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

Grant Agreement n. 224168 FP7-ICT-2007-2

WIDE

Decentralized Wireless Control of Large-Scale Systems

Starting date: 01 September 2008 Duration: 3 years

Deliverable number	**D3.2**
Title	**Decentralized and distributed model predictive control and estimation algorithms**
Work package	WP3 - Multilayer distributed control and model management (RTD)
Due date	M24
Actual submission date	31/08/2010
Lead contractor for this deliverable	Honeywell Prague Laboratory (HPL)
Author(s)	P. Trnka `pavel.trnka@honeywell.com`, L. Baramov `lubomir.baramov@honeywell.com`
With the help of	A. Bemporad `bemporad@ing.unitn.it`, D. Barcelli `barcelli@dii.unisi.it` V. Puig `vicenc.puig@upc.edu`
Revision	v1.1 (October 10, 2010)

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 documents covers results in decentralized and distributed model predictive control and distributed Kalman filtering.

Contents

1 Decentralized model predictive control

Decentralized and distributed model predictive control (DMPC) addresses the problem of controlling a multivariable dynamical process, composed by several interacting subsystems and subject to constraints, in a computation and communication efficient way. Compared to a centralized MPC setup, where a global optimal control problem must be solved on-line with respect to all actuator commands given the entire set of states, in DMPC the control problem is divided into a set of local MPCs of smaller size, that cooperate by communicating each other a certain information set, such as local state measurements, local decisions, optimal local predictions. Each controller is based on a partial (local) model of the overall dynamics, possibly neglecting existing dynamical interactions. The global performance objective is suitably mapped into a local objective for each of the local MPC problems.

This deliverable surveys some of the main contributions to DMPC appeared in the literature and reports about a method developed within the WIDE Consortium to design decentralized MPC controllers.

1.1 Introduction

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

When considering large-scale systems the presupposition of centrality fails because of the lack of a centralized information-gathering system or of centralized computing capabilities. Typical examples of such systems are power networks, water networks, urban traffic networks, cooperating vehicles, digital cellular networks, flexible manufacturing networks, supply chains, complex structures in civil engineering, and many others. In such systems the centrality assumption often fails because of geographical separation of components (spatial distribution), as the costs and the reliability of communication links cannot be neglected. Moreover, technological advances and reduced cost of microprocessors provide a new force for distributed computation. Hence the current trend for *decentralized* decision making, distributed computations, and hierarchical control.

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

Ideas for decentralizing and hierarchically organizing the control actions in industrial automation systems date back to the 70's [9, 26, 27, 33, 39], but were mainly limited to the analysis of stability of decentralized linear control of interconnected subsystems, so the interest faded. Since the late 90's, because of the advances in computation techniques like convex optimization, the interest in decentralized control raised again [13, 31], and convex formulations were developed, although limited to special classes of systems such as spatially invariant systems [4]. Decentralized control and estima-

tion schemes based on distributed convex optimization ideas have been proposed recently in [19,32] based on Lagrangean relaxations. Here global solutions can be achieved after iterating a series of local computations and inter-agent communications.

Large-scale multi-variable control problems, such as those arising in the process industries, are often dealt with model predictive control (MPC) techniques. In MPC the control problem is formulated as an optimization one, where many different (and possibly conflicting) goals are easily formalized and state and control constraints can be included. Many results are nowadays available concerning stability and robustness of MPC, see e.g. [24]. However, centralized MPC is often unsuitable for control of large-scale networked systems, mainly due to lack of scalability and to maintenance issues of global models. In view of the above considerations, it is then natural to look for *decentralized* or for *distributed* MPC (DMPC) algorithms, in which the original large-size optimization problem is replaced by a number of smaller and easily tractable ones that work iteratively and cooperatively towards achieving a common, system-wide control objective.

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

- *decentralized MPC*: Control agents take control decisions independently on each other. Information exchange (such as measurements and previous control decisions) is only allowed before and after the decision making process. There is no negotiation between agents during the decision process. The time needed to decide the control action is not affected by communication issues, such as network delays and loss of packets.
- *distributed MPC*: An exchange of candidate control decisions may also happen during the decision making process, and iterated until an agreement is reached among the different local controllers, in accordance with a given stopping criterion.

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

A DMPC control layer is often interacting with a higher-level control layer in a hierarchical arrangement, as depicted in Figure 1. The goal of the higher layer is to possibly adjust set-points and constraint specifications to the DMPC layer, based on a *global* (possibly less detailed) model of the entire system. Because of its general overview of the entire process, such a centralized decision layer allows one to reach levels of coordination and performance optimization otherwise very difficult (if not impossible) using a decentralized or distributed action. For a recent survey on decentralized, distributed and hierarchical model predictive control architectures, the reader is referred to the recent survey paper [34].

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

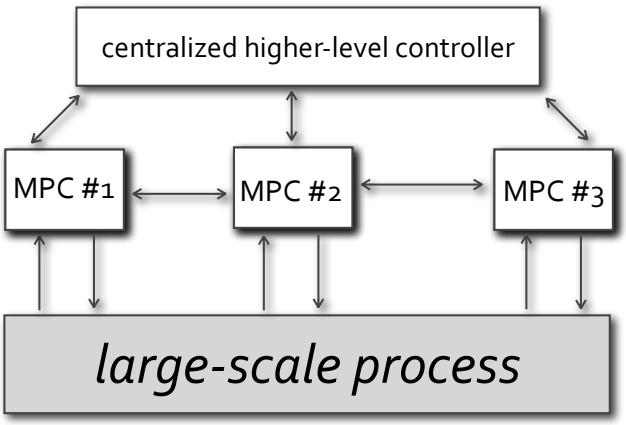


Figure 1: Hierarchical and decentralized/distributed model predictive control of a large-scale process

1.2 Model Predictive Control

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

$$\begin{cases} x(t+1) &= Ax(t)+Bu(t) \\ y(t) &= Cx(t) \end{cases} \tag{1}$$

to the origin while fulfilling the constraints

$$u_{\min} \leq u(t) \leq u_{\max} \tag{2}$$

at all time instants $t \in \mathbb{Z}_{0+}$ where $\mathbb{Z}_{0+}$ is the set of nonnegative integers, $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^p$ are the state, input, and output vectors, respectively, and the pair (A,B) is stabilizable. In (2) the constraints should be interpreted component-wise and we assume $u_{min} < 0 < u_{max}$.

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

$$V(x(t)) = \min_{U} \quad x'_{t+N} P x_{t+N} + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \tag{3a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k,\ k = 0,\dots,N-1 \tag{3b}$$

$$y_k = Cx_k,\ k = 0,\dots,N \tag{3c}$$

$$x_0 = x(t) \tag{3d}$$

$$u_{\min} \leq u_k \leq u_{\max},\ k = 0,\dots,N_u-1 \tag{3e}$$

$$u_k = Kx_k,\ k = N_u,\dots,N-1 \tag{3f}$$

is solved, where $U \triangleq \{u_0,\dots,u_{N_u-1}\}$ is the sequence of future input moves, x_k denotes the predicted state vector at time $t+k$, obtained by applying the input sequence $u_0,\dots,u_{k-1}$ to model (1), starting from $x(t)$. In (3) $N > 0$ is the prediction horizon, $N_u \leq N-1$ is the input horizon, $Q = Q' \geq 0$, $R = R' > 0$, $P = P' \geq 0$ are square weight matrices defining the performance index, and K is some terminal feedback gain. As we will discuss below, P, K are chosen in order to ensure closed-loop stability of the overall process.

Problem (3) can be recast as a quadratic programming (QP) problem (see e.g. [6,24]), whose solution $U^*(x(t)) \triangleq \{u_0^* \ \dots \ u_{N_u-1}^*\}$ is a sequence of optimal control inputs. Only the first input

$$u(t) = u_0^* \tag{4}$$

acronym	Section	submodels	constraints	intersampling iterations	broadcast predictions	state constr.	stability constr.	references
JK	1.3.1	coupled	local inputs	no	yes	yes	yes	[10, 16, 17]
VRW	1.3.2	coupled	local inputs	yes	no	no	none	[35, 36]
DM	1.3.3	decoupled	local inputs	no	yes	yes	yes	[14]
KBB	1.3.4	decoupled		no	yes	yes	none	[20]
MD	1.3.5	coupled	local inputs	yes	yes	no	none	[25]
MS	1.3.6	coupled	local inputs	no	no	no	yes	[23]
ABB	1.4	coupled	local inputs	no	no	no	none	[2, 3], [5]*, [1]*

Table 1: Classification of existing DMPC approaches.

is actually applied to system (1), as the optimization problem (3) is repeated at time $t+1$, based on the new state $x(t+1)$ (for this reason, the MPC strategy is often referred to as *receding horizon* control). The MPC algorithm (3)-(4) requires that all the n components of the state vector $x(t)$ are collected in a (possibly remote) central unit, where a quadratic program with mN_u decision variables needs to be solved and the solution broadcasted to the m actuators. As mentioned in the introduction, such a centralized MPC approach may be inappropriate for control of large-scale systems, and it is therefore natural to look for decentralized or distributed MPC (DMPC) algorithms.

1.3 A Survey of Existing DMPC Approaches

A few contributions have appeared in recent years in the context of DMPC, mainly motivated by applications of decentralized control of cooperating air vehicles [7, 21, 30]. We review in this section some of the main contributions on DMPC, summarized in Table 1, that have appeared in the scientific literature, including the contributions developed within the WIDE project that will be described in Section 1.4. An application of some of the results surveyed in this section in a problem of distributed control of power networks with comparisons among DMPC approaches is reported in [12].

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

1.3.1 DMPC approach of Jia and Krogh

In [10, 16] the system under control is composed by a number of unconstrained linear discrete-time subsystems with decoupled input signals, described by the equations

$$\begin{bmatrix} x_1(k+1) \\ \vdots \\ x_M(k+1) \end{bmatrix} = \begin{bmatrix} A_{11} & \dots & A_{1M} \\ \vdots & \ddots & \vdots \\ A_{M1} & \dots & A_{MM} \end{bmatrix} \begin{bmatrix} x_1(k) \\ \vdots \\ x_M(k) \end{bmatrix} + \begin{bmatrix} B_1 & & 0 \\ & \ddots & \\ 0 & & B_M \end{bmatrix} \begin{bmatrix} u_1(k) \\ \vdots \\ u_M(k) \end{bmatrix} \tag{5}$$

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

$$x_j(k+i+i|k) = A_{jj}x_j(k+i|k) + B_j + u_j(k+i|k) + K_j v_j(k+i|k) \tag{6}$$

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

* Work supported by the European Commission under project "WIDE - Decentralized and Wireless Control of Large-Scale Systems", contract number FP7-IST-224168.

receives the signal

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

where i is the prediction time index, from the other MPC controllers at the end of the previous time step $k-1$. The signal $v_j(k+i|k)$ is used by controller #j at time k to estimate the effect of the neighboring subsystem dynamics in (6).

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

The authors deal with state constraints in [17] by proposing a min-max approach, at the price of a possible conservativeness of the approach.

1.3.2 DMPC approach of Venkat, Rawlings, and Wright

In [35–37] the authors propose distributed MPC algorithm based on a process of negotiations among DMPC agents. The adopted prediction model is

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

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

Starting from a multiobjective formulation, the authors distinguish between a "communication-based" control scheme, in which each controller #i is optimizing his own local performance index Φ_i, and a "cooperation-based" control scheme, in which each controller #i is optimizing a weighted sum $\sum_{j=1}^{M} \alpha_j \Phi_j$ of all performance indices, $0 \leq \alpha_j \leq 1$. As performance indices depend on the decisions taken by the other controllers, at each time step k a sequence of iterations is taken before computing and implementing the input vector $u(k)$. In particular, within each sampling time k, at every iteration p the previous decisions $u_{j\neq i}^{p-1}$ are broadcast to controller #i, in order to compute the new iterate u_i^p. With the communication-based approach, the authors show that if the sequence of iterations converges, it converges to a Nash equilibrium. With the cooperation-based approach, convergence to the optimal (centralized) control performance is established. In practical situations the process sampling interval may be insufficient for the computation time required for convergence of the iterative algorithm, with a consequent loss of performance. Nonetheless, closed-loop stability is not compromised: as it is achieved even though the convergence of the iterations is not reached. Moreover, all iterations are plantwide feasible, which naturally increases the applicability of the approach including a certain robustness to transmission faults.

1.3.3 DMPC approach of Dunbar and Murray

In [14] the authors consider the control of a special class of dynamically decoupled continuous-time nonlinear subsystems

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t))$$

where the local states of each model represent a position and a velocity signal

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

State vectors are only coupled by a global performance objective

$$L(x,u) = \sum_{(i,j)\in\mathscr{E}_0} \omega\|q_i - q_j + d_{ij}\|^2 + \omega\|q_\Sigma - q_d\|^2 + \nu\|\dot{q}\|^2 + \mu\|u\|^2 \tag{7}$$

under local input constraints $u_i(t) \in U$, $\forall i = 1,\ldots,M$, $\forall t \geq 0$. In (7) $\mathscr{E}_0$ is the set of pair-wise neighbors, d_{ij} is the desired distance between subsystems i and j, $q_\Sigma = (q_1 + q_2 + q_3)/3$ is the average position of the leading subsystems 1,2,3, and $q_d = (q_1^c + q_2^c + q_3^c)/3$ the corresponding target.

The overall integrated cost (7) is decomposed in distributed integrated cost functions

$$L_i(x_i, x_{-i}, u_i) = L_i^x(x_i, x_{-i}) + \gamma\mu\|u_i\|^2 + L^d(i)$$

where $x_{-i} = (x_{j1},\ldots,x_{jk})$ collects the states of the neighbors of agent subsystem #i, $L_i^x(x_i, x_{-i}) = \sum_{j\in N_i} \frac{\gamma\omega}{2}\|q_i - q_j + d_{ij}\|^2 + \gamma\nu\|\dot{q}_i\|^2$, and

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

It holds that

$$L(x,u) = \frac{1}{\gamma}\sum_{i=1}^{N} L_i(x_i, x_{-i}, u_i)$$

Before computing DMPC actions, neighboring subsystems broadcast in a synchronous way their states, and each agent transmits and receives an "assumed" control trajectory $\hat{u}_i(\tau;t_k)$. Denoting by $u_i^p(\tau;t_k)$ the control trajectory predicted by controller #i, by $u_i^*(\tau;t_k)$ the optimal predicted control trajectory, by T the prediction horizon, and by $\delta \in (0,T]$ the update interval, the following DMPC performance index is minimized

$$\begin{aligned}
\min_{u_i^p} \quad J_i(x_i(t_k), x_{-i}(t_k), u_i^p(\cdot;t_k)) \quad = \quad \min_{u_i^p} \quad & \int_{t_k}^{t_k+T} L_i(x_i^p(s;t_k), \hat{x}_{-i}(s;t_k), u_i^p(s;t_k))ds + \gamma\|x_i^p(t_k+T;t_k) - x_i^C\|_{P_i}^2 \\
\text{s.t.} \quad & \dot{x}_i^p(\tau;t_k) = f_i(x_i^p(\tau;t_k), u_i^p(\tau;t_k)) \\
& \dot{\hat{x}}_i^p(\tau;t_k) = f_i(\hat{x}_i^p(\tau;t_k), \hat{u}_i^p(\tau;t_k)) \\
& \dot{\hat{x}}_{-i}^p(\tau;t_k) = f_{-i}(\hat{x}_{-i}^p(\tau;t_k), \hat{u}_{-i}^p(\tau;t_k)) \\
& u_i^p(\tau;t_k) \in U \\
& \|x_i^p(\tau;t_k) - \hat{x}_i(\tau;t_k)\| \leq \delta^2\kappa \\
& x_i^p(t_k+T;t_k) \in \Omega_i(\varepsilon_i)
\end{aligned}$$

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

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

1.3.4 DMPC approach of Keviczy, Borrelli, and Balas

Dynamically decoupled submodels are also considered in [20], where the special nonlinear discrete-time system structure

$$x_{k+1}^i = f^i(x_k^i, u_k^i)$$

is assumed, subject to local input and state constraints $x_k^i \in \mathscr{X}^i,\ u_k^i \in \mathscr{U}^i,\ i = 1,\ldots,M$. Subsystems are coupled by the cost function

$$l(\tilde{x},\tilde{u}) = \sum_{i=1}^{N_v} l^i(x^i,u^i,\tilde{x}^i,\tilde{u}^i)$$

and by the global constraints

$$g^{i,j}(x^i,u^i,x^j,u^j) \leq 0,\ (i,j) \in \mathscr{A}$$

where $\mathscr{A}$ is a given set. Each local MPC controller is based on the optimization of the following problem

$$\begin{aligned}
\min_{\tilde{U}_t} \quad & \sum_{k=0}^{N-1} l(\tilde{x}_{k,t},\tilde{u}_{k,t}) + l_N(\tilde{x}_{N,t}) && \text{(8a)}\\
\text{s.t.} \quad & x_{k+1,t}^i = f^i(x_{k,t}^i, u_{k,t}^i) && \text{(8b)}\\
& x_{k,t}^i \in \mathscr{X}^i, \quad u_{k,t}^i \in \mathscr{U}^i,\ k = 1,\ldots,N-1 && \text{(8c)}\\
& x_{N,t}^i \in \mathscr{X}_f^i && \text{(8d)}\\
& x_{k+1,t}^j = f^j(x_{k,t}^j, u_{k,t}^j), (i,j) \in \mathscr{A} && \text{(8e)}\\
& x_{k,t}^j \in \mathscr{X}^j, \quad u_{k,t}^j \in \mathscr{U}^j, (i,j) \in \mathscr{A}\ k = 1,\ldots,N-1 && \text{(8f)}\\
& x_{N,t}^j \in \mathscr{X}_f^j, (i,j) \in \mathscr{A} && \text{(8g)}\\
& g^{i,j}(x_{k,t}^i, u_{k,t}^i, x_{k,t}^j, u_{k,t}^j) \leq 0, (i,j) \in \mathscr{A}\ k = 1,\ldots,N-1 && \text{(8h)}\\
& x_{0,t}^i = x_t^i, \tilde{x}_{0,t}^i = \tilde{x}_t^i && \text{(8i)}
\end{aligned}$$

where (8b)–(8d) are the local model and constraints of the agent, (8e)–(8g) are the model and constraints of the neighbors, and (8h) represent interaction constraints of agent #*i* with its own neighbors.

The information exchanged among the local MPC agents are the neighbors' current states, terminal regions, and local models and constraints. As in (18), only the optimal input $u_{0,t}^i$ computed by controller #*i* is applied; the remaining inputs $u_{k,t}^j$ are completely discarded, as they are only used to enhance the prediction.

