose input $u = [u_1\ u_2\ u_3\ u_4]'$ collects the forces applied to the masses, and whose output $y = [z_1\ z_2\ z_3\ z_4]'$ are mass positions, to be controlled on the set-point $r \in \mathbb{R}^4$.

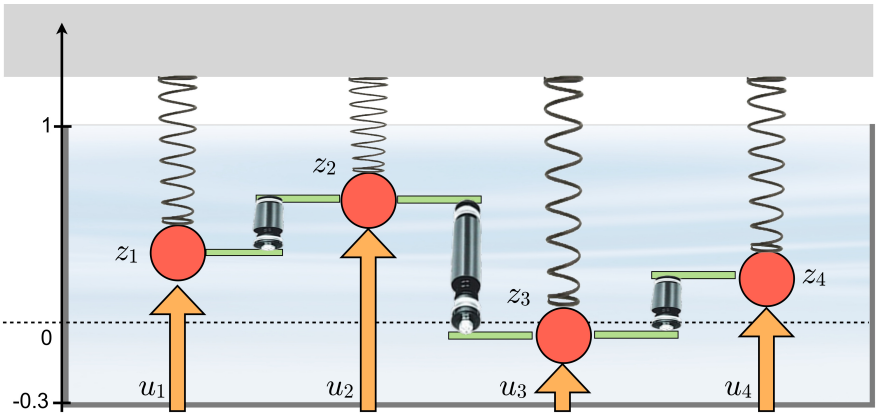


Figure 9: Dynamical process used in the simulation example

The plant is sampled every $T^L = 0.25$ s and is subject to the state constraints $x(t) \in \mathscr{X}$, where $\mathscr{X} = \{x \in \mathbb{R}^8 : -0.3 \leq x_1, x_3, x_5, x_7 \leq 1,\ x_3 \leq x_1 + 0.3\}$, corresponding to constraining mass positions between -0.3 and 1 m, and by preventing mass #1 to go below mass #2 by more than 0.3 m.

The equivalent discrete-time model (1) of the process is decentralized according to the following index selection

$$\mathscr{I}_x^1 = \{1,2,3,4\},\ \mathscr{I}_x^2 = \{5,6\}\ ,\ \mathscr{I}_x^3 = \{7,8\} \tag{44a}$$

$$\mathscr{I}_u^1 = \mathscr{I}_r^1 = \{1,2\},\ \mathscr{I}_u^2 = \mathscr{I}_r^2 = \{3\},\ \mathscr{I}_u^3 = \mathscr{I}_r^3 = \{4\} \tag{44b}$$

which clearly makes the constraint $x_3 \leq x_1 + 0.3$ local in subsystem Σ^1.

The upper level of DHiMPC is composed of a set of linear MPC controllers based on linear prediction models as in (43), and implemented using the Hybrid Toolbox [49] and the WIDE Toolbox developed by Barcelli et al. (`http://ist-wide.dii.unisi.it/toolbox/`). The prediction horizon is 10, the control horizon 5, unit weights are used on reference increments and weight equal 100 is used to penalize each mass position error. The constraints on control signals $r_i(k) \in \mathscr{R}_i$ and on their increments $|r_i^j(k) - r_i^j(k-1)| \leq \Delta r_i(N_i) - \sigma$, $j = 1,2,3,4$ are enforced for $i = 1,\ldots,m = 8$, where σ is the machine precision. The quantities

$$\Delta K^1 = \begin{bmatrix} \Delta_0 \\ \Delta_0 \\ 0.4\Delta_0 \end{bmatrix},\ \Delta K^2 = \begin{bmatrix} \Delta_0 \\ \Delta_0 \end{bmatrix},\ \Delta K^3 = \begin{bmatrix} \Delta_0 \\ \Delta_0 \end{bmatrix},\ \Delta K^4 = \begin{bmatrix} \Delta_0 \\ \Delta_0 \end{bmatrix}$$

are chosen to tighten the constraint on the reference in (32), where Δ_0 is a scaling factor to be determined.

The computation of the maximum output admissible robustly invariant sets (MOARS) $\Omega_i(0)$ defined in (36) is carried out independently for each subsystem i. Figure 10 shows the two-dimensional polytopes related to Σ^2 and Σ^3 (the MOARS for Σ^1 lies in $\mathbb{R}^4$ and is not shown). The conservatism introduced by the unmodeled dynamics $d_i(t)$ is displayed in the same figure by comparing the MOARS with the corresponding MOAS resulting by setting $d_i(t) = 0$ (which means a complete lack of interaction among the subsystems). Note that Σ^2 and Σ^3 have the same MOAS as they share the same local dynamical model, but different MOARS's. This is due to the overall plant structure, which is not symmetric for the last two subsystems, as mass #3 is surrounded by other two masses while mass #4 has only one neighboring mass.

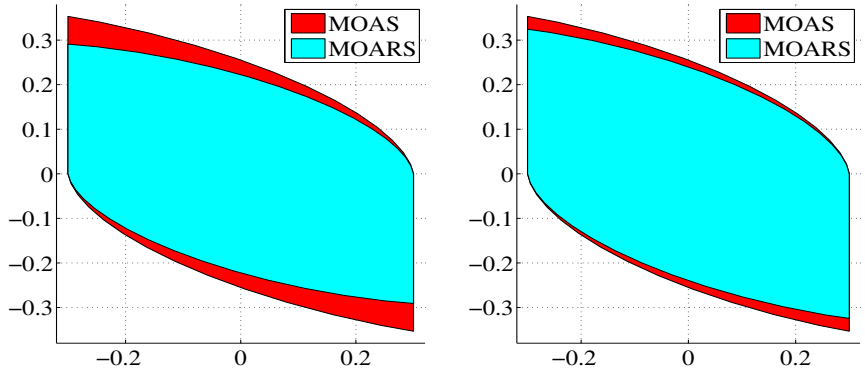


Figure 10: Invariant sets $\Omega_2(0)$ (left) and $\Omega_3(0)$ (right): with bounded dynamical coupling (MOARS, cyan) and interaction-free (MOAS, red)

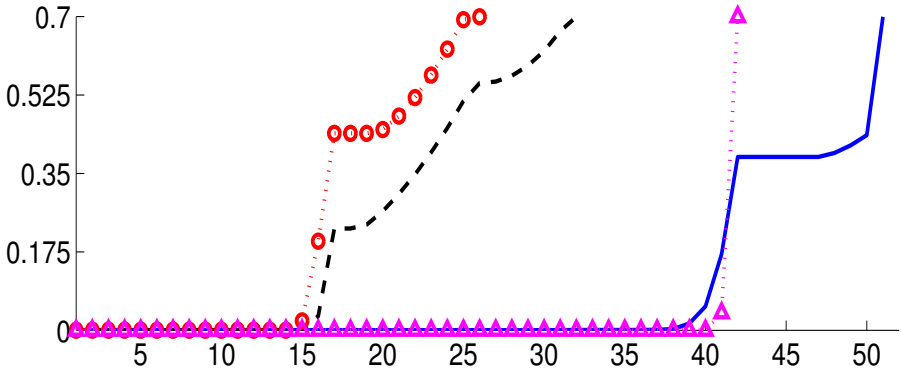


Figure 11: Plots of $\Delta r_i(N_i)$ as a function of N_i obtained by solving (39): subsystems Σ^1 (continuous blue line), Σ^2 (black dashed), and Σ^3 (red dotted with circles), and centralized approach (magenta dotted with triangles)

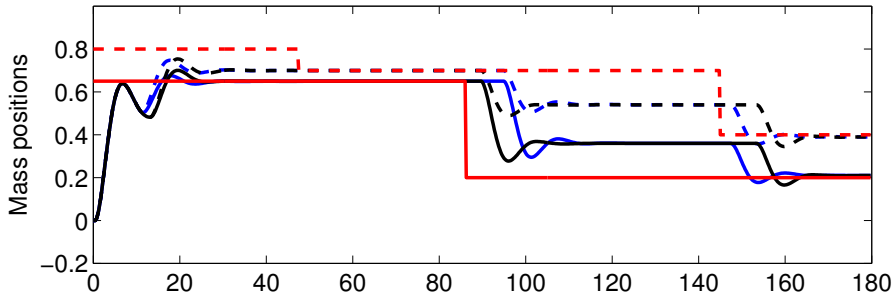
3.6.2 Computation of maximum reference rates

The relation among N_i, Δ_0 and Δr_i is investigated for each subsystem independently. Figure 3 depicts $\Delta r(N)$ for the values of N for which the optimization problem (39) admits a solution. For comparison, in Figure 3 we also report the results obtained by computing the same quantities with the centralized hierarchical approach of [60].

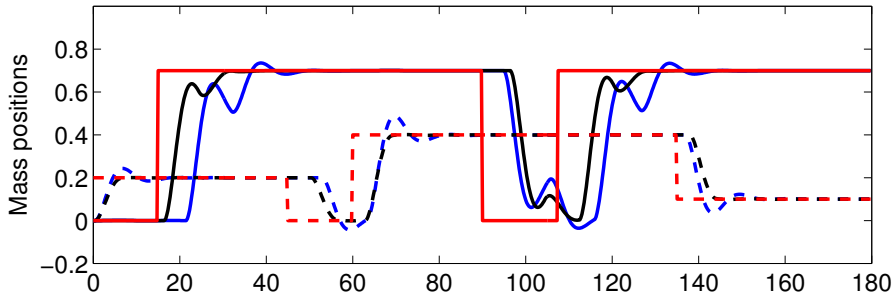
Let DHiMPC$_i$ denote the decentralized hierarchical controller designed for subsystem Σ_i, $i = 1, 2, 3$, and HiMPC the centralized hierarchical controller designed as proposed in [60]. DHiMPC$_i$ enforces constraints more conservatively than HiMPC because of the conservative way interactions are modeled. On the other hand, while HiMPC has a uniform sampling frequency on all subsystems, DHiMPC$_2$ and DHiMPC$_3$ can be implemented at smaller sampling times, which makes the reaction to changes in set points on masses #2 and #3 more quick.

All functions depicted in Figure 11 are nondecreasing, in accordance with Lemma 4. Similarly to [60], it is possible to compute the maximum reference rate that can be generated by each DHiMPC scheme by maximizing the ratio $\frac{\Delta r_i(N)}{N}$, $i = 1 \dots 3$.

From the computation viewpoint, the centralized MILP associated with HiMPC involves more state



(a) 1^{st} and 2^{nd} mass positions



(b) 3^{rd} and 4^{th} mass positions

Figure 12: Mass positions: DHiMPC (black), HiMPC (blue), mass #1 and #3 (continuous line), mass #2 and #4 (dashed line), and corresponding references (red)

variables and dynamic constraints than any of the decentralized DHiMPC's, however each of the latter needs to account for disturbances. Using the centralized MILP as a reference for comparison, the CPU time for computing the MILPs associated with the decentralized approach are get multiplied by a factor $\simeq 5$ (DHiMPC$_1$), $\simeq \frac{1}{5}$ (DHiMPC$_2$), and $\simeq \frac{1}{8}$ (DHiMPC$_3$).

3.6.3 Simulation results

We test the DHiMPC approach and compare it to the corresponding HiMPC one. Denote by $p(t)$ the desired output reference at time t, $p(0) = [0.65\ 0.8\ 0\ 0.2]'$, and let $r_i^{-1} = 0 \in \mathscr{R}_i$ and $x_i(0) = 0 \in \Omega_i(r_i^{-1})$, for $i = 1,2,3$. Let $N_1 = 51$, $N_2 = 32$, and $N_3 = 25$ for DHiMPC$_{1,2,3}$, respectively, and $N = 42$ for HiMPC. This choice leads to $\Delta r_1 = 0.7$, $\Delta r_2 = 0.7$, and $\Delta r_3 = 0.69$ for DHiMPC, and $\Delta r = 0.7$ for HiMPC.

At time $t = 0$ both HiMPC and DHiMPC$_1$ exploit the full admissible reference range, applying the maximum Δr to masses #1 and #2, see Figure 12(a). Note that the desired reference $p_2(t)$ provided by the user for the position of mass #2 is out of the bounds in (32) during the time interval $t \in [0,45]$, $p_2(0)$ also violates the reference variation constraint which is $0 \leq r_i(t) \leq 0.7$, $i = 1,\ldots,3$. Figure 12(a) shows how DHiMPC$_1$ reacts later than HiMPC to changes of $p(t)$, while instead DHiMPC$_2$ and DHiMPC$_3$ react more quickly. Note that constraints are always satisfied, even if commanded references p are infeasible, such as for $86 < t < 144.5$ s for masses #1 and #2.

3.7 Conclusions

This work has proposed a decentralized hierarchical control approach to handle state-dependent constraints in large-scale linear control systems. The control design is carried out in two steps: First, a lower-level set of decentralized linear controllers is designed to stabilize the process without

accounting for the constraints; second, each regulator is fed by an upper-level controller, running at a slower pace, that manipulates the desired references so as to guarantee the fulfillment of the constraints. Although some conservatism is introduced by treating the dynamic couplings as bounded disturbances, the proposed approach is totally scalable and therefore suitable for constrained linear systems of large size.

4 Hierarchical and Decentralized Hybrid Model Predictive Control of a Formation of Unmanned Aerial Vehicles

4.1 Introduction

The last few years have been characterized by an increasing interest in stabilizing and maneuvering a formation of multiple vehicles. Research areas include both military and civilian applications (such as intelligence, reconnaissance, surveillance, exploration of dangerous environments) where Unmanned Aerial Vehicles (UAVs) can replace humans. There are different types of UAVs: planes are suitable for long-range applications because are energy efficient, but they need wide operating spaces; rotorcrafts, such as helicopters and quadcopters, have less autonomy but can operate in limited workspaces, they can take off and land vertically and easily hover above targets. Nevertheless, VTOL (Vertical Take-Off and Landing) UAVs are more complex to control [1], because of highly nonlinear and coupled dynamics, and to limitations on actuators and pitch/roll angles. In particular, *quadcopters* are a class of VTOL vehicles for whose stabilization several approaches were proposed in literature: classical PID [2], nonlinear control [3], LQR [4], visual feedback [1], H_∞ control [5,6], and recently linear MPC (Model Predictive Control) [7].

MPC is particularly suitable for control of multivariable systems governed by costrained dynamics, as it allows to operate closer to the boundaries imposed by hard constraints; in the context of UAVs, MPC techniques have been already applied for control of formation flight in [8–13].

In the context of path planning for obstacle avoidance, several other solutions have been proposed in the literature, such as potential fields [14, 15], A^* with visibility graphs [2, 16], nonlinear trajectory generation (see e.g. the NTG software package developed at Caltech [9], and mixed-integer linear programming (MILP) [17, 18]. In particular the latter has shown the great flexibility of on-line mixed-integer optimization in real-time trajectory planning of aircrafts, as also reported in [19] where on-line MILP techniques were proved very effective in handling multiaircraft conflict avoidance problems.

