4

Distributed Model Predictive Control Based on Dynamic Games

Guido Sanchez[1], Leonardo Giovanini[2], Marina Murillo[3] and Alejandro Limache[4]

[1,2]*Research Center for Signals, Systems and Computational Intelligence*
Faculty of Engineering and Water Sciences, Universidad Nacional del Litoral
[3,4]*International Center for Computer Methods in Engineering*
Faculty of Engineering and Water Sciences, Universidad Nacional del Litoral
Argentina

1. Introduction

Model predictive control (*MPC*) is widely recognized as a high performance, yet practical, control technology. This model-based control strategy solves at each sample a discrete-time optimal control problem over a finite horizon, producing a control input sequence. An attractive attribute of *MPC* technology is its ability to systematically account for system constraints. The theory of *MPC* for linear systems is well developed; all aspects such as stability, robustness,feasibility and optimality have been extensively discussed in the literature (see, e.g., (Bemporad & Morari, 1999; Kouvaritakis & Cannon, 2001; Maciejowski, 2002; Mayne et al., 2000)). The effectiveness of *MPC* depends on model accuracy and the availability of fast computational resources. These requirements limit the application base for *MPC*. Even though, applications abound in process industries (Camacho & Bordons, 2004), manufacturing (Braun et al., 2003), supply chains (Perea-Lopez et al., 2003), among others, are becoming more widespread.

Two common paradigms for solving system-wide *MPC* calculations are centralised and decentralised strategies. Centralised strategies may arise from the desire to operate the system in an optimal fashion, whereas decentralised *MPC* control structures can result from the incremental roll-out of the system development. An effective centralised *MPC* can be difficult, if not impossible to implement in large-scale systems (Kumar & Daoutidis, 2002; Lu, 2003). In decentralised strategies, the system-wide *MPC* problem is decomposed into subproblems by taking advantage of the system structure, and then, these subproblems are solved independently. In general, decentralised schemes approximate the interactions between subsystems and treat inputs in other subsystems as external disturbances. This assumption leads to a poor system performance (Sandell Jr et al., 1978; Šiljak, 1996). Therefore, there is a need for a cross-functional integration between the decentralised controllers, in which a coordination level performs steady-state target calculation for decentralised controller (Aguilera & Marchetti, 1998; Aske et al., 2008; Cheng et al., 2007; 2008; Zhu & Henson, 2002).

Several distributed *MPC* formulations are available in the literature. A distributed *MPC* framework was proposed by Dumbar and Murray (Dunbar & Murray, 2006) for the class

of systems that have independent subsystem dynamic but link through their cost functions and constraints. Then, Dumbar (Dunbar, 2007) proposed an extension of this framework that handles systems with weakly interacting dynamics. Stability is guaranteed through the use of a consistency constraint that forces the predicted and assumed input trajectories to be close to each other. The resulting performance is different from centralised implementations in most of cases. Distributed *MPC* algorithms for unconstrained and *LTI* systems were proposed in (Camponogara et al., 2002; Jia & Krogh, 2001; Vaccarini et al., 2009; Zhang & Li, 2007). In (Jia & Krogh, 2001) and (Camponogara et al., 2002) the evolution of the states of each subsystem is assumed to be only influenced by the states of interacting subsystems and local inputs, while these restrictions were removed in (Jia & Krogh, 2002; Vaccarini et al., 2009; Zhang & Li, 2007). This choice of modelling restricts the system where the algorithm can be applied, because in many cases the evolution of states is also influenced by the inputs of interconnected subsystems. More critically for these frameworks is the fact that subsystems-based *MPC*s only know the cost functions and constraints of their subsystem. However, stability and optimality as well as the effect of communication failures has not been established.
The distributed model predictive control problem from a game theory perspective for *LTI* systems with general dynamical couplings, and the presence of convex coupled constraints is addressed. The original centralised optimisation problem is transformed in a dynamic game of a number of local optimisation problems, which are solved using the relevant decision variables of each subsystem and exchanging information in order to coordinate their decisions. The relevance of proposed distributed control scheme is to reduce the computational burden and avoid the organizational obstacles associated with centralised implementations, while retains its properties (stability, optimality, feasibility). In this context, the type of coordination that can be achieved is determined by the connectivity and capacity of the communication network as well as the information available of system's cost function and constraints. In this work we will assume that the connectivity of the communication network is sufficient for the subsystems to obtain information of all variables that appear in their local problems. We will show that when system's cost function and constraints are known by all distributed controllers, the solution of the iterative process converge to the centralised MPC solution. This means that properties (stability, optimality, feasibility) of the solution obtained using the distributed implementation are the same ones of the solution obtained using the centralised implementation. Finally, the effects of communication failures on the system's properties (convergence, stability and performance) are studied. We will show the effect of the system partition and communication on convergence and stability, and we will find a upper bound of the system performance.

2. Distributed Model Predictive Control

2.1 Model Predictive Control

MPC is formulated as solving an on-line open loop optimal control problem in a receding horizon style. Using the current state $x(k)$, an input sequence $U(k)$ is calculated to minimize a performance index $J(x(k), U(k))$ while satisfying some specified constraints. The first element of the sequence $u(k,k)$ is taken as controller output, then the control and the prediction horizons recede ahead by one step at next sampling time. The new measurements are taken to compensate for unmeasured disturbances, which cause the system output to be

different from its prediction. At instant k, the controller solves the optimisation problem

$$\begin{aligned} &\min_{U(k)} J\left(x(k), U(k)\right) \\ &st. \\ X(k+1) =& \Gamma x(k) + \mathcal{H} U(k) \\ U(k) \in& \mathcal{U} \end{aligned} \tag{1}$$

where Γ and $\mathcal{H}$ are the observability and Haenkel matrices of the system (Maciejowski, 2002) and the states and input trajectories at time k are given by

$$\begin{aligned} X(k) &= \left[x(k,k) \ \cdots \ x(k+V,k)\right]^T \quad V > M, \\ U(k) &= \left[u(k,k) \ \cdots \ u(k+M,k)\right]^T. \end{aligned}$$

The integers V and M denote the prediction and control horizon. The variables $x(k+i,k)$ and $u(k+i,k)$ are the predicted state and input at time $k+i$ based on the information at time k and system model

$$x(k+1) = Ax(k) + Bu(k), \tag{2}$$

where $x(k) \in R^{n_x}$ and $u(k) \in \mathcal{U} \subseteq R^{n_u}$. The set of global admissible controls $\mathcal{U} = \{u \in R^{n_u} \ | Du \leq d, d > 0\}$ is assumed to be non-empty, compact and convex set containing the origin in its interior.

Remark 1. *The centralised model defined in (2) is more general than the so-called composite model employed in (Venkat et al., 2008), which requires the states of subsystems to be decoupled and allows only couplings in inputs. In this approach, the centralised model can represent both couplings in states and inputs.*

In the optimisation problem (1), the performance index $J\left(x(k), U(k)\right)$ measures of the difference between the predicted and the desired future behaviours. Generally, the quadratic index

$$J\left(x(k), U(k)\right) = \sum_{i=0}^{V} x^T(k+i,k) Q_i x(k+i,k) + \sum_{i=0}^{M} u^T(k+i,k) R_i u(k+i,k) \tag{3}$$

is commonly employed in the literature. To guarantee the closed–loop stability, the weighting matrices satisfy $Q_i = Q > 0$, $R_i = R > 0 \quad \forall i \leq M$ and $Q_i = \bar{Q} \quad \forall i > M$, where $\bar{Q}$ is given by $A^T \bar{Q} A - \bar{Q} = -Q$ (Maciejowski, 2002). For this choice of the weighting matrices, the index (3) is equivalent to a performance index with an infinite horizon.

$$J_\infty\left(x(k), U(k)\right) = \sum_{i=0}^{\infty} x^T(k+i,k) Q x(k+i,k) + u^T(k+i,k) R u(k+i,k).$$

In many formulations an extra constraint or extra control modes are included into (1) to ensure the stability of the closed-loop system (Maciejowski, 2002; Rossiter, 2003).

2.2 Distributed MPC framework

Large-scale systems are generally composed of several interacting subsystems. The interactions can either be: *a*) dynamic, in the sense that the states and inputs of each subsystem influence the states of the ones to which it is connected, *b*) due to the fact that the subsystems

share a common goal and constraint, or *c*) both. Systems of this type admit a decomposition into m subsystems represented by

$$x_l(k+1) = \sum_{p=1}^{m} A_{l\,p} x_p(k) + B_{l\,j\in\mathcal{N}_p} u_{j\in\mathcal{N}_p}(k) \quad l = 1, \ldots, m \tag{4}$$

where $x_l \in R^{n_{x_l}} \subseteq R^{n_x}$ and $u_l \in \mathcal{U}_l \subseteq R^{n_{u_l}} \subset R^{n_u}$ are the local state and input respectively. The set of control inputs indices of subsystem l is denoted $\mathcal{N}_l$, and the set $\mathcal{I}$ denotes all control input indices such that $u(k) = u_{j\in\mathcal{I}}(k)$.

Remark 2. *This is a very general model class for describing dynamical coupling between subsystems and includes as a special case the combination of decentralised models and interaction models in (Venkat et al., 2008). The subsystems can share input variables such that*

$$\sum_{l=1}^{m} n_{u_l} \geq n_u. \tag{5}$$

Each subsystem is assumed to have local convex independent and coupled constraints, which involve only a small number of the others subsystems. The set of local admissible controls $\mathcal{U}_l = \{u_l \in R^{n_{u_l}} \mid D_l u_l \leq d_l, d_l > 0\}$ is also assumed to be non-empty, compact, convex set containing the origin in their interior.
The proposed control framework is based on a set of m independent agents implementing a small-scale optimizations for the subsystems, connected through a communication network such that they can share the common resources and coordinate each other in order to accomplish the control objectives.