Stability is analyzed for the problem without coupling constraints (8h), under the assumption that the following inequality holds

$$\sum_{k=1}^{N-1} 2\|Q(x_{k,t}^{j,j} - x_{k,t}^{j,i})\|_p + \|R(u_{k,t}^{j,j} - u_{k,t}^{j,i})\|_p \leq \|Qx_t^i\|_p + \|Qx_t^j\|_p + \|Q(x_t^i - x_t^j)\|_p + \|Ru_{0,t}^{i,i}\|_p + \|Ru_{0,t}^{j,i}\|_p$$

where $\|Qx\|_2 \triangleq x'Qx$, and $\|Qx\|_1$, $\|Qx\|_\infty$ are the standard q and ∞ norm, respectively.

1.3.5 DMPC approach of Mercangöz and Doyle

The distributed MPC *and* estimation problems are considered in [25] for square plants (the number of inputs equals the number of outputs) perturbed by noise, whose local prediction models are

$$\begin{cases} x_i(k+1) &= A_i x_i(k) + B_i u_i(k) + \sum_{j=1}^M B_j u_j(k) + w_i(k) \\ y_i(k) &= C_i x_i(k) + v_i(k) \end{cases} \qquad (9)$$

A distributed Kalman filter based on the local submodels (9) is used for state estimation. The DMPC approach is similar to Venkat et al.'s "communication-based" approach, although only first moves $u_j(k)$ are transmitted and assumed frozen in prediction, instead of the entire optimal sequences. Only constraints on local inputs are handled by the approach. Although general stability and convergence results are not proved in [25], experimental results on a four-tank system are reported to show the effectiveness of the approach.

1.3.6 DMPC approach of Magni and Scattolini

Another interesting approach to decentralized MPC for nonlinear systems has been formulated in [23]. The problem of regulating a nonlinear system affected by disturbances to the origin is considered under some technical assumptions of regularity of the dynamics and of boundedness of the disturbances. Closed-loop stability is ensured by the inclusion in the optimization problem of a contractive constraint. The considered class of functions and the absence of information exchange between controllers leads to some conservativeness of the approach.

1.4 WIDE Approach to Decentralized MPC

Based on preliminary ideas developed by UNISI in [2, 3], within the WIDE project a new approach to decentralized MPC design has been proposed.

The approach aims at addressing possible dynamical coupling between subprocesses in a nonconservative way, and to actually exploit such coupling to improve the degree of cooperativeness of local controllers. A (partial) decoupling assumption only appears in the *prediction* models used by different MPC controllers. The chosen degree of decoupling represents a tuning knob of the approach. Sufficient criteria for analyzing the asymptotic stability of the process model in closed loop with the set of decentralized MPC controllers are provided. If such conditions are not verified, then the structure of decentralization should be modified by augmenting the level of dynamical coupling of the prediction submodels, increasing consequently the number and type of exchanged information about state measurements among the MPC controllers. Following such stability criteria, a hierarchical scheme was proposed to change the decentralization structure on-line by a supervisory scheme without destabilizing the system. Moreover, to cope with the case of a non-ideal communication channel among neighboring MPC controllers, sufficient conditions for ensuring closed-loop stability of the overall closed-loop system when packets containing state measurements may be lost were given.

We review here the main ingredients and results of this approach, pointing the reader for deeper technical details to the results already published in [1, 5].

1.4.1 Decentralized Prediction Models

Consider again process model (1). Matrices A, B may have a certain number of zero or negligible components corresponding to a partial dynamical decoupling of the process, especially in the case of large-scale systems, or even be block diagonal in case of total dynamical decoupling. This is the case for instance of independent moving agents each one having its own dynamics and only coupled by a global performance index.

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

$$x^i = W_i' x = \begin{bmatrix} x_1^i \\ \vdots \\ x_{n_i}^i \end{bmatrix} \in \mathbb{R}^{n_i} \tag{10a}$$

where $W_i \in \mathbb{R}^{n \times n_i}$ collects the n_i columns of the identity matrix of order n corresponding to the indices in $\mathscr{I}_{xi}$, and, similarly,

$$u^i = Z_i' u = \begin{bmatrix} u_1^i \\ \vdots \\ u_{m_i}^i \end{bmatrix} \in \mathbb{R}^{m_i} \tag{10b}$$

as the vector of input signals tackled by the i-th controller, where $Z_i \in \mathbb{R}^{m \times m_i}$ collects m_i columns of the identity matrix of order m corresponding to the set of indices $\mathscr{I}_{ui} \subseteq \{1,\ldots,m\}$. Note that

$$W_i' W_i = I_{n_i},\ Z_i' Z_i = I_{m_i},\ \forall i = 1,\ldots,M \tag{11}$$

where $I_{(\cdot)}$ denotes the identity matrix of order $(\cdot)$. By definition of x^i in (10a) we obtain

$$x^i(t+1) = W_i' x(t+1) = W_i' A x(t) + W_i' B u(t) \tag{12}$$

An *approximation* of (1) is obtained by changing $W_i' A$ in (12) into $W_i' A W_i W_i'$ and $W_i' B$ into $W_i' B Z_i Z_i'$, therefore getting the new prediction reduced order model

$$x^i(t+1) = A_i x^i(t) + B_i u^i(t) \tag{13}$$

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

The size (n_i, m_i) of model (13) in general will be much smaller than the size (n,m) of the overall process model (1). The choice of the pair (W_i, Z_i) of *decoupling matrices* (and, consequently, of the dimensions n_i, m_i of each submodel) is a tuning knob of the DMPC procedure proposed in the sequel of the paper.

We want to design a controller for each set of moves u^i according to prediction model (13) and based on feedback from x^i, for all $i = 1,\ldots,M$. Note that in general different input vectors u^i, u^j may share common components. To avoid ambiguities on the control action to be commanded to the process, we impose that only a subset $\mathscr{I}_{ui}^{\#} \subseteq \mathscr{I}_{ui}$ of input signals computed by controller #i is actually applied to the process, with the following conditions

$$\bigcup_{i=1}^{M} I_{ui}^{\#} = \{1,\ldots,m\} \tag{14a}$$

$$I_{ui}^{\#} \cap I_{uj}^{\#} = \emptyset,\ \forall i,j = 1,\ldots,M,\ i \neq j \tag{14b}$$

Condition (14a) ensures that all actuators are commanded, condition (14b) that each actuator is commanded by only one controller. For the sake of simplicity of notation, since now on we assume that $M = m$ and that $I_{ui}^{\#} = i$, $i = 1,\ldots,m$, i.e., that each controller #i only controls the ith input signal. As observed earlier, in general $\mathscr{I}_{x^i} \cap \mathscr{I}_{x^j} \neq \emptyset$, meaning that controller #i may partially share the same feedback information with controller #j, and $\mathscr{I}_{ui} \cap \mathscr{I}_{uj} \neq \emptyset$. This means that controller #i may take into account the effect of control actions that are actually decided by another controller #j, $i \neq j$, $i,j = 1,\ldots,M$, which ensures a certain degree of cooperation.

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

1.4.2 Decentralized Optimal Control Problems

In order to exploit submodels (13) for formulating local finite-horizon optimal control problems that lead to an overall closed-loop stable DMPC system, let the following assumptions be satisfied.

Assumption 1. *Matrix A in (1) is strictly Hurwitz.*

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

Assumption 2. *Matrix A_i is strictly Hurwitz, $\forall i = 1,\dots,M$.*

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

$$V_i(x(t)) = \min_{\{u_k^i\}_{k=0}^{\infty}} \sum_{k=0}^{\infty} (x_k^i)' W_i' Q W_i x_k^i + (u_k^i)' Z_i' R Z_i u_k^i = \tag{15a}$$

$$= \min_{u_0^i} (x_1^i)' P_i x_1^i + x^i(t)' W_i' Q W_i x^i(t) + (u_0^i)' Z_i' R Z_i u_0^i \tag{15b}$$

$$\text{s.t. } x_1^i = A_i x^i(t) + B_i u_0^i \tag{15c}$$

$$x_0^i = W_i' x(t) = x^i(t) \tag{15d}$$

$$u_{\min}^i \le u_0^i \le u_{\max}^i \tag{15e}$$

$$u_k^i = 0,\ \forall k \ge 1 \tag{15f}$$

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

$$A_i' P_i A_i - P_i = -W_i' Q W_i \tag{16}$$

that exists by virtue of Assumption 2. Problem (15) corresponds to a finite-horizon constrained problem with control horizon $N_u = 1$ and linear stable prediction model. Note that only the local state vector $x^i(t)$ is needed to solve Problem (15).

At time t, each controller MPC #i measures (or estimates) the state $x^i(t)$ (usually corresponding to local and neighboring states), solves problem (15) and obtains the optimizer

$$u_0^{*i} = [u_0^{*i1}, \dots, u_0^{*ii}, \dots, u_0^{*im_i}]' \in \mathbb{R}^{m_i} \tag{17}$$

In the simplified case $M = m$ and $I_{ui}^{\#} = i$, only the i-th sample of u_0^{*i}

$$u_i(t) = u_0^{*ii} \tag{18}$$

will determine the i-th component $u_i(t)$ of the input vector actually commanded to the process at time t. The inputs u_0^{*ij}, $j \ne i$, $j \in \mathscr{I}_{ui}$ to the neighbors are discarded, their only role is to provide a better prediction of the state trajectories x_k^i, and therefore a possibly better performance of the overall closed-loop system.

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

$$u(t) = [u_0^{*11} \ \dots \ u_0^{*ii} \ \dots \ u_0^{*mm}]' \tag{19}$$

While usually a matrix A is called *Hurwitz* if all its eigenvalues have strictly negative real part (continuous-time case), in this paper we say that a matrix A is *Hurwitz* if all the eigenvalues λ_i of A are such that $|\lambda_i| < 1$ (discrete-time case).

is the actual input commanded to process (1). The optimizations (15) are repeated at time $t+1$ based on the new states $x^i(t+1) = W_i' x(t+1)$, $\forall i = 1,\ldots,M$, according to the usual receding horizon control paradigm. The degree of coupling between the DMPC controllers is dictated by the choice of the decoupling matrices (W_i, Z_i). Clearly, the larger the number of interactions captured by the submodels, the more complex the formulation (and, in general, the solution) of the optimization problems (15) and hence the computations performed by each control agent. Note also that a higher level of information exchange between control agents requires a higher communication overhead. We are assuming here that the submodels are constant at all time steps.

1.4.3 Convergence Properties

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

The following theorem, proved in [1, 2], summarizes the closed-loop convergence results of the proposed DMPC scheme using the cost function $V(x(t)) \triangleq \sum_{i=1}^{M} V_i(W_i' x(t))$ as a Lyapunov function for the overall system.

Theorem 1. *Let Assumptions 1, 2 hold and define P_i as in* (16) $\forall i = 1,\ldots,M$. *Define*

$$\begin{array}{ll} \Delta u^i(t) \triangleq u(t) - Z_i u_0^{*i}(t), & \Delta x^i(t) \triangleq (I - W_i W_i')x(t) \\ \Delta A^i \triangleq (I - W_i W_i')A, & \Delta B^i \triangleq B - W_i W_i' B Z_i Z_i' \end{array} \tag{20}$$

Also, let

$$\Delta Y^i(x(t)) \triangleq W_i W_i' (A \Delta x^i(t) + B Z_i Z_i' \Delta u^i(t)) + \Delta A^i x(t) + \Delta B^i u(t) \tag{21a}$$

and

$$\Delta S^i(x(t)) \triangleq \left(2(A_i W_i' x(t) + B_i u_0^{*i}(t))' + \Delta Y^i(x(t))' W_i\right) P_i W_i' \Delta Y^i(x(t)) \tag{21b}$$

If the condition

$$(i)\ x' \left(\sum_{i=1}^{M} W_i W_i' Q W_i W_i' \right) x - \sum_{i=1}^{M} \Delta S^i(x) \geq 0, \quad \forall x \in \mathbb{R}^n \tag{22a}$$

is satisfied, or the condition

$$(ii)\ x' \left(\sum_{i=1}^{M} W_i W_i' Q W_i W_i' \right) x - \alpha x' x - \sum_{i=1}^{M} \Delta S^i(x) + \sum_{i=1}^{M} u_0^{*i}(x)' Z_i' R Z_i u_0^{*i}(x) \geq 0, \quad \forall x \in \mathbb{R}^n \tag{22b}$$

is satisfied for some scalar $\alpha > 0$, then the decentralized MPC scheme defined in (15)–(19) in closed loop with (1) is globally asymptotically stable.

By using the explicit MPC results of [6], each optimizer function $u_0^{*i} : \mathbb{R}^n \mapsto \mathbb{R}^{m_i}$, $i = 1,\ldots,M$, can be expressed as a piecewise affine function of x

$$u_0^{*i}(x) = F_{ij} x + G_{ij} \quad \text{if} \quad H_{ij} x \leq K_{ij}, \quad j = 1,\ldots,n_{ri} \tag{23}$$

Hence, both condition (22a) and condition (22b) are a composition of quadratic and piecewise affine functions, so that *global stability* can be tested through linear matrix inequality relaxations [18] (cf. the approach of [15]).

As $u_{\min} < 0 < u_{\max}$, there exists a ball around the origin $x = 0$ contained in one of the regions, say $\{x \in \mathbb{R}^n : \ H_{i1} x \leq K_{i1}\}$, such that $G_{i1} = 0$. Therefore, around the origin both (22a) and (22b) become

a quadratic form $x'(\sum_{i=1}^{M} E_i)x$ of x, where matrices E_i can be easily derived from (20), (21) and (22). Hence, *local stability* of (15)–(19) in closed loop with (1) can be simply tested by checking the positive semidefiniteness of the square $n \times n$ matrix $\sum_{i=1}^{M} E_i$. Note that, depending on the degree of decentralization, in order to satisfy the sufficient stability criterion one may need to set $Q > 0$ in order to dominate the unmodeled dynamics arising from the terms ΔS^i.

In the absence of input constraints, Assumptions 1, 2 can be removed to extend the previous DMPC scheme to the case where (A,B) and (A_i,B_i) may not be Hurwitz, although stabilizable.

Theorem 2 ([1,3]). *Let the pairs (A_i,B_i) be stabilizable, $\forall i = 1,\ldots,M$. Let Problem (15) be replaced by*

$$V_i(x(t)) = \min_{\{u_k^i\}_{k=0}^{\infty}} \sum_{k=0}^{\infty} (x_k^i)' W_i' Q W_i x_k^i + (u_k^i)' Z_i' R Z_i u_k^i = \tag{24a}$$

$$= \min_{u_0^i} (x_1^i)' P_i x_1^i + x^i(t)' W_i' Q W_i x^i(t) + (u_0^i)' Z_i' R Z_i u_0^i \tag{24b}$$

$$\text{s.t. } x_1^i = A_i x^i(t) + B_i u_0^i \tag{24c}$$

$$x_0^i = W_i' x(t) = x^i(t) \tag{24d}$$

$$u_k^i = K_{LQ_i} x_k^i, \ \forall k \geq 1 \tag{24e}$$

where $P_i = P_i' \geq 0$ is the solution of the Riccati equation

$$W_i' Q W_i + K_{LQ_i}' Z_i' R Z_i K_{LQ_i} + (A_i + B_i K_{LQ_i})' P_i (A_i + B_i K_{LQ_i}) = P_i \tag{25}$$

and $K_{LQ_i} = -(Z_i' R Z_i + B_i' P_i B_i)^{-1} B_i' P_i A_i$ is the corresponding local LQR feedback.

Let $\Delta Y^i(x(t))$ and let $\Delta S^i(x(t))$ be defined as in (21)*, in which P_i is defined as in* (25)*.*

If condition (22a) *is satisfied, or condition* (22b) *is satisfied for some scalar $\alpha > 0$, then the decentralized MPC scheme defined in (24), (19) in closed-loop with (1) is globally asymptotically stable.*

1.4.4 Decentralized MPC over Wireless Networks

So far we assumed that the communication model among neighboring MPC controllers is faultless, so that each MPC agent successfully receives the information about the states of its corresponding submodel. However, one of the main issues in networked control systems is the unreliability of communication channels, which may result in data packet dropout.

In line with the main *leitmotif* of the WIDE project, a sufficient condition for ensuring convergence of the DMPC closed-loop in the case packets containing measurements are lost for an arbitrary but upper-bounded number N of consecutive time steps was proved in [1,5], and is reported in Deliverable D4.2.

1.5 Conclusions

This section of the deliverable has surveyed the state of the art in decentralized model predictive control (DMPC), and has proposed a novel approach developed by UNISI/UNITN within the WIDE project. The problem addressed is the one of controlling a distributed, possibly large-scale, process through the cooperation of multiple decentralized model predictive controllers. Each controller is based on a submodel of the overall process, and different submodels may share common states and inputs, to possibly decrease modeling errors in case of dynamical coupling, and to increase the level of cooperativeness of the controllers. The DMPC approach is suitable for control of large-scale

systems subject to constraints: the possible loss of global optimal performance is compensated by the gain in controller scalability, reconfigurability, and maintenance.

Although a few contributions have been given in the last few years, the theory of DMPC is not yet mature and homogenous. In this deliverable we have tried to highlight similarities and differences among the various approaches that have been proposed, to place the contributions of the WIDE consortium in the right perspective.

The algorithms associated with the approach proposed in this deliverable have been included in the MATLAB/Simulink WIDE Toolbox.

2 Distributed model predictive control

2.1 Introduction

Distributed MPC is characteristic by communication between individual controllers during computation of control action in each sampling period. This type of control is communication intensive; however, it does not sacrifice control performance as decentralized control and can achieve practically the same optimality results as centralized MPC.

Distributed MPC will be demonstrated in the first part of this section on an example of controlling two interconnected tanks. Then this example will be extended to multiple tanks and finally it will be applied to the model of Barcelona water distribution network.

The distributed MPC is based on a classical dual decomposition of constrained optimization problem [11]. The dual problem is solved by Nesterov accelerated gradient method [28, 29] for control without coordinator and by limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method [8, 22].