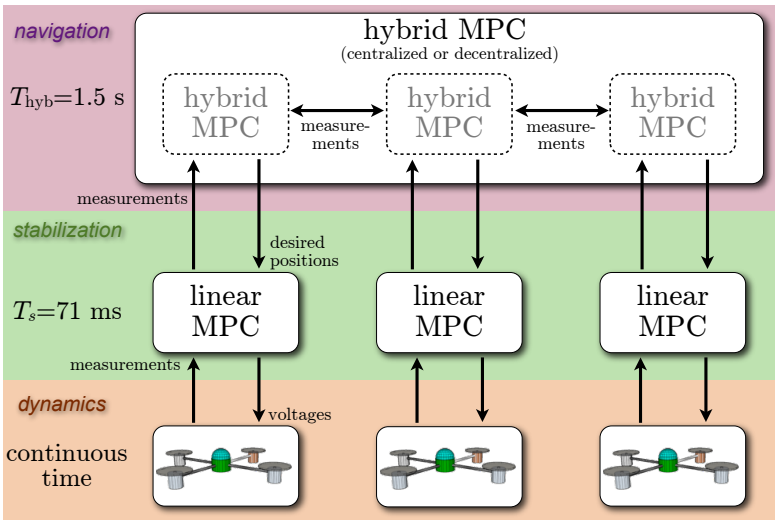


Figure 13: Hierarchical control structure for UAV navigation

Here we adopt the two-layer MPC approach to quadcopter stabilization and on-line trajectory generation for autonomous navigation with obstacle avoidance presented earlier in [7]. A linear constrained MPC controller with integral action takes care of quadcopter stabilization and offset-free tracking of desired set-points. At a higher hierarchical level and lower sampling rate, a hybrid MPC controller

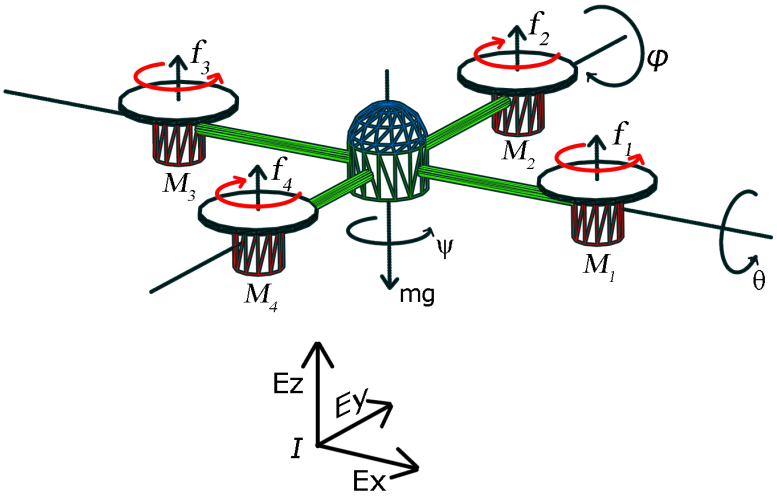


Figure 14: Quadcopter model

generates on line the path to follow to reach a given target position/orientation while avoiding obstacles. In this work we extend the approach in two ways: first, we let the underlying linear MPC algorithm controlling directly motor voltages, rather than torques; second, the higher-level hybrid MPC are organized in a decentralized control scheme, based on a leader-follower approach. We assume that target and obstacle positions may be time-varying, and in this case that only their current coordinates are known (not the future ones), so that off-line (optimal) planning cannot be easily accomplished.

In Section 4.2 we describe the layers of the hierachical structure of each controlled quadcopter and its nonlinear dynamics, whose linearization provides the prediction model for linear MPC design in Section 4.3 for stabilization under constraints and tracking of generated trajectories. Then, the higher-level hybrid MPC controller for path planning and obstacle avoidance is described in Section 4.3.2. In Section 4.6 we give the results of simulations, comparing our approach with the potential fields method described in [15]. Finally, some conclusions are drawn in Section 4.7.

4.2 Hierarchical MPC of each UAV

Consider the hierarchical control system depicted in Figure 13. At the top layer a hybrid MPC controller generates on-line the desired positions (x_d, y_d, z_d), in order to accomplish the main mission, namely reach a given target position (x_t, y_t, z_t) while avoiding collisions with obstacles and other UAVs. This references are tracked in real-time by a linear MPC controller placed at middle layer of the architecture. At the bottom layer of the closed-loop system lies the nonlinear dynamics of the quadcopter, commanded by linear MPC. In the next subsections we describe in details each layer of the architecture.

4.2.1 Nonlinear quadcopter dynamics

A quadcopter air vehicle is an underactuated mechanical system with six degrees of freedom and only four control inputs (see Figure 14). We denote by x, y, z the position of the vehicle and by θ, ϕ, ψ its rotations around the Cartesian axes, relative to the "world" frame I. In particular, x and y are the coordinates in the horizontal plane, z is the vertical position, ψ is the yaw angle (rotation around the z-axis), θ is the pitch angle (rotation around the x-axis), and ϕ is the roll angle (rotation around the y-axis). The dynamical model adopted in this contribution is mainly based on the model

proposed in [20], simplified to reduce the computational complexity and to make easier the design of the controller. As described in Figure 14, each of the four motors M_1, M_2, M_3, M_4 generate four thrust forces f_1, f_2, f_3, f_4, and four torques τ_1, τ_2, τ_3, τ_4, respectively, which are adjusted by manipulating the voltages applied to the motors. We use the linear equations obtained in [4]

$$f_i = \frac{9.81(22.5V_{Mi} - 9.7)}{1000},\ i = 1,\ldots,4$$

approximated to consider all motors identical, to express the relationships between motors thrustes and their voltages. A total force f and three torques τ_θ, τ_ϕ, τ_ψ around their corresponding axes are generated

$$f = f_1 + f_2 + f_3 + f_4 \tag{45a}$$
$$\tau_\theta = (f_2 - f_4)l \tag{45b}$$
$$\tau_\phi = (f_3 - f_1)l \tag{45c}$$
$$\tau_\psi = \sum_{i=1}^{4} \tau_i \tag{45d}$$

where l is the distance between each motor and the center of gravity of the vehicle, that allow changing the position and orientation coordinates of the quadcopter freely in the three-dimensional space. The dynamical system is described by following equations

$$\ddot{x} = (-f\sin\theta - \beta\dot{x})\frac{1}{m} \tag{46a}$$
$$\ddot{y} = (f\cos\theta\sin\phi - \beta\dot{y})\frac{1}{m} \tag{46b}$$
$$\ddot{z} = -g + (f\cos\theta\cos\phi - \beta\dot{z})\frac{1}{m} \tag{46c}$$
$$\ddot{\theta} = \frac{\tau_\theta}{I_{xx}} \tag{46d}$$
$$\ddot{\phi} = \frac{\tau_\phi}{I_{yy}} \tag{46e}$$
$$\ddot{\psi} = \frac{l}{I_{zz}}(-f_1 + f_2 - f_3 + f_4) \tag{46f}$$

where the damping factor β takes into account realistic friction effects, m denotes the mass of the vehicle, and I_{xx}, I_{yy}, I_{zz} are the moments of inertia around the body frame axes x, y, z. The nonlinear dynamical model has twelve states (six positions and six velocities) and four inputs (the motors voltages V_{Mi}), largely coupled through the nonlinear relations in (46). The parameters used in this work are reported in Table 3.

Table 3: Parameters of quadcopter model

m [kg]	l [m]	β [Ns/m]	I_{xx} [Nms^2]	I_{yy} [Nms^2]	I_{zz} [Nms^2]
0.408	0.136	0.2	0.0047	0.0047	0.0089

4.3 Linear MPC for Stabilization

In order to design a linear MPC controller for the quadcopter air vehicle, we first linearize the nonlinear dynamical model (46) around an equilibrium condition of hovering. The resulting linear continuous-

time state-space system is converted to discrete-time with sampling time T_s

$$\begin{cases} \xi_L(k+1) &= A\xi_L(k) + Bu_L(k) \\ y_L(k) &= \xi_L(k) \end{cases} \tag{47}$$

where $\xi_L(k) = [\theta, \phi, \psi, x, y, z, \dot{\theta}, \dot{\phi}, \dot{\psi}, \dot{x}, \dot{y}, \dot{z}, z_{\mathrm{I}}]' \in \mathbb{R}^{13}$ is the state vector, $u_L(k) = [u, \tilde{\tau}_\theta, \tilde{\tau}_\phi, \tilde{\tau}_\psi]' \in \mathbb{R}^4$ is the input vector, $y_L(k) \in \mathbb{R}^{13}$ is the output vector (that we assume completely measured or estimated), and A, B, C, D are matrices of suitable dimensions obtained by the linearization process. The additional state, $z_{\mathrm{I}} = \int (z - z_d)$ is included to provide integral action on the altitude z, so that offset-free tracking of the desired setpoint z_d is guaranteed in steady-state. The integral action is mainly due to counteract effect of the gravity force acting against the force developed by the collective input u.

The linear MPC formulation of the Model Predictive Control Toolbox for Matlab [21] is used here, where the MPC control action at time k is obtained by solving the optimization problem

$$\min_{\begin{bmatrix} \Delta u_L(k|k) \\ \vdots \\ \Delta u_L(m-1+k|k) \\ \varepsilon \end{bmatrix}} \sum_{i=0}^{N_L-1} \left(\sum_{j=1}^{n_y} |w_j^y [y_{Lj}(k+i+1|k) - r_{Lj}(k+i+1)]|^2 + \sum_{j=1}^{n_u} |w_j^{\Delta u} \Delta u_{Lj}(k+i|k)|^2 + \rho_\varepsilon \varepsilon^2 \right) \tag{48a}$$

$$\begin{aligned} \text{s.t.} \quad & u_{Lj}^{\min} \le u_{Lj}(k+i|k) \le u_{Lj}^{\max}, \ j = 1, \ldots, n_u \\ & y_{Lj}^{\min} - \varepsilon V_{Lj}^{\mathrm{y,min}} \le y_{Lj}(k+i+1|k) \le y_{Lj}^{\max} + \varepsilon V_{Lj}^{\mathrm{y,max}} \quad j = 1, \ldots, n_y \\ & \Delta u(k+h|k) = 0, \ h = N_{Lu}, \ldots, N_L \\ & \varepsilon \ge 0 \end{aligned} \tag{48b}$$

for all $i = 0, \ldots, N_L - 1$, with respect to the sequence of input increments $\{\Delta u(k|k), \ldots, \Delta u(N_{Lu} - 1 + k|k)\}$ and the slack variable ε. In (48a) the subscript "$()_j$" denotes the jth component of a vector, "$(k+i|k)$" denotes the value predicted for time $k+i$ based on the information available at time k, $r_L(k)$ is the current sample of the output reference, $V^{\mathrm{y,min}}$, $V^{\mathrm{y,max}}$ are constant vectors with nonnegative entries which represent the designer's concern for relaxing the corresponding output constraint, $n_y = 13$ is the number of outputs, $n_u = 4$ is the number of inputs. The linear MPC controller sets $u(k) = u(k-1) + \Delta u^*(k|k)$, where $\Delta u^*(k|k)$ is the first element of the optimal sequence.

4.3.1 Linear MPC tuning and validation

The linear MPC controller is tuned according to the following setup. Regarding input variables, we set $u_{Lj}^{\min} = -6$ Nm, $u_{Lj}^{\max} = 6$ Nm, $j = 1, 2, 3$, $u_{L4}^{\min} = -6$ N, $u_{L4}^{\max} = 6$ N, $w_{i,j}^{\Delta u} = 0.1$, $\forall j = 1, \ldots, 4$, $i = 0, \ldots, N_L - 1$. For output variables we set a lower bound $y_{L6}^{\min} = 0$ on altitude z, and upper and lower bounds $y_{L1-2}^{\max} = -y_{L1-2}^{\min} = \frac{\pi}{12}$ on pitch θ and roll ϕ angles. The output weights are $w_j^y = 1$, $j \in \{4, 5, 11, 12\}$, on x, y, $\dot{x}$, $\dot{y}$, respectively, and $w_j^y = 10$ on the remaining output variables. The chosen set of weights ensures a good trade-off between fast system response and actuation energy. The prediction horizon is $N_L = 20$, the control horizon is $N_{Lu} = 3$, which, together with the choice of weights, allow obtaining a good compromise between tracking performance, robustness, and limited computational complexity. The sampling time of the controller is $T_s = \frac{1}{14}$ s. The remaining parameters $V^{\mathrm{y,min}}$, $V^{\mathrm{y,max}}$, ρ_ε are defaulted by the Model Predictive Control Toolbox [25].

The closed-loop performance is tested by simulating the nonlinear quadcopter model (46) under the effect of the linear MPC controller (48) using Simulink and the Model Predictive Control Toolbox. Additional blocks were designed to generate reference signals from either the computer keyboard or an external four-axes joystick. Moreover, the numerical signals are connected to an animation block based on FlightGear [55] for 3D visual inspection on the regulation performance, connected through a

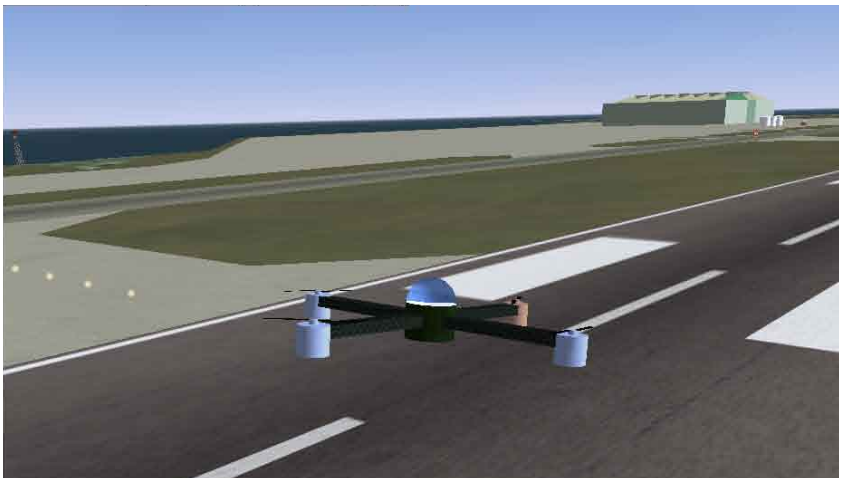


Figure 15: Visualization of quadcopter dynamics in FlightGear (chase view)

graphical user interface designed for easy human-machine interaction. FlightGear allows one to very intuitively move the quadcopter around the virtual world to any desired direction by sending desired set-point commands directly from the keyboard or joystick, and assess closed-loop performance visually. An animation movie can be retrieved at `http://www.dii.unisi.it/hybrid/aerospace/quadcopter`. Due to different angle representation systems, the following coordinate transformation

$$\begin{bmatrix} \theta' \\ \phi' \\ \psi' \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix}$$

is used to map the signals (θ, ϕ, ψ) generated by model (46) into the coordinates (θ', ϕ', ψ') defining the angle coordinates of the quadcopter in the virtual environment:

Figure 16 shows simulation results obtained by tracking a generic reference signal. The corresponding input plots are shown in Figure 17, where the thick solid line represents the collective activity u. Note that u is nonzero at steady-state due to the task of keeping the quadcopter in hovering. The other lines represent, respectively, the actuations on pitch $\tilde{\tau}_\theta$, roll $\tilde{\tau}_\phi$, and yaw $\tilde{\tau}_\psi$ angles.

Note that because of the constraints imposed on θ and ϕ, the nonlinear dynamics of the vehicle is maintained close enough to the linearized model used in the MPC design. As performance is rather satisfactory, the possible use of multiple MPC controllers based on models linearized at different conditions has not been explored here.

The results were obtained on a Core 2 Duo running Matlab R2008a and the MPC Toolbox under MS Windows. The average CPU time to compute the MPC control action is about 13 ms per time step, which is about 1/6 of the sampling time T_s. No attempt was done to speed up computations, such as using fast on-line MPC implementations [53] or explicit off-line MPC solutions [54].