Assumption 1. *The local states of each subsystem $x_l(k)$ are accessible.*

Assumption 2. *The communication between the control agents is synchronous.*

Assumption 3. *Control agents communicates several times within a sampling time interval.*

This set of assumption is not restrictive. In fact, if the local states are not accessible they can be estimated from local outputs $y_l(k)$ and control inputs using a Kalman filter, therefore **Assumption 1** is reasonable. As well, **Assumptions 2** and **3** are not so strong because in process control the sampling time interval is longer with respect the computational and the communication times.
Under these assumptions and the decomposition, the cost function (3) can be written as follows

$$J\left(x(k), U(k), A\right) = \sum_{l=1}^{m} \alpha_l J_l \left(x(k), U_{j\in\mathcal{N}_l}(k), U_{j\in\mathcal{I}-\mathcal{N}_l}(k)\right), \tag{6}$$

where $A = [\alpha_l]$, $\alpha_l \geq 0$, $\sum_{l=1}^{m} \alpha_l = 1$, $U_j(k)$ is the j-th system input trajectory. This decomposition of the cost function and input variable leads to a decomposition (1) into m coupled optimisation problems

$$\min_{U_{j\in\mathcal{N}_l}(k)} J\left(x(k), U(k), A\right)$$

$$
\begin{aligned}
&st.\\
&X(k+1) = \Gamma x(k) + \mathcal{H}U(k)\\
&U_{j\in\mathcal{N}_l}(k) \in \mathcal{U}_{j\in\mathcal{N}_l}\\
&U_{j\in\mathcal{I}-\mathcal{N}_l}(k) \in \mathbf{U}_{j\in\mathcal{I}-\mathcal{N}_l}
\end{aligned}
\tag{7}
$$

where $\mathbf{U}_{j\in\mathcal{I}-\mathcal{N}_l}$ denotes the assumed inputs of others agents. The goal of the decomposition is to reduce the complexity of the optimisation problem (1) by ensuring that subproblems (7) are smaller than the original problem (fewer decision variables and constraints), while they retain the properties of the original problem. The price paid to simplify the optimisation problem (1) is the needs of coordination between the subproblems (7) during their solution. In this way, the optimisation problem (1) has been transformed into a *dynamic game* of m agents where each one searches for their optimal decisions through a sequence of *strategic games*, in response to decisions of other agents.

Definition 1. *A dynamic game* $\langle m, \mathcal{U}, J_l\left(x(k), U^q(k), A\right), \mathcal{D}(q,k)\rangle$ *models the interaction of* m *agents over iterations* q *and is composed of: i)* $m \in \mathbb{N}$ *agents; ii) a non empty set* $\mathcal{U}$ *that corresponds to the available decisions* $U_l^q(k)$ *for each agent; iii) an utility function* $J_l\left(x(k), U^q(k), A\right) : x(k) \times U^q(k) \to \mathbb{R}^+$ *for each agent; iv) an strategic game* $\mathcal{G}(q,k)$ *that models the interactions between agents at iteration* q *and time* k*; v) a dynamic process of decision adjustment* $\mathcal{D}(q,k) : (U^q(k), \mathcal{G}(q,k), q) \to U^{q+1}(k)$.

At each stage of the dynamic game, the joint decision of all agents will determine the outcome of *the strategic game* $\mathcal{G}(q,k)$ and each agent has some preference $U^q_{j\in\mathcal{N}_l}(k)$ over the set of possible outcomes $\mathcal{U}$. Based on these outcomes and the adjustment process $\mathcal{D}(q,k)$, which in this framework depends on the cost function $J_l(\cdot)$ and constraints, the agents reconcile their decisions. More formally, a strategic game is defined as follows (Osborne & Rubinstein, 1994)

Definition 2. *A finite strategic game* $\mathcal{G}(q,k) = \langle m, \mathcal{U}_l, J_l\left(x(k), U^q(k), A\right)\rangle$ *models the interactions between* m *agents and is composed of: i) a non empty finite set* $\mathcal{U}_l \subseteq \mathcal{U}$ *that corresponds the set of available decisions for each agent; ii) an utility function* $J_l\left(x(k), U^q(k), A\right) : x(k) \times U^q(k) \to \mathbb{R} \quad U^q(k) \in \mathcal{U}$ *for each agent.*

In general, one is interested in determining the choices that agents will make when faced with a particular game, which is sometimes referred to as the *solution of the game*. We will adopt the most common solution concept, known as *Nash equilibrium* (Nash, 1951): a set of choices where no individual agent can improve his utility by unilaterally changing his choice. More formally, we have:

Definition 3. *A group of control decisions* $U(k)$ *is said to be Nash optimal if*

$$
J_l\left(x(k), U^q_{j\in\mathcal{N}_l}(k), U^{q-1}_{j\in\mathcal{I}-\mathcal{N}_l}(k)\right) \leq J_l\left(x(k), U^{q-1}_{j\in\mathcal{N}_l}(k), U^{q-1}_{j\in\mathcal{I}-\mathcal{N}_l}(k)\right)
$$

where $q > 0$ *is the number of iterations elapsed during the iterative process.*

If Nash optimal solution is achieved, each subproblem does not change its decision $U^q_{j\in\mathcal{N}_l}(k)$ because it has achieved an equilibrium point of the coupling decision process; otherwise the local performance index J_l will degrade. Each subsystem optimizes its objective function

using its own control decision $U^q_{j\in\mathcal{N}_l}(k)$ assuming that other subsystems' solutions $U^q_{j\in\mathcal{I}-\mathcal{N}_l}(k)$ are known. Since the mutual communication and the information exchange are adequately taken into account, each subsystem solves its local optimisation problem provided that the other subsystems' solutions are known. Then, each agent compares the new solution with that obtained in the previous iteration and checks the stopping condition

$$\left\| U^q_{j\in\mathcal{N}_l}(k) - U^{q-1}_{j\in\mathcal{N}_l}(k) \right\|_\infty \leq \varepsilon_l \quad l = 1, \ldots, m. \tag{8}$$

If the algorithm is convergent, condition (8) will be satisfied by all agents, and the whole system will arrive to an equilibrium point. The subproblems m (7) can be solved using the following iterative algorithm

Algorithm 1

Given $Q_l,\ R_l, 0 < q_{max} < \infty, \varepsilon_l > 0 \quad \forall l = 1, \cdots, m$
For each agent $l \quad l = 1, \cdots, m$
Step 1 Initialize agent l
1.*a* Measure the local state $x_l(k), q = 1,$
$\rho_l = \phi\varepsilon_l \quad \phi \gg 1$
1.*b* $U^0(k) = [u(k, k-1) \ \cdots \ u(k+M, k-1)\ 0\ ]$
Step 2 **while** $\rho_l > \varepsilon_l$ and $q < q_{max}$
2.*a* Solve problem (7) to obtain $\tilde{U}^q_{j\in\mathcal{N}_l}(k)$
2.*b* **for** $p = 1, \cdots, m$ and $p \neq l$
Communicate $\tilde{U}^q_{j\in\mathcal{N}_l}(k)$ to agent p
end
2.*c* Update the solution iterate $q\ \forall j \in \mathcal{N}_l$

$$U^q_j(k) = \sum_{p=1}^m \alpha_p \tilde{U}^q_{i\in\mathcal{N}_p\cap j}(k) + \left(1 - \sum_{p=1}^m \alpha_p\, card\,(\, j \cap \mathcal{N}_l)\right) U^{q-1}_j(k)$$

2.*d* $\rho_l = \left\| U^q_{j\in\mathcal{N}_l}(k) - U^{q-1}_{j\in\mathcal{N}_l}(k) \right\|_\infty$
$q = q + 1$
end
Step 3 Apply $u_l(k, k)$
Step 4 $k = k + 1$ and goto Step 1

At each k, q_{max} represents a design limit on the number of iterates q and ε_l represents the stopping criteria of the iterative process. The user may choose to terminate Algorithm 1 prior to these limits.

3. Properties of the framework

3.1 Performance

Given the distributed scheme proposed in the previous Section, three fundamental questions naturally arise: *a*) the behavior of agent's iterates during the negotiation process, *b*) the

location and number of equilibrium points of the distributed problem and *c*) the feasibility of the solutions. One of the key factors in these questions is the effect of the cost function and constraints employed by the distributed problems. Therefore, in a first stage we will explore the effect of the performance index in the number and position of the equilibrium points.
Firstly, the optimality conditions for the centralised problem (1) are derived in order to have a benchmark measure of distributed control schemes performance. In order to make easy the comparison, the performance index (3) is decomposed into m components related with the subsystems, like in the distributed problems (7), as follows

$$J\left(x(k), U(k), \Theta\right) = \sum_{l=1}^{m} \theta_l J_l\left(x(k), U(k)\right), \theta_l \geq 0, \sum_{l=1}^{m} \theta_l = 1. \tag{9}$$

This way writing the performance index corresponds to multiobjective characterization of the optimisation problem (1). Applying the first–order optimality conditions we obtain

$$\sum_{l=1}^{m} \theta_l \frac{\partial J_l\left(x(k), U(k)\right)}{\partial U_{j\in\mathcal{N}_p}(k)} + \lambda^T D^{j\in\mathcal{N}_p} = 0 \qquad p = 1, \ldots, m, \tag{10a}$$

$$\lambda^T \left(D^{j\in\mathcal{N}_p} U_{j\in\mathcal{N}_p}(k) - b\right) = 0, \tag{10b}$$

where D^j is the j–th column vector of D. The solution of this set of equations $U^*(k)$ is the optimal solution of the optimisation problem (1) and belongs to *Pareto set*, which is defined as (Haimes & Chankong, 1983).

Definition 4. *A solution $U^*(k) \in \mathcal{U}$ is said to be Pareto optimal of the optimisation problem (1) if there exists no other feasible solution $\forall U(k) \in \mathcal{U}$ such that $J_l\left(x(k), U(k)\right) \leq J_l\left(x(k), U^*(k)\right) \quad \forall l = 1, \ldots, m$.*

In distributed control the agents coordinate their decisions, through a negotiation process. Applying the first–order optimality conditions to decentralised cost (6) we obtain

$$\sum_{l=1}^{m} \alpha_l \frac{\partial J_l\left(x(k), U(k)\right)}{\partial U_{j\in\mathcal{N}_p}(k)} + \lambda^T D^{j\in\mathcal{N}_p} = 0 \qquad p = 1, \ldots, m, \tag{11a}$$

$$\lambda^T \left(D^{j\in\mathcal{N}_p} U_{j\in\mathcal{N}_p}(k) - b\right) = 0. \tag{11b}$$

By simple inspection of (10) and (11) we can see that these equations have the same structure, they only differ on the weights. Therefore, the location of the distributed schemes equilibrium will depend on the selection of $\alpha_l \quad l = 1, \ldots, m$. There are two options:

- If $\alpha_l = 1, \alpha_{p\neq l} = 0$ the optimality condition (11) becomes

$$\frac{\partial J_l\left(x(k), U(k)\right)}{\partial U_{j\in\mathcal{N}_l}(k)} + \lambda^T D^{j\in\mathcal{N}_l} = 0 \qquad l = 1, \ldots, m, \tag{12a}$$

$$\lambda^T \left(D^{j\in\mathcal{N}_l} U_{j\in\mathcal{N}_l}(k) - b\right) = 0. \tag{12b}$$

  This condition only evaluates the effect of $U_{j\in\mathcal{N}_l}$, given $U_{j\in\mathcal{I}-\mathcal{N}_l}$, in subsystem l without taking into account its effects in the remaining agents (**selfish behavior**). This configuration of the distributed problem leads to an **incomplete and perfect information game** that can achieve Nash optimal solutions for a pure strategy (**Cournot equilibrium**)

(Osborne & Rubinstein, 1994). By simple comparison of (10) and (12) we can conclude that the solution of this equations lies outside of the Pareto set (Dubey & Rogawski, 1990; Neck & Dockner, 1987). The reason of Nash equilibrium inefficiency lies in the fact that the information of each agent decision variable effects' on the remaining agents is neglected ($\alpha_{p\neq l} = 0$ incomplete information game). Therefore, each agent minimizes their performance index, accommodating the effects of other agents' decisions, without taking in account its effects on the rest of the system. Besides the lack of optimality, the number of equilibrium points generated by the optimality condition (12) can grow with the number of agents (Bade et al., 2007).