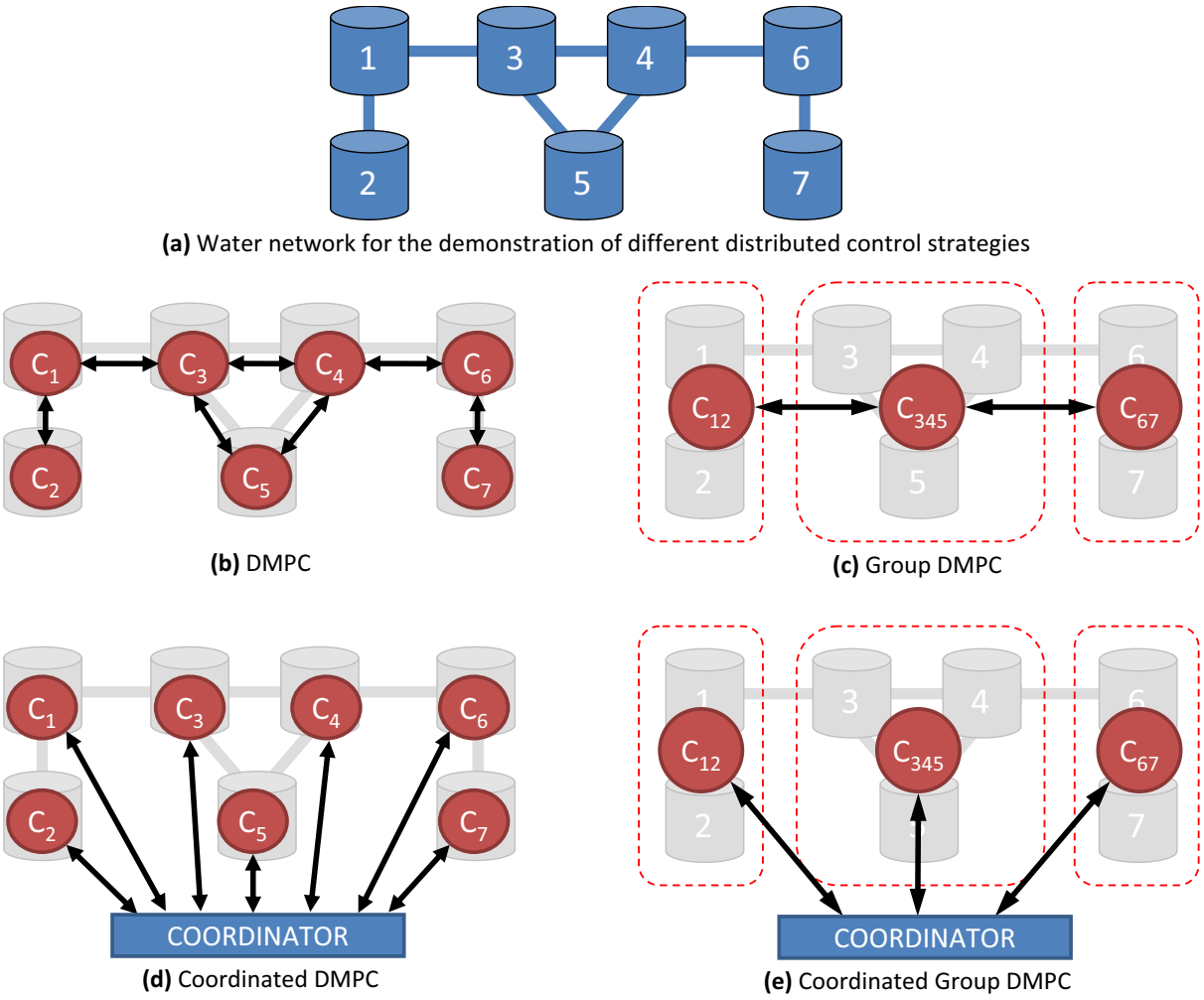


Figure 2: Demonstration of the different types of distributed control on a simple water network.

Four different types of distribution schemes are considered. This will be illustrated on a simple water network (Figure 2a). The first strategy denoted simply as "Distributed control" assumes that each tank has its own controller, which communicates only with neighboring tank controllers (Figure 2b). Neighboring is meant in the sense of having interconnection by valve or pump. The second strategy denoted as "Distributed group control" is similar to the first one, but the tanks are aggregated to groups. Each group has its own controller, which communicates with another group controller only if

there is any interconnection between groups (Figure 2c). The third strategy denoted as "Coordinated distributed control" assumes a controller on every tank. There is no direct communication between these controllers. They communicate only with central coordinator (Figure 2d). The fourth strategy denoted as "Coordinated distributed group control" mixes tanks aggregation and groups controllers communication only with coordinator (Figure 2e).

The advantages and disadvantages of different control strategies are summarized in Table 2.

	number of iterations	modularity	computation time ()
DMPC	+++	+++	+
Group DMPC	+	+	++
Coordinated DMPC	+	++	- -
Coordinated Group DMPC	- -	+	-

Table 2: Comparison of different DMPC strategies

2.2 Two-tanks distributed control

This section demonstrates the concept of distributed control on a simple example of two interconnected tanks (Figure 3). The objective is to fulfill water demands ($\mathbf{d}_1$,$\mathbf{d}_2$) while minimizing costs of fresh water pumping ($\mathbf{u}_1$,$\mathbf{u}_2$) and pumping between tanks ($\mathbf{f}$). The variables $\mathbf{d}_i$, $\mathbf{u}_i$ and $\mathbf{f}$ are vectors of trajectories on prediction horizon.

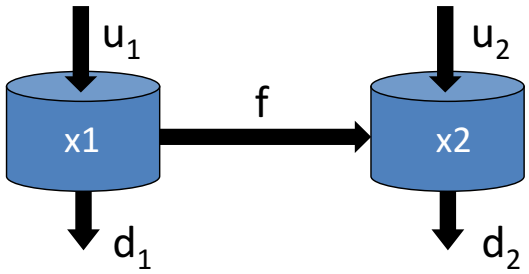


Figure 3: Two tanks example.

The objective can be written as

$$\min_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{f}} J(\mathbf{u}_1,\mathbf{u}_2,\mathbf{f}), \tag{26}$$

where $J(\mathbf{u}_1,\mathbf{u}_2,\mathbf{f})$ is a global convex cost function. This cost can be separated for individual tanks as

$$\min_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{f}} J(\mathbf{u}_1,\mathbf{f}) + J(\mathbf{u}_2,\mathbf{f}). \tag{27}$$

If tanks would not be interconnected by flow $\mathbf{f}$ then this minimization problem would be easily separable to two convex minimization problems. Flow $\mathbf{f}$ complicates this separation and it is therefore called **complicating variable**. Dual decomposition deals with complicating variable by duplicating it into both cost functions and imposing additional equivalence constraint

$$\min_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{f}_1,\mathbf{f}_2} J_1(\mathbf{u}_1,\mathbf{f}_1) + J_2(\mathbf{u}_2,\mathbf{f}_2) \qquad \text{s.t.} \qquad \mathbf{f}_1 = \mathbf{f}_2, \tag{28}$$

and switching to dual problem by introducing Lagrangian with Lagrange multiplier λ

$$L(\mathbf{u}_1,\mathbf{f}_1,\mathbf{u}_2,\mathbf{f}_2,\lambda) = J_1(\mathbf{u}_1,\mathbf{f}_1) + J_2(\mathbf{u}_2,\mathbf{f}_2) + \lambda^T(\mathbf{f}_1 - \mathbf{f}_2). \tag{29}$$

Computational time without time required for communication.

The dual function is

$$g(\lambda) = \min_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{f}_1,\mathbf{f}_2} L(\mathbf{u}_1,\mathbf{f}_1,\mathbf{u}_2,\mathbf{f}_2,\lambda), \tag{30}$$

which is for given λ separable to

$$g(\lambda) = g_1(\lambda) + g_2(\lambda), \tag{31}$$

where

$$\begin{aligned} g_1(\lambda) &= \min_{\mathbf{u}_1,\mathbf{f}_1} J_1(\mathbf{u}_1,\mathbf{f}_1) + \lambda^T \mathbf{f}_1, \\ g_2(\lambda) &= \min_{\mathbf{u}_2,\mathbf{f}_2} J_2(\mathbf{u}_2,\mathbf{f}_2) - \lambda^T \mathbf{f}_2. \end{aligned}$$

Solving dual problem (maximization of $g(\lambda)$) can be done by sub-gradient methods. For example

$$\lambda_{k+1} = \lambda_k + \alpha_k \left(\frac{\partial g_1}{\partial \lambda} + \frac{\partial g_2}{\partial \lambda} \right) = \lambda_k + \alpha_k \left(\mathbf{f}_1^*(\lambda_k) - \mathbf{f}_2^*(\lambda_k) \right), \tag{32}$$

which uses the favorable fact that separated dual functions gradients are equal to optimal values of complicating variables for given Lagrange multiplier

$$\frac{\partial g_1}{\partial \lambda} = \mathbf{f}_1^*(\lambda), \qquad \frac{\partial g_2}{\partial \lambda} = \mathbf{f}_2^*(\lambda). \tag{33}$$

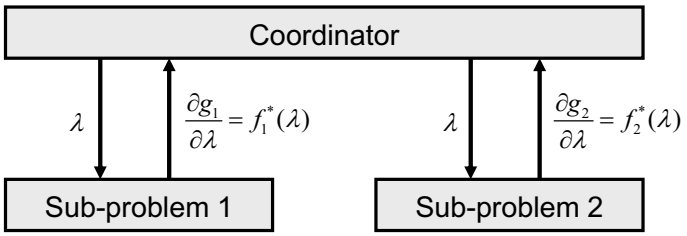


Figure 4: Dual price consensus coordination.

The distributed control scheme is depicted in Figure4. For two tanks it consists of two optimizers and a coordinator. The algorithm outline is following:

1. coordinator chooses initial value λ_1
2. λ_k is distributed to optimizers
3. optimizers compute their optimization problem and return gradient of their dual function $g_i(\lambda)$ with respect to λ, which is equal to optimal flow $\mathbf{f}_i^*(\lambda)$
4. coordinator updates dual prices $\lambda_{k+1} = \lambda_k + \alpha_k \left(\mathbf{f}_1^*(\lambda_k) - \mathbf{f}_2^*(\lambda_k) \right)$
5. stop if $\| \mathbf{f}_1^*(\lambda_k) - \mathbf{f}_2^*(\lambda_k) \| > \varepsilon$, where ε is error threshold, otherwise continue to step 2

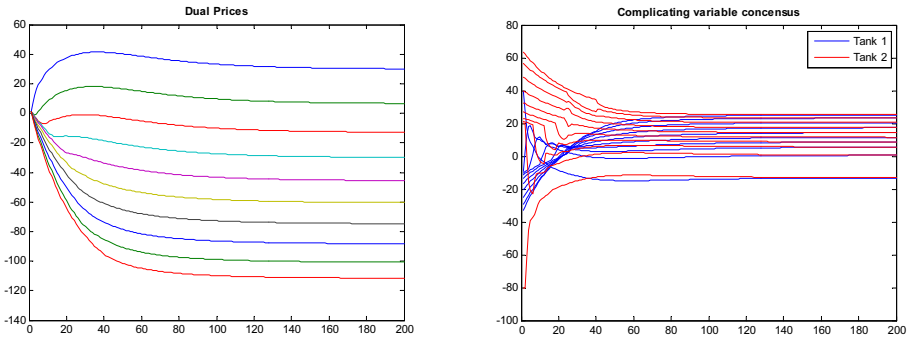
Coordinator can be physically combined with one optimizer. Initial value of λ is selected from the last value in the previous sampling period – warm start.

Two-tanks distributed control - example

Requirements for optimal tanks control (fresh water prices, pumping between tanks price, tank minimum water level soft limit price, MV changes penalty,...) are formulated as MPC with 10 steps prediction horizon. Consensus iterations on the flow between tanks in the first sampling period can

be seen in Figure 5 (cold start). Figure 5a shows convergence of dual prices to stationary values (one price for each step on prediction horizon). Figure 5b shows convergence toward consensus between flows demanded by individual tanks. Second and later sampling periods can start dual prices from the result of previous sampling period and achieve faster convergence.

The comparison between results of centralized and distributed MPC are in Figure 6. The results are practically the same.



(a) Dual prices / Lagrange multipliers convergence

(b) Complicating variable consensus

Figure 5: Consensus iterations in the first sampling period (cold start). Figure **(a)** shows convergence of dual prices and figure **(b)** shows convergence toward consensus between flows demanded by individual tanks on prediction horizon (10 steps).

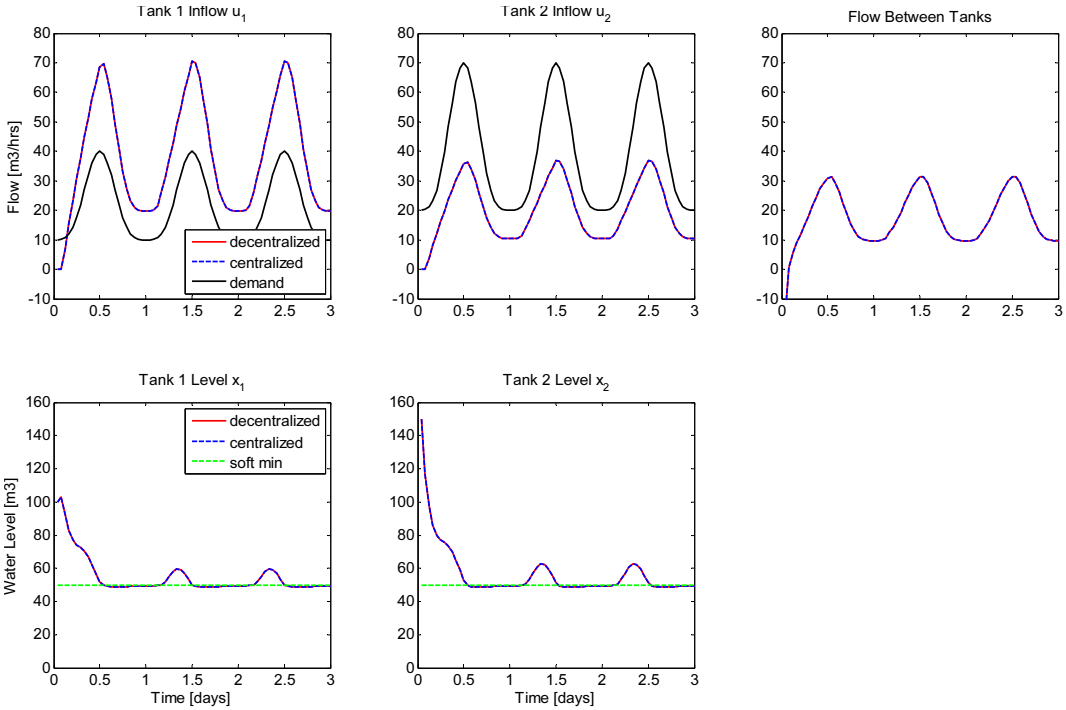


Figure 6: Comparison of trajectories for centralized and distributed MPC. The first two figures show inflow of fresh water to both tanks. The third figure show flow between tanks and the last two figures show water levels in both tanks.

Extending two tanks to arbitrary water network

Two tanks example is very simple; however, it forms the basic principle for distributed control of large water network. This section shows the changes that are needed to extend two tanks example to arbitrary water network.

It is straightforward to extend the example from the previous section to large network where every

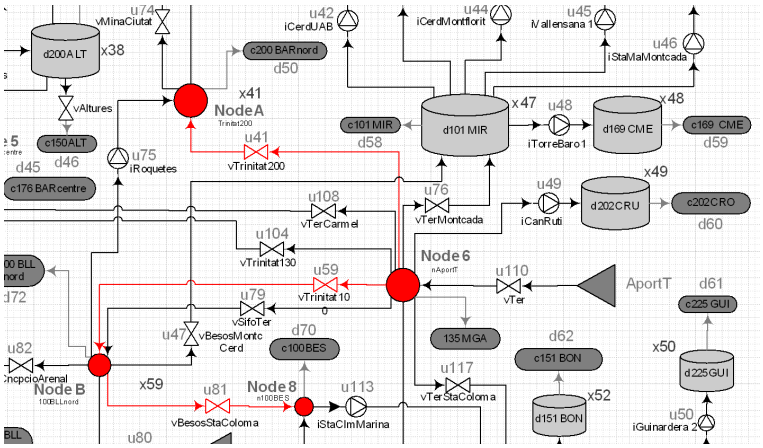


Figure 7: Complex nodes and their interconnections in Barcelona WN.

pump or valve interconnects only two tanks. This type of interconnection is most common; however, large water networks have several nodes, which interconnect together multiple tanks (Figure 7). The flows into these nodes have to preserve mass flow consistency. Assuming that there is no direct interconnection between any two nodes (Figure 8a) it is a simple extension of two tanks by consistency constraint

$$\sum_{i=1}^{N} f_i = 0 \tag{34}$$

which leads to sub-problems

$$\begin{aligned} g_1(\lambda) &= \min_{\mathbf{u}_1,\mathbf{f}_1} J_1(\mathbf{u}_1,\mathbf{f}_1) + \lambda^T \mathbf{f}_1, \\ &\vdots \\ g_N(\lambda) &= \min_{\mathbf{u}_N,\mathbf{f}_N} J_N(\mathbf{u}_N,\mathbf{f}_N) + \lambda^T \mathbf{f}_N, \end{aligned}$$

and coordination update

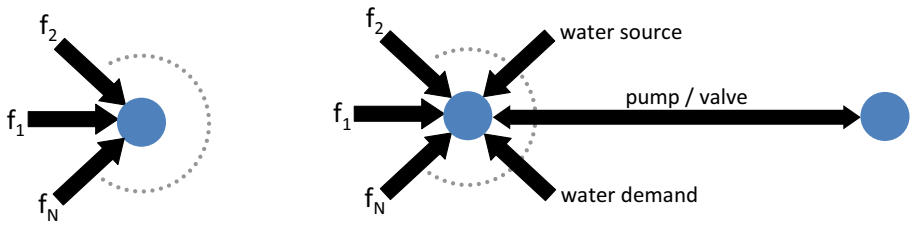
$$\lambda_{k+1} = \lambda_k + \alpha_k \sum_{i=1}^{N} \mathbf{f}_i^*(\lambda_k). \tag{35}$$

The complication is when two nodes are connected by valve or pump with flow limitation and cost function for flow and flow changes (Figure 8b). These limits and costs cannot be incorporated into consistency constraints of interconnected nodes (34). Similar problem is with nodes having their own water source and water demand. There are two solutions:

1. Complex nodes can be modeled as tanks with zero maximum and minimum water level limits.
2. Nodes interconnecting pumps / valves are modeled as stand-alone subsystem withs own cost function, limits and two external flows.