4.3.2 Hybrid MPC for collision avoidance

The proposed approach consists of constructing an abstract hybrid model of the controlled aerial vehicle and of the surrounding obstacles, and then use a hybrid MPC strategy for on-line generation of the desired positions. The closed-loop dynamics composed by the quadcopter and the linear MPC controller can be very roughly approximated as a 3-by-3 diagonal linear dynamical system, whose inputs are (x_d, y_d, z_d) and whose outputs are (x, y, z). Accordingly, the dynamics is formulated

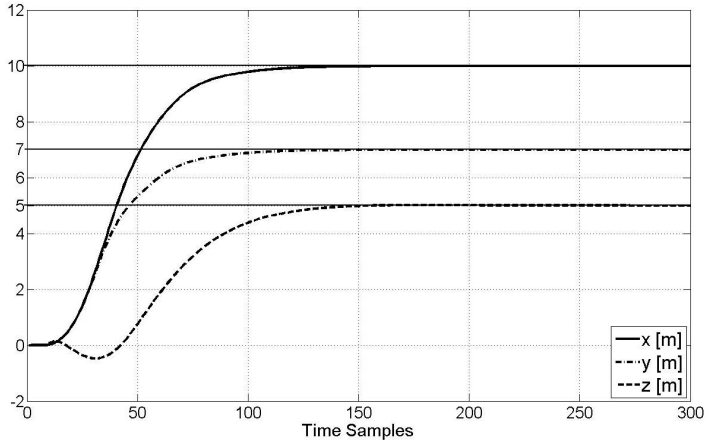


Figure 16: Linear MPC for tracking generic position references

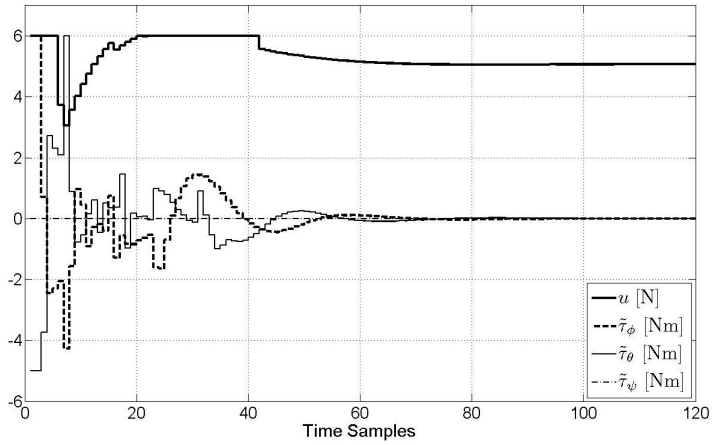


Figure 17: Linear MPC: commanded inputs

in discrete-time as

$$\begin{cases} x(k+1) &= \alpha_{1x}x(k)+\beta_{1x}(x_d(k)+\Delta x_d(k)) \\ y(k+1) &= \alpha_{1y}y(k)+\beta_{1y}(y_d(k)+\Delta y_d(k)) \\ z(k+1) &= \alpha_{1z}z(k)+\beta_{1z}(z_d(k)+\Delta z_d(k)) \end{cases} \tag{49}$$

where $\Delta(\cdot)_d(k)$ is the increment of desired $(\cdot)$-coordinate commanded at time kT_{hyb}, and $T_{\text{hyb}} > T_s$ is the sampling time of the hybrid MPC controller. Input increments $\Delta x_d(k)$, $\Delta y_d(k)$, $\Delta z_d(k)$ are upper and lower bounded by a quantity Δ

$$-\Delta \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \leq \begin{bmatrix} \Delta x_d(k) \\ \Delta y_d(k) \\ \Delta z_d(k) \end{bmatrix} \leq \Delta \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{50}$$

Constraint (50) is a tuning knob of the hybrid MPC controller, as it allows one to directly control the speed of maneuver of the quadcopter by imposing constraints on the reference derivatives $\left\| \begin{bmatrix} \dot{x}_d(t) \\ \dot{y}_d(t) \\ \dot{z}_d(t) \end{bmatrix} \right\|_\infty \leq \Delta \cdot T_{\text{hyb}}$.

Obstacles are modeled as polyhedral sets. For minimizing complexity, the ith obstacle, $i = 1,\ldots,M$, is modeled as the tetrahedron

$$A_{\text{obs}} k_i \begin{bmatrix} x(k)-x_i(k) \\ y(k)-y_i(k) \\ z(k)-z_i(k) \end{bmatrix} \leq B_{\text{obs}} \tag{51}$$

where $A_{\text{obs}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq B_{\text{obs}}$ is a fixed hyperplane representation of a reference tetrahedron, k_i is a fixed scaling factor, and $\begin{bmatrix} x_i(k) \\ y_i(k) \\ z_i(k) \end{bmatrix}$ is a reference point of the obstacle. In this work we choose A_{obs}, B_{obs} such that the corresponding polyhedron is the convex hull of vectors $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 5/2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 5/2 \\ 0 \end{bmatrix}, \begin{bmatrix} 5/6 \\ 5/6 \\ 5/2 \end{bmatrix}$, which makes the reference point $\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ its vertex with smallest coordinates.

Equation (51) can be rewritten as

$$A_{\text{obs}} k_i \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} \leq C_{\text{obs}}(k) \tag{52}$$

where $C_{\text{obs}}(k) = B_{\text{obs}} + A_{\text{obs}} k_i \begin{bmatrix} x_i(k) \\ y_i(k) \\ z_i(k) \end{bmatrix} \in \mathbb{R}^4$ is a quantity that may vary in real-time. Although we model here the *predicted* evolution of C_{obs} as

$$C_{\text{obs}}(k+h+1) = C_{\text{obs}}(k+h) \tag{53}$$

non-constant dynamics may be used as well if obstacle velocities and/or accelerations were estimated.

Finally, to represent the obstacle avoidance constraint, define the following binary variables $\delta_{ij} \in \{0,1\}$, $i = 1,\ldots,M$, $j = 1,\ldots,4$

$$[\delta_{ij}(k) = 1] \leftrightarrow [A^j_{\text{obs}} k_i \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} \leq C^j_{\text{obs}}(k)] \tag{54}$$

where ^{j} denotes the jth row (component) of a matrix (vector). The following logical constraints

$$\bigvee_{j=1}^{4} \neg\delta_{ij}(k) = 1, \ \forall i = 1,\ldots,M \tag{55}$$

impose that at least one linear inequality in (52) is violated, therefore enforcing the quadcopter position $\begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix}$ to lie outside each obstacle.

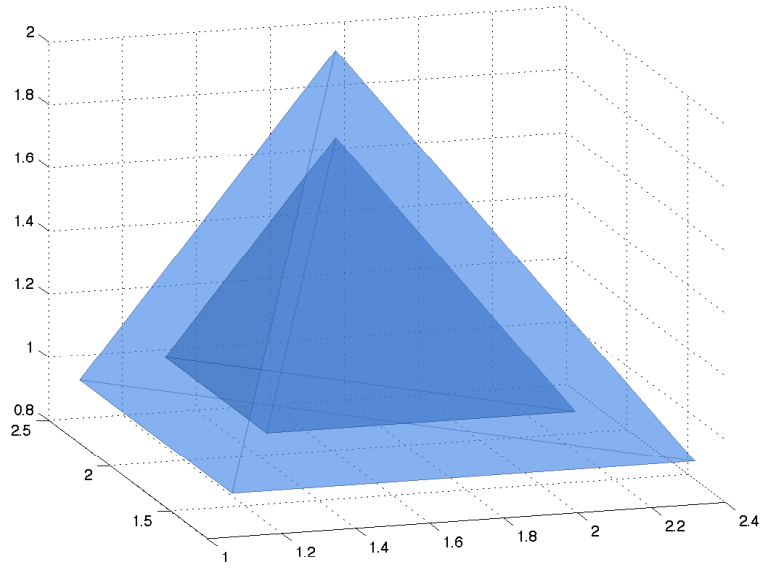


Figure 18: Example of "safety area" surrounding an obstacle

Differently from [7], we want to penalize here that vehicles fly too close to obstacles. To this end, each obstacle seen by a UAV is surrounded by a "safety area", represented by a larger tetrahedron containing the one defined by (52) (see Figure 18). Also for such safety areas we define binary variables $\delta_{\ell j}(k)$, $\ell = 1,\dots,S$, as in (54), where S is the number of safety areas ($S \leq M$ in case of one UAV flying in an area with M obstacles), but without imposing the logical constraints as in (55) to allow the UAV entering those areas. Such an event is penalized by defining for each "safety area" a variable $\gamma_\ell(k)$, $\ell = 1,\dots,S$

$$\gamma_\ell(k) = \begin{cases} 1 & \text{if } \bigwedge_{j=1}^{4} \delta_{\ell j}(k) = 1 \quad \forall \ell = 1,\dots,S \\ 0 & \text{otherwise} \end{cases} \tag{56}$$

that the controller will try to keep at zero. In this way, the vehicle will tend to avoid passing through the safety areas that have been set around the obstacles. Note that although γ_ℓ can only assume values 0 and 1, we treat it as a real variable to ease the hybrid MPC computations that will be defined in the next paragraphs. The sampling time T_{hyb} must be chosen large enough to neglect fast transient dynamics, so that the lower and upper MPC designs can be conveniently decoupled. On the other hand, the obstacle avoidance constraint (55) is only imposed at multiples of T_{hyb}, and hence an excessively large sampling time may lead to trajectories that go through the obstacles during intersampling intervals.

4.4 Simulation results: Hierarchical MPC of a Single UAV

To test the behavior of the overall system we cascade the linear MPC controller described in Section 4.3 with the hybrid MPC controller designed in this section, according to the hierarchical scheme of Figure 13. The simulation consists of avoiding three obstacles (tetrahedra) of different dimensions, placed along the path between the quadcopter and the target point (see Figure 19). The following parameters are employed: $\alpha_{1x} = \alpha_{1y} = \alpha_{1z} = 0.6$, $\beta_{1x} = \beta_{1y} = \beta_{1z} = 0.4$ for the approximation of the lower level dynamics; N_H=10 (prediction horizon), T_{hyb}=2 s, and $\Delta = \frac{1}{5}T_{hyb}$; $k_1 = k_3 = 1$, $k_2 = \frac{2}{3}$ are the

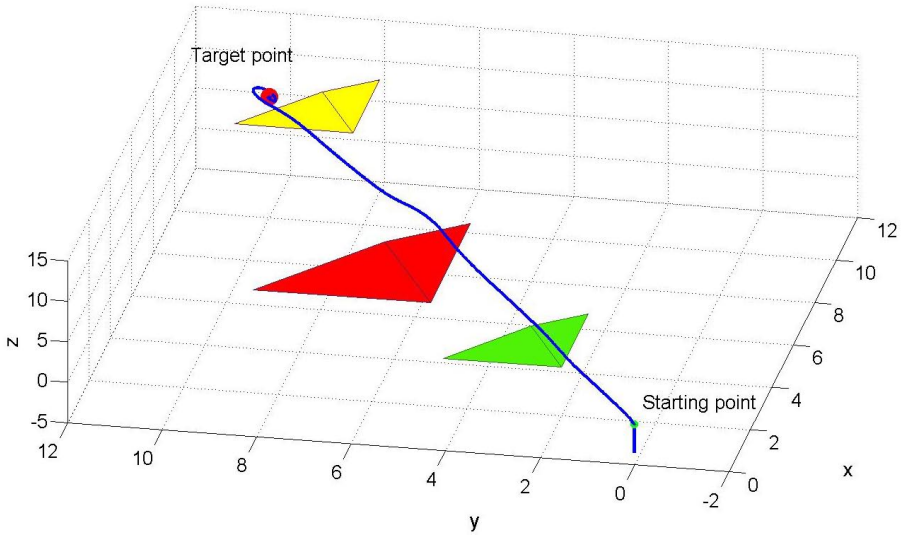


Figure 19: Obstacle avoidance maneuvers commanded by the hybrid MPC controller

scaling factors for the tetrahedra; $Q = I_{3\times 3}$ and $R = 0.1 \cdot I_{3\times 3}$ are the weight matrices; the initial position is $x(0) = y(0) = z(0)$=0 and the target point is located at $x_t = y_t = z_t$=10.

The overall performance is quite satisfactory: The trajectory generated on-line circumvents the obstacles without collisions (see Figure 19), and finally the quadcopter settles at the target point (see Figure 20). Note that in the first transient the altitude z first drops by about 3 m due to the effect of gravity.

In the simulations we assumed that the positions of the obstacles were known at each sample step. In more realistic applications with several obstacles it may be enough to only know the locations of the obstacles which are closest to the vehicle, for a threefold reason. First, because of the finite-horizon formulation, remote obstacles will not affect the optimal solution, and may be safely ignored to limit the complexity of the optimization model. Second, because of the receding horizon mechanism, the optimal plan is continuously updated, which allows one to change the maneuvers to avoid new obstacles. Third, in a practical context obstacles may be moving in space, and since such dynamics is not modeled here, taking into account remote obstacles in their current position has a weak significance. The number M of obstacles to be taken into account in the hybrid model clearly depends on the density of the obstacles and the speed of the vehicle. Note that, depending on the sensor system on board of the vehicle, it may be even impossible to measure the position of remote obstacles.

The average CPU time to compute the hybrid MPC action for set-point generation is about 17 ms per time step on the same platform used for linear MPC, using the mixed-integer quadratic programming solver of CPLEX 11.2 [26].

4.4.1 Hybrid model for UAV navigation in formation

The hierachical structure described above for one quadcopter is extended to coordinate a formation of V cooperating UAVs, $V > 1$. We use a leader-follower approach with decentralized scheme to manage the formation; according to such an approach, one of the vehicles (Leader) is chosen to direct the formation, following a prescribed path, and all the other aircraft (Followers) are expected to maintain a constant relative distance reference from the Leader. Each UAV is equipped with its own hybrid MPC controller and takes decisions autonomously, measuring its own state and the

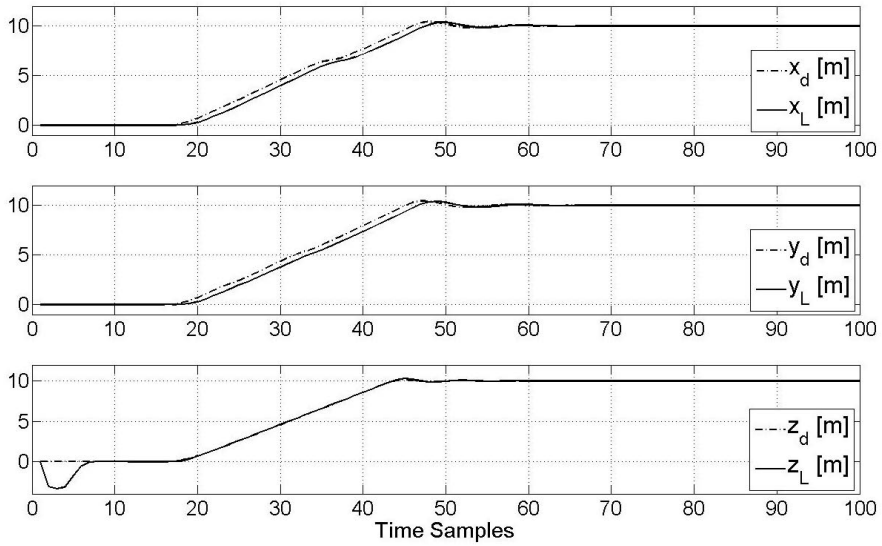


Figure 20: Position and reference signals

positions of the neighboring vehicles and obstacles, planning its own path with obstacle avoidance. The whole formation must be capable of reconfiguring, making decisions (for instance, changing relative distances to modify the formation shape), and achieving mission goals (e.g., target tracking). Moreover, each follower must avoid collision with the other one. In this case the other vehicles in the formation are treated as obstacles, so that M accounts now for both real obstacles and other (neighboring) vehicles. We take account the real dimensions of the vehicles and avoid that they move too close to each other and to the obstacles to minimize risk of collision.