- If $\alpha_l > 0$ the optimality condition (11) becomes

$$\sum_{l=1}^{m} \alpha_l \frac{\partial J_l\left(x(k), U(k)\right)}{\partial U_{j\in\mathcal{N}_p}(k)} + \lambda^T D^{j\in\mathcal{N}_p} = 0 \qquad p = 1, \ldots, m \tag{13a}$$

$$\lambda^T \left(D^{j\in\mathcal{N}_p} U_{j\in\mathcal{N}_p}(k) - b \right) = 0. \tag{13b}$$

  This condition evaluates the effect of $U_{j\in\mathcal{N}_l}$, given $U_{j\in\mathcal{I}-\mathcal{N}_l}$, in the entire system, taking in account the effect of interactions between the subsystems (**cooperative behavior**), leading to a **complete and perfect information game**. By simple comparison of (10) and (13) it is easy to see that these two equations have a similar structure, therefore we can conclude that their solutions lie in the Pareto set. The position of distributed *MPC* solutions will depend on the values of α_l. In the particular case of $\alpha_l = \theta_l \quad l = 1, \ldots, m$ the solution of the centralised and distributed schemes are the same.

The value of weights $\alpha_l \quad l = 1, \ldots, m$ depends on the information structure; that is the information of the cost function and constraints available in each agent. If the cost function and constraints of each agent are known by all the others, for example a retailer company, α_l can be chosen like the second distributed scheme ($\alpha_l > 0 \quad \forall l = 1, \ldots, m$). In this case the centralised optimisation problem is distributed between m independent agents that coordinate their solutions in order to solve the optimisation problem in a distributed way. For this reason we call this control scheme **distributed MPC**. On the other case, when the local cost function and constraints are only known by the agents, for example a power network where several companies compete, the weights α_l should be chosen like the first scheme ($\alpha_l = 1, \alpha_{p\neq l} = 0 \quad \forall l, p = 1, \ldots, m$). In this case the centralised optimisation problem is decentralised into m independent agents that only coordinate the effects of their decisions to minimize the effect of interactions. For this reason we call this control scheme **coordinated decentralised MPC**.

Remark 3. *The fact that agents individually achieve Nash optimality does not imply the global optimality of the solution. This relationship will depend on the structure of agents' cost function and constraints, which depends on the value of weights α_l, and the number of iterations allowed.*

The structure of $U_{j\in\mathcal{N}_l}$ determine the structure of constraints that can be handled by the distributed schemes. If the subproblems share the input variables involved in the coupled constraints ($\mathcal{N}_l \cap \mathcal{N}_{p\neq l} \neq \emptyset$), the distributed MPC schemes can solve optimisation problems with coupled constraints. On the other hand, when subproblems do not include the input variables of coupled constraints ($\mathcal{N}_l \cap \mathcal{N}_{p\neq l} = \emptyset$), the distributed MPC schemes can only solves optimisation problems with independent constraints (Dunbar, 2007; Jia & Krogh, 2001; Venkat et al., 2008). These facts become apparent from optimality conditions (12) and (13).

3.2 Convergence

During the operation of the system, the subproblems (7) can compete or cooperate in the solution of the global problem. The behavior of each agent will depend on the existence, or not, of conflictive goals that can emerge from the characteristics of the interactions, the control goals and constraints. The way how the system is decomposed is one of the factors that defines the behavior of the distributed problem during the iterations, since it defines how the interactions will be addressed by distributed schemes.

The global system can be partitioned according to either the physical system structure or on the basis of an analysis of the mathematical model, or a combination of both. Heuristic procedures for the partitioning the system based on input–output analysis (see (Goodwin et al., 2005; Henten & Bontsema, 2009; Hovd & Skogestad, 1994)), an state–space analysis based (see (Salgado & Conley, 2004; Wittenmark & Salgado, 2002) or on performance metric for optimal partitioning of distributed and hierarchical control systems (see (Jamoom et al., 2002; Motee & Sayyar-Rodsari, 2003)) have been proposed. In all these approaches the objective is to simplify the control design by reducing the dynamic couplings, such that the computational requirements are evenly distributed to avoid excessive communication load. It is important to note that the partitioning of a state–space model can lead to overlapping states both due to coupled dynamics in the actual continuous system and due to discrete-time sampling, which can change the sparsity structure in the model.

Assumption 4. *The model employed by the distributed MPC algorithms are partitioned following the procedures described in (Motee & Sayyar-Rodsari, 2003).*

To analysed the effect of the system decomposition on the distributed constrained scheme, firstly we will analysed its effects on unconstrained problem. Solving the optimality condition (11) for an unconstrained system leads to

$$U^q(k) = \mathcal{K}_0 U^{q-1}(k) + \mathcal{K}_1 x(k) \quad \forall q > 0, \tag{14}$$

which models the behavior of the distributed problem during the iterative process. Its stability induces the convergence of the iterative process and it is given by

$$|\lambda\left(\mathcal{K}_0\right)| < 1. \tag{15}$$

The gain $\mathcal{K}_1$ is the decentralised controller that computes the contribution of $x(k)$ to $U(k)$ and has only non–zero elements on its main diagonal $\mathcal{K}_1 = [\mathcal{K}_{ll}] \quad l = 1, \ldots, m$. On other hand, $\mathcal{K}_0$ models the interaction between subsystems during the iterative process, determining its stability, and has non zero elements on its off diagonal elements

$$\mathcal{K}_0 = \begin{bmatrix} 0 & \mathcal{K}_{12} & \cdots & \mathcal{K}_{1m} \\ \mathcal{K}_{21} & 0 & & \mathcal{K}_{2m} \\ \vdots & & \ddots & \vdots \\ \mathcal{K}_{m1} & \cdots & \mathcal{K}_{mm-1} & 0 \end{bmatrix}. \tag{16}$$

The structure of the components of $\mathcal{K}_0$ and $\mathcal{K}_1$ depends on the value of the weights α_l:

- If the *coordinated decentralised MPC* is adopted ($\alpha_l = 1, \alpha_{p \neq l} = 0$) the elements of $\mathcal{K}_0$ are given by given by

$$\mathcal{K}_{lp} = -K_{ll}\mathcal{H}_{lp} \quad l, p = 1, \ldots, m \tag{17}$$

where $K_{ll} = \left(\mathcal{H}_{ll}^T Q_{ll} \mathcal{H}_{ll} + R_{ll}\right)^{-1} \mathcal{H}_{ll}^T Q_{ll}$. Therefore, the way in which the global problem (1) was partitioned and how the controllers' parameters were tuned defines the convergence of the *coordinated decentralised MPC*. Under **Assumption 2** the convergence of the algorithm can be guaranteed for those systems that exhibit weak interactions.

- On the other hand, when the *distributed MPC* is adopted ($\alpha_l > 0$) the gain $\mathcal{K}_0$ is given by

$$\mathcal{K}_{lp} = -K_{lp} \mathcal{H}_{lp} \quad l, p = 1, \ldots, m \tag{18}$$

where the controller gains are given by $K_{lp} = \left(\mathcal{H}_{lp}^T Q_{lp} \mathcal{H}_{lp} + R_{lp}\right)^{-1} \mathcal{H}_{lp}^T Q_{lp}$. Since the *distributed MPC* is designed to guarantee the stability of the entire system, its convergence is guaranteed independently of the way of partitioning the system.

Now, we will consider constrained systems. In this case, under Assumption 2, the convergence for constrained systems can be analysed using Lyapunov arguments. The key idea is to show the contractivity of the sequence of global cost functions $J\left(x(k), U^q(k)\right), A)$ generated by **Algorithm 1** along the iterative process.

Lemma 1. *Let's assume that the system has been partitioned following a decentralised design procedure and the distributed MPC problems (7) $\forall l = 1, \ldots, m$ are feasible, then the sequence of cost functions $J\left(x(k), U^q(k)\right), A)$ generated by **Algorithm 1** during the iterative process is non increasing $\forall q > 0$ at any time k.*

Proof. See appendix 8.A. □

3.3 Feasibility

Although in current literature it is typically assumed that an initial centralised feasible solution exist and is available, in this Section we will provide a simple and implementable way of constructing it in a distributed way assuming that the global initial state is available in advanced.

An initial feasible solution input $U^0_{j \in \mathcal{N}_l}(k)$ at $k = 0$ can be computed locally by using an inner approximation of the global feasible set $\mathcal{U}$ based on all the constraints appearing in (1) and the global initial state $x(0)$, which is assumed to be available. Consider an inner-hyperbox approximation Ω of $\mathcal{U}$, which then takes the form of a Cartesian product

$$\Omega = \Omega_1 \times \cdots \Omega_m \subset \mathcal{U}. \tag{19}$$

This approximation essentially decomposes and decouples the constraints among subsystems by performing constraint tightening. Each subsystem l will thus have to include Ω_l in their local problem setup. Since the Cartesian product of these local constraint sets are included in the globally feasible set $\mathcal{U}$, any combination of local solutions within Ω_l will be globally feasible as well. The local constraint sets that arise from this inner-hyperbox approximation will be in general quite conservative, but at the same time will allow the construction of a feasible solution locally to initialize **Algorithm 1**.