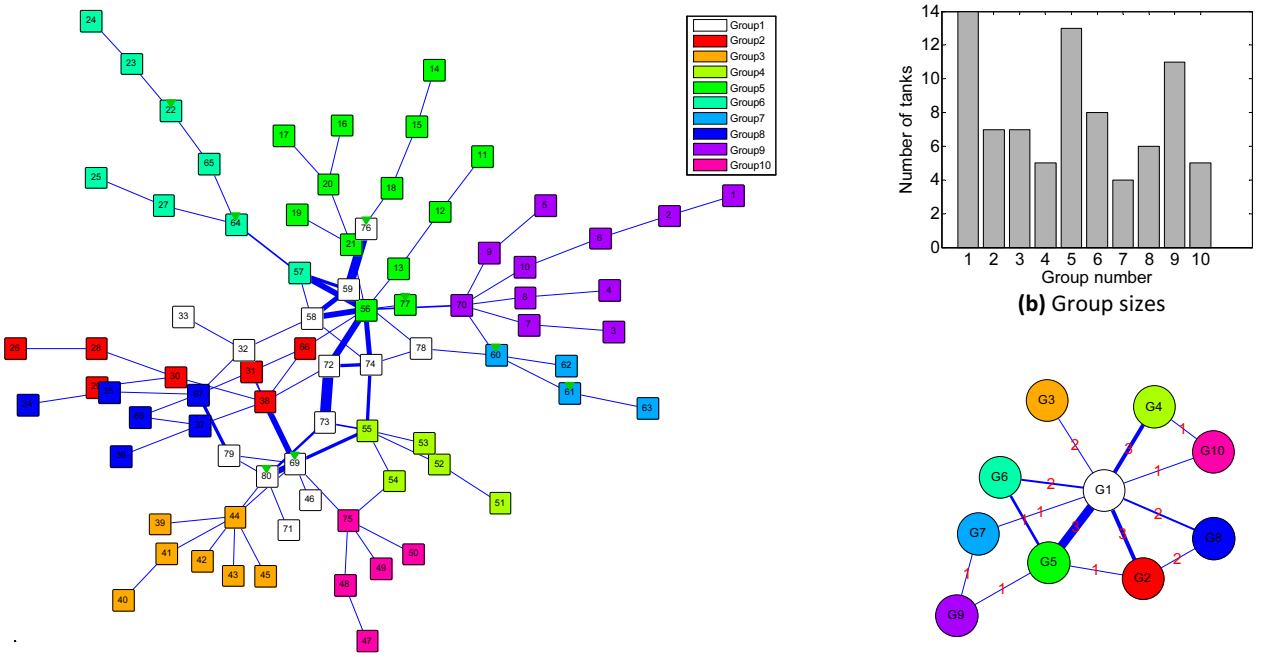
Another consideration for complex water networks is that water demands can be infeasible. The demands can be beyond resources capacities especially in the case of failures. This infeasibility would cause failure of distributed MPC convergence. The solution is to replace tanks low limit hard constraints by soft constraints. The virtual negative water level in case of infeasibility would be an indicator of missing water to cover user demands.

Another consideration is zero price of flow through valves. This can bring solution non-uniqueness as two tanks can be interconnected by multiple paths with zero prices. This solution non-uniquness can prevent distributed MPC to converge. Therefore flow through valve must have some minimum



(a) Simple Node **(b)** Complex Node

Figure 8: Nodes interconnecting tanks only **(a)**. Nodes can be also interconnected with another nodes by pump/valve with limits and cost function. Node can also have its own water source and water demand **(b)**.



(a) Tanks grouping for Barcelona water network

(b) Group sizes

(c) Aggregated groups interconnections

Figure 9: Aggregation of tanks to groups for control.

price. This will force the optimizer to choose the shortest possible path. The problem of non-unique solution can be further emphasizes by linear pricing

$$\min_{f_1,f_2} f_1^2 + f_2^2 \quad \text{s.t.} \quad f_1 \geqslant 0, \quad f_2 \geqslant 0, \quad f_1 + f_2 = 1, \tag{36}$$

vs.

$$\min_{f_1,f_2} f_1 + f_2 \quad \text{s.t.} \quad f_1 \geqslant 0, \quad f_2 \geqslant 0, \quad f_1 + f_2 = 1, \tag{37}$$

where quadratic pricing (36) has unique solution though linear pricing (37) does not.

2.3 Large Scale Model Partitioning

DMPC control strategies with grouping (Figure 2c and 2e) require methods for aggregation of subsystems into groups. The quality measure of grouping can be selected in multiple different ways. The best selection for DMPC with consensus iterations is to choose such partitioning, which gives minimum number of iterations or such partitioning, which gives minimum computing time (as a sum of worst computation times in all iterations). The partitioning methods with such quality measures are so far unsolved problems and they are currently replaced by heuristics based on graph algorithms.

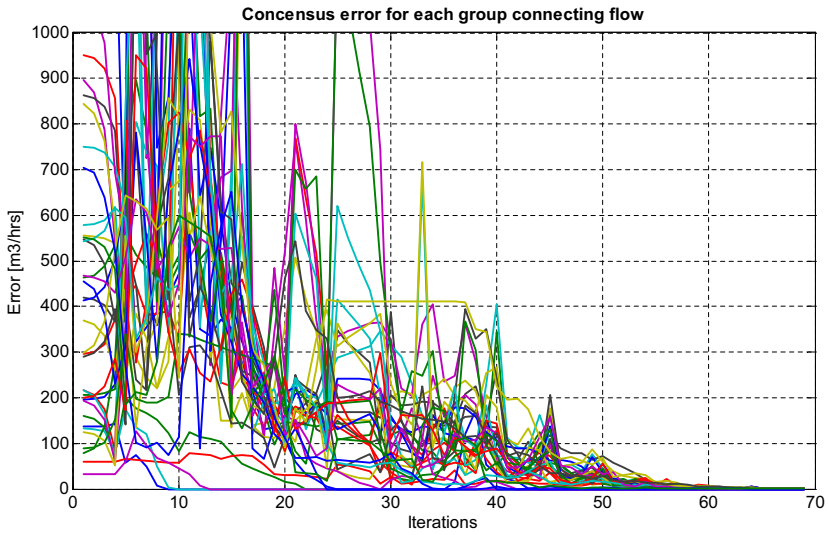


Figure 10: Example of consensus error in iterations during one sampling period for coordinated DMPC. There are 38 flows

The algorithm we use is based on graph condensation and epsilon decomposition [38]. The vertexes of the graph represent tanks and complex nodes and edges represent valves/pumps. The weigh of each edge is given by total flow through this valve/pump (from archive data). The algorithm is following:

1. **Leafs condensation** - graph leaf is vertex, which has only one edge. In the first step all leafs are merged to their neighboring vertexes until graph has no leaf.
2. **Epsilon decomposition** - condensed graph is decomposed to a selected number of groups by epsilon-decomposition [38].

The result of the algorithm for Barcelona water network and target of 10 groups is in Figure 9a. Figure 9b shows the sizes of individual groups and Figure 9c shows graph of interconnections between aggregated tank groups together with the number of interconnections between groups (red number on each edge).

2.4 Consensus Iterations

In each sampling period the controllers exchange data to reach consensus, where duplicate instances of complicating variables are equal. Although communication schemes may differer, the underlying problem (in dual decomposition) is in general concave maximization problem without constraints

$$\max_{\lambda} g(\lambda) \tag{38}$$

where gradient $\nabla g(\lambda)$ or sub-gradient $\partial g(\lambda)$ is known in each iteration. If DMPC strategy has coordinator than complete (sub-)gradient is collected by the coordinator and it can be used for Quasi-Newton methods. In DMPC strategies without coordinator each element of λ has its own coordinator, which knowns only appropriate element of (sub-)gradient – therefore only (sub-)gradient based methods can be used.

The algorithms used in our solutions are:

DMPC without coordinator uses Nesterov accelerated gradient method [28, 29].

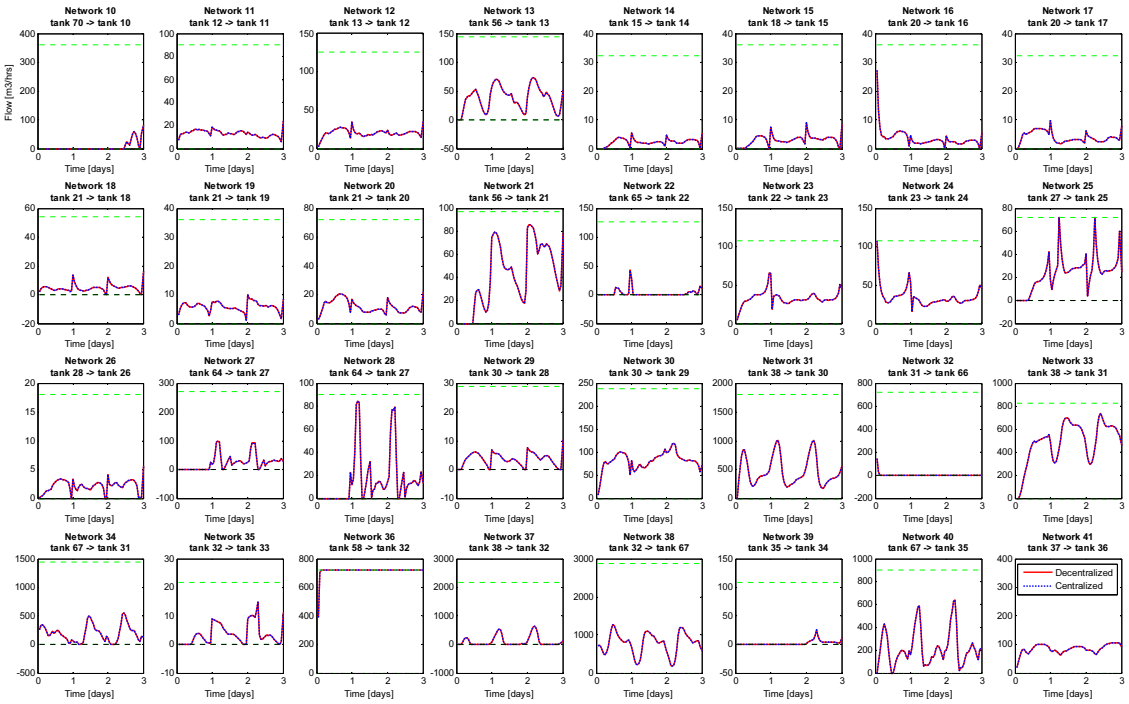


Figure 11: Figure shows 40 flows from total ∼110 flows. Red lines are nominal trajectories from centralized MPC (10 hrs predictions), Blue lines are trajectories computed by distributed MPC - practically identical results.

DMPC with coordinator uses limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method [8, 22], which is quasi–Newton method, which does not store or directly form Hessian estimate, but works with last m gradients only.

Example of consensus error in iterations during one sampling period is in Figure 10.

2.5 DMPC Simulation Results

For DMPC simulations we had a scheme of Barcelona water network together with limits on tank volumes and limits on pump/valve flows. We also had three days data with all demands and all flows in the network with 1 hour sampling period.

The goal of the simulation was to compare performance of centralized and decentralized MPC. For the purpose of this comparison we ignored demands prediction problem as demand prediction algorithm makes no difference in this comparison. We used historic demand data as measurable disturbances.

Optimality conditions

Loss function considers the following optimality measures:

- water pumping prices (time variable electricity price)
- water sources prices
- penalties for MV changes (pumps, valves and water sources)
- penalties for long water storing (chlorine concentration decrease)
- penalties for water levels below safety limits

Nominal solution

Nominal trajectories were obtained by centralized MPC. To make centralized MPC computable, the prediction horizon had to be reduced to 10 hours, which was the maximum length allowing computation in Matlab. The parameters of quadratic programming problem for centralized MPC were following:

Centralized MPC	
no. of variables	1910
no. of inequalities	3820
avg. optimization time per iteration [s]	120

Results comparison

Figure 11 compares flow trajectories on valves and pumps for solutions from centralized and distributed MPC. It shows 40 flows from total $\sim$110 flows. It can be seen that the results are practically identical. All four distributed control strategies (Figure 2) give very similar results. The difference is in the sizes of sub-problems and in the number of iterations. The summary of these differences is in the following table:

	iterations	seq. comp. [s]	parallel comp. [s]
Coordinated Group DMPC	$\sim$100	35	8
Group DMPC	$\sim$300	85	20
Coordinated DMPC	$\sim$300	20	$\sim$2
DMPC	$\sim$1000	60	5

Sequential computation represents time needed to iterate to global solution while computing distributed algorithm on a single computer only. Parallel computation represents time required by totally parallel computation - longest sub-problem optimization time (in one sampling period). The stopping condition for iterations is consensus error bellow 1 m^3/hrs. This error is then reduced by projection of dual solution to feasible set. The comparison of computation times for different DMPC strategies is in Figure 12.

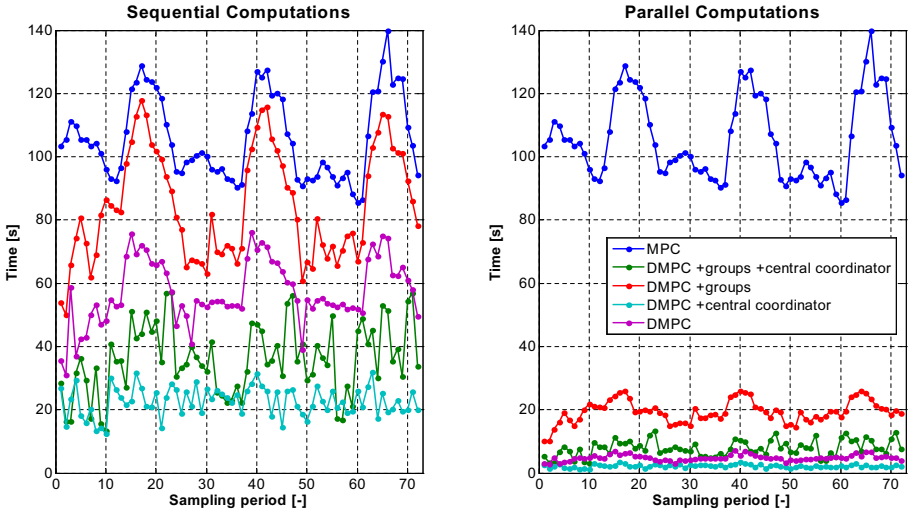


Figure 12: Comparison of computation times for different DMPC strategies. Times do not include time required for communication.

2.6 Distributed Control Modularity

Distributed control based on dual decomposition and especially the strategies without coordination is perfectly modular solution. It is very simple to reflect changes in network configuration without any global adjustments while preserving centralized MPC optimality. The changes can be:

- adding or removing new tank / group of tanks / pump from the system
- interconnecting independent networks
- setting any pump / valve to constant flow (MAN mode)

An example is in Figure 13. It shows two networks operating separately for 2 days, then these networks are connected together and after 4 days the connecting pump is switched to manual flow $-10\ \mathrm{m}^3/\mathrm{hrs}$. The figures on the right side of Figure 13 shows smooth (price optimal) transition between optimal regimes of disconnected and interconnected networks and finally optimal control for valve in manual.

Nice feature is that connecting that two networks means only that tank 5 starts to exchange information with node 3 coordinator (tank controllers and node coordinators can be physically aggregated). Reflecting the same change in centralized MPC would mean to completely change the prediction model, update constraints, etc.

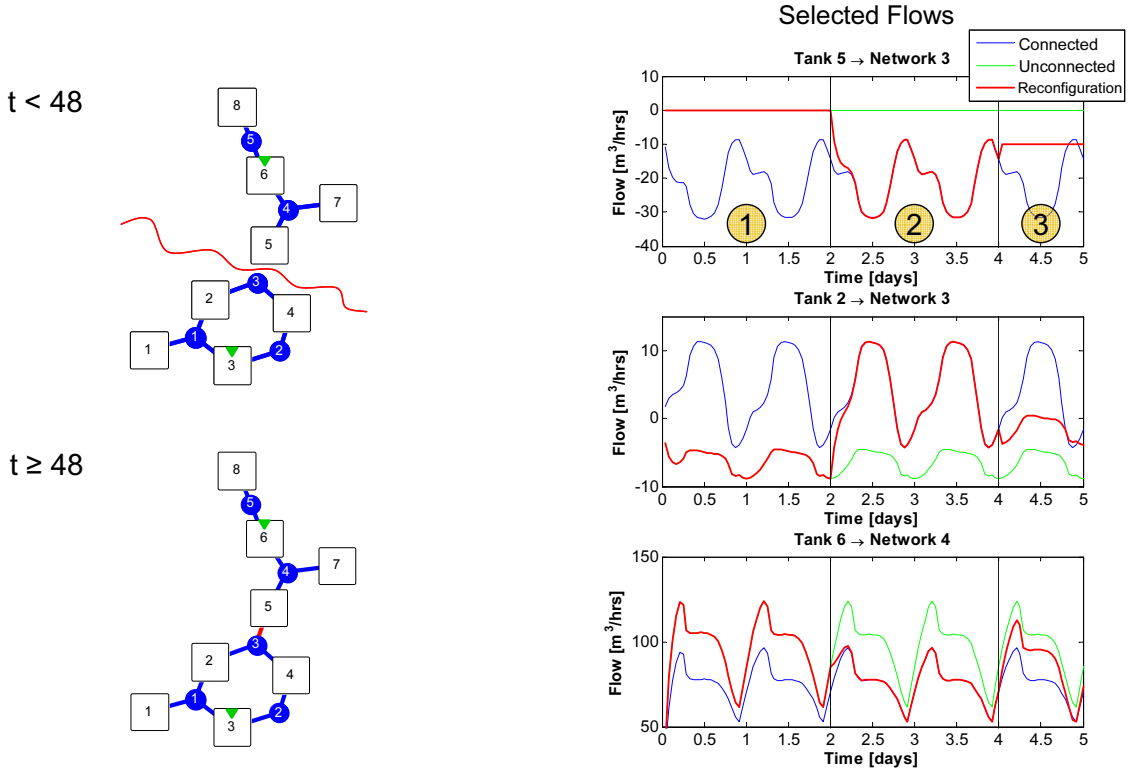


Figure 13: Example of DMPC modularity. Two water networks operate separately for 2 days, then they are connected together and after 4 days the pump from tank 5 to node 3 is switched to manual flow $-10\ \mathrm{m}^3/\mathrm{hrs}$ (red lines). Figures on the right show flows in three selected pipes. Green lines are optimum flows for disconnected networks and blue lines are optimum flows for connected networks (on the whole interval).

2.7 Conclusion

Distributed MPC or more generally dual decomposition of optimization problems can be used to reduce size and computational requirements of large scale convex non-differentiable optimization

problems. The distribution can be especially efficient for systems with obvious structure, such as systems interconnected by mass flows / energy flows / etc. Well designed distributed optimization can then have superior performance to centralized computation even on a single controller.

Although dual decompositions are theoretically straightforward, it is not trivial to transform them to practically useful algorithm with a reasonable number of consensus iterations and guaranteed solution robustness. The presented solution together with the state-of-the-art numerical methods delivers such an algorithm, which is moreover perfectly modular. The drawback is obviously the requirement for iterative information exchanges. This designates the algorithm for applications with sampling times in the order of tens of minutes.

Our contribution to distributed model predictive control is a general full scale application of dual decomposition based DMPC to water distribution networks, novel algorithm for tanks grouping for control and application of Nesterov Accelerated gradient method to consensus iterations.