The overall hybrid dynamical model is obtained by collecting (49), (53), (54), (55), (56). These are modeled through the modeling language HYSDEL [22] and converted automatically by the Hybrid Toolbox [23] into a mixed logical dynamical (MLD) system form [24]

$$\xi_H(k+1) = A\xi_H(k) + B_1\Delta u_H(k) + B_2\delta(k) + B_3\gamma(k) \tag{57a}$$

$$y_H(k) = C\xi_H(k) + D_1\Delta u_H(k) + D_2\delta(k) + D_3\gamma(k) \tag{57b}$$

$$E_2\delta(k) + E_3\gamma(k) \leq E_1\Delta u_H(k) + E_4\xi_H(k) + E_5, \tag{57c}$$

where $\xi_H(k) = [x(k)\ y(k)\ z(k)\ C_1'(k)\ \ldots C_M'(k)\ x_d(k-1)\ y_d(k-1)\ z_d(k-1)]' \in \mathbb{R}^{6+4M}$ is the state vector, $\Delta u_H(k) = [\Delta x_d(k)\ \Delta y_d(k)\ \Delta z_d(k)]' \in \mathbb{R}^3$ is the input vector, $y_H(k) = [x(k)\ y(k)\ z(k)] \in \mathbb{R}^3$ is the output vector, $\delta(k) \in \{0,1\}^{4M+4S}$ and $\gamma(k) \in \mathbb{R}^S$ are respectively the vector of binary (defined in (54)) and continuous auxiliary variables. The inequalities (57c) include a big-M representation [24] of (54) and a polyhedral inequality representation of (55). Matrices A, B_1, C, E_1, E_2, E_4, E_5 have suitable dimensions and are generated by the HYSDEL compiler. In order to design a hybrid MPC controller, consider the finite-time optimal control problem

$$\min_{\{\Delta u(k)\}_{k=0}^{N_H-1}} \sum_{j=0}^{N_H-1} (y_H(k+j+1) - y_{Ht})' Q_y (y_H(k+j+1) - y_{Ht}) + \Delta u_H'(k+j) R \Delta u_H(k+j) + \gamma_H'(k+j) Q_\gamma \gamma_H(k+j) \tag{58a}$$

s.t. MLD dynamics (57) (58b)

constraints (50) (58c)

where N_H is the prediction horizon, $y_{Ht} = [\bar{x}_t\ \bar{y}_t\ \bar{z}_t]'$ is the desired position (e.g., the position of the

target or of the leading vehicle), γ_H are the auxiliary continuous variables with reference γ_{Ht}, $Q_y \geq 0$, $R > 0 \in \mathbb{R}^{3\times3}$ and $Q_\gamma \geq 0 \in \mathbb{R}^{S\times S}$ are weight matrices.

The MLD hybrid dynamics (57) has the advantage of making the optimal control problem (58) solvable by mixed-integer quadratic programming (MIQP). At each sample step k, given the current reference values $y_H(k)$ and the current state $\xi(t)$, Problem (58) is solved to get the first optimal input sample $\Delta u_H^*(k)$, which is commanded as the increment of desired set-point (x_d, y_d, z_d) to the linear MPC controller at the lower hierarchical level.

As an alternative, to manage the formation in this approach we use a *centralized scheme* consisting of a single hybrid MPC controller based on a macromodel (including the dynamics and obstacles of the entire formation) generates the references for all UAVs, that are passed to the individual linear MPC controllers designed for stabilization. Clearly, this centralized approach has the drawback of needing the solution of a single MIQP optimization problem for the entire team, which typically requires significant computation. We will compare the performance of the centralized and decentralized hybrid navigation schemes in the next section.

4.5 Potential Fields Method

For comparing the hybrid MPC approach with other existing navigation schemes, we consider the 3D potential fields method proposed in [15] for a formation of helicopters, adapted for the formation of quadcopters defined earlier. In this case, tetrahedra are replaced by spherical obstacles. This approach generates a potential field for each UAV depending on formation pattern, desired and actual position, and obstacle positions, for collision and obstacle avoidance and target tracking. The total field

$$F^{tot} = F_t + F_{ca}^{tot} + F_{oa}^{tot} \tag{59}$$

for each vehicle, used for generating the references for the lower-level stabilizing linear MPC, is composed by the three components F_t (target tracking), F_{ca}^{tot} (collision avoidance), and F_{oa}^{tot} (obstacle avoidance). The contribution for tracking the position of the target is $F_t = K_t(t-p)$, where K_t is the gain for target potential, t is the target position and p the vehicle position. For vehicle Leader, the target is a given fixed position p_t, for the Followers is a given distance p_d from the Leader. To avoid collision between vehicles or with obstacles, a safety space is defined around each vehicle, defined as a sphere with positive radius r_{sav}. The additional field component for vehicle i whose safety sphere is invaded by vehicle j is defined by

$$F_{ca}(k) = \begin{cases} \left(\frac{K_{ca} r_{sav}}{||d_{ji}||} - K_{ca}\right) \frac{d_{ji}}{||d_{ji}||} & \text{if } ||d_{ji}|| \leq r_{sav} \\ 0 & \text{otherwise} \end{cases} \tag{60}$$

where K_{ca} is the gain for collision avoidance and d_{ji}, $i \neq j$, is the distance between vehicles. The total amount of the collision avoidance term is given by

$$F_{ca}^{tot} = \sum_{j=1}^{N} F_{ca}^{ij} \quad \text{for } i \neq j \tag{61}$$

Eqs. (60) and (61) change to

$$F_{oa}(k) = \begin{cases} \left(\frac{K_{oa}}{||d_{ki}||} - \frac{K_{oa}}{r_{sav}}\right) \frac{d_{ki}}{||d_{ki}||} & \text{if } ||d_{ki}|| \leq r_{sav} \\ 0 & \text{otherwise} \end{cases} \tag{62}$$

$$F_{oa}^{tot} = \sum_{k=1}^{M} F_{oa}^{ik} \quad \text{for } i \neq k \tag{63}$$

for obstacle avoidance. Here, d_{ki} represents the distance between vehicle i and the center of obstacle k, and K_{oa} is the gain for obstacle avoidance. In both cases, to increase performance, r_{sav} is chosen dynamically, depending on the vehicle's velocity $\dot{p}$:

$$r_{sav} = r_{sav}^{min} + K_{sav}||\dot{p}|| \tag{64}$$

using K_{sav} as a gain and r_{sav}^{min} as the minimum distance for a save avoidance. For collision avoidance $r_{sav}^{min} = 2r_q$, for obstacle avoidance $r_{sav}^{min} = r_{obs}(k) + r_q$, where r_q and $r_{obs}(k)$ are respectively the radius of quadcopter and of the obstacle.
Finally, the reference trajectory $p_{i,r}$ for vehicle i is given by

$$p_{i,r} = p_i + F_i^{tot} \tag{65}$$

where p_i is the position of vehicle.

4.6 Simulation results

The overall system is tested by cascading the linear MPC controller with the hybrid MPC designed for navigation, according to the hierarchical scheme of Figure 13. The simulation consists of avoiding four obstacles (tetrahedra) of different dimensions, placed along the path between the quadcopters and the target points.

The linear MPC controller is tuned according to the following setup. Regarding input variables, we set $u_{Lj}^{\min} = 0$ V, $u_{Lj}^{\max} = 11.1$ V, $w_{i,j}^{\Delta u} = 0.1$, $\forall j = 1,\ldots,4$, $i = 0,\ldots,N_L - 1$. For output variables we set a lower bound $z_L^{\min} = 0$ on altitude, and upper and lower bounds $y_{L1-2}^{\max} = -y_{L1-2}^{\min} = \frac{\pi}{6}$ on pitch θ and roll ϕ angles. The output weights are $w_j^y = 0$, $j \in \{1,2\}$, on θ and ϕ, $w_j^y = 1$, $j \in \{7,8\}$, on $\dot{\theta}$ and $\dot{\phi}$, and $w_j^y = 10$ on the remaining output variables. The chosen set of weights ensures a good trade-off between fast system response and actuation energy. The prediction horizon is $N_L = 20$, the control horizon is $N_{Lu} = 3$, which, together with the choice of weights, allow obtaining a good compromise between tracking performance, robustness, and limited computational complexity. The sampling time of the controller is $T_s = \frac{1}{14}$ s. The remaining parameters $V^{\mathrm{y,min}}$, $V^{\mathrm{y,max}}$, ρ_ε are defaulted by the Model Predictive Control Toolbox [25].

The following parameters are employed for hybrid MPC: $\alpha_{1x} = \alpha_{1y} = \alpha_{1z} = 0.6$, $\beta_{1x} = \beta_{1y} = \beta_{1z} = 0.4$ for the approximation of the lower level dynamics; N_H=10 (prediction horizon), T_{hyb}=1.5 s, and $\Delta = \frac{1}{5}T_{hyb}$; $k_1 = k_4 = \frac{5}{11}$, $k_2 = \frac{10}{27}$, $k_3 = \frac{5}{16}$ are the scaling factors for tetrahedra (obstacles), and for each of them we have the corresponding "safety area" with scaling factor $k_s = \frac{k_i}{1+k_i 15/20}$, $i = 1,\ldots,4$; $Q_y = 0.01 \cdot I_{3\times 3}$, $Q_\gamma = 10 \cdot I_{4\times 4}$ and $R = 0.1 \cdot I_{3\times 3}$ are the weight matrices.

The initial positions of the UAVs are $p_L(0) \triangleq [x_L(0)\ y_L(0)\ z_L(0)]' = [0\ 0\ 0]'$ for the leader, $p_{F1}(0) \triangleq [x_{F1}(0)\ y_{F1}(0)\ z_{F1}(0)]' = [-2\ -2\ 0]'$, and $p_{F2}(0) \triangleq [x_{F2}(0)\ y_{F2}(0)\ z_{F2}(0)]' = [-4\ -4\ 0]'$ for the followers; the target point for the Leader is located at $[\bar{x}_t\ \bar{y}_t\ \bar{z}_t]_L = [35\ 35\ 6]$; the followers take off with a delay of 2.5 and 5 seconds respectively, and should follow the leader at given distances $[\bar{x}_t\ \bar{y}_t\ \bar{z}_t]'_{F1} = p_L - p_{d1}$, $[\bar{x}_t\ \bar{y}_t\ \bar{z}_t]'_{F2} = p_{d1} = [6\ 1\ 0]'$, and $p_L - p_{d2}$, $p_{d2} = [1\ 6\ 0]'$. The results were obtained on a Core 2 Duo running Matlab R2009b, the Model Predictive Control and the Hybrid Toolbox under MS Windows, using the MIQP solver of CPLEX 11.1 [26].

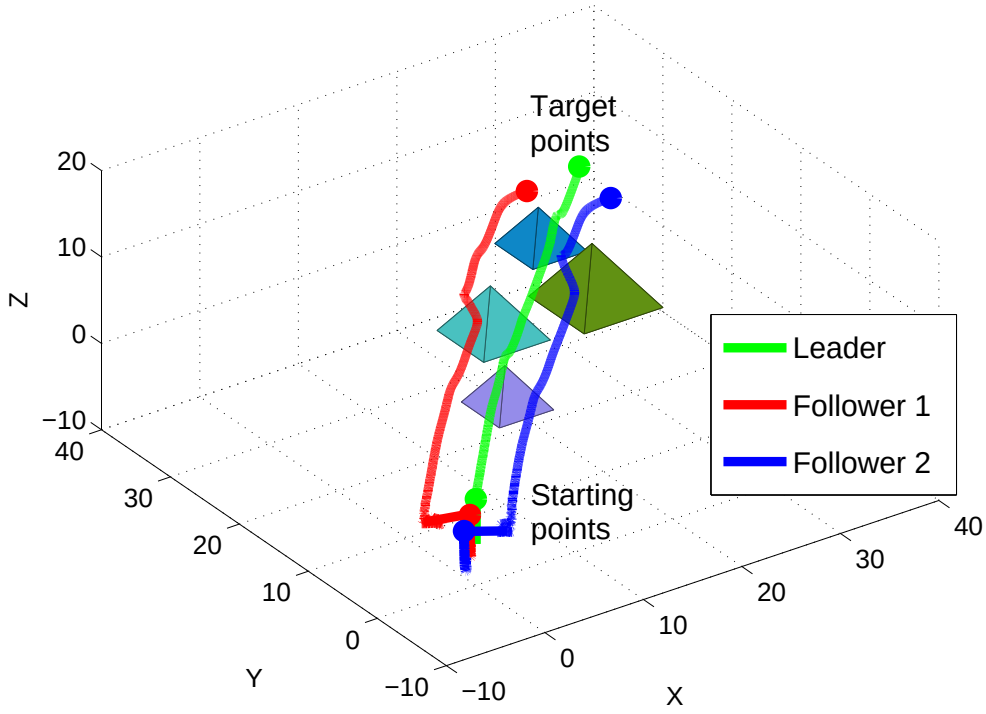


Figure 21: Trajectories of formation with obstacle avoidance, hybrid approach

4.6.1 Decentralized hierarchical hybrid + linear MPC

The trajectories obtained by using the decentralized hierarchical hybrid + linear MPC are shown in Figure 21. The performance is quite satisfactory: trajectories circumvent obstacles without collisions and finally the quadcopters settle at the target point, while maintaining the desired formation as much as the obstacles allow it. The lower-level tracking of references achieved by using the linear MPC controller is shown in Figure 22 for the leader.

4.6.2 Centralized hybrid MPC + decentralized linear MPC

Next, we compare the results with the trajectories obtained by a single centralized hybrid MPC planner cascaded by the decentralized set of linear MPC controllers for stabilization. The performance is very similar, but with an increment of computational complexity: while with the decentralized scheme the average CPU time to compute the hybrid MPC action for set-point generation is about 0.073 s per time step (T_{hyb}=1.5 s), with the centralized scheme is about 0.466 s.

4.6.3 Comparison with potential fields method

In order to assess further the performance of the proposed hybrid MPC approach to formation flying control, we compare it with the potential fields method described in Section 4.5 to generate on-line the desired positions, while maintaining the lower-level linear MPC controllers for stabilization and reference tracking. In this case the tetrahedra are replaced by four spherical obstacles. The parameters used for simulation are reported in Table 4.