Calculation of the inner-hyperbox approximation can be performed a priori and the local Ω_l constraints distributed to each subsystem. A polynomial-time procedure to compute a maximum volume inner box of could follow the procedure described in (Bemporad et al., 2004). Obtaining the local component-wise constraints Ω_l is then straightforward. For time steps $k > 0$, we construct a feasible solution by performing **Step 1** of **Algorithm 1**

$$U^0_{j \in \mathcal{N}_l}(k) = \begin{bmatrix} u_{j \in \mathcal{N}_l}(k, k-1) & \cdots & u_{j \in \mathcal{N}_l}(k+M, k-1) & 0 \end{bmatrix}$$

The feasibility throughout the iterations is maintained because in **step 2**.*a* m feasible solutions $\tilde{U}^q_{j\in\mathcal{N}_l}(k)$ are obtained. Then, in **step 2**.*c*, the new control profile $U^q_{j\in\mathcal{N}_l}(k)$ is built as a convex combination of these solutions. Since problem (7) is a convex constrained QP, any convex combination of $U^q_{j\in\mathcal{N}_l}(k)$ also satisfies the convex constraint set. Therefore $U^q(k)$ is a feasible solution of optimisation problem (7) for all l.

3.4 Stability

Showing nominal stability of the resulting closed-loop system follows standard arguments for the most part (Mayne et al., 2000). The proof in this section is closely related to the stability proof of the FC-MPC method in (Venkat et al., 2008) with the addition of Assumption 2. The key idea is to show the contractivity of the sequence of global cost functions J generated by **Algorithm 1** along the system operation and the stability of the origin.

Theorem 1. *Let us assume that the system has been partitioned following a decentralised design procedure and the optimisation problem (7) solved using Algorithm 1 is feasible, then the origin is an exponentially stable equilibrium point.*

Proof. See appendix 8.B. □

4. System behavior under communication failures

In the proposed framework agents coordinate their actions by exchanging information through the communication network. Since the agents extensively use the communication network some questions related to the system behavior arise if communications fail: Which are the conditions for the convergence of the iterative process? How closed-loop stability is affected? How does system performance change?

In a first stage, the failures in the communication system are modeled introducing three matrices: *i*) the *connection matrix* $\mathcal{C}$ which represents the communication structure, *ii*) the *transmission failure matrix* $\mathcal{T}$ which models the transmission failures and *iii*) *reception failure matrices* $\mathcal{R}$ that models the reception failures in the system. The matrix $\mathcal{C}$ is defined as

$$\mathcal{C} = \left[c_{lp}\right],\ c_{lp} = \begin{cases} 0 & l = p, \\ 1 \text{ or } 0 & l \neq p, \end{cases} \tag{20}$$

where $c_{lp} = 1$ indicates the connection between agents l and p, while $c_{lp} = 0$ shows no connection between these agents. Then, the failures in the communication system can be modeled combining the connection matrix with the others matrices that models the reception ($\mathcal{R}$) and transmission ($\mathcal{T}$) failures, $\mathcal{RCT}$, which are given by

$$\mathcal{R} = \left[r_{lp}\right],\ r_{lp} = \begin{cases} 1 & l = p, \\ 0 & l \neq p, \end{cases} \tag{21a}$$

$$\mathcal{T} = \left[t_{lp}\right],\ t_{lp} = \begin{cases} 1 & l = p, \\ 0 & l \neq p. \end{cases} \tag{21b}$$

An element $t_{ll} = 1$ ($r_{ll} = 1$) corresponds to a perfect transmission (reception) of agent l, while $t_{ll} = 0$ ($r_{ll} = 0$) corresponds to a transmission (reception) failure of agent l. A failure between agents l and p is represented with the transition from $1 \to 0$ of the corresponding elements of $\mathcal{R}$ and $\mathcal{T}$.

Following the same procedure like in Section 3.2, the solution for the distributed problem at each iteration is

$$U^q(k) = \mathcal{K}_0\mathcal{RCT}U^{q-1}(k) + \mathcal{K}_1\mathcal{RCT}x(k) \quad q \geq 1$$

Its behavior is related with its stability, which is given by

$$\|\lambda\left(\mathcal{K}_0\mathcal{RCT}\right)\|_1 < 1 \tag{22}$$

Under communication failure each agent cannot exchange information properly, which will modify the iterative process driving it to another solution. In this case, the agent with communication failure will become a decentralised controller that will receive information about the decision of the others agents through the system. This will deteriorate the stability margins and performance due to the presence of the interactions not accounted during the iterative process, which will act like non measurable disturbances. In the extreme case $\mathcal{K}_0\mathcal{RCT} = 0$, the control structure will correspond to the *fully decentralised control architecture*, and the stability will depend on the way that the system was partitioned. If the controllers are designed following a decentralised design procedure (Wittenmark & Salgado, 2002), the stability of the closed-loop system can be guaranteed.
Once the convergence of the iterates can be guaranteed, the next issue to be addressed is the effect of the communication failures on the closed-loop stability. In order to establish their on the closed-loop behavior, the control action

$$\tilde{U}(k) = \left(I - \mathcal{K}_0\mathcal{RCT}\right)^{-1}\mathcal{K}_1\mathcal{RCT}\Gamma x(k)$$

is replaced in the open-loop model of the system, leading to the closed-loop system

$$x(k+1) = \left(A - B\mathcal{I}\left(I - \mathcal{K}_0\mathcal{RCT}\right)^{-1}\mathcal{K}_1\mathcal{RCT}\Gamma\right)x(k).$$

Then, the stability of the closed–loop system under communication failures is determined by

$$\left|\lambda\left(A - B\mathcal{I}\left(I - \mathcal{K}_0\mathcal{RCT}\right)^{-1}\mathcal{K}_1\mathcal{RCT}\Gamma\right)\right| < 1. \tag{23}$$

Under the communication failure, each agent can not exchange information properly therefore the stability of the closed–loop system will depend on the dynamic characteristics of the interactions between subsystems. In the extreme case $\mathcal{RCT} = 0$, the stability condition is always satisfied corresponding to the full decentralised architecture. The interactions act like non measurable disturbances for the controllers, reducing the stability margins and degrading the system performance.

Theorem 2. *Let us assume that the system has been partitioned in a way that the* convergence condition *(15) is satisfied, its performance at time instant k under the local communication failure is*

$$\tilde{J}(k) \leq \left(1 + \frac{\|\mathcal{W}(k)\|}{\lambda_{min}(\mathcal{F})}\right)J^*, \tag{24}$$

where the performance degradation is bounded by

$$\frac{\tilde{J}(k) - J^*}{J^*} \leq \frac{\|\mathcal{W}_{max}\|}{\lambda_{min}(\mathcal{F})} \tag{25}$$

where $\lambda_{min}\left(\mathcal{F}\right)$ *denotes the minimal eigenvalue of*

$$\mathcal{F} = \left(\mathcal{K}_1^{-1}\left(I-\mathcal{K}_0\right)-\mathcal{H}\right)^T \mathcal{Q}\left(\mathcal{K}_1^{-1}\left(I-\mathcal{K}_0\right)-\mathcal{H}\right)+R,$$
$$\mathcal{W}_{\max} = \mathcal{S}_{\max}^T\left(\mathcal{H}^T\mathcal{Q}\mathcal{H}+R\right)\mathcal{S}_{\max},$$
$$\mathcal{S}_{\max} = 2I-\left[I+\left(I-\mathcal{K}_0\right)^{-1}\left(I+\mathcal{K}_0\right)\right]^{-1}.$$

Proof. See appendix 8.C. □

5. Simulations

In this Section, we will illustrate the applicability and limitations of the theoretical results presented in this paper through two problems: *i*) a *LTI MIMO* system and *ii*) the operation of a heat–exchanger network (HEN). In the first case we analyse and evaluate the ideas discussed in previous sections through the control of a strongly coupled MIMO LTI system. In the second problem we will evaluate the applicability and limitations of the proposed framework to system with complex dynamic.

5.1 LTI System

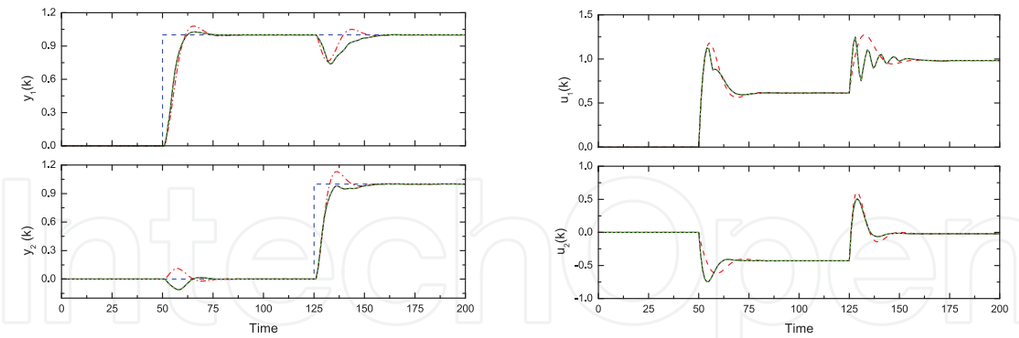
To explore the ideas discussed in previous Sections, let's consider the following *MIMO* linear system

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{1}{10s+1} & \frac{-1}{7.5s+1} \\ \frac{1.5}{11s+1} & \frac{1.5}{8s+1} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \tag{26}$$

This system shows a strong interaction with a non cooperative behavior between both subsystem due to the difference in the sign of the gain. Besides, the interaction between y_1 and u_2 is faster than the dynamic between y_1 and u_1. The models for the distributed and coordinated decentralized *MPC* algorithms were obtained by dividing the system in two agents

$$\textbf{Agent 1}: y_1(s) = \frac{1}{10s+1}u_1(s) - \frac{1}{7.5s+1}u_2(s), \tag{27a}$$
$$\textbf{Agent 2}: y_2(s) = \frac{1.5}{11s+1}u_1(s) + \frac{1.5}{8s+1}u_2(s). \tag{27b}$$

Agent 1 solves the optimization problem using u_1 as decision variable, while **agent 2** solves its optimization problem using u_2. The parameters of the predictive control algorithms are $M=5$, $V=20$, $Q_i = I_{2\times 2}$, $R_i = 2I_{2\times 2}$ $\quad i=0,\cdots,M-1$, and the stopping condition for the decentralized and distributed *MPC* algorithms was fixed to $\varepsilon_1=\varepsilon_2=0.005$ and $q_{max}=30$. Figure 1 shows the closed–loop responses for different *MPC* schemes. In this figure we can see that the performance of *centralized* and *distributed MPC* are similar, while the performance of the *coordinated decentralized MPC* is worst than the others control schemes. In general, the response obtained by the *coordinated decentralized MPC* shows stronger interactions that deteriorate the overall system performance. This phenomenon is due to the fact that the *coordinated decentralized MPC* does not optimize the effect of the agents decision variable on the performance of the other agent. Figure 1.*b* shows the resulting manipulated variables where we can see the behavior of the three MPC schemes. Even though all algorithms achieve



(a) System output (b) Manipulated variables

Fig. 1. System responses to different *MPC* schemes (- - Centralized MPC, — Distributed MPC and -.- Coordinated decentralized MPC).