3. Distributed Kalman filter

3.1 Introduction

3.1.1 Problem description and motivation

This section describes the effort in HPL in distributed state estimation. This is related to inferring process states that are not accessible for direct measurements, based on process output observation and process model. The value of process states is necessary for advanced process control algorithms, real-time optimization, process monitoring and fault detection and isolation. Our activities in distributed state estimation complement our efforts in decentralized and coordinated MPC control in Task 3.2 as well as to support real-time optimization in T3.3.

WIDE aims at development of a scaleable, distributed state estimator for large interconnected systems, consisting of a network of local estimators whose topology is consistent with those of the process. Local estimators are generalized Kalman filters associated with particular subsystems of the process. Further, the inter-filter communication is limited to data exchanges between the network neighbours. The local estimators are assumed to know model topology as well as local models of other subsystem within a certain network neighbourhood. The distributed estimator can respond to the process reconfiguration by updating its internal representation of the network neighbourhood. The uncertainties are mapped to covariances of coupling variables that are exchanged between the neighbours.

3.1.2 State of the art

Several strategies for distributed state estimation have been proposed. One class of problems involves the case when several nodes of local filters having their own observations of the process produce local estimates of the whole process state. The problem is to fuse the information from these estimators to get a global estimate – either in a 'central information fusion' node, or by exchanging the information on the estimated state with neighbouring states in the filter network topology. The former approach with the central fusion node uses the information filter implementation of the local estimators–see e.g. [52], [50]. This allows obtaining results equivalent to the central KF; it requires, however, an intensive information exchange. The latter strategy is decentralized: individual agents exchange estimates with their neighbours and modify their own estimates to be close to the estimates received from their peers. It has been shown that, under certain assumptions, the local state estimates converge to each other, i.e., there is a consensus on the state estimate over the network, [41]. Also in these cases, estimates of the global state are present in each local filter. This may require a large computational load in the case when the state dimension is large. A version of the distributed, consensus-based filtering for continuous-time systems, where the local nodes estimate different but overlapping subspaces of the global state appeared in the recent paper [48]. An alternative approach to distributed estimation where local agents estimate overlapping subspaces of global state was proposed in [46], [47]. Their algorithm uses a special form of consensus strategy formulation for observation vectors (using bi-partite fusion graphs) and distributed inverse computation of an L-banded approximation of the global state covariance matrix. In [45], several distributed estimation strategies were compared. It was found that the bi-partite graph-based KF has the best performance but is sensitive to (random) communication loss. Another suboptimal distributed estimator intended for applications in complex chemical processes was proposed in [49], proposing to

approximate the optimal filter by a suboptimal multi-rate one where the internode communication proceeds at a slower rate than the local filter prediction.

3.1.3 Solution at a glance

In a paper [53], currently under review, we proposed a novel distributed KF for interconnected discrete-time dynamical systems. A local KF (agent) is assigned to each subsystem, and communication proceeds (bi-directionally) between agents whose subsystems are connected by an interconnection. Each local filter estimates the mean and the covariance of the local state of interconnection inputs. These inputs represent the overlaps with state spaces of neighbouring subsystems. Local filters can have an access to measurements taken by subsystems within a certain network distance. Further, agents use the estimates of coupling variables from their neighbours as 'measurements'. An iterative algorithm is proposed as a series of stochastic optimizations within one time step of the filter, where the local estimates are repeatedly refined using the updated coupling variables produced by network neighbours. The overall process is converging fast. This approach resembles, the consensus algorithms – the discrepancies in the shared variable estimates are used for adjusting local estimates. The adjustment is done here in the KF setting, not, e.g., by an ad-hoc replacement of the local estimate of the shared variable by a (weighted) average of local estimates. Thus, the agents are optimal KFs, while the overall distributed estimator is not, in general. Under certain conditions, the estimate equivalent to the central KF is recovered. On the other hand, this algorithm needs some data to be passed from one agent to another across the network; also the local agents need to have the knowledge of other subsystems' models as well as the overall network topology. This increases the communication overhead and restricts the class of problems, for which this solution is suitable.

This problem is addressed by a newly proposed sub-optimal distributed filter (paper prepared for publication [54]) so that the agents require the data as well as the model knowledge only within a limited network radius. The full global state covariance matrix is approximated by an upper-bounding multi-band matrix. This is done by introducing fictitious noises in the algorithm that vary from one 'consensus' iteration to another. These noises are assumed to be of zero mean; their covariance matrices are designed to de-correlate local measurements and couplings with the corresponding variables in the specific network distance.

3.2 Problem formulation

3.2.1 Process and noise models

This paper addresses state estimation of interconnected systems as in Figure . The overall model consists of a network of N interconnected subsystems whose state-space representation is given by (deterministic external inputs being omitted):

$$\begin{aligned} x_i(k+1) &= A_i x_i(k) + \sum_{j \in N_i} B_{ij} i_{ij}(k) + \xi_i(k) \\ y_i(k) &= C_{yi} x_i(k) + \eta_i(k) \\ o_{is}(k) &= C_{ois} x_i(k), \qquad s \in M_i . \end{aligned} \tag{39}$$

There, $x_i \in R^{n_i}$ and $y_i \in R^{m_i}$ are local process state and measurement output, respectively. Sets N_i and M_i are sets of subsystems that, respectively, provide an interconnection input i_{ij} to/receive an interconnection output o_{is} from subsystem i. Notice, that there is strictly no feed-through from interconnection inputs to interconnection outputs; this assumption

needs to be enforced for the problem to be tractable. On the other hand, the non-existence of feed-throughs in the measurement equation is merely for convenience and can be removed. The state is disturbed by process noise ξ_i; measurements are corrupted by noise η_i. Both noises are zero-mean, Gaussian and white, with *positively definite* covariance matrices Q_{oi} and R_i, respectively. Moreover, they are assumed to be independent. The overall process model is thus given by the collection of local models (39) as well as by interconnection equations

$$i_{ij}(k) = o_{ji}(k) \quad \forall k = 0,1,\ldots; \forall i = 1,\ldots,N; \forall j \in N_i. \tag{40}$$

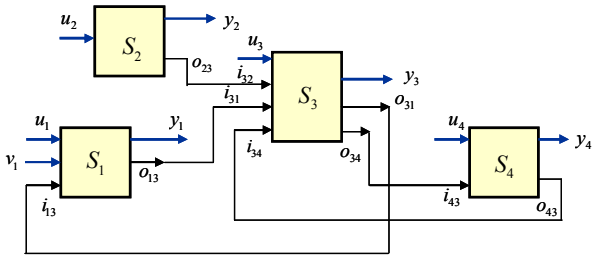


Figure 14 Interconnected system.

3.2.2 Distributed estimator structure

Distributed estimator consists of a network of Kalman filters—one KF agent per subsystem. We shall assume here that the agents rely on local process measurements. Coupling inputs are modelled as stochastic variables whose mean and covariance are supplied by neighbouring agents. In particular,

$$i_{ij}(k) = \overline{o}_{ji}(k \mid k-1) \; + \vartheta_{ij}(k) \; \forall k = 0,1,\ldots; \forall i; \forall j \in N_i. \tag{41}$$

There, $\overline{o}_{ji}(k \mid k-1)$ denotes the mean of $\overline{o}_{ji}(k)$ conditioned by data up to time $k-1$. Further, ϑ_{ij} is Gaussian noise with zero mean and conditional covariance $C_{oji}\,\mathrm{E}\left\{x_j(k)x_j(k)^T \mid k-1\right\}C_{oji}^T$ that is computed by j^{th} KF; symbol $\mathrm{E}\{\ \}$ denotes here the expected value. Noise ϑ_{ij} is treated as process noise by i^{th} KF. Note that it is correlated with local state x_i; therefore, it is estimated in i^{th} KF along with x_i. Variables ϑ_{ij}, for all $j \in N_i$, thus augment the local state.

Local KF then estimates both interconnection outputs o_{ip} (as functions of their state) and inputs i_{ij} (as sums of their means supplied by j^{th} *KF* and noise ϑ_{ij} as an element of the local augmented state). Hence, interconnection variables are estimated simultaneously by two neighbouring filters. The idea behind the distributed filter is that coupling variables are treated by local KFs as measured process variables. Instead of the real process data, the estimates provided by neighbouring filters are used. After one filtering step, local KFs modify their states using data from their neighbours. The local estimates of interconnection variables change; then, the new estimates are exchanged and the update is repeated, until a network-wide equilibrium is reached. The KF filtering step thus proceeds in an iterative way; let the iteration within one time step k be indexed by variable s.

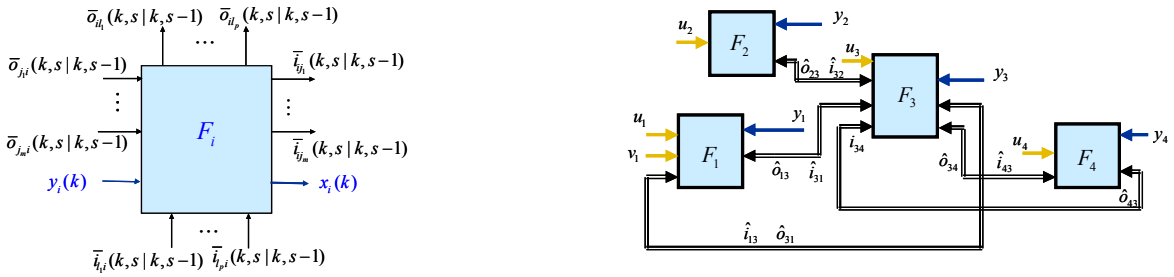


Figure 15 Local Kalman filter (left) and a network of filters (right)

The interaction of the iterative filtering algorithm is shown in Figure 15. It follows from this scheme, local KF must take into account the correlation between coupling variables and the local state in order to update the local state covariance correctly. In order to keep track on this correlation, local KF must record and update covariance matrices of the local state with all other states in the network, conditioned by data up to time k and iteration s.

After the termination of *s*-iterations, a standard prediction step occurs by incrementing time-step *k*. To predict the local state mean and covariance, process model (39) is used to predict covariance matrices with other states, models of other subsystems in the network are needed.

The algorithm outlined above shall be described in the following section in more detail. It shall be noted that although the local agents are optimal Kalman filters, the overall distributed filter is generally not -- although in certain cases, the globally optimal KF can be achieved in this distributed framework. The main drawback of this approach is a significant communication overhead and the need for local filters to know the global model and the network topology. The goal of this paper is to restrict the communications among agents as well as the model knowledge to a pre-specified network radius.

3.3 Baseline algorithm of distributed estimation

3.3.1 An overview

A crude flowchart of a local filter is shown in Figure 16. The main difference relative to the centralized Kalman filter is the repeated performance of the data step to update the local state by newly received data. It needs to be repeated as some of the data are computed by other agents and may differ from one s-iteration to another. This step is repeated until the network-wide equilibrium is reached. The initialization performed upon start-up and/or upon a structural change (network topology) involves collecting information about models of other subsystems and network map (at this point we do not consider the restricted network radius for sharing data and model knowledge). The necessary data structures depending on the topology and other agent's internal data are built here as well us pointers into the neighbours data structures saying which portion of data are to be imported. Details about this function are omitted here.

The outer filtering loop indexed by time-step k starts by re-initialization: exchanging predicted interconnection outputs and their covariance to build the extended state. Then, the execution passes to the inner loop, indexed by time index s in which data update is performed. This involves multiple interactions with the neighbours; Kalman gains and state-to-output covariance matrices are passed across the network from one neighbour to another. An important point is that the algorithm processes data upon their receiving and does not have to wait idling before it receives the next batch.

The prediction step is non-iterative; it is similar to the prediction step of standard KFs.

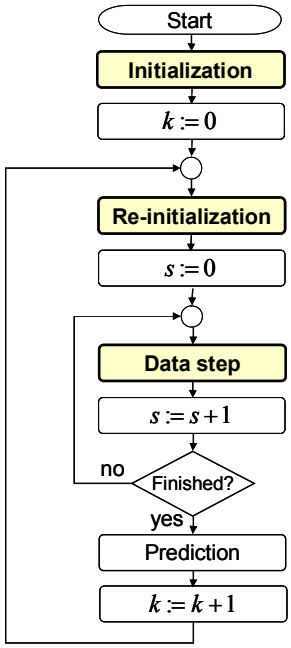


Figure 16 Local filter flowchart; highlighted block involve communication with neighbors.

3.3.2 Re-initialization

The filtering step of each KF starts from data predicted in the previous time-step. A local KF computes predicted values of the local state mean $\overline{x}_i(k \mid k-1)$, its covariance $P_{ii}(k \mid k-1)$ and cross-covariance matrices with other subsystem states $P_{ij}(k \mid k-1)$. Then, as was mentioned above, the local state is augmented by the estimate of the coupling input as,

$$X_i = \left[x_i^T \,\middle|\, \vartheta_{ij_1}^T \;\cdots\; \vartheta_{ij_n}^T \right]^T, \; j_1, \ldots j_n \in N_i \tag{42}$$

We shall use a projection operator to obtain the coupling input variation from the augmented state as $\vartheta_{ij} = E_{\vartheta ij} X_i$. Means of these variables are initialized to zero, $\overline{\vartheta}_{ij}(k \mid k-1) = 0.$ However, it is assumed that predicted coupling inputs, i.e., $\overline{i}_{ij}(k \mid k-1) = C_{oji}\overline{x}_j(k \mid k-1)$, are obtained from all 'upstream' neighbours.

Cross-covariance matrices of augmented state variables are denoted as $M_{ij}(k \mid k-1)$. In particular, we shall have, for all m, for $j_1, .., j_k \in N_i$ and for $l_1, .., l_n \in N_m$, the following block matrix (the argument $(k \mid k-1)$ being omitted)

$$M_{im} = \left[\begin{array}{cccc} P_{im} & P_{is_1} C_{os_1 m}^T & \cdots & P_{is_n} C_{os_n m}^T \\ \hline C_{ol_1 i} P_{l_1 m} & C_{ol_1 i} P_{l_1 s_1} C_{os_1 m}^T & \cdots & C_{ol_1 i} P_{l_1 s_n} C_{os_n m}^T \\ \vdots & \vdots & \ddots & \vdots \\ \hline C_{ol_k i} P_{l_k m} & C_{ol_k i} P_{l_k s_1} C_{os_1 m}^T & \cdots & C_{ol_k i} P_{l_k s_n} C_{os_n m}^T \end{array}\right] \tag{43}$$

The first block of rows in this equation is available from local data; further blocks of rows are obtained from network neighbours l_1 to l_k.

3.3.3 Data update in step s

The data step of the local filter proceeds iteratively for $s = 1, 2, \ldots..$ by updating the conditional mean $\overline{X}_i(k, s \mid k, s-1)$ and covariance $M_{ij}(k, s \mid k, s-1)$. The starting values for $s = 0$ are

$$\overline{X}_i(k,0\,|\,k,-1)=\overline{X}_i(k\,|\,k-1)\ \text{ and }\ M_{ij}(k,0\,|\,k,-1)=M_{ij}(k\,|\,k-1)\quad\forall j. \tag{44}$$

The data used for the local update in s^{th} step (measurements plus estimates of coupling inputs/outputs from adjacent agents) are given by

$$Y_i(k,s)=\begin{bmatrix} y_i(k) \\ \hline \overline{o}_{l_1 i}(k,s\,|\,k,s-1)-\overline{o}_{l_1 i}(k\,|\,k-1) \\ \vdots \\ \overline{o}_{l_k i}(k,s\,|\,k,s-1)-\overline{o}_{l_k i}(k\,|\,k-1) \\ \hline \overline{\vartheta}_{j_1 i}(k,s\,|\,k,s-1) \\ \vdots \\ \overline{\vartheta}_{j_m i}(k,s\,|\,k,s-1) \end{bmatrix}. \tag{45}$$

The output error can be expressed as

$$\tilde{Y}_i(k,s\,|\,k,-1)=Y_i(k,s)-\overline{Y}_i(k,s\,|\,k,s-1)=\begin{bmatrix} y_i(k)-C_{yi}\overline{x}_i(k,s\,|\,k,s-1) \\ \hline \vdots \\ \delta_{il}(k,s\,|\,k,s-1) \\ \vdots \\ \hline \vdots \\ \delta_{ji}(k,s\,|\,k,s-1) \\ \vdots \end{bmatrix}. \tag{46}$$

There, variables $\delta_{ij}(k,s\,|\,k,s-1)$ are defined as

$$\delta_{ij}(k,s\,|\,k,s-1)=C_{oji}\left(\overline{X}_j(k,s\,|\,k,s-1)-\overline{X}_j(k\,|\,k-1)\right)-E_{\vartheta ij}\overline{X}_i(k,s\,|\,k,s-1). \tag{47}$$

Notice that as follows from the above initialization rules, there holds $\delta_{ij}(k,0\,|\,k,-1)=0$. Finally, let us introduce the following notation for covariance matrices:

$$\begin{aligned} H_{ij}(k,s\,|\,k,s-1) &= \mathrm{E}\left(\left(X_i(k)-\overline{X}_i(k,s\,|\,k,s-1)\right)\tilde{Y}_j(k,s\,|\,k,s-1)^T\,|\,k,s-1\right), \\ W_{ij}(k,s\,|\,k,s-1) &= \mathrm{E}\left(\tilde{Y}_i(k,s\,|\,k,s-1)\tilde{Y}_j(k,s\,|\,k,s-1)^T\,|\,k,s-1\right). \end{aligned} \tag{48}$$

For notational considerations, we shall partition these matrices compatibly with (46) as

$$H_{ij}(k,s\,|\,k,s-1)=\left[H_{iy_j}(k,s\,|\,k,s-1)\ \ H_{i\delta_{j\bullet}}(k,s\,|\,k,s-1)\ \ H_{i\delta_{\bullet j}}(k,s\,|\,k,s-1)\right], \tag{49}$$

$$W_{ij}(k,s\,|\,k,s-1)=\begin{bmatrix} W_{y_i y_j}(k,s\,|\,k,s-1) & W_{y_i\delta_{j\bullet}}(k,s\,|\,k,s-1) & W_{y_i\delta_{\bullet j}}(k,s\,|\,k,s-1) \\ W_{\delta_{i\bullet}y_j}(k,s\,|\,k,s-1) & W_{\delta_{i\bullet}\delta_{j\bullet}}(k,s\,|\,k,s-1) & W_{\delta_{i\bullet}\delta_{\bullet j}}(k,s\,|\,k,s-1) \\ W_{\delta_{\bullet i}y_j}(k,s\,|\,k,s-1) & W_{\delta_{\bullet i}\delta_{j\bullet}}(k,s\,|\,k,s-1) & W_{\delta_{\bullet i}\delta_{\bullet j}}(k,s\,|\,k,s-1) \end{bmatrix} \tag{50}$$