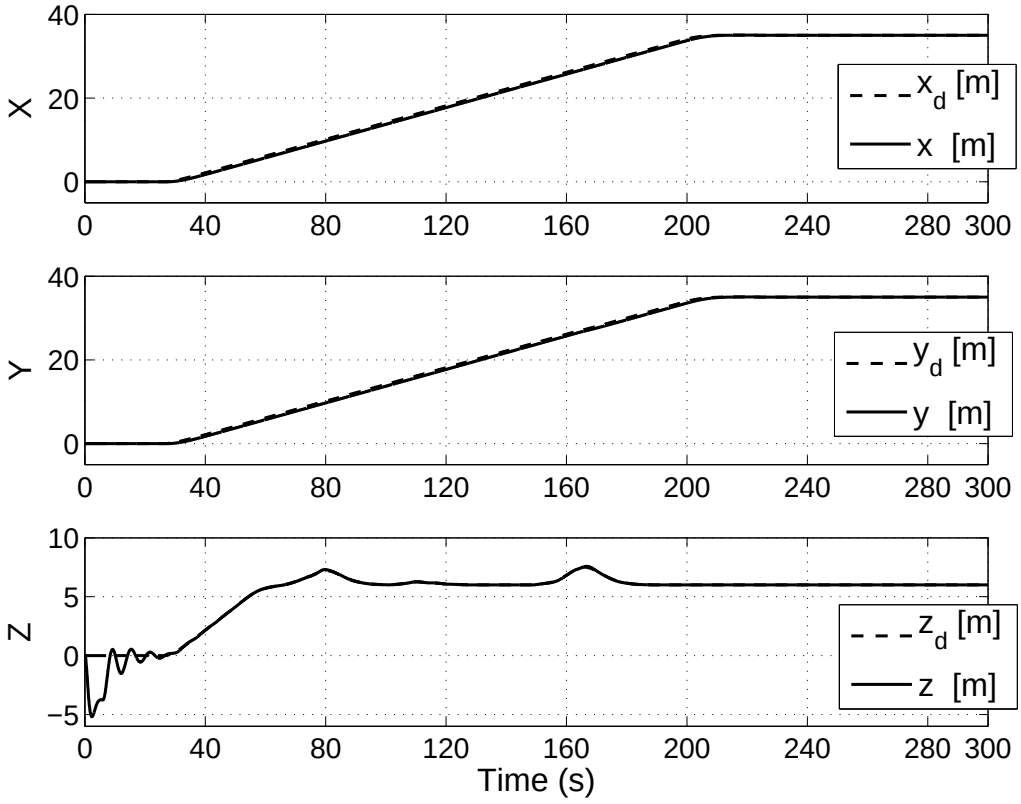


Figure 22: Leader's position and reference signals generated by the decentralized hybrid MPC approach, tracked by linear MPC

Table 4: Parameters for the potential fields method

K_t	0.15	Gain for target
r_q	1.3	Quadcopter radius
K_{ca}	10	Gain for collision avoidance
K_{oa}	40	Gain for obstacle avoidance
K_{sav}	2	Gain for a save avoidance
$r_{obs}(1)$	2.5	Radius of obstacle 1
$r_{obs}(2)$	3	Radius of obstacle 2
$r_{obs}(3)$	3.5	Radius of obstacle 3
$r_{obs}(4)$	2.5	Radius of obstacle 4

The use of the potential field reduces the computational complexity thanks to the absence of the overall hybrid model; the CPU time to calculate the desired positions is of the order of milliseconds. Even if the performance of obstacle avoidance (see Figure 23) and reference tracking (see Figure 24) is satisfactory, it takes a longer time to reach the target. Moreover, it is necessary to impose an upper bound $z_L^{\max} = 22.4$ on altitude in the linear MPC formulation, to avoid undesired overshoots due to fast variations of the references. Finally, to make a quantitative comparison of the different control strategies, we show in Table 5 different performance indices and compare them on the different navigation algorithms: decentralized hybrid MPC, centralized hybrid MPC, and potential fields method. We consider the following three indices defined on the simulation interval 25÷300 s (i.e., 350÷4200 samples)

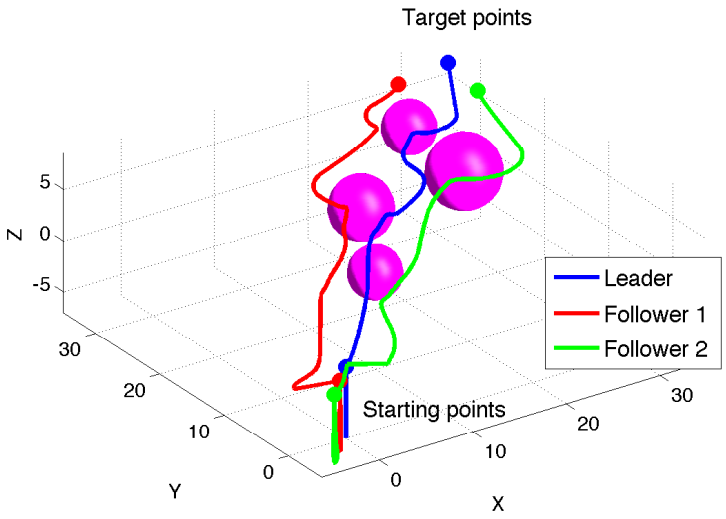


Figure 23: Trajectories of formation with obstacle avoidance, potential fields method

Table 5: Comparison of different approaches

	J_{tt}	J_{fpt}	J_{u}
centralized hybrid MPC	1	1	1
decentralized hybrid MPC	-0.09%	-0.51%	-0.03%
potential fields	+210.32%	+294,30%	+212.75%

$$
\begin{aligned}
J_{\text{tt}} &= \sum_{k=250}^{4200} \|p_L(k) - p_t\|_2^2 \\
J_{\text{fpt}} &= \sum_{k=250}^{4200} \|p_L(k) - p_{F1}(k) - p_{d1}\|_2^2 + \|p_L(k) - p_{F2}(k) - p_{d2}\|_2^2 \\
J_{\text{u}} &= \sum_{k=250}^{4200} \|u(k) - u(k-1)\|_1
\end{aligned}
$$

where J_{tt} represents the *target tracking* Integral Square Error (ISE) index, J_{fpt} the *formation pattern tracking* ISE index, and J_{u} the *absolute derivative of input signals* (IADU) index for checking the smoothness of control signals [6]. The indices are normalized with respect to the values obtained using centralized hybrid MPC. It is apparent that the hybrid MPC approach outperforms the potential fields method. Note also that the decentralized and the centralized hybrid MPC schemes have almost equal performance, actually the decentralized scheme is even slightly better. This maybe due to the receding-horizon mechanism of MPC and to the fact that the MPC weights were tuned for the decentralized approach and used for both schemes.

In the simulations we assumed that the positions of the obstacles are known at each sample step. In more realistic applications with several obstacles it may be enough to only know the locations of the obstacles which are closest to the vehicle, for a threefold reason. First, because of the finite-horizon formulation, remote obstacles will not affect the optimal solution, and may be safely ignored to limit the complexity of the optimization model. Second, because of the receding horizon mechanism, the optimal plan is continuously updated, which allows one to change the maneuvers to avoid new obstacles. Third, in a practical context obstacles may be moving in space, and since such dynamics is not

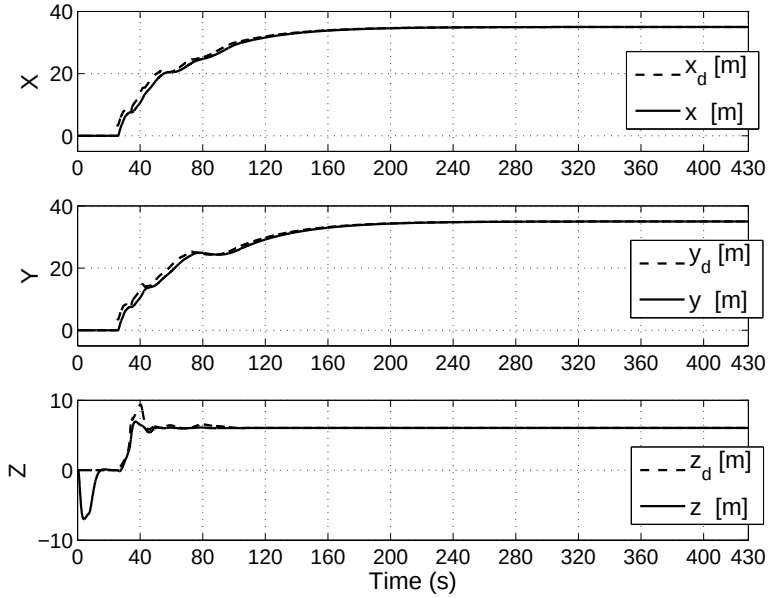


Figure 24: Leader's position and reference signals generated by the potential fields method, tracked by linear MPC

modeled here, taking into account remote obstacles in their current position has a weak significance. The number M of obstacles to be taken into account in the hybrid model clearly depends on the density of the obstacles and the speed of the vehicle. Note that, depending on the sensor system on board of the vehicle, it may be even impossible to measure the position of remote obstacles.

4.7 Conclusions

In this deliverable we have proposed an approach to hierarchical multi-rate control design for *linear* systems that enforces constraints on the variables of the process and guarantees closed-loop stability. By constraining both the magnitude and the variation of the reference signals applied to the lower control layer, we have provided quantitative guidelines for selecting the ratio between the sampling rates of the upper and lower layers, driven by the idea that the state of the process must always lie in an invariant set at the sampling instants of the higher-level controller. We believe that the approach provides valuable insight in the design of hierarchical schemes for decentralized control systems.

To investigate the use of decentralized and hierarchical model predictive control based on *hybrid* dynamical models, we have examined the case of for autonomous navigation of formation of unmanned aerial vehicles, such as quadcopters. A linear MPC controller takes care of vehicles stabilization and reference tracking, and a hybrid MPC generates the path to follow in real-time to reach a given target while avoiding obstacles. The simulation results have shown the reduced computational load of the decentralized hybrid MPC compared to the corresponding centralized hybrid MPC scheme, and the better performance of hybrid MPC in comparison to on-line planning methods like potential fields. Compared to off-line planning methods, such a feature of on-line generation of the 3D path to follow is particularly appealing in realistic scenarios where the positions of the target and of the obstacles are not known in advance, but rather acquired (and possibly time-varying) during flight operations.

5. Distributed RTO with Parametric Coordination

5.1 Introduction

Many practical optimization problems in control engineering, manufacturing, resources distribution, operations research and other fields can be formulated as Quadratic Programming (QP). Typical example in control engineering is **Model Predictive Control** (MPC), which is the most widely implemented advanced process control strategy.

Quadratic programming is an optimization problem involving minimization of quadratic cost function $J(\mathbf{x})$ under linear equality and inequality constraints

$$\min_{\mathbf{x}} J(\mathbf{x}), \qquad \text{s.t.} \qquad \mathbf{A}\mathbf{x} \le \mathbf{b}, \;\; \mathbf{A}_e\mathbf{x} = \mathbf{b}_e. \tag{66}$$

There are many solvers for this class of problems; however, time and memory demands of general QP solvers increase rapidly with the size of optimization problem. Large scale general QP problems involving many thousands of variables and constraints are computationally very demanding.

The large size QP problems arise surprisingly quickly in the formulation of optimization problems involving dynamic optimization of multiple systems and in optimization of so called **systems of systems**. Typical examples are:

- Optimal control of **complex distribution networks** for drinking water or gas involving prediction of demands on 24 hours horizon
- Optimal control of **electric power distribution**
- Coordination of **renewable electric power sources**
- Energy efficient control in **building automation** involving heating, ventilating, and air conditioning of large buildings or multiple buildings
- Optimal control of irrigation channels and urban drainage systems

The systems in the previous examples can be seen as a set of interconnected sub-systems (Figure 25). The formulation of their optimization problem usually follows similar pattern. The cost function is composed as a sum of cost functions corresponding to individual sub-systems and a set of consistency conditions representing mass or energy balances for interactions between sub-systems

$$\min_{\mathbf{x}_i,\mathbf{y}_i} \sum_{i=1}^{N} J_i(\mathbf{x}_i,\mathbf{y}_i), \qquad \text{s.t.} \qquad \begin{matrix} (\mathbf{x}_i,\mathbf{y}_i) \in C_i, \\ \mathbf{M}\mathbf{y} = \mathbf{n}, \end{matrix} \tag{67}$$

where $\mathbf{x}_i$ are private variables of each sub-system, variables $\mathbf{y}_i$ are interaction variables, J_i are the cost functions of individual subsystems, C_i are constraints of individual sub-systems and $\mathbf{M},\mathbf{n}$ define sub-systems interaction constraints.

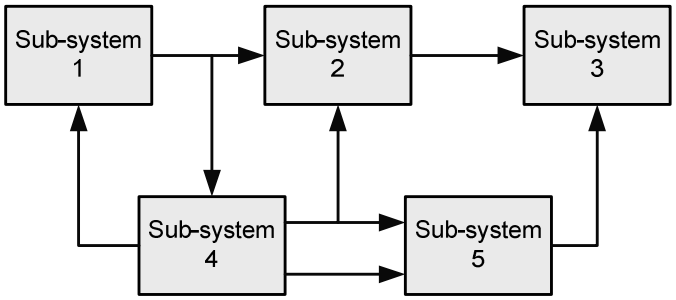


Figure 25 Large-scale system as a set of sub-systems and their interactions.

The class of large-scale problems defined by (67) can be solved by **decomposition methods**. These methods lead to algorithms where multiple relatively small separated sub-problems are **iteratively** solved and coordinated together. These methods use the fact that solving multiple small optimization problems is significantly faster than solving a single large optimization problem.

A general scheme of decomposition methods is in Figure 26. Typical steps are following:

1) Distribution of coordination variables to sub-problems
2) Solve sub-problems and return (part of) solution to coordinator
3) Update coordination variables
4) Continue to step 1 until interaction variables are coordinated

Advantages of decomposition methods are:

- Allows to solve truly large scale optimization problems
- Sub-problems are independent and may be solved in parallel
- Optimization time may be reduced even on a single computer

Disadvantages:

- The number of iteration may be very high
- Optimal solution is obtained asymptotically

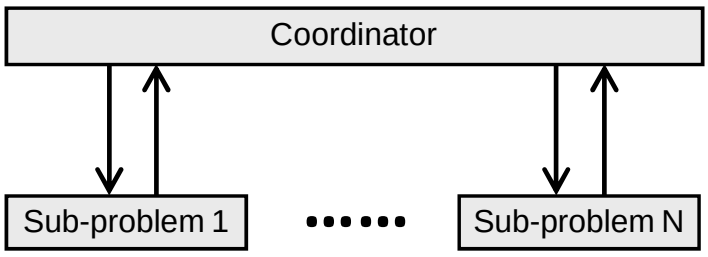


Figure 26 Coordinated optimization.

We propose a novel method for solving the class of large-scale problems defined by (67) with the following properties:

- **Exact optimal solution is obtained in a finite number of iterations**
- **The number of iterations and hence optimization time is low**
- Only some sub-problems need to be resolved in each iteration → **communication reduction** between coordinator and sub-problem solvers

An illustration of the novel method performance is in Figure 27. It shows optimization time as a function of optimization problem size for three different algorithms. The black line is time needed to get solution without decomposition (centralized solution), the blue line is

the solution by the decomposition method based on L-BFGS algorithm and the red line (denoted as MP) is the performance of the proposed algorithm. The results were obtained for optimization problem imposed by model predictive control of interconnected water tanks. The values are mean times required to get solution for different number of tanks (different number of variables). Note that precise solutions are obtained only by centralized and MP method.

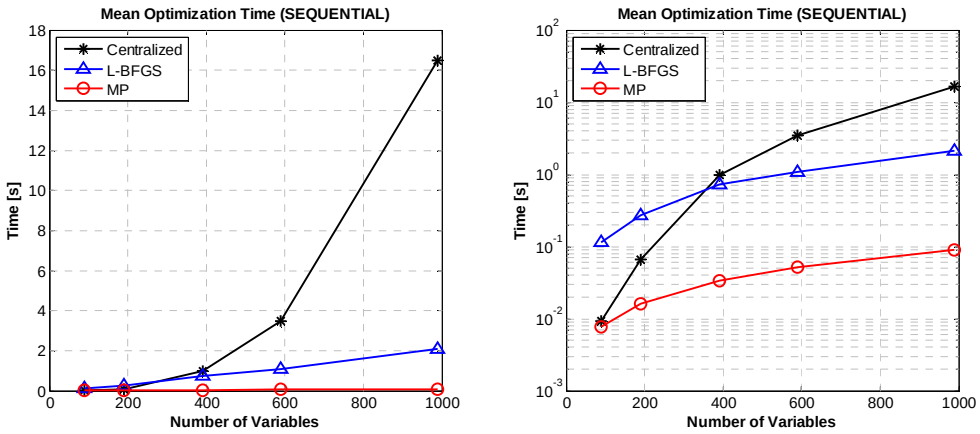


Figure 27 Optimization time for different methods as a function of problem size. Left figure has linear and right figure logarithmic scale.

The properties of novel algorithm help to:

- solve large scale optimization problems in real time (model predictive control), where centralized computation is impossible or extensively time demanding
- solve large scale optimization problems without sacrificing precision
- physically distribute computation while keeping communication traffic low

5.2 Method

This section describes the state-of-the-art decomposition algorithm and novel algorithm. Both approaches are then compared as functional block diagrams in Figure 30.