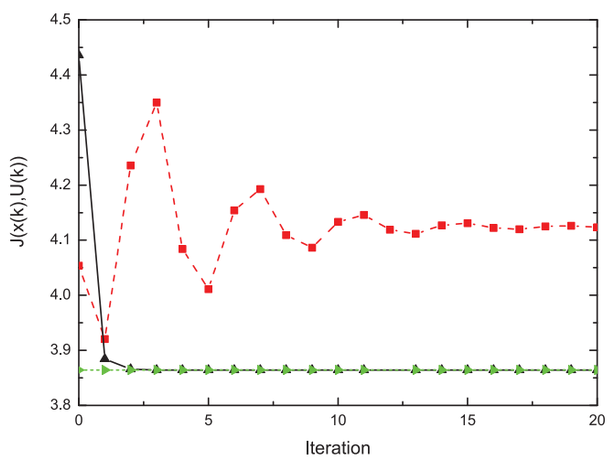


Fig. 2. Behavior of the controllers' cost functions during the iterative procedure (- - Centralized MPC, — Distributed MPC and -.- Coordinated decentralized MPC).

the same steady–state conditions, the inputs / outputs trajectories followed by the *coordinated decentralized MPC* are different from the *centralized* and *distributed MPC*.

Figure 2 shows the behavior of *MPC* controllers cost functions during the iterative procedure for the first set point change. The first thing to see is the oscillatory behavior of the iterative process of the *coordinated decentralized MPC*, in contrast with the monotonous behavior of the *distributed MPC*. This behavior is due to the nature interactions and the characteristics Balderud et al. (2008). The cost function of the *distributed MPC* converges to the solution of the *centralized MPC*, which is globally optimal, in few iterations. If the iterative process is stopped before ($q < 4$), the resulting solution will be suboptimal however it will lead to a stable closed–loop system. The earlier the iterative process is stopped, the bigger the difference to the centralized solution.

5.2 Heat-exchanger network

The heat-exchanger network (*HEN*) system studied here is represented schematically in Figure 3. It is a system with only three recovery exchangers (I_1, I_2 and I_3) and three service (S_1, S_2 and S_3) units. Two hot process streams (h_1 and h_2) and two cold process streams (c_1 and c_2) take part of the heat exchange process. There are also three utility streams (s_1, s_2 and s_3) that can be used to help reaching the desired outlet temperatures.

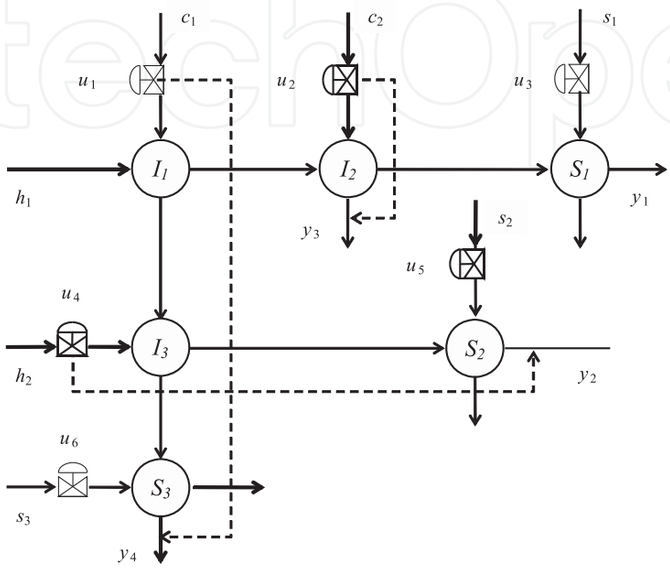


Fig. 3. Schematic representation of the HEN system.

The main purpose of a *HEN* is to recover as much energy as necessary to achieve the system requirements from high–temperature process streams (h_1 and h_2) and to transfer this energy to cold–process streams (c_1 and c_2). The benefits are savings in fuels needed to produce utility streams s_1, s_2 and s_3. However, the *HEN* has to also provide the proper thermal conditioning of some of the process streams involved in the heat transfer network. This means that a control system must *i*) drive the exit process–stream temperatures (y_1, y_2, y_3 and y_4) to the desired values in presence of external disturbances and input constraints while *ii*) minimizes the amount of utility energy.

The usual manipulated variables of a *HEN* are the flow rates at bypasses around heat exchangers (u_1, u_2 and u_4) and the flow rates of utility streams in service units (u_3, u_5 and u_6), which are constrained

$$0 \leq u_j(k) \leq 1.0 \qquad j = 1, \ldots, 6.$$

A fraction $0 < u_j < 1$ of bypass j means a fraction u_j of corresponding stream goes through the bypass and a fraction $1 - u_j$ goes through the exchangers, exchanging energy with other streams. If $u_j = 0$ the bypass is *completely closed* and the whole stream goes through the exchangers, maximizing the energy recovery. On the other hand, a value of $u_j = 1$ the bypass is *completely open* and the whole stream goes through the bypass, minimizing the energy recovery.

The *HEN* studied in this work has more control inputs than outlet temperatures to be controlled and so, the set of input values satisfying the output targets is not unique. The

possible operation points may result in different levels of heat integration and utilities consumption. Under nominal conditions only one utility stream is required (s_1 or s_3) for the operation of the *HEN*, the others are used to expand the operational region of the *HEN*.
The inclusion of the control system provides new ways to use the extra utility services (s_2 and s_3) to achieve control objectives by introducing new interactions that allow the redirection of the energy through the *HEN* by manipulating the flow rates. For example, any change in the utility stream s_3 (u_6) has a direct effect on output temperature of c_1 (y_4), however the control system will redirect this change (through the modification of u_1) to the output temperature of h_1 (y_1), h_2 (y_2), and c_2 (y_3). In this way, the *HEN* has energy recycles that induces feedback interaction, whose strength depends on the operational conditions, and leads to a complex dynamic: *i*) small energy recycles induce weak couplings among subsystems, whereas *ii*) large energy recycles induce a time scale separation, with the dynamics of individual subsystems evolving in a fast time scale with weak interactions, and the dynamics of the overall system evolving in a slow time scale with strong interactions Kumar & Daoutidis (2002).
A complete definition of this problem can be found in Aguilera & Marchetti (1998). The controllers were developed using the following linear model

$$Y = A(s) * U,$$

where

$$A(s) = \begin{bmatrix} \frac{20.6\ e^{-61.3s}}{38.8s+1} & \frac{19.9\ e^{-28.9s}}{25.4s+1} & \frac{17.3\ e^{-4.8s}}{23.8s+1} & 0 & 0 & 0 \\ \frac{4.6\ e^{-50.4s}}{48.4s+1} & 0 & 0 & 79.1\frac{31.4s+0.8}{31.4s+1.0} & \frac{20.1\ e^{-4.1s}}{25.6s+1.0} & 0 \\ \frac{16.9\ e^{-24.7s}}{39.5s+1} & -39.2\frac{22.8s+0.8}{22.8s+1.0} & 0 & 0 & 0 & 0 \\ 24.4\frac{48.2s^2+4.0s+0.05}{48.2s^2+3.9s+0.06} & 0 & 0 & -8.4\frac{e^{-18.8s}}{27.9s+1} & 0 & \frac{16.3\ e^{-3.5s}}{20.1s+1.0} \end{bmatrix}$$

and

$$U = [\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6\ ]^T,$$
$$Y = [\ y_1\ y_2\ y_3\ y_4\ ]^T.$$

The first issue that we need to address in the development of the distributed controllers is selection of the input and output variables associated to each agent. The decomposition was carried after consideration of the multi-loop rules (Wittenmark & Salgado, 2002). The resulting decomposition is given in Table 1: **Agent 1** corresponds to the first and third rows of $A(s)$, while **agents 2** and **3** correspond to the second and fourth rows of $A(s)$ respectively. Agents 1 and 2 will mainly interact between them through the process stream c_1.
For a *HEN* not only the dynamic performance of the control system is important but also the cost associated with the resulting operating condition must be taken into account. Thus, the performance index (3) is augmented by including an economic term J_U, such that the global cost is given by $J + J_U$, defined as follows

$$J_U = u_{SS}^T R_U u_{SS}. \tag{28}$$

where $u_{SS} = [u_3(k+M,k)\ u_5(k+M,k)\ u_6(k+M,k)]$ for the *centralized MPC*. In the case of the *distributed* and *coordinated decentralized MPC*, u_{SS} is decomposed among the agents of the control schemes ($u_{SS} = u_3(k+M,k)$ for Agent 1, $u_{SS} = u_5(k+M,k)$ for Agent 2 and $u_{SS} = u_6(k+M,k)$ for Agent 3). Finally, the tuning parameters of the *MPC* controllers are: $t_s =$

0.2 min; $V_l = 50$; $M_l = 5$; $\varepsilon_l = 0.01$; $q_{max} = 10 \quad l = 1, 2, 3$, the cost functions matrices are given in Table 2.

MATLAB based simulation results are carried out to evaluate the proposed *MPC* algorithms (*coordinated decentralized* and *distributed MPC*) through performance comparison with a *centralized* and *decentralized MPC*. The *MPC* algorithms used the same routines during the simulations, which were run in a computer with an Intel Quad-core Q9300 CPU under Linux operating system. One of the processors was used to execute the *HEN* simulator, while the others were used to execute the *MPC* controllers. Only one processor was used to run the *centralized MPC* controller. In the case of the distributed algorithms, the controllers were distributed among the other processors. These configurations were adopted in order to make a fair comparison of the computational time employed for each controller.

We consider the responses obtained for disturbance rejection. A sequence of changes is introduced into the system: after stabilizing at nominal conditions, the inlet temperature of h_1 ($T^{in}_{h_1}$) changes from 90°C to 80°C; 10 min later the inlet temperature of h_2 ($T^{in}_{h_2}$) goes from 130°C to 140°C and after another 10 min the inlet temperature of c_1 ($T^{in}_{c_1}$) changes from 30°C to 40°C.