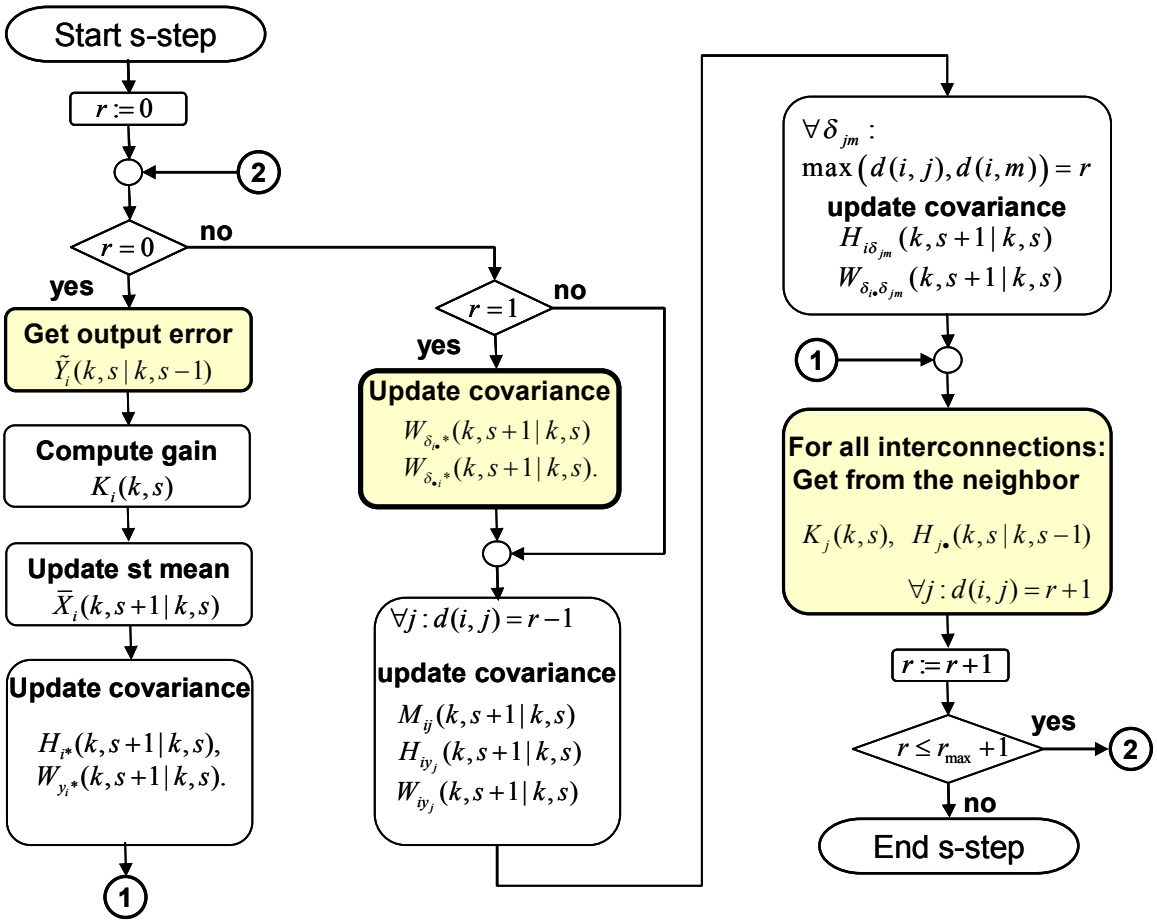


Figure 17 Flow chart for one s-step. Highlighted blocks are those which import data from network neighbors.

One s-step is described in Figure 17. Here we shall expand some of the blocks, if needed. Note, that the agents have to update cross-covariance matrices of local state X_i with other states X_j for all other agents j; for this, it needs the Kalman gain K_j and also the state-to-output covariance from previous iteration that need to be passed through the network. Thus, the s-step is computed in a loop where the number of passes (also called r-iteration) equals to the maximum network distance plus 2. Network distance between two agents, i and j, denotes the minimum number of edges between these nodes in the (non-oriented) interconnection graph. The first pass uses only local data for the covariance update; the data for the next passes are downloaded from the neighbors at the end of the loop. At each pass a different part of the local data is updated.

In the first pass (for $r = 0$), the output error $\tilde{Y}_i(k,s \mid k,-1)$ is computed as in (46). Then, Kalman gain $K_i(k,s)$ is determined as a (minimum norm) solution of the possibly singular equation

$$K_i(k,s)W_{ii}(k,s \mid k,s-1) = H_{ii}(k,s \mid k,s-1). \tag{51}$$

Then, the local state is updated as

$$\bar{X}_i(k,s+1 \mid k,s) = \bar{X}_i(k,s \mid k,s-1) + K_i(k,s)Y_i(k,s \mid k,s-1). \tag{52}$$

Then, an intermediate result towards the updated matrices H_{i*} and W_{i*} is obtained as

$$\begin{aligned} H^0_{i*}(k,s+1 \mid k,s) &= H_{i*}(k,s \mid k,s-1) - K_i(k,s)W_{i*}(k,s \mid k,s-1) \\ W^0_{y_i*}(k,s+1 \mid k,s) &= W_{y_i*}(k,s \mid k,s-1) - C_{m_i}K_i(k,s)W_{i*}(k,s \mid k,s-1) \end{aligned} \tag{53}$$

The first pass is concluded by downloading the newly computed Kalman gains $K_j(s,k)$ from network neighbours $j \in M_i \cup N_i$. The old state-to output covariance matrix $H_{j\bullet}(k,s \mid k,s-1)$ is downloaded from the same source (not necessarily at the same time) – but only selected columns that may be used locally, or by other agents that may download this in next *r*-iterations; the particular columns to be imported are pre-determined in the initialization step.

The second pass starts the row operations over covariance sub-matrix $W_{\delta_{\bullet\bullet}*}$ as

$$\begin{aligned} W^0_{\delta_{ij}*}(k,s+1 \mid k,s) &= W^0_{\delta_{ij}*}(k,s+1 \mid k,s) + C_{oij}K_i(k,s)W_{i*}(k,s \mid k,s-1) \\ &\quad -E_{\vartheta ji}K_j(k,s)W_{j*}(k,s \mid k,s-1), \qquad \forall j \in M_i \\ W^0_{\delta_{mi}*}(k,s+1 \mid k,s) &= W^0_{\delta_{mi}*}(k,s+1 \mid k,s) + C_{omi}K_m(k,s)W_{m*}(k,s \mid k,s-1) \\ &\quad -E_{\vartheta im}K_i(k,s)W_{m*}(k,s \mid k,s-1), \qquad \forall m \in N_i \end{aligned} \tag{54}$$

These row operations complement those in (53) for $r=0$. The agent *i* imports, from an upstream neighbour *m*, the quantity $C_{omi}K_m(k,s)W_{m*}(k,s \mid k,s-1)$ as well as $E_{\vartheta ji}K_j(k,s)W_{j*}(k,s \mid k,s-1)$ from a downstream agent *j*. These data can be prepared for export by the neighbours already in the previous pass (for $r=0$; not shown in the flowchart).

Then, the second pass continues by steps common for all $r=1,2,...,r_{\max+1}$, (where $r_{\max+1}$ is the maximum network distance from i^{th} agent): a series of cross-covariance updates is done for variables related to all j, $d(i,j)=r-1$. First, the state covariance matrix is updated as

$$\begin{aligned} M_{ij}(k,s+1 \mid k,s) = M_{ij}(k,s \mid k,s-1) - K_i(k,s)H_{ji}(k,s \mid k,s-1)^T - \\ H_{ij}(k,s \mid k,s-1)K_j(k,s)^T + K_i(k,s)W_{ij}(k,s \mid k,s-1)K_j(k,s \mid k,s-1)^T. \end{aligned} \tag{55}$$

Then, cross-covariance related to output y_j are computed as

$$\begin{bmatrix} H_{iy_j}(k,s+1 \mid k,s) \\ W_{iy_j}(k,s+1 \mid k,s) \end{bmatrix} = \begin{bmatrix} H^0_{iy_j}(k,s+1 \mid k,s) \\ W^0_{iy_j}(k,s+1 \mid k,s) \end{bmatrix} - \begin{bmatrix} H^0_{ij}(k,s+1 \mid k,s) \\ W^0_{ij}(k,s+1 \mid k,s) \end{bmatrix} K_j(k,s)^T C^T_{mj} \tag{56}$$

Then, cross-covariances with consensus variables δ_{pq}, where $\max\left(d(i,p),d(i,q)\right)=r$ are obtained as

$$\begin{aligned} \begin{bmatrix} H_{i\delta_{pq}}(k,s+1 \mid k,s) \\ W_{i\delta_{pq}}(k,s+1 \mid k,s) \end{bmatrix} &= \begin{bmatrix} H^0_{i\delta_{pq}}(k,s+1 \mid k,s) \\ W^0_{i\delta_{pq}}(k,s+1 \mid k,s) \end{bmatrix} + \begin{bmatrix} H^0_{ip}(k,s+1 \mid k,s) \\ W^0_{ip}(k,s+1 \mid k,s) \end{bmatrix} K_p(k,s)^T C^T_{opq} \\ &\quad - \begin{bmatrix} H^0_{iq}(k,s+1 \mid k,s) \\ W^0_{iq}(k,s+1 \mid k,s) \end{bmatrix} K_q(k,s)^T E^T_{\vartheta qp}. \end{aligned} \tag{57}$$

At the end any r-iteration, Kalman gain $K_j(s,k)$ and relevant columns of $H_{j\bullet}(k,s \mid k,s-1)$ are imported from specific neighbours, for all j, $d(i,j)=r+1$.

Interestingly, updating covariance matrices $H_{i*}(k,s+1 \mid k,s)$ and $W_{i*}(k,s+1 \mid k,s)$ does not use $M_{i*}(k,s \mid k,s-1)$ for $s>0$; it is thus possible to do update only H and W matrices at each

s-iteration and after terminating the s-loop do a ‘cumulative update’ of .the state covariance matrix.

It can be proven, see [53], that the s-loop converges, and that the network-wide equilibrium is $K_j(k,\infty)=0$ for all $j=1,\ldots,N$. This can be used as a termination test in the s-loop.

3.3.4 Prediction step

The state mean prediction proceeds, after termination of the s-loop for $s=s_e$, as follows:

$$x_i(k+1,k)=\tilde{A}_i X_i(k,s_e+1\,|\,k,s_e)+\sum_{p\in N_i} B_{ip}\left(\overline{o}_{pi}(k\,|\,k-1)+\delta_{ij}(k,s_e+1\,|\,k,s_e)\right), \tag{58}$$

where

$$\tilde{A}_i=\left[\,A_i\,|\,B_{ip_1}\ \cdots\ B_{ip_m}\,\right],\quad p_*\in N_i\,. \tag{59}$$

The predicted covariance is given by

$$\begin{aligned} P_{im}(k+1\,|\,k)=\tilde{A}_i M_{im}(k,s_e+1\,|\,k,s_e)\tilde{A}_m^T+Q_{0im}+\sum_{p\in N_i} B_{ip}H^T_{m\delta_{pi}}(k,s_e+1\,|\,k,s_e)\tilde{A}_m^T+\\ \tilde{A}_i\sum_{q\in N_m} H_{i\delta_{qm}}(k,s_e+1\,|\,k,s_e)B_{mq}^T+\sum_{p\in N_i}\sum_{q\in N_m} B_{ip}W_{\delta_{pi}\delta_{qm}}(k,s_e+1\,|\,k,s_e)B_{mq}^T. \end{aligned} \tag{60}$$

Matrix Q_{0im} is the process noise covariance.

Finally, predicted values of coupling inputs $\overline{o}_{is}(k+1\,|\,k)$ and their covariance with predicted (augmented) states are computed and made accessible for the respective neighbours in order to be used in the re-initialization of the s-iteration process.

3.4 Distributed estimator with restricted communication radius

3.4.1 Overview

The goal of this section is to modify the above algorithm to obtain a distributed estimator that is possibly of lower performance but has no need for communication and model sharing between filters whose network distance is greater than a pre-defined network radius R. For this, we need to have

$$M_{ij}(k \mid k-1) = 0 \quad \text{for } \forall j : d(i,j) \geq R \tag{61}$$

and, for all $s = 0,1,....$

$$H_{ij}(k,s \mid k,s-1) = 0,\ W_{ij}(k,s \mid k,s-1) = 0 \quad \text{for } \forall j : d(i,j) \geq R \tag{62}$$

This is sufficient for having $M_{ij}(k,s \mid k,s-1) = 0$ for all j with radius R or greater and for all iterations s.This is achieved by approximating the real covariance matrix by a matrix that is larger than or equal to (in the LMI sense) the original one and satisfying the above conditions on de-correlation of variables in a certain network distance. A geometrical interpretations of this idea for a pair of variables is in Figure 18. Formally, this is done by introducing fictitious noises whose covariance varies from one s-iteration to another.

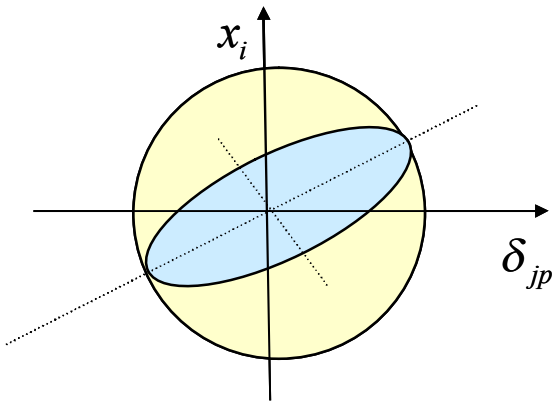


Figure 18 Geometrical interpretation of the decorrelation technique.

3.4.2 Modified algorithm: introducing fictitious noise

Let us assume that, for a given iteration s, there holds . $M_{ij}(k,s \mid k,s-1) = 0$ for all j such that $d(i,j) \geq R$; further, assume that (62) holds. Then, the data step is performed as described above, for $r_{\max} = R$. Mark the newly updated covariance matrices by the superscript '$-$', in particular, $M_{ii}^{-}(k,s+1 \mid k,s)$, $H_{ij}^{-}(k,s+1 \mid k,s)$ and $W_{ij}^{-}(k,s+1 \mid k,s)$ We end up with updated state-to-output and output-to-output covariance matrices, that no longer satisfy assumption (62); there holds only

$$H_{ij}^{-}(k,s+1 \mid k,s) = 0,\ W_{ij}^{-}(k,s+1 \mid k,s) = 0 \quad \text{for } \forall j : d(i,j) > R \tag{63}$$

First, we would like to get rid of the non-zero cross-covariance $H_{ij}^{-}(k,s+1 \mid k,s)$ for $d(i,j) = R$. (In fact, only some columns of this matrix are non-zero: those corresponding to all δ_{jl} and δ_{qj} for which $d(i,j) = d(i,q) = R-1$). We shall introduce a fictitious noise varying from one s-iteration to another which is added to the local state and the output

$$\begin{aligned} X_i(k,s+1) &= X_i^-(k,s+1)+\xi_{xi}(k,s) \\ Y_i(k,s+1) &= Y_i^-(k,s+1)+\xi_{Yi}(k,s)+\eta_{Yi}(k,s). \end{aligned} \tag{64}$$

These noises are assumed to be Gaussian, zero-mean, white (with respect to both k and s); moreover, noises ξ_{Xi} and ξ_{Yj} are independent with η_{Yi} and mutually independent for all i and j, except of those where $d(i,j)=R$. First, we shall discuss the role of ξ_{Xi} and ξ_{Yj}, the role of η_{Yi} will be clarified later. In particular, the joint covariance will be set to

$$\mathrm{E}\left\{\begin{bmatrix}\xi_{Xi}(k,s)\\ \xi_{Yj}(k,s)\end{bmatrix}\begin{bmatrix}\xi_{Xi}(k,s)^T & \xi_{Yj}(k,s)^T\end{bmatrix}\right\}=\begin{bmatrix}Q_{Xi}(k,s) & -H_{ij}^-(k,s+1|k,s)\\ -H_{ij}^-(k,s+1|k,s)^T & R_{Yj}(k,s)\end{bmatrix}. \tag{65}$$

Variances Q_{Xi} and R_{Yji} are chosen as the smallest ones so that the joint covariance matrix (65) is positive semi-definite. In particular, if the singular value decomposition of the cross-covariance term is given by

$$H_{ij}^-(k,s+1|k,s)=U_{ij}S_{ij}V_{ij}^T,\;\; H_{ji}^-(k,s+1|k,s)=U_{ji}S_{ji}V_{ji}^T,\;\; d(i,j)=R \tag{66}$$

then

$$Q_{Xi}=\sum_{j:d(i,j)=R}U_{ij}S_{ij}U_{ij}^T,\;\; R_{Yi}=\sum_{j:d(i,j)=R}V_{ji}S_{ji}V_{ij}^T. \tag{67}$$

Note that the non-zero columns of H_{ji}^- are available from neighboring agents. Thus, this de-correlates states X_i and outputs Y_j for $d(i,j)=R$.

Further, it is necessary to take into account that variables $Y_i(k,s+1|k,s)$ and $Y_k(k,s+1|k,s)$ for all $k:d(i,k)=1$ depend on $\xi_{xi}(k,s)$, and hence, we have to map the matrix Q_{Xi} on H_{ii}^-, H_{ik}^-, W_{ii}^{T-}, W_{ik}^-, W_{ki}^- and W_{ki}^- (the argument omitted for brevity). This involves then certain information exchange between neighbouring agents. Similarly, the de-correlation terms $\mathrm{E}\,\xi_{Xi}\xi_{Y_j}^T=-H_{ij}^-$ is mapped, for all l, $d(j,l)=1$, on matrices W_{pq}^- and W_{qp}^- where $p\in\{i,k\}$ and $q\in\{j,l\}$. Details are omitted here.

After finishing these updates, matrix $H_{i*}^-(k,s+1|k,s)$ satisfies (62); it is then renamed to $H_{i*}(k,s+1|k,s)$ for the use in the next s-iteration. Further, set

$$M_{ii}(k,s+1|k,s)=M_{ii}^-(k,s+1|k,s)+Q_{Xi} \tag{68}$$

In the next step, we need to de-correlate output vectors Y_i and Y_j for $d(i,j)=R$. For this, the fictitious noise η_{Yi} will be used. Again, it is zero-mean, Gaussian noise and, η_{Yi} and η_{Yj} are independent except of those where $d(i,j)=R$. The joint covariance is given by

$$\mathrm{E}\left\{\begin{bmatrix}\eta_{Yi}\\ \eta_{Yj}\end{bmatrix}\begin{bmatrix}\eta_{Yi}^T & \eta_{Yj}^T\end{bmatrix}\right\}=\begin{bmatrix}P_{Yi} & -W_{ij}^-\\ -W_{ji}^{T-} & P_{Yj}\end{bmatrix} \tag{69}$$

Let us make the decomposition $W_{ij}^-=U_{Yij}S_{Yij}V_{Yij}^T$. Then

$$\mathrm{P}_{Yi}=\sum_{j:d(i,j)=R}U_{Yij}S_{Yij}U_{Yij}^T. \tag{70}$$

Then, $W_{ii}(k,s+1|k,s)=W_{ii}^-(k,s+1|k,s)+P_{Yi}$ and $W_{ij}(k,s+1|k,s)=0$. Note that the rows in W_{i*} and This concludes the s-iteration.