5.2.1 Decomposition methods

One class of decomposition methods is based on price coordination of dual decomposition, where the objective of coordinator (Figure 28) is to maximize function $g(\lambda)$, which is dual to original cost function $J(\mathbf{x})$, by manipulating prices λ

$$\max_{\lambda} g(\lambda). \tag{68}$$

The key property of dual decomposition is that the coordinator gets a gradient of function $g(\lambda)$ in the point λ by sending λ to sub-problem solvers and aggregating returned information. This gradient is then used to update variable λ by using gradient type methods and the process is repeated until the coordination error is bellow given threshold. The general description of dual decomposition methods steps is following:

0) Initialize $\lambda^{(0)}$ (upper index indicates iteration index)
1) Coordinator distributes $\lambda^{(k)}$ to sub-problem solvers

2) Solve sub-problems and send back part of the gradient $\nabla g_i(\lambda^{(k)})$ computed in point $\lambda^{(k)}$
3) Coordinator computes dual function gradient $\nabla g\left(\lambda^{(k)}\right)=\sum_{i=1}^{N}\nabla g_i\left(\lambda^{(k)}\right)$
4) Gradient $\nabla g_i(\lambda^{(k)})$ is used to update $\lambda^{(k)}$ to $\lambda^{(k+1)}$ by gradient type methods.
5) Continue to step 1 if coordination error is above selected threshold.

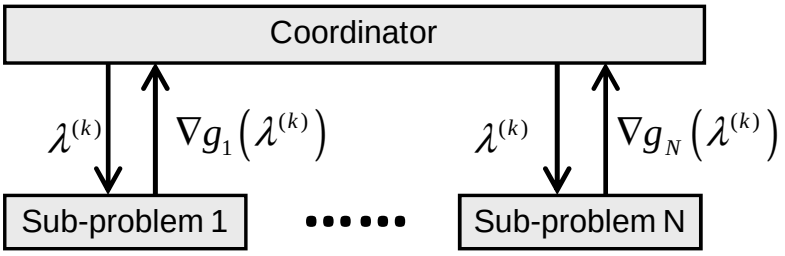


Figure 28 Gradient based coordinated optimization.

5.2.2 Parametric Coordination

The novel algorithm is based on the fact, that in the case of quadratic programming the sub-problem solvers are able to return not only a solution in a single point (as in the previous algorithm), but also a **parametric solution, which is valid on a region around given point**.

With these solutions the coordinator is able to form dual function value, gradient and **hessian**, which are valid on an intersection of sub-problem validity regions. This information allows coordinator to apply highly effective Newton type methods for $\lambda^{(k)}$ update and to test if exact solution lies within current validity region (then it can be instantly reached).

Another advantage of having parametric solution with the validity region is that only those sub-problems have to be recomputed where updated $\lambda^{(k)}$ lies out of their validity region of the last solution. This saves a lot of computation effort as in the most iterations only a fraction of sub-problems need to be recomputed.

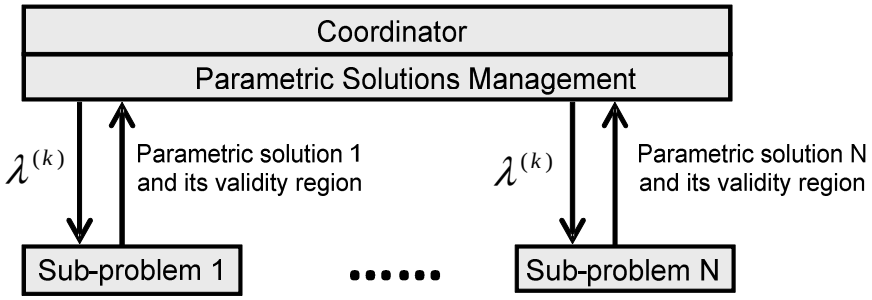


Figure 29 Parametric coordination.

The algorithm with a block scheme in Figure 29 has the following steps:

0) Initialize $\lambda^{(0)}$ and distribute it to all sub-problem solvers
1) Sub-problems receiving new $\lambda^{(k)}$ are solved and parametric solutions together with their validity regions P_i are returned to coordinator

2) Coordinator computes dual function value, gradient and hessian in the point $\lambda^{(k)}$ and corresponding validity region P as an intersection of P_i
3) The exact solution is achieved if undamped Newton step solution lies within P, otherwise continue.
4) $\lambda^{(k)}$ is updated to $\lambda^{(k+1)}$ by Newton type method.
5) $\lambda^{(k+1)}$ is sent to sub-problems where last solution is not valid for new $\lambda^{(k+1)}$
6) Continue to step 1

5.2.3 Methods Comparison

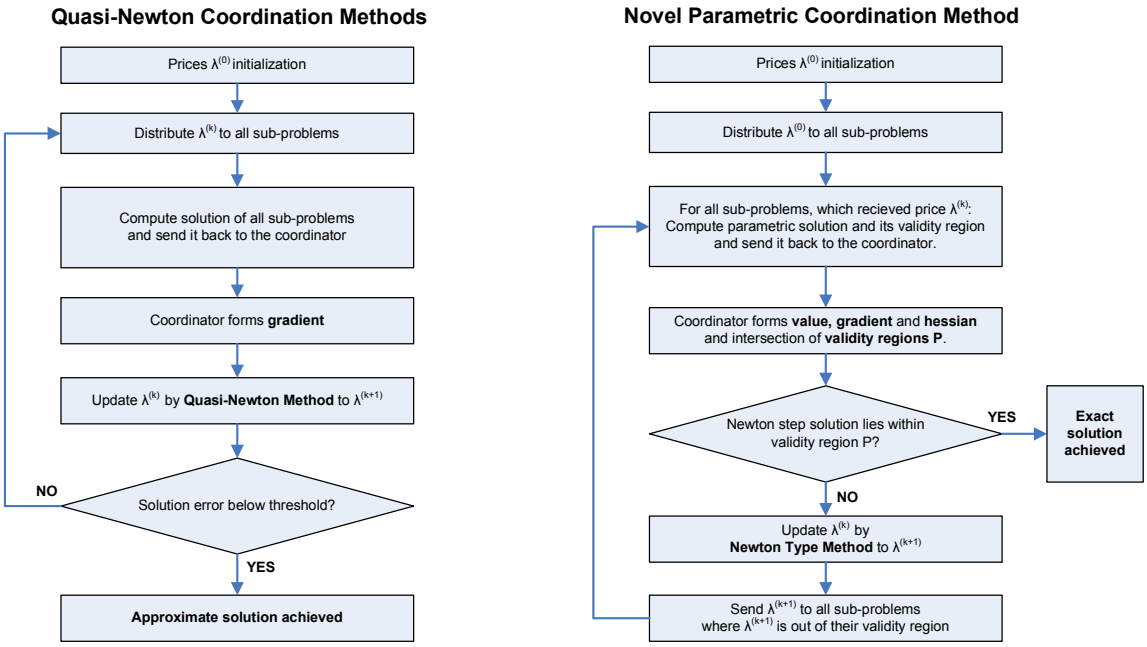


Figure 30 Comparison state-of-the-art and novel methods.

5.3 Method details

The target class of optimization problems is quadratic programming

$$\min_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{N} \mathrm{J}_i\left(\mathbf{x}_i,\mathbf{y}_i\right), \qquad \text{s.t.} \qquad \begin{matrix}\left(\mathbf{x}_i,\mathbf{y}_i\right)\in C_i,\\ \mathbf{My}=\mathbf{n},\end{matrix} \tag{69}$$

where J_i are quadratic functions, C_i are linear constraints, $\mathbf{x}_i$ and $\mathbf{y}_i$ are parts of $\mathbf{x}$ and $\mathbf{y}$ respectively. Corresponding dual function

$$\mathrm{g}(\lambda)=\min_{\mathbf{x},\mathbf{y}}\left(\left(\sum_{i=1}^{N}\mathrm{J}_i\left(\mathbf{x}_i,\mathbf{y}_i\right)\right)+\lambda^T\left(\mathbf{My}-\mathbf{n}\right)\right) \qquad \text{s.t.} \qquad \left(\mathbf{x}_i,\mathbf{y}_i\right)\in C_i, \tag{70}$$

is for given λ separable as

$$\mathrm{g}(\lambda)=\left(\sum_{i=1}^{N}\mathrm{g}_i(\lambda)\right)-\lambda^T\mathbf{n} \qquad \text{s.t.} \qquad \left(\mathbf{x}_i,\mathbf{y}_i\right)\in C_i, \tag{71}$$

where g_i are independent optimization sub-problems

$$\mathrm{g}_i(\lambda) = \min_{\mathbf{x}_i, \mathbf{y}_i} \mathrm{J}_i\left(\mathbf{x}_i, \mathbf{y}_i\right) + \lambda^T \mathbf{M}_i \mathbf{y}_i \qquad \text{s.t.} \qquad \left(\mathbf{x}_i, \mathbf{y}_i\right) \in C_i, \tag{72}$$

and where $\mathbf{M}_i$ are parts of $\mathbf{M}$ corresponding to $\mathbf{y}_i$. The key point is that g_i is a piecewise smooth quadratic function and can be on a polytopic region P_i expressed as

$$\mathrm{g}_i(\lambda) = \lambda^T \mathbf{A}_i \lambda + \mathbf{b}_i^T + c_i, \quad \lambda \in P_i, \quad P_i = \left\{\lambda \mid \mathbf{H}_i \lambda \le \mathbf{k}_i\right\}, \tag{73}$$

where $\mathbf{A}_i, \mathbf{b}_i, c_i, \mathbf{H}_i, \mathbf{k}_i$ are obtained from parametric optimization of (72). The coordinator can then compute dual function value, gradient and hessian as

$$\begin{aligned} g(\lambda) &= \left(\sum_{i=1}^{N} \lambda^T \mathbf{A}_i \lambda + \mathbf{b}_i^T \lambda + c_i\right) - \lambda^T \mathbf{m}, \\ \nabla g(\lambda) &= \left(\sum_{i=1}^{N} \left(\mathbf{A}_i + \mathbf{A}_i^T\right) \lambda + \mathbf{b}_i\right) - \mathbf{m}, \\ \nabla^2 g(\lambda) &= \sum_{i=1}^{N} \mathbf{A}_i + \mathbf{A}_i^T, \end{aligned} \tag{74}$$

with validity region P given as an intersection of local solutions validity regions

$$P = \bigcap_{i=1}^{N} P_i. \tag{75}$$

Dual function value, gradient and hessian are used by effective Newton type methods (for example adaptively damped Newton method) to solve dual problem and therefore also the original problem (69).

Notes:

- Parameters $\mathbf{A}_i, \mathbf{b}_i, c_i, \mathbf{H}_i, \mathbf{k}_i$ are **highly sparse** (most elements are zeros). Only values, which are non-zero (by problem structure) needs to be communicated to the coordinator. Similarly sub-problems require only part of λ.
- Sub-problem solvers do not need to enumerate all polytopic regions, but only a **single polytop** around given λ.
- The convergence is guaranteed by convexity of the original problem.

5.4 Method steps

Algorithm **Distributed Optimization with Parametric Coordination**

0) Coordinator initializes $\lambda^{(0)}$
1) Coordinator requests new solution from sub-problem solvers, where the last solution is invalid for $\lambda^{(k)}$
2) All sub-problem solvers with invalid solution compute their parametric solution and its validity region (73) and send back parameters $\mathbf{A}_i, \mathbf{b}_i, c_i, \mathbf{H}_i, \mathbf{k}_i$
3) Coordinator forms dual function value, gradient and hessian in $\lambda^{(k)}$ as (74) and its appropriate validity region (75).
4) Exact solution is achieved if un-damped Newton step lies in the validity region (75).
5) Coordinator updates $\lambda^{(k)}$ to $\lambda^{(k+1)}$ by Newton type method (for example adaptively damped Newton method).
6) Continue to step 1

5.5 Application to Model Predictive Control

Following sections describe application of novel coordination method to Model Predictive Control (MPC) of large-scale systems (Figure 31). Application of novel method is roughly 100x faster than MPC formulated as a single optimization problem and roughly 10x faster than Quasi-Newton coordination methods for large-scale systems, while giving an exact solution. The method can be implemented in a single controller or it can be physically distributed to multiple controllers with a single coordinator (Figure 34). The method is demonstrated on control of Barcelona water distribution network.

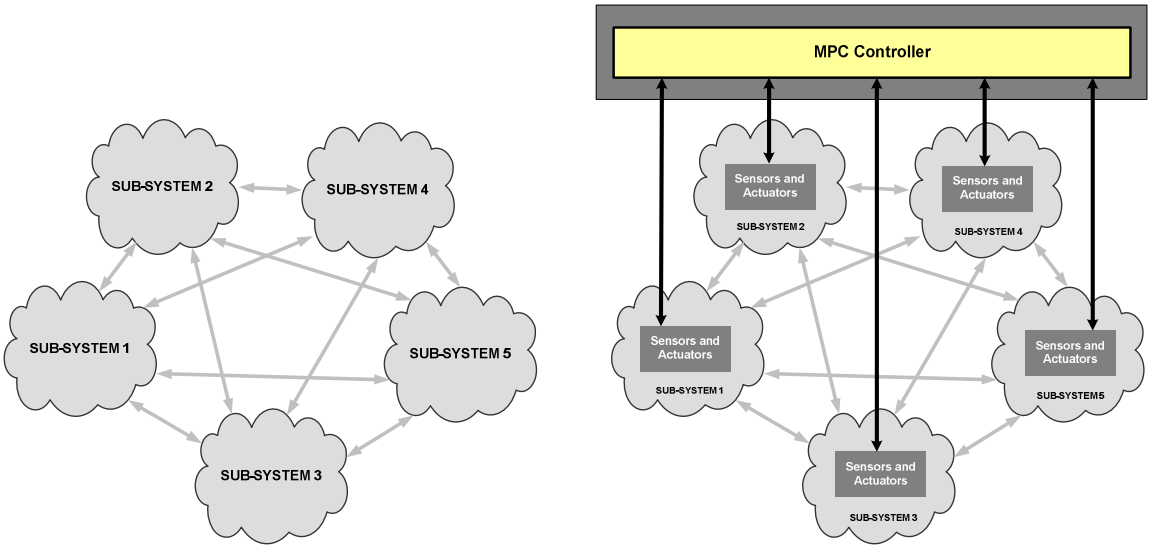


Figure 31 Large-scale system as a set of interacting sub-systems (left figure); control of large-scale system by a single MPC controller (right figure).

The main problem with the application of MPC to large-scale systems is that the computational complexity increases quickly with the size of the system and the length of prediction horizon.

MPC computes quadratic optimization (QP) problem in each sampling period. To illustrate the size of optimization problems consider MPC for Barcelona water distribution network, which will be shown in more details in Section 5.6.2. The parameters of MPC and appropriate QP problem are following:

Prediction horizon	24 hours
Sampling period	1 hour
Number of variables	**~5.000**
Number of constraints	**~10.000**

MPC for large-scale systems can be reformulated to the form of (69) and parametric coordination method can be directly applied. To follow notation usual in control community the sub-problem solver is denoted as Local MPC Controller. The operation block diagram of Distributed MPC with Parametric Coordination is in Figure 32.