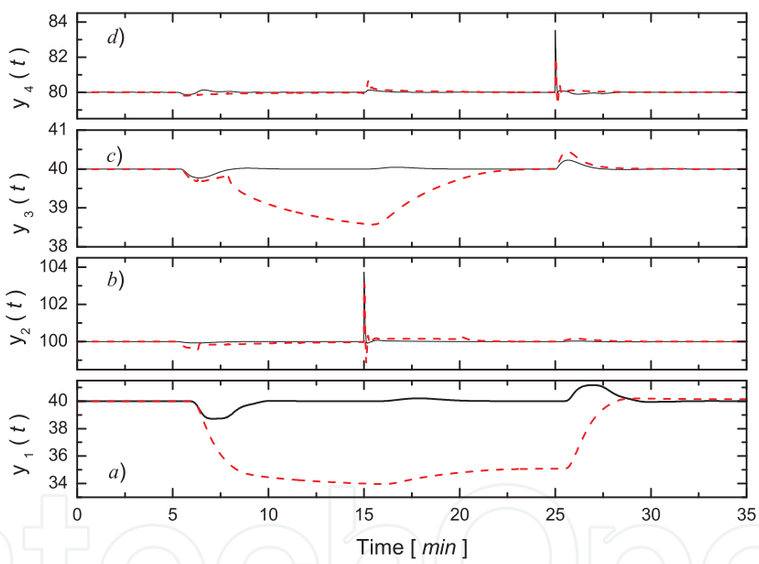
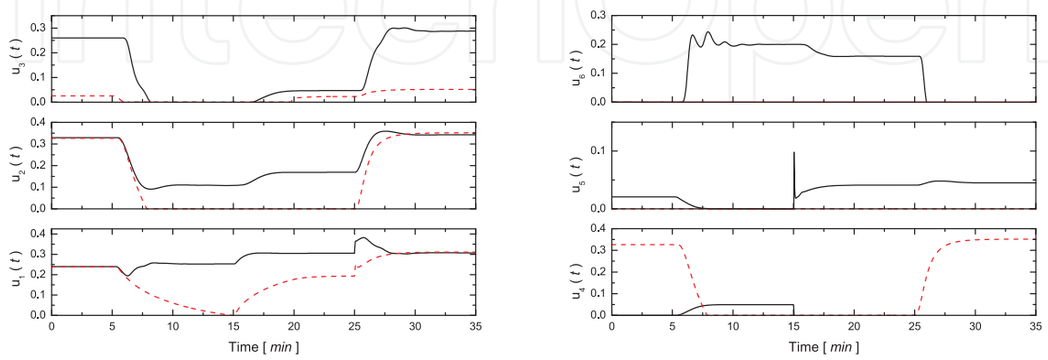


Fig. 4. Controlled outputs of the HEN system using (—) distributed MPC and (-.-) coordinated decentralized MPC.

Figures 4 and 5 show the dynamic responses of the *HEN* operating with a *distributed MPC* and a *coordinated decentralized MPC*. The worse performance is observed during the first and second load changes, most notably on y_1 and y_3. The reasons for this behavior can be found by observing the manipulated variables. The first fact to be noted is that under nominal steady-state conditions, u_4 is completely closed and y_2 is controlled by u_5 (see Figures 5.*b*), achieving the maximum energy recovery. Observe also that u_6 is inactive since no heating service is necessary at this point. After the first load change occurs, both control variables u_2 and u_3 fall rapidly (see Figures 5.*a*). Under this conditions, the system activates the heater flow rate u_6 (see Figures 5.*b*). The dynamic reaction of the heater to the cool disturbance is

also stimulated by u_2, while u_6 takes complete control of y_1, achieving the maximum energy recovery. After the initial effect is compensated, y_3 is controlled through u_2 –which never saturates–, while u_6 takes complete control of y_1. Furthermore, Figure 5.*b* show that the cool perturbation also affects y_2, where u_5 is effectively taken out of operation by u_4. The ensuing pair of load changes are heat perturbations featuring manipulated movements in the opposite sense to those indicated above. Though the input change in h_2 allows returning the control of y_1 from u_6 to u_3 (see Figures 5.*a*).



(a) $u_1(t), u_2(t), u_3(t)$ (b) $u_4(t), u_5(t), u_6(t)$

Fig. 5. Manipulated inputs of the HEN system using (—) distributed MPC and (-.-) coordinated decentralized MPC.

In these figures we can also see that the *coordinated decentralized MPC* fails to reject the first and second disturbances on y_1 and y_3 (see Figures 4.*a* and *c*) because it is not able to properly coordinate the use of utility service u_6 to compensate the effects of active constraints on u_2 and u_3. This happens because the *coordinated decentralized MPC* is only able to address the effect of interactions between agents but it can not coordinate the use of utility streams s_2 and s_3 to avoid the *output-unreachability under input constraint* problem. The origin of the problem lies in the cost function employed by the *coordinated decentralized MPC*, which does not include the effect of the local decision variables on the other agents. This fact leads to different steady–state values in the manipulated variables to those ones obtained by the *distributed MPC* along the simulation.

Figure 6 shows the steady–state value of the recovered energy and utility services used by the system for the distributed *MPC* schemes. As mentioned earlier, the *centralized* and *distributed MPC* algorithms have similar steady–state conditions. These solutions are *Pareto optimal*, hence they achieve the best plant wide performance for the combined performance index. On the other hand, the *coordinated decentralized MPC* exhibited a good performance in energy terms, since it employs less service energy, however it is not able of achieving the control objectives, because it is not able of properly coordinate the use of utility flows u_5 and u_6. As it was pointed out in previous Sections, the fact that the agents achieve the Nash equilibrium does not implies the optimality of the solution.

Figure 7 shows the CPU time employed for each *MPC* algorithm during the simulations. As it was expected, the *centralized MPC* is the algorithm that used more intensively the CPU. Its CPU time is always larger than the others along the simulation. This fact is originated on the size of the optimization problem and the dynamic of the system, which forces the

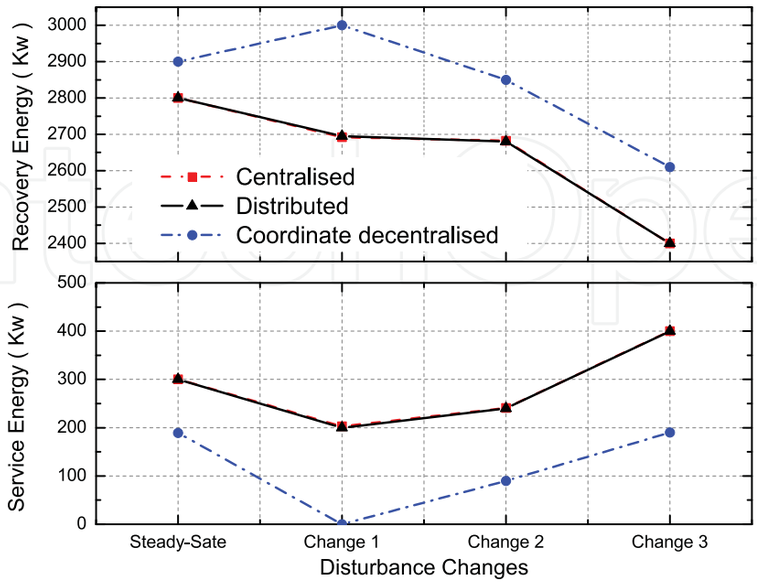


Fig. 6. Steady-state conditions achieved by the HEN system for different MPC schemes.

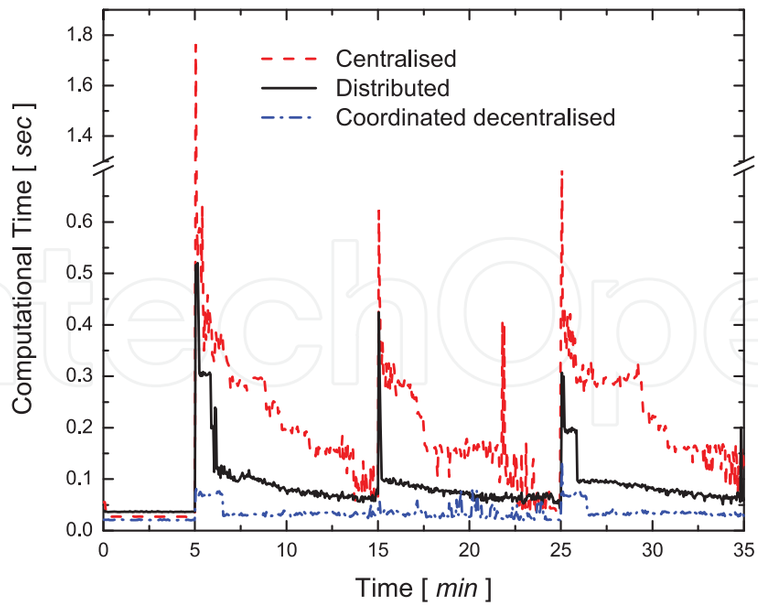


Fig. 7. CPU times for different MPC schemes.

centralized MPC to permanently correct the manipulated variable along the simulation due to the system interactions. On the other hand, the *coordinated decentralized MPC* used the CPU less intensively than the others algorithms, because of the size of the optimization problem. However, its CPU time remains almost constant during the entire simulation since it needs to compensate the interactions that had not been taken into account during the computation. In general, all algorithms show larger CPU times after the load changes because of the recalculation of the control law. However, we have to point out that the value of these peak are smaller than sampling time.

6. Conclusions

In this work a distributed model predictive control framework based on dynamic games is presented. The *MPC* is implemented in distributed way with the inexpensive agents within the network environment. These agents can cooperate and communicate each other to achieve the objective of the whole system. Coupling effects among the agents are taken into account in this scheme, which is superior to other traditional decentralized control methods. The main advantage of this scheme is that the on-line optimization can be converted to that of several small-scale systems, thus can significantly reduce the computational complexity while keeping satisfactory performance. Furthermore, the design parameters for each agent such as prediction horizon, control horizon, weighting matrix and sample time, etc. can all be designed and tuned separately, which provides more flexibility for the analysis and applications. The second part of this study is to investigate the convergence, stability, feasibility and performance of the distributed control scheme. These will provide users better understanding to the developed algorithm and sensible guidance in applications.

7. Acknowledgements

The authors wishes to thank: the *Agencia Nacional de Promoción Científica y Tecnológica*, the *Universidad Nacional de Litoral* and the *Consejo Nacional de Investigaciones Científicas y Técnicas* (CONICET) from Argentina, for their support.