After the termination of the s-iterations, we shall proceed with the prediction and re-initialization steps as described above; first, $M_{ik}(k+1,0\,|\,k,-1)=0$ for all k such that $d(i,k)>R$; for $d(i,k)=R$, this cross-covariance will be non-zero. The correction will be done, again, by introducing a fictitious process noise

$$X_i(k+1)=X_i^-(k+1)+\varphi_{Xi}(k); \tag{71}$$

As before, the added process noises are zero-mean, Gaussian and white; they are independent, except of pairs $\varphi_{Xi}(k)$ and $\varphi_{Xm}(k)$ for all pairs i and m such that $d(i,m)=R$. Then, $E\left(\varphi_{Xi}(k)\varphi_{Xm}(k)^T\right)=-P_{im}^-(k+1\,|\,k)$. Designing variances $E\left(\varphi_{Xi}(k)\varphi_{Xi}(k)^T\right)$ and $E\left(\varphi_{Xm}(k)\varphi_{Xm}(k)^T\right)$ is done analogously to the previous cases. Having done this, condition (61) is satisfied. Further, we obtain state-to-output and output-to-output covariance matrices $H_{ij}^-(k,0\,|\,k,-1)$ and $W_{ij}^-(k,0\,|\,k,-1)$, respectively, that do not satisfy condition (62). Their modification proceeds in the same way as it was described above for $H_{ij}^-(k,s+1\,|\,k,s)$ and $W_{ij}^-(k,s+1\,|\,k,s)$ for $s=0,1,\ldots$. This concludes the *k*-loop of the modified algorithm for distributed estimation with a limited communication radius.

3.4.3 Remarks on the algorithm

The modified algorithm described in this section reduces the overall computational load as well as the communication overhead. On the other hand, it affect the performance of the filter from the global view (in terms, for instance, of the trace of the global state covariance). The degree of performance degradation with respect to the communication radius is hard to establish a priori -- sensitivity of the performance to the radius depends significantly on the dimension and the distribution of process measurements.

The network-wide equilibrium at the end of the s-iterations is such that all Kalman gains K_i are zero for all i and that all variances of the fictitious noises are zero as well. Although we have not yet a rigorous proof for the convergence to this equilibrium, it has converged in our simulational tests. Compared to the case without restrictions for the communication radius, the convergence is slower – the former case converges usually in a finite number of s-steps. Another point that needs to be mentioned, covariance matrices $M_{ii}(k,s+s\,|\,k,s)$ have not all eigenvalues decreasing during the iterations – this is caused by adding the correction term in (68) in order to achieve the global multi-band structure. In other words, the global uncertainty ellipsoid given by the overall state covariance approximation shrinks in some directions and grows in other ones. From this we can derive an ad-hoc test for possible on-line adjustment of the communication radius; if the largest positive eigenvalue of matrix

$$\Delta_{ii}(k,s)=M_{ii}(k,0\,|\,k,-1)-M_{ii}(k,s_{end}\,|\,k,s_{end}-1) \tag{72}$$

is not significantly smaller that the opposite of the smallest (negative) eigenvalue (say by an order of magnitude), then an increase of the communication radius is recommended.

3.5 Illustrative example

Here, we present an example of coupled spring-mass systems. We merge three spring-mass systems into one subsystem; the coupling inputs are displacement and velocity at one end and force on the other; the outputs are the same, in reversed order. This subsystem is discretized using zero-order hold; we are fully aware that this introduces some error, when more of those discrete-time subsystems are connected, compared to other discretization

methods; the point is to have no feedthrough term between the coupling inputs and outputs. Therefore, we chose a sampling rate so that the error was not significant in the simulation for a cascade of 9 subsystems; moreover, we used a low-pass filter at the force input. Parameters of the elements are not all the same.

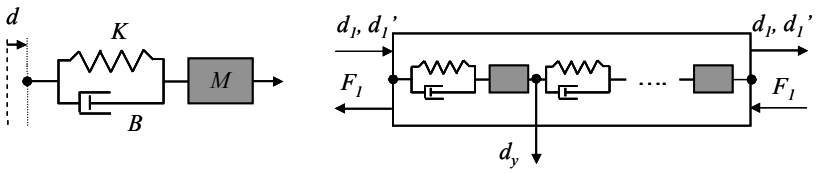


Figure 19 Elementary spring mass system; multiple spring/mass elements merged into a subsystem.

The cascade of 9 subsystems is shown in Figure 20. It has total 61 states and 5 scalar measurements distributed among subsystems with an odd index; hence, every other local KF has no direct access to process data and uses the information received from the neighbours. The measurements are either displacement or velocity chosen somewhat arbitrarily. In the following we address only the evolution of the covariance matrices. First, we shall take the case without restrictions on the communication radius.

Some illustrative plots of the evolution of certain filter variables during s-iterations are shown in Figure 21. It is for a sufficiently large time k, where the filter trajectories in s-iteration did not change much from one time step to another; Local KFs were thus near-periodic systems. We can observe some interesting properties: First; the network settles in 8 s-iterations (not incidentally, the largest network distance between subsystems). Kalman gains are pulsating -- this is due to the fact that the filter receives new relevant information on the process every other iteration, due to the distribution of the measurements. Further, the local state-to-global output covariance vanishes completely for each subsystem; this means that this distributed estimator is equivalent to the centralized KF. This can inferred from the fact that the measurements are of dimension smaller than/equal to the coupling variables, and that a measurement $y_l(k)$ is mapped on the coupling error estimate $\delta_{ij}(k, s_r \mid k, s_r - 1)$, where $d(i,l) = s_r$,.without any rank loss. It can be seen that the agents reach the zero state-to-output covariance in the iteration corresponding to the largest network distance to a process measurement. Finally, the bottom plot in Figure 21 shows the norm of the covariance matrix between a local state to the global set of consensus variables. It can be observed that this norm reduces, for all states, by an order of magnitude in 3 steps; this is sufficient, in this particular case, to terminate the s-iterations after 3 steps with only a small degradation of performance. If we had performed only one s-iteration, then the dissensus is large and the filter becomes unstable.

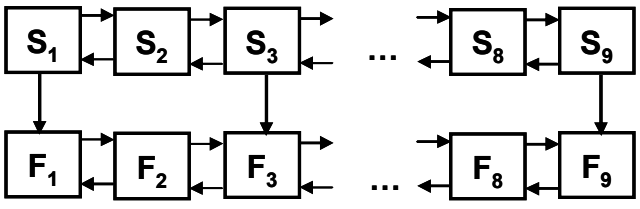


Figure 20 Process and filter networks

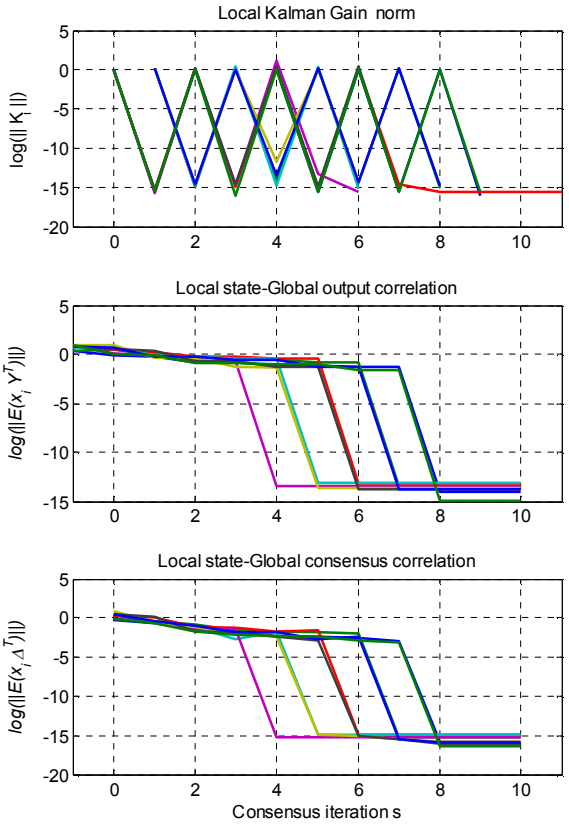


Figure 21 Top: $\log\|K_i(k,s)\|$; center: logs of the norms of local states to the global output covariance; Bottom: logarithms of the norms of the cross-covariances from the local states to the global set of coupling variable errors.

Now, we shall restrict the communication radius from 8 to 6. The evolution of filter variables is then shown in Figure 22. We can observe that the Kalman gains take much longer to vanish; further, the state-to-output covariance does not vanish at all—the filter is not optimal; nevertheless, the s-iteration still reduce the correlation significantly. Figure 23 shows that this communication restriction does not hurt the performance badly. However, reducing the communication radius further by one causes a significant degradation of performance. One would conclude that the proposed method does not allow much of the communication reduction. Note however, that the process measurements available globally in the network are fairly scarce, and thus the local filters heavily rely on data from distant subsystems. For a richer and evenly distributed measurement set, the communication radius can be reduced much more.

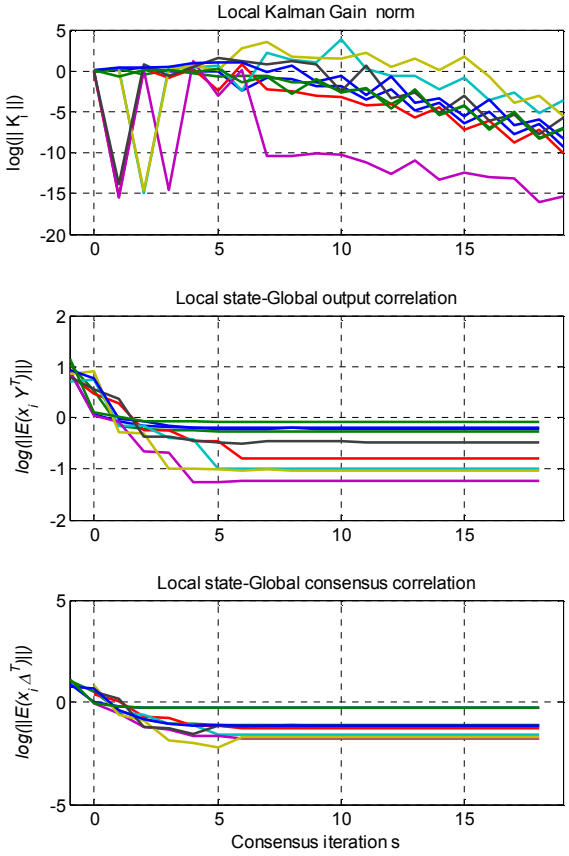


Figure 22 Evolution of filter variables during s-iterations analogous to Figure 21 for restricted communication radius.

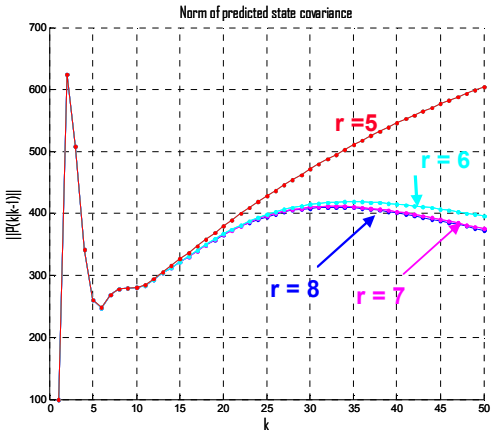


Figure 23 Norm of the predicted state covariance for distributed estimator with different communication radii

3.6 Conclusion

We have introduced a novel suboptimal distributed state estimator for interconnected systems. It is scalable and on-line reconfigurable; it admits a time-varying process models. Local filters estimate local state augmented by interconnection inputs. These inputs represent the only overlaps with state spaces of neighboring filters. Date update of the conditional mean and covariance is done iteratively within one time period, using interconnection variable estimates from the neighbors. The estimation uncertainty is propagated, from one local filter to another in the form of certain covariance matrices. To manage this uncertainty, local filters have to estimate covariance of local states with other states in the network; this requires model knowledge of other subsystems as well as the knowledge of certain intermediate results. A technique was proposed to limit the need for the data related to other subsystems to a certain network neighborhood.

4 Performance evaluation of decentralized versus centralized MPC on the benchmark problem

The decentralized MPC law proposed in Section 1.4 is compared in this section against the corresponding centralized version on the simulation benchmark example described in D1.1. The benchmark is about a linear, relatively large-scale system representing the temperature at different passenger areas in a railcar. The results are also reported in Section 5 of Deliverable D4.2, where the emphasis is on testing the control algorithms in the presence of packet loss.

We investigate different simulation outcomes of the comparison depending on three ingredients:

- type of controller (centralized/decentralized);
- changes in reference values;
- changes of external temperature (acting as a measured disturbance).

4.1 Simulation setup

The initial condition is 17°C for all seat-area temperatures, except for the antechamber, which is 15°C. The sample time used for discretizing the continuous time differential equations describing the dynamics of the system and obtain the prediction model is 9 minutes, which is also the sample time of the controller.

The temperature profile used in the simulation is the discretization of a typical sunny day of January in center Italy. Figure 14 shows the temperature in Celsius degrees in the vertical axis, as a function of the number of samples starting from midnight, *i.e.* 00:00 AM = 0 samples and 12:00 PM = 160 samples. Since a change in the external temperature requires a centralized update of the controller bounds, the temperature profile is discretized with a step of 20 samples.

User defined references for areas temperatures are described in the following:

- $\#1 : \#4$ constant at 18 degrees;
- $\#5 : \#8$ start at 18. then change to 18.1 at sample time 40;
- $\#9 : \#12$ start at 17.8, then change to 17.9 at sample time 120;
- $\#13 : \#16$ constant at 18.1.

Figure 15 gives a graphical interpretation of the used references, showing the time instants at which the setpoints change. Note that setpoints variations are chosen so that reference values differ in each adjacent area, which makes the control task more challenging.

4.2 Simulation results

Since the number of outputs variables of interest is considerably high, we show in Figure 16 only the first state and input trajectories obtained with both DMPC and CMPC, in order to permit a better understanding.

The closed-loop trajectories of centralized MPC feedback vs. decentralized MPC are shown in Figure 16(a), while Figure 16(b) shows the trajectory of the first applied input. In both the decentralized and centralized MPC case the temperature values of the four-seat areas converge to the set-point asymptotically.

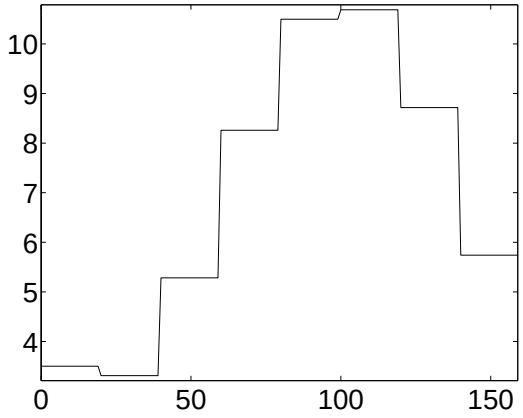


Figure 14: External temperature profile

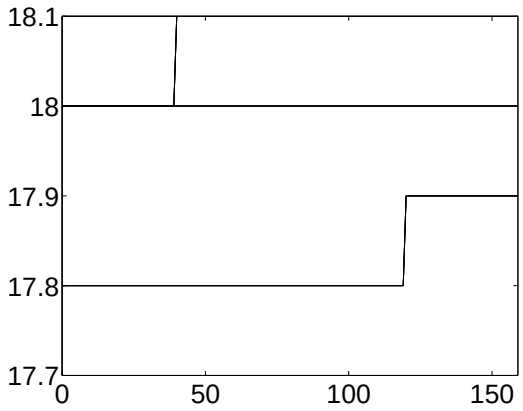
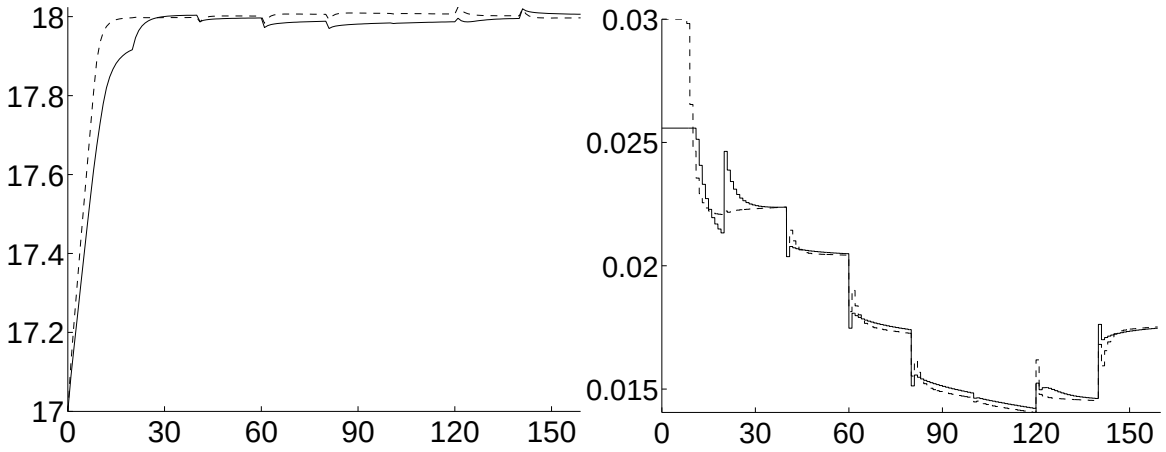


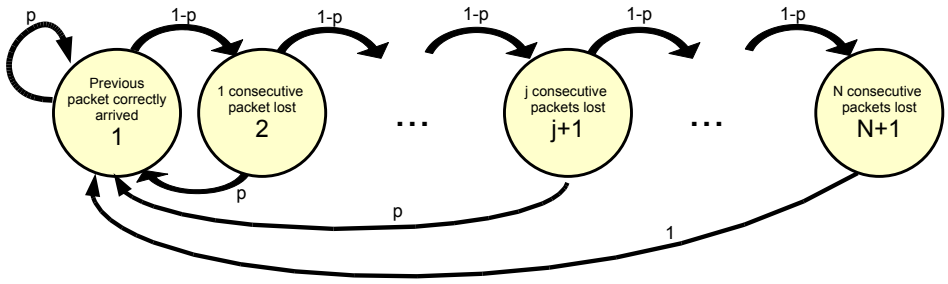
Figure 15: User defined references used in simulations



(a) Output trajectory (b) Applied input

Figure 16: Comparison between centralized MPC (dashed lines) and decentralized MPC (continuous lines): output h_1 (left hand side plots) and input v_1 (right hand side plots).

The comparison of centralized vs. decentralized MPC becomes more interesting when packet loss are present on the wireless feedback channels. To simulate packet loss, we assume that the probability of losing a packet depends on the state of the Markov chain depicted below.



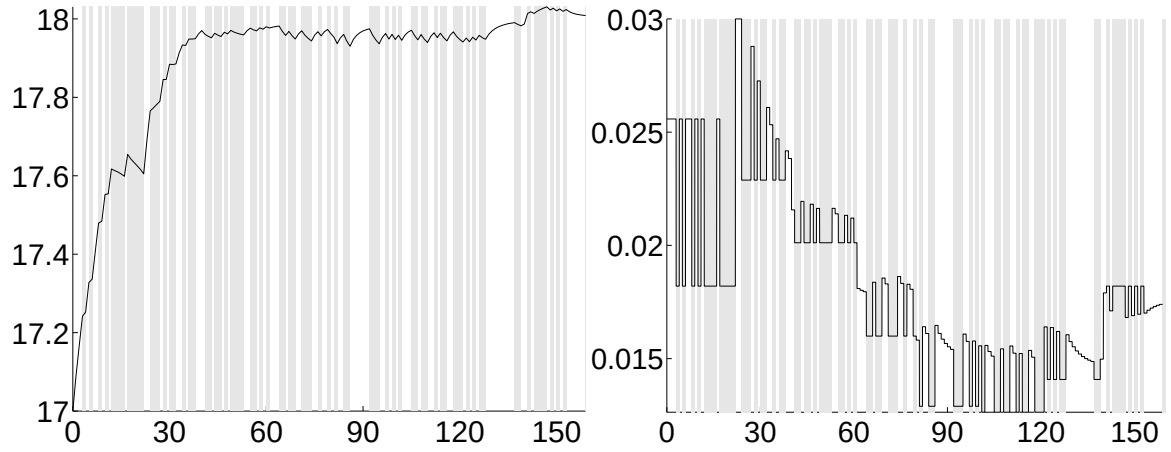
The Markov chain is in the jth state if $j-1$ consecutive packets have been lost. The probability of losing a further packet is $1-p$, $0 \leq p \leq 1$, except for the $(N+1)$th state where no packet can be lost any more. Such a probability model is partially confirmed by the experimental results on relative frequencies of packet failure burst length observed in [55].

The simulation results obtained with $p = 0.5$ are shown in Figure 17 and Figure 18. The stability condition reported in Deliverable D4.2 was tested and proved satisfied for values of j up to 160 (corresponding to a one-day lack of measurements, more than enough for any practical scenario).

It is evident that, despite the intermittent lack of information, the decentralized controller achieves tracking of the user defined references. Note that in instants where losses occurs the input is set to the asymptotical value, thanks to the coordinate shift, and thus the convergence is slower. In facts, the output get farther to the reference with respect to the previous instant. Moreover, the centralized controller, whose trajectories are not reported for brevity, behaves similarly during the "blind" instants.

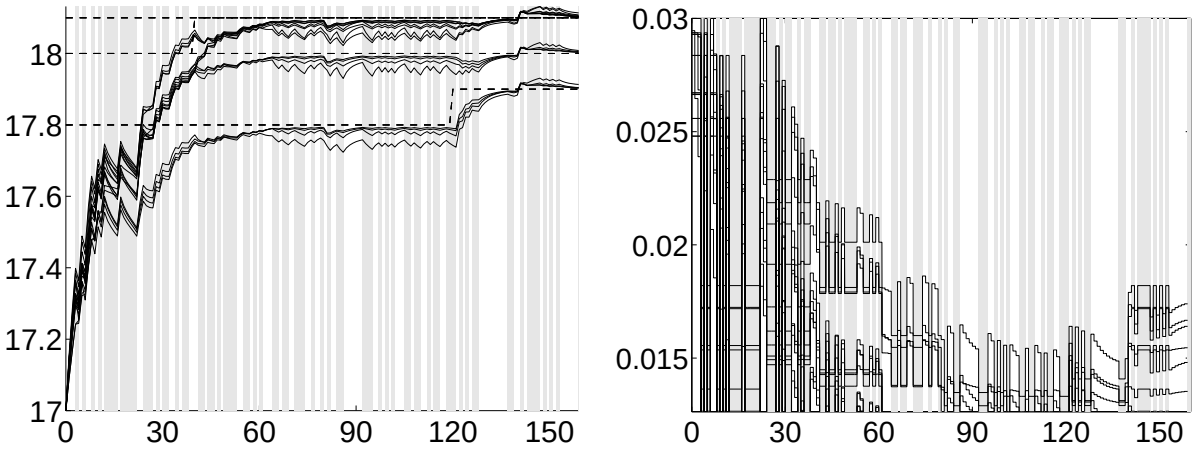
In order to compare closed-loop performances in different simulation scenarios, define the following performance index

$$J = \sum_{t=1}^{N_{\text{sim}}} e_z'(t) Q e_z(t) + e_v'(t) R e_v(t) \tag{73}$$



(a) Output trajectory (b) Applied input

Figure 17: Decentralized MPC output (left hand side plots) and input (right hand side plots), with loss probability $p = 0.5$, where grey background indicates a full blackout of the transmission system, which implies that all packets are lost at that instant.



(a) Output trajectories (b) Applied inputs

Figure 18: Decentralized MPC results. Left hand side plots: output variables (continuous lines) and references (dashed lines). Right hand side plots: command inputs. Gray areas denote packet drop intervals.

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

Figure 19 shows that the performance index J defined in Eq. (73) increases as the packet-loss probability grows, implying performance to deteriorate due to the conservativeness of the backup control action $u = 0$ (that is, $v = v_r$). The results of Figure 19 are averaged over 10 simulations per probability sample.

It is apparent, as expected, that centralized MPC dominates over decentralized MPC. However, for certain values of p the average performance of decentralized MPC is slightly better, probably due to the particular packet loss sequences that have realized. However, the loss of performance due to decentralization, with regard to the benchmark simulation example, is largely negligible. This is due to the weak dynamical coupling between non-neighboring passenger areas.

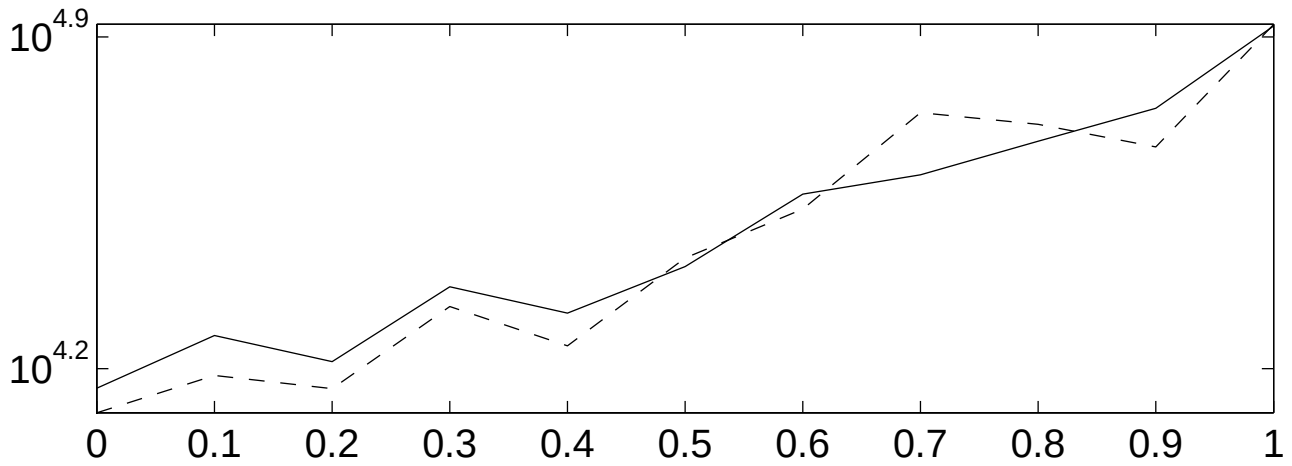


Figure 19: Performance indices of Centralized MPC (dashed line) and Decentralized MPC (solid line)

4.2.1 Computation complexity

The simulations were performed on a MacBook Air 1.86 GHz running Matlab R2008a under OS X 10.5.6 and the Hybrid Toolbox for Matlab [56]. The average CPU time for solving the centralized QP problem associated with (3) is 6.0 ms (11.9 ms in the worst case). For the decentralized case, the average CPU time for solving the QP problem associated with (15) is 3.3 ms (7.4 ms in the worst case).

4.3 Conclusion

In this section a comparison between centralized and decentralized MPC was presented for the benchmark example described in D1.1 and D4.2. Simulation results showed that closed-loop performance and complexity of MPC computations can be traded off. Moreover, the comparison was tested simulating an unreliable communication channel, where the lack of information makes the distinction between optimality and suboptimal more fuzzy.

Further comparisons of decentralized versus centralized MPC strategies will be reported in deliverable D3.3, where centralized hierarchical schemes are compared against decentralized ones, and hybrid decentralized MPC versus hybrid centralized MPC is tested on a UAV formation flight application. Moreover, in D5.6 the decentralized and centralized MPC approaches will be compared on the simulation demo related to the Barcelona water distribution network.

References

[1] A. Alessio, D. Barcelli, and A. Bemporad. Decentralized model predictive control of dynamically-coupled linear systems. Journal of Process Control. Conditionally accepted for publication.

[2] A. Alessio and A. Bemporad. Decentralized model predictive control of constrained linear systems. In *Proc. European Control Conf.*, pages 2813–2818, Kos, Greece, 2007.

[3] A. Alessio and A. Bemporad. Stability conditions for decentralized model predictive control under packet dropout. In *Proc. American Contr. Conf.*, pages 3577–3582, Seattle, WA, 2008.

[4] B. Bamieh, F. Paganini, and M. A. Dahleh. Distributed control of spatially invariant systems. *IEEE Trans. Automatic Control*, 47(7):1091–1107, 2002.

[5] D. Barcelli and A. Bemporad. Decentralized model predictive control of dynamically-coupled linear systems: Tracking under packet loss. In *1st IFAC Workshop on Estimation and Control of Networked Systems*, pages 204–209, Venice, Italy, 2009.

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

[7] F. Borrelli, T. Keviczky, K. Fregene, and G.J. Balas. Decentralized receding horizon control of cooperative vechicle formations. In *Proc. 44th IEEE Conf. on Decision and Control and European Control Conf.*, pages 3955–3960, Sevilla, Spain, 2005.

[8] R. H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16:1190–1208, 1995.

[9] F.M. Callier, W.S. Chan, and C.A. Desoer. Input-output stability theory of interconnected systems using decomposition techniques. *IEEE Trans. Circuits and Systems*, 23(12):714–729, 1976.

[10] E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, pages 44–52, February 2002.

[11] R. Cheng, J. F. Forbes, and W. S. Yip. Price-driven coordination method for solving plant-wide mpc problems. *Journal of Process Control*, 17:429–438, 2006.

[12] A. Damoiseaux, A. Jokic, M. Lazar, P.P.J. van den Bosch, I.A. Hiskens, A. Alessio, and A. Bemporad. Assessment of decentralized model predictive control techniques for power networks. In *16th Power Systems Computation Conference*, Glasgow, Scotland, 2008.

[13] R. D'Andrea. A linear matrix inequality approach to decentralized control of distributed parameter systems. In *Proc. American Contr. Conf.*, pages 1350–1354, 1998.

[14] W.B. Dunbar and R.M. Murray. Distributed receding horizon control with application to multi-vehicle formation stabilization. *Automatica*, 42(4):549–558, 2006.

[15] P. Grieder and M. Morari. Complexity reduction of receding horizon control. In *Proc. 42th IEEE Conf. on Decision and Control*, pages 3179–3184, Maui, Hawaii, USA, 2003.

[16] D. Jia and B. Krogh. Distributed model predictive control. In *Proc. American Contr. Conf.*, pages 2767–2772, Arlington, VA, 2001.

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

[18] M. Johannson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.

[19] B. Johansson, T. Keviczky, M. Johansson, and K.H. Johansson. Subgradient methods and consensus algorithms for solving convex optimization problems. In *Proc. 47th IEEE Conf. on Decision and Control*, pages 4185–4190, Cancun, Mexico, 2008.

[20] T. Keviczky, F. Borrelli, and G.J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12):2105–2115, 2006.

[21] W. Li and C.G. Cassandras. Stability properties of a receding horizon contoller for cooperating UAVs. In *Proc. 43th IEEE Conf. on Decision and Control*, pages 2905–2910, Paradise Island, Bahamas, 2004.

[22] Dong C. Liu, Jorge Nocedal, and Dong C. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[23] L. Magni and R. Scattolini. Stabilizing decentralized model predictive control of nonlinear systems. *Automatica*, 42(7):1231–1236, 2006.

[24] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[25] M. Mercangöz and F.J. Doyle III. Distributed model predictive control of an experimental four-tank system. *Journal of Process Control*, 17(3):297–308, 2007.

[26] A.N. Michel. Stability analysis of interconnected systems. *SIAM J. Contr.*, 12:554–579, August 1974.

[27] A.N. Michel and R.D. Rasmussen. Stability of stochastic composite systems. *IEEE Trans. Automatic Control*, 21:89–94, 1976.

[28] Y. Nesterov. A method for solving a convex programming problem with convergence rate o($1/k^2$). *Soviet Math. Dokl.*, 27:372–376, 1983.

[29] Y. Nesterov. *Introductory lectures on convex opimization: A basic course*. Kluwer Academic Publishers, 2003.

[30] A. Richards and J.P. How. Decentralized model predictive control of cooperating UAVs. In *Proc. 43th IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, 2004.

[31] M. Rotkowitz and S. Lall. A characterization of convex problems in decentralized control. *IEEE Trans. Automatic Control*, 51(2):1984–1996, 2006.

[32] S. Samar, S. Boyd, and D. Gorinevsky. Distributed estimation via dual decomposition. In *Proc. European Control Conf.*, pages 1511–1516, Kos, Greece, 2007.

[33] N.R. Sandell, P. Varaiya, M. Athans, and M.G. Safonov. Survey of decentralized control methods for large scale systems. *IEEE Trans. Automatic Control*, 23(2):108–128, 1978.

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

[35] A.N. Venkat, I.A. Hiskens, J.B. Rawlings, and S.J. Wright. Distributed MPC strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, 2008.

[36] A.N. Venkat, J.B. Rawlings, and J.S. 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.

[37] A.N. Venkat, J.B. Rawlings, and J.S. Wright. Implementable distributed model predictive control with guaranteed performance properties. In *Proc. American Contr. Conf.*, pages 613–618, Minneapolis, MN, 2006.

[38] D. D. Šiljak. *Large-Scale Dynamic Systems: Stability and Structure*. North-Holland, New York, 1978.

[39] S. Wang and E. J. Davison. On the stabilization of decentralized control systems. *IEEE Trans. Automatic Control*, 18(5):473–478, 1973.

[40] S. Samar, S. Boyd, and D.S. Gorinevsky. Distributed estimation via dual decomposition. In *Proceedings of European Control Conference*, Kos, Greece, 2007.

[41] R. Olfati-Saber. Distributed Kalman Filtering for Sensor Networks. In *Proc of 46th IEEE Conf on Decision and Control*, New Orleans, LA, USA, 2007.

[42] R. Olfati-Saber and R.M. Murray. Consensus Problem in Network Agents with Switching Topology and Time Delays. *IEEE Trans on Automatic Control*, Vol. 49, pp. 1520–1553, 2004.

[43] R. Olfati-Saber. Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory. *IEEE Trans on Automatic Control*, Vol 51, pp. 401–420, 2006.

[44] R. Carli, R. Chiuso, L. Schenato, and S. Zampieri. Distributed Kalman Filtering Using Consensus Strategies. In *Proc. of 46th IEEE Conf on Decision and Control*, New Orleans, LA, USA, 2007.

[45] Sijs,J., Lazar, M. van der Bosch, P.P.J and Papp, Z. An overview of non-centralized Kalman Filters. In *Proceedings of 17th International Conf on Control Applications*, Part of 2008 IEEE Multi-conference on Systems and Control, San Antonio, TX, USA, 2008.

[46] Khan, U.A. and Moura, J.M.F. Distributed Kalman Filters in Sensor Networks: Bipartite Fusion Graphs. In *Proc. of 15th IEEE Workshop on Statistical Signal processing*, Madison, WI, 2007.

[47] Khan, U.A. and Moura, J.M.F. Distributing the Kalman Filter for large-Scale Systems. *IEEE Trans on Signal Processing*, Vol 56, Oct. 2008, pp 4919-4935.

[48] Stankovic, S. S., Stankovic, M. S. and Stipanovic, D. M. Consensus-based Overlapping Decentralized Estimator. *IEEE Transactions on Automatic Control*, Vol. 54, Feb 2009, pp 410-412.

[49] Vadigepalli, R. and Doyle, F. J. III. A distributed State Estimation and Control Algorithm for Plantwide processes. *IEEE Transactions on Control Technology*, l Vol 2, Jan 2003, pp 119-127.

[50] Zhu,Y., You, Z., Zhao, J., Zhang, K. and Rong Li, X. The optimality for the distributed Kalman filtering fusion with feedback. *Automatica*, Vol 37, 2001, pp 1489-1493.

[51] Sanders, C.W., Tacker, E.C., Linton, T.D. and Ling R. Y.-S. Specific structures for Large-Scale State Estimation Algorithms having Information Exchange. *IEEE Trans on Automatic Control*, Vol. AC-23, April 1978, pp 255-261.

[52] Hashemipour, H.R., Roy, S. and Laub, A.J. Decentralized Structures for parallel Kalman Filtering. *IEEE Transactions on Automatic Control*, Vol.33, January 1988, pp 88-94.

[53] Baramov, L. and Havlena. V. Distributed Kalman Filter for interconnected systems with consensus. Under review, *International Journal of Control*, 2009.

[54] Baramov, L. Pachner, D. and Havlena, V. Approximate distributed Kalman filter for interconnected systems. Under preparation.

[55] A. Willig and R. Mitschke. Results of bit error measurements with sensor nodes and casuistic consequences for design of energy-efficient error control schemes. In *Proc. 3rd European Workshop on Wireless Sensor Networks (EWSN)*, volume 3868 of *Lecture Notes in Computer Science*, pages 310–325. Springer-Verlag, 2006.

[56] A. Bemporad. *Hybrid Toolbox – User's Guide*. December 2003. `http://www.dii.unisi.it/hybrid/toolbox`.