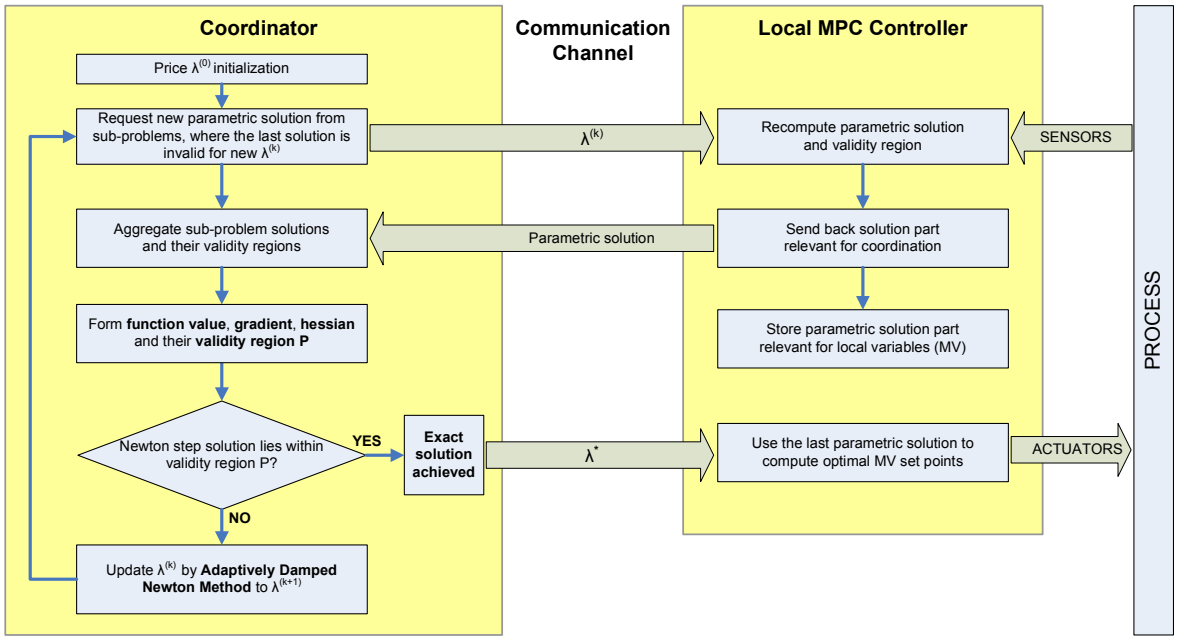


Figure 32 Block diagram of Distributed MPC with Parametric Coordination.

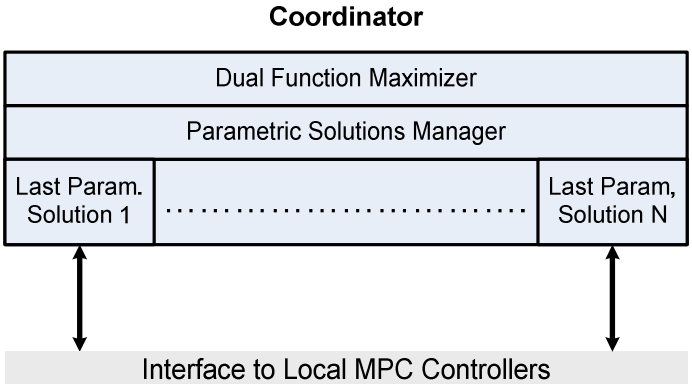


Figure 33 Functional block scheme of parametric coordinator.

5.5.1 Algorithm Implementation

The proposed iterative algorithm can be either implemented on a single controller or it can be physically distributed to local controllers and one coordinator. Both cases are in Figure 34.

Implementation on a single controller allows implementing MPC for large-scale system, where legacy control is already realized as centralized control architecture. This configuration eliminates data transfer delays between coordinator and local MPC controllers. The algorithm can be computed in parallel even in a single controller with multiple computing units.

Distributed implementation brings improved robustness as local MPC controllers can continue operating even in the case of communication failure.

Proposed method allows to implement MPC strategy to large-scale processes, where classical MPC is not possible due to their size.

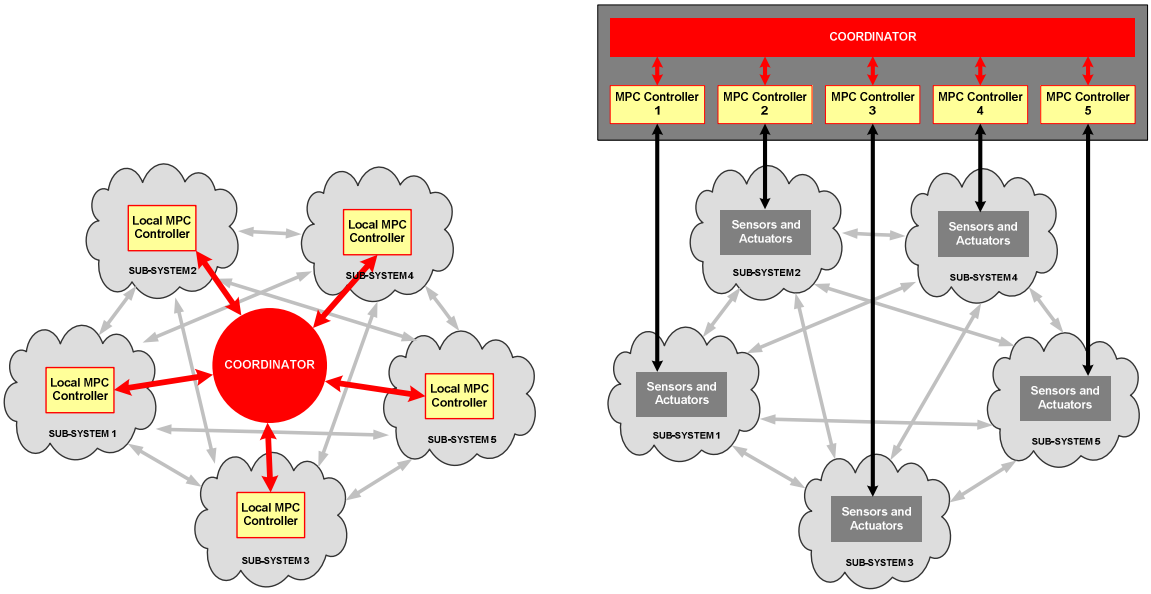


Figure 34 Coordinated Distributed MPC implementation either as physically distributed controllers with central coordinator (left figure); or as a single controller computing solution by internal coordination of multiple sub-problems (right figure).

5.6 Application Example

The performance of the proposed method is demonstrated on two examples. Both examples are control of water distribution networks. The first example is control of a medium-scale network and the second example is control of water distribution network in Barcelona, Spain.

Water distribution networks are complex interconnections of tanks, pump, valves, interconnection nodes, water sources and points of water consumption. The basic control objective of water networks is to fully satisfy water demands in all points of consumption. To achieve this objective water is pumped from sources across the network to water tanks, where each tank usually represents also a point of consumption. This basic objective can be achieved in multiple ways. The objective of optimal control is to select the one with the following properties:

- Minimize electricity price for water pumping (time variable electricity price)
- Minimize price of fresh water (different prices of water from different sources: wells, rivers,…)
- Keep tank levels above safety level
- Minimize manipulated variables changes (to reduce wear of pumps and valves)
- Minimize long water storage in tanks (requires additional chlorination)
- Fulfill all constraints (limits on pumps, valves, tanks, water sources)

These objectives can be well achieved by MPC. The application of MPC requires good predictions of water demands in points of consumption, which can be with satisfying accuracy obtained from history process data.

5.6.1 Example 1 - Medium-Scale Water Distribution Network

The scheme of water network used in this example is in Figure 25. The network has the following parameters:

Number of tanks	12
Number of pumps and valves	16
Number of source	3
Number of complex nodes	3
Number of points of demand	15

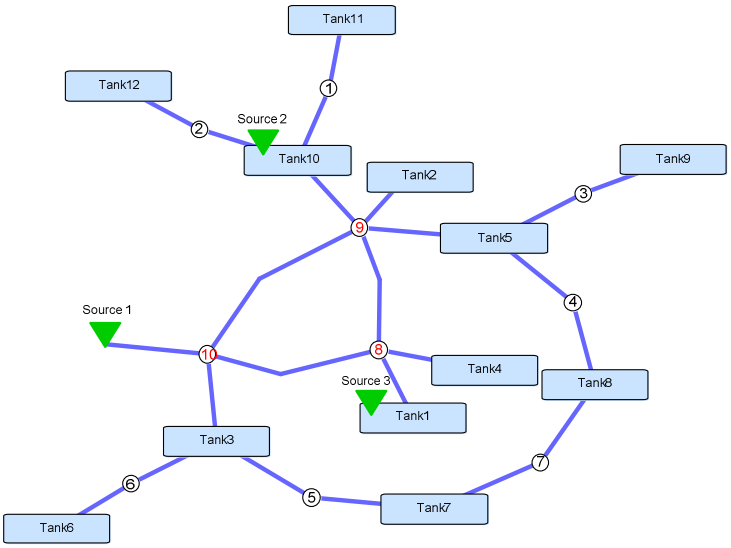


Figure 35 Example 1 - Medium-scale water distribution network. Green triangles indicate water source. Each tank and each complex node (red color) has also a point of consumption.

MPC is designed with 1 hour sampling period and 24 hours prediction horizon. Three methods are compared:

1) **Centralized MPC** – formulation as a single QP problem (Centralized)
2) Distributed MPC with **Quasi-Newton coordination** (L-BFGS) (state-of-the-art method)
3) Distributed MPC with **Parametric coordination** (Parametric)

The flow trajectories (manipulated variables) obtained by Centralized MPC and Parametric Coordinated MPC are the same as can be seen in Figure 36 and Figure 37 (top-right). The stopping condition for L-BFGS coordination was maximum flow error lower than $0.01\,\mathrm{m}^3/\mathrm{hour}$. The number of iterations of distributed algorithms is compared in Figure 37 (top-left). Effective number of parametric iterations refines the number of real iterations to match the fact that parametric method does not re-compute all local MPC problems as compared to other coordination methods. Optimization times of all three methods are compared in Figure 37 (bottom). “Sequential” means time required on a single computing unit and “parallel” means time required by parallel computing units.

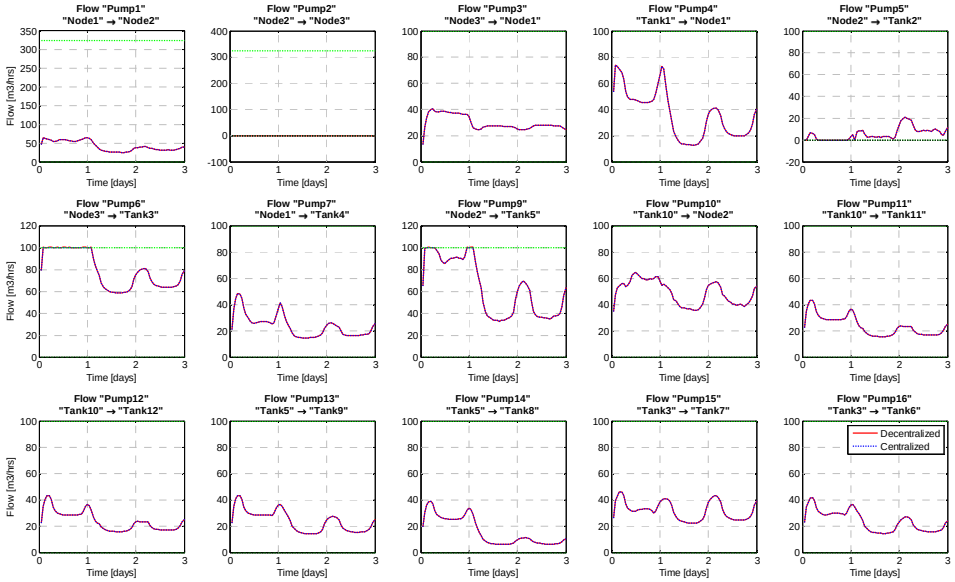


Figure 36 Valves and Pumps flow trajectories computed by centralized MPC (blue lines) and distributed MPC with parametric coordination (red dashed lines).

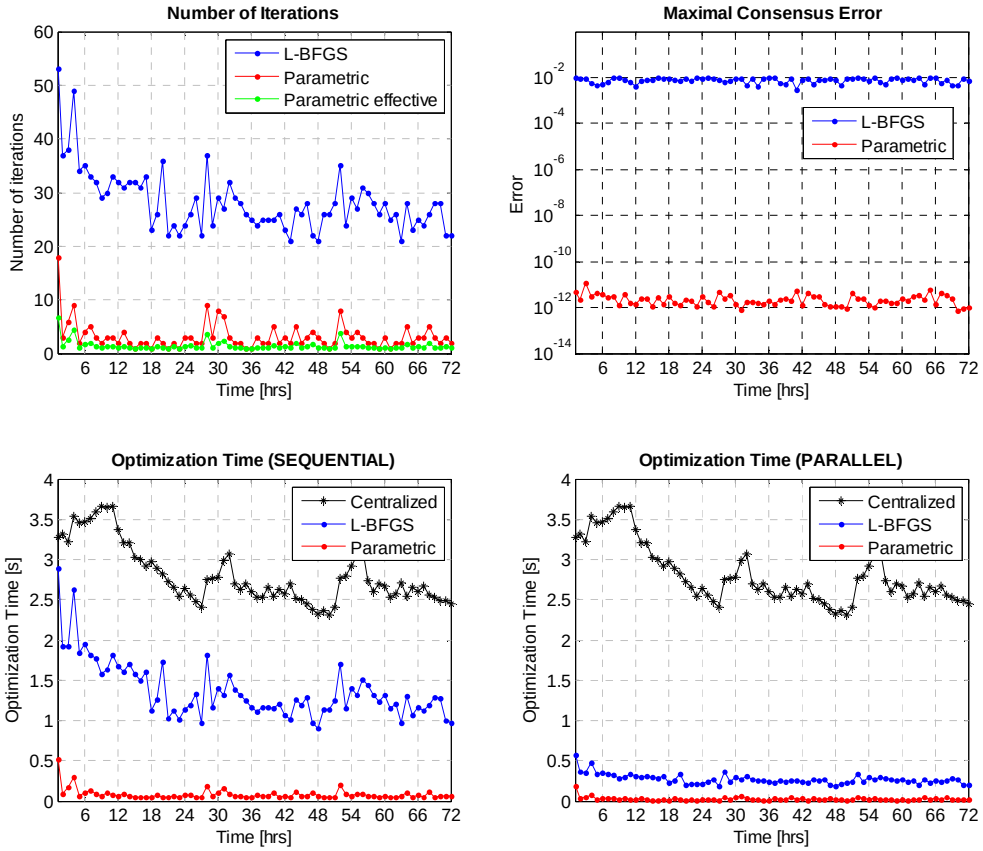


Figure 37 Control results comparison.

5.6.2 Example 2 - Barcelona Drinking Water Distribution Network

Barcelona drinking water distribution network scheme is in Figure 38. The network has the following parameters:

Number of tanks	67
Number of pumps and valves	111
Number of source	10
Number of complex nodes	15
Number of points of demand	88

Nominal solution was computed by centralized MPC with 10 hours prediction horizon, which was the maximum length allowing computation of centralized solution in Matlab. The parameters of quadratic programming problem for centralized MPC were following:

Prediction horizon	10 hours
Number of variables	**1.920**
Number of constraints	**3.820**

Tanks in the network were partitioned into groups. Each group is controlled by its own local MPC controller. Three methods are compared:

1) **Centralized MPC** – formulation as a single QP problem (Centralized)
2) Distributed MPC with **Quasi-Newton coordination** (L-BFGS) (state-of-the-art method)
3) Distributed MPC with **Parametric coordination** (Parametric)

The comparison of control method performance is in Figure 39. The top figure compares the number of iterations for each distributed method. Bottom two figures compare optimization times in each iteration for sequential (bottom-left) and parallel computations (bottom-right). The results are summarized in the following table:

	Mean Number of Iterations	**Mean Optimization Time [s]**	
		Sequential	**Parallel**
Centralized MPC	-	120	120
Distributed L-BFGS	118	36.2	7.9
Distributed Parametric	40 (9 effective)	4.8	2.1

The table shows the reason why centralized MPC is not used to control large-scale water networks despite the fact that it can perfectly fulfill all main performance criterions. Even though the prediction horizon is only 10 hours (optimal would be 24 hours) the computation time for centralized MPC is already very long and the size of optimization problems is on the edge of practical computational limits (in the number of variables and constraints).

On the other hand distributed MPC with parametric coordination offers method, which will not have problems with 24 hours prediction horizon (in terms of computational demands and number of iterations).