8. Appendices

A. Proof of Lemma 1

Proof. From definition of the $J(\cdot)$ we have

$$J\left(x(k), U^q(k), A\right) = J\left(x(k), \left[U^q_{j\in\mathcal{N}_1}(k)\cdots U^q_{j\in\mathcal{N}_m}(k)\right], A\right) \tag{29}$$

From definition of $U^q_{j\in\mathcal{N}_l}$.we have

$$\begin{aligned} J\left(x(k), U^q(k), A\right) =& J\left(x(k), \left[\alpha_{j\in\mathcal{N}_1}\tilde{U}^q_1(k) + \left(1-\alpha_{j\in\mathcal{N}_1}\right)U^{q-1}_{j\in\mathcal{N}_1}(k)\cdots\right.\right. \\ & \left.\left.\alpha_{j\in\mathcal{N}_m}\tilde{U}^q_{j\in\mathcal{N}_m}(k) + \left(1-\alpha_{j\in\mathcal{N}_m}\right)U^{q-1}_{j\in\mathcal{N}_m}(k)\right], A\right) \\ =& J\left(x(k), \left[\alpha_{j\in\mathcal{N}_1}\left[\tilde{U}^q_{j\in\mathcal{N}_1}(k)\cdots U^{q-1}_{j\in\mathcal{N}_m}(k)\right]\cdots\right.\right. \\ & \left.\left.\alpha_{j\in\mathcal{N}_m}\left[U^{q-1}_{j\in\mathcal{N}_1}(k)\cdots\tilde{U}^q_{j\in\mathcal{N}_m}(k)\right]\right], A\right) \end{aligned}$$

By convexity of $J(\cdot)$ we have

$$J\left(x(k), U^q(k), A\right) \leq \sum_{l=1}^{m} \alpha_l J\left(x(k), \tilde{U}^q_{j\in\mathcal{N}_l}(k), U^{q-1}_{j\in\mathcal{I}-\mathcal{N}_l}(k), A\right) \tag{30}$$

and from Algorithm 1 we know that

$$J\left(x(k), \tilde{U}^q_{j\in\mathcal{N}_l}(k), U^{q-1}_{j\in\mathcal{I}-\mathcal{N}_l}(k), A\right) \leq J\left(x(k), U^{q-1}(k), A\right),$$

then

$$J\left(x(k), U^q(k), A\right) \leq J\left(x(k), \tilde{U}^q_{j\in\mathcal{N}_l}(k), U^{q-1}_{j\in\mathcal{I}-\mathcal{N}_l}(k), A\right) \leq J\left(x(k), U^{q-1}(k), A\right). \tag{31}$$

Subtracting the cost functions at $q-1$ and q we obtain

$$\Delta J\left(x(k), U^{q-1}(k), A\right) \leq -\Delta U^{q-1}_{j\in\mathcal{N}_l}(k)^T R\, \Delta U^{q-1}_{j\in\mathcal{N}_l}(k).$$

This shows that the sequence of cost $\left\{J^q_l(k)\right\}$ is non-increasing and the cost is bounded below by zero and thus has a non-negative limit. Therefore as $q \to \infty$ the difference of cost $\Delta J^q(k) \to 0$ such that the $J^q(k) \to J^*(k)$. Because $R > 0$, as $\Delta J^q(k) \to 0$ the updates of the inputs $\Delta U^{q-1}(k) \to 0$ as $q \to \infty$, and the solution of the optimisation problem $U^q(k)$ converges to a solution $\bar{U}(k)$. Depending on the cost function employed by the distributed controllers, $\bar{U}(k)$ can converge to $U^*(k)$ (see Section 3.1). □

B. Proof of Theorem 1

Proof. First it is shown that the input and the true plant state converge to the origin, and then it will be shown that the origin is an stable equilibrium point for the closed-loop system. The combination of convergence and stability gives asymptotic stability.

Convergence. Convergence of the state and input to the origin can be established by showing that the sequence of cost values is non-increasing.

Showing stability of the closed-loop system follows standard arguments for the most part Mayne et al. (2000), Primbs & Nevistic (2000). In the following, we describe only the most important part for brevity, which considers the nonincreasing property of the value function. The proof in this section is closely related to the stability proof of the *FC-MPC* method in Venkat et al. (2008).

Let $q(k)$ and $q(k+1)$ stand for iteration number of **Algorithm 1** at time k and $k+1$ respectively. Let $J(k) = J\left(x(k), U(k), A\right)$ and $J(k+1) = J\left(x(k+1), U(k+1), A\right)$ denote the cost value associated with the final combined solution at time k and $k+1$. At time $k+1$, let $J_l(k+1) = J\left(x(k+1), U^q_{j\in\mathcal{N}_l}(k), U^{q-1}_{j\in\mathcal{I}-\mathcal{N}_l}(k), A\right)$ denote the global cost associated with solution of subsystem l at iterate q.

The global cost function $J\left(x(k), U(k)\right)$ can be used as a Lyapunov function of the system, and its non-increasing property can be shown following the chain

$$J\left(x(k+1), U(k+1), A\right) \leq \cdots \leq J\left(x(k+1), U^q(k+1), A\right) \leq \cdots$$
$$\cdots \leq J\left(x(k+1), U^1(k+1), A\right) \leq J\left(x(k), U(k), A\right) - x(k)^T Q x(k) - u(k)^T R u(k)$$

The inequality $J\left(x(k+1), U^q(k+1), A\right) \leq J\left(x(k+1), U^{q-1+}(k+1), A\right)$ is consequence of Lemma 1. Using this inequality we can trace back to $q = 1$

$$J\left(x(k+1), U(k+1), A\right) \leq \cdots \leq J\left(x(k+1), U^q(k+1), A\right) \leq \cdots$$
$$\cdots \leq J\left(x(k+1), U^1(k+1), A\right).$$

At time step $q = 1$, we can recall the initial feasible solution $U^0(k+1)$. At this iteration, the distributed MPC optimizes the cost function with respect the local variables starting from $U^0(k+1)$, therefore $\forall l = 1, \ldots, m$

$$\begin{aligned} J\left(x(k+1), U^1_{j\in\mathcal{N}_l}(k), U^0_{j\in\mathcal{I}-\mathcal{N}_l}(k), A\right) &\leq J\left(x(k+1), U^0(k), A\right) \\ &\leq \sum_{i=1}^{\infty} x(k+i,k^T Q x(k+i,k) + u(k+i,k)^T R u(k+i,k) \\ &\leq J\left(x(k), U(k), A\right) - x(k)^T Q x(k) - u(k)^T R u(k) \end{aligned}$$

Due to the convexity of J and the convex combination up date (Step 2.*c* of Algorithm 1), we obtain

$$J\left(x(k), U^1(k), A\right) \leq \sum_{l=1}^{m} \alpha_l J\left(x(k+1), U^1_{j\in\mathcal{N}_l}(k), U^0_{j\in\mathcal{I}-\mathcal{N}_l}(k), A\right) \tag{32}$$

then,

$$\begin{aligned} J\left(x(k), U^1(k), A\right) &\leq \sum_{l=1}^{m} \alpha_l \left[J\left(x(k), U(k), A\right) - x(k)^T Q x(k) - u(k)^T R u(k)\right], \\ &\leq J\left(x(k), U(k), A\right) - x(k)^T Q x(k) - u(k)^T R u(k). \end{aligned}$$

Subtracting $J^*(k)$ from $J^*(k+1)$

$$J^*(k+1) - J^*(k) \leq -x(k)^T Q x(k) - u(k)^T R u(k) \quad \forall k. \tag{33}$$

This shows that the sequence of optimal cost values $\{J^*(k)\}$ decreases along closed-loop trajectories of the system. The cost is bounded below by zero and thus has a non–negative limit. Therefore as $k \to \infty$ the difference of optimal cost $\Delta J^*(k+1) \to 0$. Because Q and R are positive definite, as $\Delta J^*(k+1) \to 0$ the states and the inputs must converge to the origin $x(k) \to 0$ and $u(k) \to 0$ as $k \to \infty$.

Stability. Using the QP form of (6), the feasible cost at time $k = 0$ can be written as follows $J(0) = x(0)^T \bar{Q} x(0)$, where $\bar{Q}$ is the solution of the Lyapunov function for dynamic matrix $\bar{Q} = A^T Q A + Q$.

From equation (33) it is clear that the sequence of optimal costs $\{J^*(k)\}$ is non-increasing, which implies $J^*(k) \leq J^*(0) \quad \forall k > 0$. From the definition of the cost function it follows that $x^T(k) Q x(k) \leq J^*(k) \quad \forall k$, which implies

$$x^T(k) Q x(k) \leq x(0)^T \bar{Q} x(0) \quad \forall k.$$

Since Q and $\bar{Q}$ are positive definite it follows that

$$\|x(k)\| \leq \gamma \|x(0)\| \quad \forall k > 0$$

where

$$\gamma = \sqrt{\frac{\lambda_{\max}(\bar{Q})}{\lambda_{\min}(Q)}}.$$

Thus, the closed-loop is stable. The combination of convergence and stability implies that the origin is asymptotically stable equilibrium point of the closed-loop system. □

C. Proof of Theorem 2

Proof. The optimal solution of the distributed control system with communications faults is given by

$$\tilde{U}(k) = (I - \mathcal{K}_0\mathcal{R}\mathcal{C}\mathcal{T})^{-1}\mathcal{K}_1\mathcal{R}\mathcal{C}\mathcal{T}\Gamma x(k). \tag{34}$$

Using the matrix decomposition technique, it gives

$$(I - \mathcal{K}_0\mathcal{R}\mathcal{C}\mathcal{T})^{-1} = (I - \mathcal{K}_0)^{-1}\Big(2I - \Big[I + (I - \mathcal{K}_0)^{-1} (I + \mathcal{K}_0 - 2\mathcal{K}_0\mathcal{R}\mathcal{C}\mathcal{T})]^{-1}\Big) + (I - \mathcal{K}_0)^{-1}$$