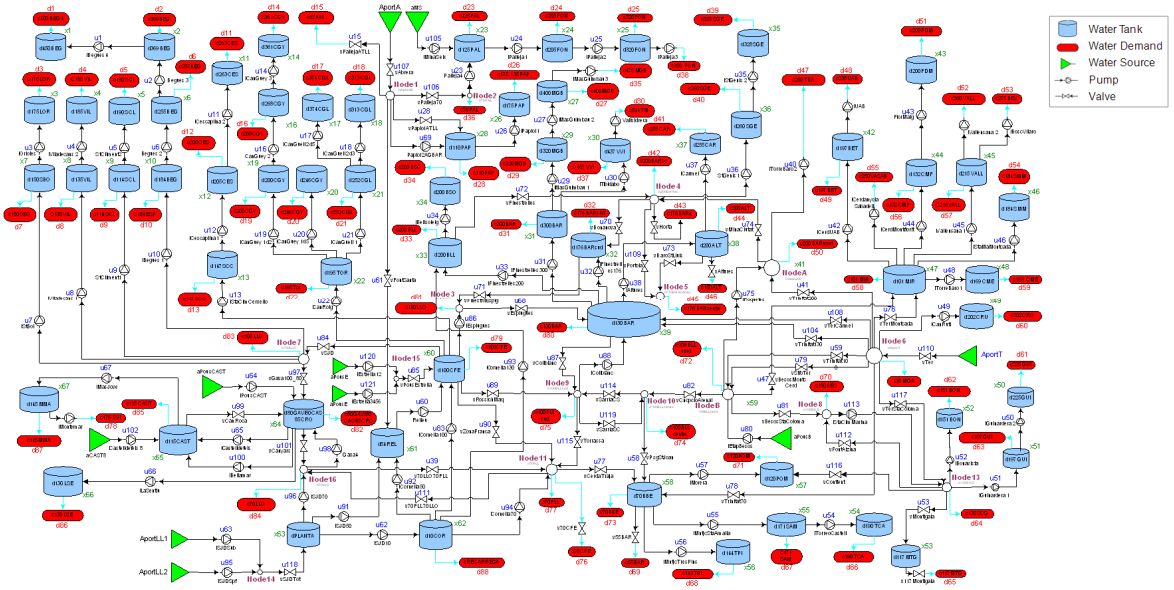


Figure 38 Scheme of Barcelona drinking water distribution network.

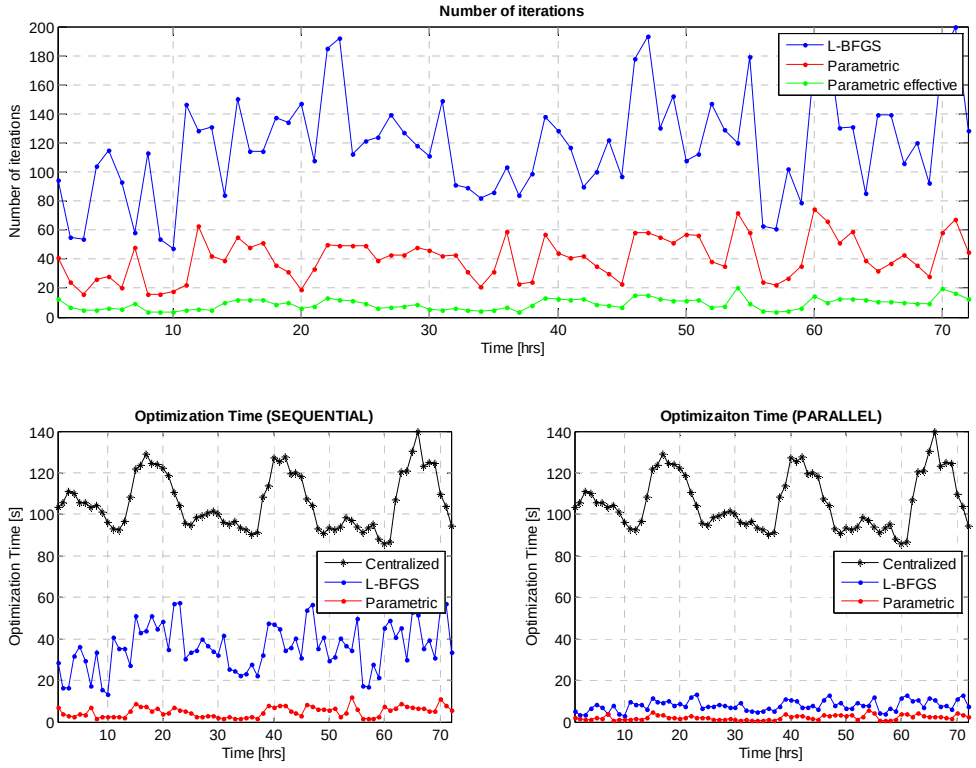


Figure 39 Control methods performance comparison for Barcelona Water Distribution Network.

5.7 Conclusion

The presented method was filled for patent in European Patenting Office:

Patent Application Title:	“OPTIMIZATION PROBLEM SOLVING”
Application No.:	11157244.2
Filing Date:	March 7, 2011
Inventors:	TRNKA PAVEL; PEKAR JAROSLAV

References

[1] E. Altug, J. Ostrowski, and C. Taylor, "Control of a quadrotor helicopter using dual camera visual feedback," *The International Journal of Robotics Research*, vol. 24, no. 5, pp. 329–341, 2005.

[2] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *Proc. AIAA Guidance, Navigation, and Control Conf., Honolulu, HI*, 2008.

[3] P. Castillo, A. Dzul, and R. Lozano, "Real-time stabilization and tracking of a four-rotor mini rotorcraft," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 4, pp. 510–516, 2004.

[4] A. Kivrak, "Design of control systems for a quadrotor flight vehicle equipped with inertial sensors," Master's thesis, Atilim University, Turkey, 2006.

[5] M. Chen and M. Huzmezan, "A combined MBPC/2 DOF H_∞ controller for a quad rotor UAV," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2003.

[6] G. Raffo, M. Ortega, and F. Rubio, "An integral predictive/nonlinear h_∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, pp. 29–39, 2010.

[7] A. Bemporad, C. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," in *3rd IFAC Conference on Analysis and Design of Hybrid Systems, Zaragoza, Spain*, 2009.

[8] W. Dunbar, "Model predictive control: extension to coordinated multi-vehicle formations and real-time implementation," CDS Technical Memo CIT-CDS 01-016, California Institute of Technology, Pasadena, CA 91125, Tech. Rep., 2001.

[9] W. Dunbar and R. Murray, "Model predictive control of coordinated multi-vehicle formations," in *IEEE Conference on Decision and Control*, vol. 4, 2002, pp. 4631–4636.

[10] F. Borrelli, T. Keviczky, K. Fregene, and G. Balas, "Decentralized receding horizon control of co-operative vechicle formations," in *Proc. 44th IEEE Conf. on Decision and Control and European Control Conf.*, Sevilla, Spain, 2005, pp. 3955–3960.

[11] W. Li and C. Cassandras, "Centralized and distributed cooperative receding horizon control of autonomous vehicle missions," *Mathematical and computer modelling*, vol. 43, no. 9-10, pp. 1208–1228, 2006.

[12] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *Proc. 43rd IEEE Conf. on Decision and Control*, 2004, pp. 4286–4291.

[13] V. Manikonda, P. Arambel, M. Gopinathan, R. Mehra, F. Hadaegh, S. Inc, and M. Woburn, "A model predictive control-based approach for spacecraft formation keeping and attitude control," in *American Control Conference, 1999. Proceedings of the 1999*, vol. 6, 1999.

[14] J. Chuang, "Potential-based modeling of three-dimensional workspace for obstacle avoidance," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 778–785, 1998.

[15] T. Paul, T. Krogstad, and J. Gravdahl, "Modelling of UAV formation flight using 3D potential field," *Simulation Modelling Practice and Theory*, vol. 16, no. 9, pp. 1453–1462, 2008.

[16] J. Latombe, *Robot motion planning*. Kluwer academic publishers, 1991.

[17] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3, 2002.

[18] F. Borrelli, T. Keviczky, G. Balas, G. Stewart, K. Fregene, and D. Godbole, "Hybrid decentralized control of large scale systems," ser. Lecture Notes in Computer Science, M. Morari and L. Thiele, Eds. Springer-Verlag, 2005, pp. 168–183.

[19] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 3–11, 2002.

[20] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," Master's thesis, Department of Automatic Control, Lund University, October 2008.

[21] A. Bemporad, N. Ricker, and J. Owen, "Model Predictive Control-New tools for design and evaluation," in *American Control Conference*, vol. 6, Boston, MA, 2004, pp. 5622–5627.

[22] F. Torrisi and A. Bemporad, "HYSDEL-A tool for generating computational hybrid models for analysis and synthesis problems," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 235–249, 2004.

[23] A. Bemporad, *Hybrid Toolbox – User's Guide*, jan 2004, http://www.dii.unisi.it/hybrid/toolbox.

[24] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[25] A. Bemporad, M. Morari, and N. Ricker, *Model Predictive Control Toolbox for Matlab – User's Guide*. The Mathworks, Inc., 2004, http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[26] ILOG, Inc., *CPLEX 11.2 User Manual*, Gentilly Cedex, France, 2008.

[27] N. Sandell, P. Varaiya, M. Athans, and M. Safonov, "Survey of decentralized control methods for large scale systems," *IEEE Trans. Automatic Control*, vol. 23, no. 2, pp. 108–128, 1978.

[28] E. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed., ser. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag, 2004.

[29] R. Scattolini, "Architectures for distributed and hierarchical model predictive control – a review," *Journal of Process Control*, vol. 19, pp. 723–731, 2009.

[30] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, pp. 44–52, Feb. 2002.

[31] A. Venkat, J. Rawlings, and J. Wright, "Stability and optimality of distributed model predictive control," in *Proc. 44th IEEE Conf. on Decision and Control and European Control Conf.*, Seville, Spain, 2005.

[32] W. Dunbar and R. Murray, "Distributed receding horizon control with application to multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549–558, 2006.

[33] T. Keviczky, F. Borrelli, and G. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006.

[34] A. Alessio and A. Bemporad, "Decentralized model predictive control of constrained linear systems," in *Proc. European Control Conf.*, Kos, Greece, 2007, pp. 2813–2818.

[35] R. Negenborn, B. De Schutter, and J. Hellendoorn, "Multi-agent model predictive control: A survey," *Arxiv preprint arXiv:0908.1076*, 2009.

[36] E. Gilbert, I. Kolmanovsky, and K. T. Tan, "Discrete-time reference governors and the nonlinear control of systems with state and control constraints," *Int. J. Robust Nonlinear Control*, vol. 5, no. 5, pp. 487–504, 1995.

[37] A. Bemporad, A. Casavola, and E. Mosca, "Nonlinear control of constrained linear systems via predictive reference management," *IEEE Trans. Automatic Control*, vol. AC-42, no. 3, pp. 340–349, 1997.

[38] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Trans. Automatic Control*, vol. AC-43, no. 3, pp. 415–419, 1998.

[39] E. Gilbert and I. Kolmanovsky, "Fast reference governors for systems with state and control constraints and disturbance inputs," *Int. J. Robust Nonlinear Control*, vol. 9, no. 15, pp. 1117–1141, Dec. 1999.

[40] K. Hirata and M. Fujita, "Set of admissible reference signals and control of systems with state and control constraints," in *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, AZ, 1999, pp. 1427–1432.

[41] K. Kogiso and K. Hirata, "Reference governor for constrained systems with time-varying references," *Robotics and Autonomous Systems*, vol. 57, no. 3, pp. 289–295, 2009.

[42] R. Scattolini and N. Schiavoni, "A multirate model based predictive controller," in *Proc. 33rd IEEE Conf. on Decision and Control*, vol. 1, 1994.

[43] P. Menchinelli and A. Bemporad, "Hybrid model predictive control of a solar air conditioning plant," *European Journal of Control*, vol. 14, no. 6, pp. 501–515, 2008.

[44] A. Bemporad, C. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," in *3rd IFAC Conference on Analysis and Design of Hybrid Systems*, Zaragoza, Spain, 2009.

[45] C. Crusius and A. Trofino, "Sufficient LMI Conditions for Output Feedback Control Problems," *IEEE Trans. Automatic Control*, vol. 44, no. 5, pp. 1053–1057, 1999.

[46] E. Gilbert and K. T. Tan, "Linear systems with state and control constraints: the theory and applications of maximal output admissible sets," *IEEE Trans. Automatic Control*, vol. 36, no. 9, pp. 1008–1020, 1991.

[47] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.

[48] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, ser. Lecture Notes in Control and Information Sciences, D. R. L. Magni, F. Allgower, Ed., vol. 384. Berlin Heidelberg: Springer-Verlag, 2009, pp. 345–369.

[49] A. Bemporad, "Hybrid Toolbox - User's Guide," 2004, http://www.ing.unitn.it/~bemporad/hybrid/toolbox.

[50] M. Kvasnica, P. Grieder, and M. Baotić, *Multi Parametric Toolbox (MPT)*, 2006, http://control.ee.ethz.ch/~mpt/.

[51] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://control.ee.ethz.ch/~joloef/yalmip.php

[52] ILOG, Inc., *Cplex 9.0 User Manual*, Gentilly Cedex, France.

[53] Wang, W. and Boyd, S. "Fast model predictive control using online optimization", 2008, pp. 6974–6979.

[54] A. Bemporad, M. Morari, V. Dua and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems", *Automatica*, 2002, pp. 38(1), 3–20.

[55] M. Basler, M. Spott, et al. *The FlightGear Manual*, 2008, http://www.flightgear.org/Docs/getstart/getstart.html.

[56] D.D. Šiljak, *Decentralized control of complex systems*, 1991, Academic Press.

[57] D. Jia and B. Krogh, *Min-max feedback model predictive control for distributed control with communication*, 2002, Proc. American Contr. Conf., pp 4507–4512.

[58] J. Liu and D. Muñoz de la Peña and P.D. Christofides, *Distributed model predictive control of nonlinear process systems*, 2009, AIChE Journal, vol. 55, pp. 1171–1184.

[59] A. Bemporad and D. Barcelli, *Decentralized model predictive control* In A. Bemporad, W.P.M.H. Heemels, and M. Johansson, editors, *Networked Control Systems*, Eds. Springer- Verlag, 2010. http://ist-wide.dii.unisi.it/school09.

[60] D. Barcelli and A. Bemporad and G. Ripaccioli, *Hierarchical Multi-Rate Control Design for Constrained Linear Systems*, 2010, Proc. 49th IEEE Conf. on Decision and Control.

[61] D.D. Šiljak and A.I. Zečević, *Control of large-scale systems: Beyond decentralized feedback*, 2005, Annual Reviews in Control, vol. 29, pp. 169–179.

[62] A. Alessio and D. Barcelli and A. Bemporad, *Decentralized Model Predictive Control of Dynamically-Coupled Linear Systems*, 2011, Journal of Process Control, in press.

[63] E.G. Gilbert and I. Kolmanovsky, *Maximal output admissible sets for discrete-time systems with disturbance inputs*, 1995, Proc. American Contr. Conf., pp. 2000–2005.

[64] H.P. Williams, *Model Building in Mathematical Programming*, 1993, John Wiley & Sons Third Edition.

[65] Federica Garin and Luca Schenato, *A Survey on Distributed Estimation and Control Applications Using Linear Consensus Algorithms* In A. Bemporad, W.P.M.H. Heemels, and M. Johansson, editors, *Networked Control Systems*, Eds. Springer- Verlag, 2010. http://ist-wide.dii.unisi.it/school09.

[66] B. Yang and M. Johansson, *Distributed Optimization and Games: A Tutorial Overview*, In A. Bemporad, W.P.M.H. Heemels, and M. Johansson, editors, *Networked Control Systems*, Eds. Springer- Verlag, 2010. http://ist-wide.dii.unisi.it/school09.