In general $(I - \mathcal{K}_0)^{-1}$ and $(I + \mathcal{K}_0 - 2\mathcal{K}_0\mathcal{R}\mathcal{C}\mathcal{T})^{-1}$ all exist, therefore the above equation holds. Now, from (34) we have $\mathcal{K}_1\Gamma x(k) = (I - \mathcal{K}_0)\,U(k)$, then $\tilde{U}(k)$ can be written as a function of the optimal solution $U(k)$ as follows

$$\tilde{U}(k) = (\mathcal{S} + I)\,U(k)$$

where $\mathcal{S} = 2I - \left[I + (I - \mathcal{K}_0)^{-1}(I + \mathcal{K}_0 - 2\mathcal{K}_0\mathcal{R}\mathcal{C}\mathcal{T})\right]^{-1}$.
The cost function of the system free of communication faults J^* can be written as function of $U(k)$ as follows

$$J^* = \left\|\mathcal{K}_1^{-1}(I - \mathcal{K}_0)\,U(k) - \mathcal{H}U(k)\right\|_Q^2 + \|U(k)\|_R^2 = \|U(k)\|_{\mathcal{F}}^2 \tag{35}$$

where

$$\mathcal{F} = \left(\mathcal{K}_1^{-1}(I - \mathcal{K}_0) - \mathcal{H}\right)^T Q\left(\mathcal{K}_1^{-1}(I - \mathcal{K}_0) - \mathcal{H}\right) + R.$$

In the case of the system with communication failures we have

$$\tilde{J} \leq J^* + \|U(k)\|_{\mathcal{W}}^2 \tag{36}$$

where $\mathcal{W} = \mathcal{S}^T\left(\mathcal{H}^T Q\mathcal{H} + R\right)\mathcal{S}$. Finally, the effect of communication can be related with J^* through

$$\|U(k)\|_{\mathcal{W}}^2 \leq \frac{\|\mathcal{W}\|}{\lambda_{min}(\mathcal{F})}J^*, \tag{37}$$

where λ_{min} denotes the minimal eigenvalue of $\mathcal{F}$. From the above derivations, the relationship between $\tilde{J}$ and J^* is given by

$$\tilde{J} \leq \left(1 + \frac{\|\mathcal{W}\|}{\lambda_{min}(\mathcal{F})}\right)J^*. \tag{38}$$

and the degradation is

$$\frac{\tilde{J} - J^*}{J^*} \leq \frac{\|\mathcal{W}\|}{\lambda_{min}(\mathcal{F})}. \tag{39}$$

Inspection of (36) shows that $\|\mathcal{W}\|$ depends on $\mathcal{R}$ and $\mathcal{T}$. So in case of all communication failures existed, $\|\mathcal{W}\|$ can arrive at the maximal value

$$\mathcal{W}_{\max} = \left(2I - \left[I + (I - \mathcal{K}_0)^{-1}(I + \mathcal{K}_0)\right]^{-1}\right)^T \left(\mathcal{H}^T Q \mathcal{H} + R\right) \left(2I - \left[I + (I - \mathcal{K}_0)^{-1}(I + \mathcal{K}_0)\right]^{-1}\right),$$

and the upper bound of performance deviation is

$$\frac{\tilde{J} - J^*}{J^*} \leq \frac{\|\mathcal{W}_{\max}\|}{\lambda_{min}(\mathcal{F})}. \tag{40}$$

□

9. References

Aguilera, N. & Marchetti, J. (1998). Optimizing and controlling the operation of heat exchanger networks, *AIChe Journal* 44(5): 1090–1104.

Aske, E., Strand, S. & Skogestad, S. (2008). Coordinator mpc for maximizing plant throughput, *Computers and Chemical Engineering* 32(1-2): 195–204.

Bade, S., Haeringer, G. & Renou, L. (2007). More strategies, more nash equilibria, *Journal of Economic Theory* 135(1): 551–557.

Balderud, J., Giovanini, L. & Katebi, R. (2008). Distributed control of underwater vehicles, *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 222(2): 95–107.

Bemporad, A., Filippi, C. & Torrisi, F. (2004). Inner and outer approximations of polytopes using boxes, *Computational Geometry: Theory and Applications* 27(2): 151–178.

Bemporad, A. & Morari, M. (1999). Robust model predictive control: A survey, in robustness in identification and control, *Lecture Notes in Control and Information Sciences* 245: 207–226.

Braun, M., Rivera, D., Flores, M., Carlyle, W. & Kempf, K. (2003). A model predictive control framework for robust management of multi-product, multi-echelon demand networks, *Annual Reviews in Control* 27(2): 229–245.

Camacho, E. & Bordons, C. (2004). *Model predictive control*, Springer.

Camponogara, E., Jia, D., Krogh, B. & Talukdar, S. (2002). Distributed model predictive control, *IEEE Control Systems Magazine* 22(1): 44–52.

Cheng, R., Forbes, J. & Yip, W. (2007). Price-driven coordination method for solving plant-wide mpc problems, *Journal of Process Control* 17(5): 429–438.

Cheng, R., Fraser Forbes, J. & Yip, W. (2008). Dantzig–wolfe decomposition and plant-wide mpc coordination, *Computers and Chemical Engineering* 32(7): 1507–1522.

Dubey, P. & Rogawski, J. (1990). Inefficiency of smooth market mechanisms, *Journal of Mathematical Economics* 19(3): 285–304.

Dunbar, W. (2007). Distributed receding horizon control of dynamically coupled nonlinear systems, *IEEE Transactions on Automatic Control* 52(7): 1249–1263.

Dunbar, W. & Murray, R. (2006). Distributed receding horizon control for multi-vehicle formation stabilization, *Automatica* 42(4): 549–558.

Goodwin, G., Salgado, M. & Silva, E. (2005). Time-domain performance limitations arising from decentralized architectures and their relationship to the rga, *International Journal of Control* 78(13): 1045–1062.

Haimes, Y. & Chankong, V. (1983). *Multiobjective decision making: Theory and methodology*, North Holland, New York.

Henten, E. V. & Bontsema, J. (2009). Time-scale decomposition of an optimal control problem in greenhouse climate management, *Control Engineering Practice* 17(1): 88–96.

Hovd, M. & Skogestad, S. (1994). Sequential design of decentralized controllers, *Automatica* 30: 1601–1601.

Jamoom, M., Feron, E. & McConley, M. (2002). Optimal distributed actuator control grouping schemes, *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 2, pp. 1900–1905.

Jia, D. & Krogh, B. (2001). Distributed model predictive control, *American Control Conference, 2001. Proceedings of the 2001*, Vol. 4.

Jia, D. & Krogh, B. (2002). Min-max feedback model predictive control for distributed control with communication, *American Control Conference, 2002. Proceedings of the 2002*, Vol. 6.

Kouvaritakis, B. & Cannon, M. (2001). *Nonlinear predictive control: theory and practice*, Iet.

Kumar, A. & Daoutidis, P. (2002). Nonlinear dynamics and control of process systems with recycle, *Journal of Process Control* 12(4): 475–484.

Lu, J. (2003). Challenging control problems and emerging technologies in enterprise optimization, *Control Engineering Practice* 11(8): 847–858.

Maciejowski, J. (2002). *Predictive control: with constraints*, Prentice Hall.

Mayne, D., Rawlings, J., Rao, C. & Scokaert, P. (2000). Constrained model predictive control: Stability and optimality, *Automatica* 36: 789–814.

Motee, N. & Sayyar-Rodsari, B. (2003). Optimal partitioning in distributed model predictive control, *Proceedings of the American Control Conference*, Vol. 6, pp. 5300–5305.

Nash, J. (1951). Non-cooperative games, *Annals of mathematics* pp. 286–295.

Neck, R. & Dockner, E. (1987). Conflict and cooperation in a model of stabilization policies: A differential game approach, *J. Econ. Dyn. Cont* 11: 153–158.

Osborne, M. & Rubinstein, A. (1994). *A course in game theory*, MIT press.

Perea-Lopez, E., Ydstie, B. & Grossmann, I. (2003). A model predictive control strategy for supply chain optimization, *Computers and Chemical Engineering* 27(8-9): 1201–1218.

Primbs, J. & Nevistic, V. (2000). Feasibility and stability of constrained finite receding horizon control, *Automatica* 36(7): 965–971.

Rossiter, J. (2003). *Model-based predictive control: a practical approach*, CRC press.

Salgado, M. & Conley, A. (2004). Mimo interaction measure and controller structure selection, *International Journal of Control* 77(4): 367–383.

Sandell Jr, N., Varaiya, P., Athans, M. & Safonov, M. (1978). Survey of decentralized control methods for large scale systems, *IEEE Transactions on Automatic Control* 23(2): 108–128.

Vaccarini, M., Longhi, S. & Katebi, M. (2009). Unconstrained networked decentralized model predictive control, *Journal of Process Control* 19(2): 328–339.

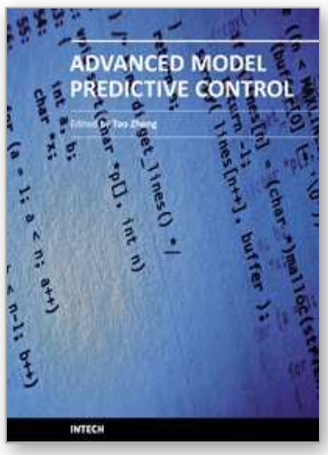
Venkat, A., Hiskens, I., Rawlings, J. & Wright, S. (2008). Distributed mpc strategies with application to power system automatic generation control, *IEEE Transactions on Control Systems Technology* 16(6): 1192–1206.

Šiljak, D. (1996). Decentralized control and computations: status and prospects, *Annual Reviews in Control* 20: 131–141.

Wittenmark, B. & Salgado, M. (2002). Hankel-norm based interaction measure for input-output pairing, *Proc. of the 2002 IFAC World Congress*.

Zhang, Y. & Li, S. (2007). Networked model predictive control based on neighbourhood optimization for serially connected large-scale processes, *Journal of process control* 17(1): 37–50.

Zhu, G. & Henson, M. (2002). Model predictive control of interconnected linear and nonlinear processes, *Industrial and Engineering Chemistry Research* 41(4): 801–816.



Advanced Model Predictive Control

Edited by Dr. Tao ZHENG

ISBN 978-953-307-298-2
Hard cover, 418 pages
Publisher InTech
Published online 24, June, 2011
Published in print edition June, 2011

Model Predictive Control (MPC) refers to a class of control algorithms in which a dynamic process model is used to predict and optimize process performance. From lower request of modeling accuracy and robustness to complicated process plants, MPC has been widely accepted in many practical fields. As the guide for researchers and engineers all over the world concerned with the latest developments of MPC, the purpose of "Advanced Model Predictive Control" is to show the readers the recent achievements in this area. The first part of this exciting book will help you comprehend the frontiers in theoretical research of MPC, such as Fast MPC, Nonlinear MPC, Distributed MPC, Multi-Dimensional MPC and Fuzzy-Neural MPC. In the second part, several excellent applications of MPC in modern industry are proposed and efficient commercial software for MPC is introduced. Because of its special industrial origin, we believe that MPC will remain energetic in the future.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen