OPTIMAL CONTROL APPLICATIONS AND METHODS
Optim. Control Appl. Meth. 2011; **32**:153–176
Published online 7 May 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/oca.940

Distributed model predictive control based on a cooperative game

J. M. Maestre, D. Muñoz de la Peña*,† and E. F. Camacho

Departamento de Ingeniería de Sistemas y Automática, Escuela Superior de Ingenieros, Camino de los Descubrimientos sn., 41092 Seville, Spain

SUMMARY

In this work we propose a distributed model predictive control scheme based on a cooperative game in which two different agents communicate in order to find a solution to the problem of controlling two constrained linear systems coupled through the inputs. We assume that each agent only has partial information of the model and the state of the system. In the proposed scheme, the agents communicate twice each sampling time in order to share enough information to take a cooperative decision. We provide sufficient conditions that guarantee practical stability of the closed-loop system as well as an optimization-based procedure to design the controller so that these conditions are satisfied. The theoretical results and the design procedure are illustrated using two different examples. Copyright © 2010 John Wiley & Sons, Ltd.

Received 18 February 2009; Revised 5 November 2009; Accepted 2 March 2010

KEY WORDS: model predictive control; distributed optimization; multi-agents systems; game theory

1. INTRODUCTION

Model predictive control (MPC) is an optimization-based control strategy that has been successfully applied to process control, specially to constrained multivariable systems exhibiting dead times [1]. At each sampling time, a finite-horizon optimal control problem based on the model of the system is solved. Only the first control move of the optimal solution is applied and the rest of the sequence is discarded repeating the procedure when a new measurement is available (the so-called receding horizon scheme). In many cases, MPC cannot be applied to large-scale systems due to the computational requirements or to the impossibility of obtaining a centralized model of the whole system. Typical examples of large-scale systems are transportation systems such as traffic, water or power networks [2]. In addition, there is an increasing interest in networked control systems, where dedicated local control networks can be augmented with additional networked (wired and/or wireless) actuator/sensor devices which have become cheap and easy-to-install [3, 4].

Most large-scale and networked control systems are based on a decentralized architecture; that is, the system is divided into several subsystems, each controlled by a different agent that does not share information with the rest. Each of the agents implements an MPC based on a reduced model of the system and on partial state information, which in general results in an optimization problem with a lower computational burden. Figures 1 and 2 show a centralized and a decentralized

*Correspondence to: D. Muñoz de la Peña, Departamento de Ingeniería de Sistemas y Automática, Escuela Superior de Ingenieros, Camino de los Descubrimientos sn., 41092 Seville, Spain.
†E-mail: davidmps@cartuja.us.es

Contract/grant sponsor: European Commission; contract/grant number: INFSO-ICT-223854
Contract/grant sponsor: MEC-Spain; contract/grant number: DPI2008-05818

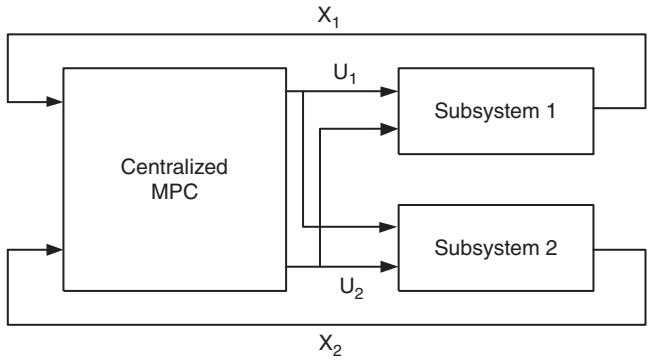

Figure 1. Centralized MPC scheme.

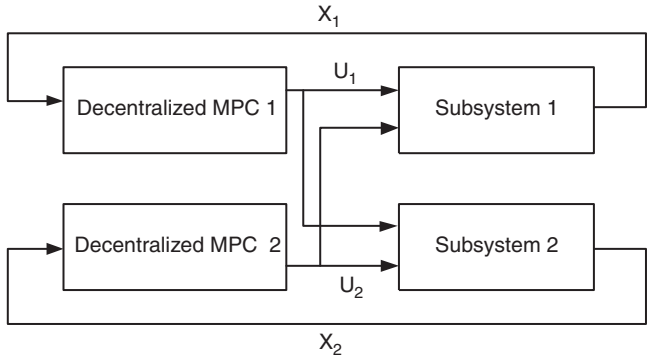

Figure 2. Decentralized MPC scheme.

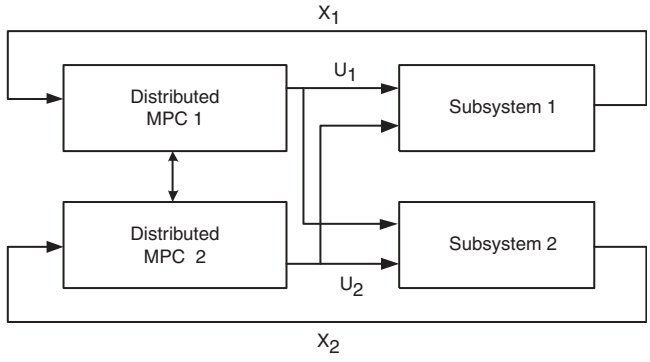

Figure 3. Distributed MPC scheme.

MPC control scheme for two subsystems coupled through the inputs. For example, in [5], a MPC algorithm was proposed under the main assumptions that the system is nonlinear, discrete-time and no information is exchanged between local controllers. The stability of this class of systems, from an input-to-state stability point of view, was studied in [6].

While this paradigm to process control has been successful, there is an increasing interest in developing distributed model predictive control (DMPC) schemes, where agents share information in order to improve closed-loop performance, robustness and fault-tolerance. See Figure 3 for an example of this control scheme. The performance of the closed-loop system depends on the decisions that all the agents take; hence, cooperation and communication policies become very

DOI: 10.1002/oca

important issues. In this context, several distributed MPC schemes have been proposed in the literature that deal with the coordination of separate MPC controllers that communicate in order to obtain optimal input trajectories in a distributed manner; see [7, 8] for reviews of results in this area. In [9], sufficient conditions that guarantee stability of a class of distributed controllers are given. In [10], basic collaboration algorithms are provided with an extensive list of conditions to ensure convergence and stability. In [11], the problem of distributed control of dynamically coupled nonlinear systems that are subject to decoupled constraints was considered. In [12, 13], the effect of the coupling was modeled as a bounded disturbance compensated using a robust MPC formulation. In [14] distributed MPC of decoupled systems (a class of systems of relevance in the framework of multi-agents systems) was studied. Several works in the literature assume that each agent has access to the model of the full system. For example, in [15], it was proven that through multiple communications between the distributed controllers and using system-wide control objective functions, stability of the closed-loop system can be guaranteed. In [16, 17] a decentralized control architecture for nonlinear systems with continuous and asynchronous measurements was presented. Following up on this work, in [18] a distributed MPC method for the design of networked control systems based on Lyapunov-based MPC was presented. In both cases, each agent had access to the full system model. However, sometimes it is not possible to obtain a model that describes a system globally or even if it is possible, it cannot be computed in a reasonable time or its very expensive, for example, in large-scale systems.

Motivated by these issues, in this paper we propose a DMPC algorithm for two agents based on game theory in which two different agents communicate in order to find a solution to the problem of controlling two constrained linear systems coupled through the inputs. We assume that each agent only has partial information of the model and the state of the system. Game theory is a theoretical framework that allows one to study the problem of cooperation of different agents with, maybe, conflicting control goals, from a mathematical point of view [19, 20]. In the proposed scheme, the coordination problem between the agents is reduced to a cooperative game where they have to choose one out of three options, and only two communication cycles are needed to reach an agreement. In addition, we provide sufficient conditions that guarantee practical stability of the closed-loop system as well as an optimization-based procedure to design the controller so that these conditions are satisfied. The theoretical results and the design procedure are illustrated using two different examples.

2. PROBLEM FORMULATION

Consider the following class of distributed linear systems in which two subsystems coupled with the neighbor subsystem through the inputs are defined

$$\begin{aligned} x_1(t+1) &= A_1 x_1(t) + B_{11} u_1(t) + B_{12} u_2(t) \\ x_2(t+1) &= A_2 x_2(t) + B_{21} u_1(t) + B_{22} u_2(t) \end{aligned} \tag{1}$$

where $x_i \in \mathbb{R}^{n_i}$, $i = 1, 2$, are the states of each subsystem and $u_i \in \mathbb{R}^{m_i}$, $i = 1, 2$, are the different inputs. This class of systems is of relevance when identifications techniques are used to obtain the transfer function of a process. We consider the following linear constraints in the state and the inputs:

$$x_i \in \mathscr{X}_i, \quad u_i \in \mathscr{U}_i, \quad i = 1, 2$$

where $\mathscr{X}_i$ and $\mathscr{U}_i$ with $i = 1, 2$ are defined by a set of linear inequalities.

The control objective is to regulate the system to the origin while guaranteeing that the constraints are satisfied. Centralized MPC solves a single optimization problem to decide the optimal sequences of the inputs u_1 and u_2 with respect to a given performance index based on the full model of the system and on the measurements from all the sensors, see Figure 1. In distributed and decentralized schemes, two independent controllers (hereby denoted agents) are defined. Agent 1 has access to the model of subsystem 1, its state x_1 and decides the value of u_1. On the other hand, agent 2

DOI: 10.1002/oca

has access to the model of subsystem 2, its state x_2 and decides the value of u_2. This implies that neither agent has access to the full model or state information and that in order to find a cooperative solution, they must communicate.

A control system is decentralized if there is no communication among the agents, see Figure 2. This is the worst scenario from the performance point of view, because each agent has to cope alone with its control problem with the risk that the absence of coordination in the agents' decisions may lead to the instability of the system. The control system is distributed if there is communication between agents, see Figure 3. The degree of communication depends on the control problem and the communication constraints. In this section, we present a distributed MPC controller based on a cooperative game scheme between two different agents.

The objective of the DMPC scheme is to minimize a performance index that depends on the future evolution of both states and inputs. At each sampling time, each agent solves a sequence of reduced dimension optimization problems based on the model of its subsystem and assuming a given fixed input trajectory for its neighbor. In order to describe the algorithm, we need to introduce the following definitions:

- U_i: Future input sequence of agent i. These are the decision variables of the optimization problems solved by both agents

$$U_1 = \begin{bmatrix} u_{1,0} \\ u_{1,1} \\ \vdots \\ u_{1,N-1} \end{bmatrix}, \quad U_2 = \begin{bmatrix} u_{2,0} \\ u_{2,1} \\ \vdots \\ u_{2,N-1} \end{bmatrix}$$

- ni: Neighboring agent of agent i; that is, $U_{n1} = U_2$ and $U_{n2} = U_1$.
- J_i: Local cost function of agent i based on the predicted trajectories of its state defined as follows:

$$J_1(x_1, U_1, U_2) = \sum_{k=0}^{N-1} L_1(x_{1,k}, u_{1,k}) + F_1(x_{1,N})$$

$$J_2(x_2, U_2, U_1) = \sum_{k=0}^{N-1} L_2(x_{2,k}, u_{2,k}) + F_2(x_{2,N})$$

  where $L_i(\cdot)$ and $F_i(\cdot)$ with $i = 1, 2$ are the stage and terminal cost functions, respectively, defined as

$$L_i(x, u) = x^{\mathrm{T}} Q_i x + u^{\mathrm{T}} R_i u$$

$$F_i(x) = x^{\mathrm{T}} P_i x$$

  with $Q_i, P_i > 0, R_i \geqslant 0$. The prediction horizon is N. We use the notation $x_{i,k}$ to denote the k-steps ahead future predicted state obtained from the initial state x_i applying the input trajectories defined by U_1 and U_2. Note that the second and third parameters of functions J_1 and J_2 are switched. This will allow us to simplify the algorithm definition.
- $U_i^d(t)$: Optimal input sequence of agent i at time t, denoted $U_i^d(t)$, defined as:

$$U_1^d(t) = \begin{bmatrix} u_{1,0}^d \\ u_{1,1}^d \\ \vdots \\ u_{1,N-1}^d \end{bmatrix}, \quad U_2^d(t) = \begin{bmatrix} u_{2,0}^d \\ u_{2,1}^d \\ \vdots \\ u_{2,N-1}^d \end{bmatrix}$$

DOI: 10.1002/oca

- $U_i^s(t)$: Shifted optimal input sequence of agent i obtained from the optimal input sequence of agent i at time $t-1$, denoted $U_i^d(t-1)$, as follows:

$$U_1^s(t)=\begin{bmatrix} u_{1,1}^d \\ u_{1,2}^d \\ \vdots \\ u_{1,N-1}^d \\ K_1 x_{1,N} \end{bmatrix}, \quad U_2^s(t)=\begin{bmatrix} u_{2,1}^d \\ u_{2,2}^d \\ \vdots \\ u_{2,N-1}^d \\ K_2 x_{2,N} \end{bmatrix}$$

where $x_{1,N}, x_{2,N}$ are the N-steps ahead predicted state obtained from $x_1(t-1), x_2(t-1)$, respectively, applying the input trajectories $U_1^d(t-1), U_2^d(t-1)$ and K_1, K_2 are two known feedback gains.

The proposed DMPC algorithm is the following:

1. At time step t, each agent i receives its corresponding partial state measurement $x_i(t)$.
2. Each agent i minimizes J_i assuming that the neighbor keeps applying the optimal trajectory evaluated at the previous time step; that is, $U_{ni}=U_{ni}^s(t)$. Agent 1 solves the following optimization problem:

$$\begin{aligned} U_1^*(t)=\arg\min_{U_1} \quad & J_1(x_1(t), U_1, U_2^s(t)) \\ & x_{1,k+1}=A_1 x_{1,k}+B_{11}u_{1,k}+B_{12}u_{2,k} \\ & x_{1,0}=x_1(t) \\ & x_{1,k}\in\mathscr{X}_1, \quad k=0,\ldots,N \\ & u_{1,k}\in\mathscr{U}_1, \quad k=0,\ldots,N-1 \\ & x_{1,N}\in\Omega_1 \end{aligned} \tag{2}$$

Agent 2 solves the following optimization problem:

$$\begin{aligned} U_2^*(t)=\arg\min_{U_2} \quad & J_2(x_2(t), U_2, U_1^s(t)) \\ & x_{2,k+1}=A_2 x_{2,k}+B_{22}u_{2,k}+B_{21}u_{1,k} \\ & x_{2,0}=x_2(t) \\ & x_{2,k}\in\mathscr{X}_2, k=0,\ldots N \\ & u_{2,k}\in\mathscr{U}_2, k=0,\ldots N-1 \\ & x_{2,N}\in\Omega_2 \end{aligned} \tag{3}$$

The sets Ω_1 and Ω_2 define the terminal region constraints that are necessary to prove closed-loop practical stability following a terminal region/terminal cost approach. Note that in both optimization problems the free variable is U_i while the neighbor input trajectory U_{ni} is fixed.
3. Each agent i minimizes J_i optimizing the neighbor input assuming that it applies the input trajectory computed in the previous optimization problem U_i^*. Agent 1 solves the following

DOI: 10.1002/oca

optimization problem:

$$\begin{aligned} U_2^w(t) = \arg\min_{U_2} \quad & J_1(x_1(t), U_1^*(t), U_2) \\ & x_{1,k+1} = A_1 x_{1,k} + B_{11} u_{1,k} + B_{12} u_{2,k} \\ & x_{1,0} = x_1(t) \\ & x_{1,k} \in \mathscr{X}_1, \quad k = 0, \ldots, N \\ & u_{2,k} \in \mathscr{U}_2, \quad k = 0, \ldots, N-1 \\ & x_{1,N} \in \Omega_1 \end{aligned} \tag{4}$$

Agent 2 solves the following optimization problem:

$$\begin{aligned} U_1^w(t) = \arg\min_{U_1} \quad & J_2(x_2(t), U_2^*(t), U_1) \\ & x_{2,k+1} = A_2 x_{2,k} + B_{22} u_{2,k} + B_{21} u_{1,k} \\ & x_{2,0} = x_2(t) \\ & x_{2,k} \in \mathscr{X}_2, \quad k = 0, \ldots, N \\ & u_{1,k} \in \mathscr{U}_1, \quad k = 0, \ldots, N-1 \\ & x_{2,N} \in \Omega_2 \end{aligned} \tag{5}$$

In this optimization problem, the free variable is U_{ni} (the input trajectory U_i is fixed). Solving this optimization problem, agent i defines an input trajectory for its neighbor that optimizes its local cost function J_i.

4. Both agents communicate. Agent 1 sends $U_1^*(t)$ and $U_2^w(t)$ to agent 2 and receives $U_2^*(t)$ and $U_1^w(t)$.
5. Each agent evaluates the local cost function J_i for each of the nine different possible combinations of input trajectories; that is $U_1 \in \{U_1^s(t), U_1^w(t), U_1^*(t)\}$ and $U_2 \in \{U_2^s(t), U_2^w(t), U_2^*(t)\}$.
6. Both agents communicate and share the information of the value of the local cost function for each possible combination of input trajectories. In this step, both agents receive enough information to take a cooperative decision.
7. Each agent applies the input trajectory that minimizes $J = J_1 + J_2$. Because both agents have access to the same information after the second communication cycle, both agents choose the same optimal input sets. We denote the chosen set of input trajectories as $U_1^d(t), U_2^d(t)$.
8. The first input of each optimal sequence is applied and the procedure is repeated the next sampling time.

From a game theory point of view, at each time step both agents are playing a cooperative game. This game can be synthesized in strategic form by a 3×3 matrix. Each row represents one of the three possible decisions of agent 1, and each column represents one of the three possible decisions of agent 2. The cells contain the sum of the cost functions of both agents for a particular choice of future inputs. At each time step, the option that yields a lower global cost is chosen. Note that both agents share this information, hence, they both choose the same option. The nine possibilities are shown in Table I.

Remark

At each sampling time, the controllers decide among three different options. The shifted optimal input trajectory $U_i^s(t)$ keeps applying the latest optimal trajectory. The *selfish* option $U_i^*(t)$ provides the best improvement in J_i if the rest of the system's manipulated variables stay unchanged. The *altruist* option $U_i^w(t)$ provides the best improvement for the neighbor agent cost function J_2. In this case, the agent i sacrifices its own welfare in order to improve the overall performance.

DOI: 10.1002/oca

Table I. Cost function table used for the decision making.

	$U_2^s(t)$	$U_2^*(t)$	$U_2^w(t)$
$U_1^s(t)$	$J_1(x_1(t), U_1^s(t), U_2^s(t))$ $+J_2(x_2(t), U_2^s(t), U_1^s(t))$	$J_1(x_1(t), U_1^s(t), U_2^*(t))$ $+J_2(x_2(t), U_2^*(t), U_1^s(t))$	$J_1(x_1(t), U_1^s(t), U_2^w(t))$ $+J_2(x_2(t), U_2^w(t), U_1^s(t))$
$U_1^*(t)$	$J_1(x_1(t), U_1^*(t), U_2^s(t))$ $+J_2(x_2(t), U_2^s(t), U_1^*(t))$	$J_1(x_1(t), U_1^*(t), U_2^*(t))$ $+J_2(x_2(t), U_2^*(t), U_1^*(t))$	$J_1(x_1(t), U_1^*(t), U_2^w(t))$ $+J_2(x_2(t), U_2^w(t), U_1^*(t))$
$U_1^w(t)$	$J_1(x_1(t), U_1^w(t), U_2^s(t))$ $+J_2(x_2(t), U_2^s(t), U_1^w(t))$	$J_1(x_1(t), U_1^w(t), U_2^*(t))$ $+J_2(x_2(t), U_2^*(t), U_1^w(t))$	$J_1(x_1(t), U_1^w(t), U_2^w(t))$ $+J_2(x_2(t), U_2^w(t), U_1^w(t))$

Remark
Centralized MPC solves a single large-scale problem based on the model of the whole system, see Figure 1. In the example section we will compare the performance of the proposed approach with a centralized MPC controller based on the following optimization problem:

$$
\begin{aligned}
\{U_1^c(t), U_2^c(t)\} = \arg\min_{U_1, U_2} \quad & J_1(x_1(t), U_1, U_2) + J_2(x_1(t), U_2, U_1) \\
& x_{1,k+1} = A_1 x_{1,k} + B_{11} u_{1,k} + B_{12} u_{2,k} \\
& x_{1,0} = x_1(t) \\
& x_{1,k} \in \mathscr{X}_1, \quad k = 0, \ldots, N \\
& u_{1,k} \in \mathscr{U}_1, \quad k = 0, \ldots, N-1 \\
& x_{1,N} \in \Omega_1 \\
& x_{2,k+1} = A_2 x_{2,k} + B_{22} u_{2,k} + B_{21} u_{1,k} \\
& x_{2,0} = x_2(t) \\
& x_{2,k} \in \mathscr{X}_2, \quad k = 0, \ldots, N \\
& u_{2,k} \in \mathscr{U}_2, \quad k = 0, \ldots, N-1 \\
& x_{2,N} \in \Omega_2
\end{aligned}
\tag{6}
$$

The centralized MPC provides, in general, the best closed-loop performance, but can only be applied when it is possible to control the system with a single controller that has access to the full model and state of the same.

Remark
In general, the minimum number of communication steps needed to implement a cooperative control scheme is two. In the first step each agent informs of its intentions to its neighbors and during the second it can confirm if it accepts its neighbors' intentions. In the best case, an agreement can be achieved in the second step, but in general an iterative procedure will be needed to reach an agreement.

Remark
The proposed controller scheme is cooperative from a game theory point of view because each agent chooses the solution that optimizes a cost function that depends on both subsystems, not only on the future trajectories of its subsystem. If the decision taken does not depend on a global performance index, the solution is not cooperative. In the simulation section, we will compare the proposed distributed controller with a distributed scheme in which the two agents communicate, but do not take a cooperative decision. They iterate until an agreement is obtained. In this case, the solution is a Nash equilibrium [15] of the multi-objective optimization problem defined by the

DOI: 10.1002/oca

cost functions of both agents. At each iteration, agent 1 solves the following optimization problem:

$$\begin{aligned} U_1^{l+1} = \arg\min_{U_1} \quad & J_1(x_1, U_1, U_2^l) \\ & x_{1,k+1} = A_1 x_{1,k} + B_{11} u_{1,k} + B_{12} u_{2,k} \\ & x_{1,0} = x_1 \\ & x_{1,k} \in \mathscr{X}_1, \quad k = 0, \ldots, N \\ & u_{1,k} \in \mathscr{U}_1, \quad k = 0, \ldots, N-1 \\ & x_{1,N} \in \Omega_1 \end{aligned} \tag{7}$$

and agent 2 solves the following optimization problem:

$$\begin{aligned} U_2^{l+1} = \arg\min_{U_2} \quad & J_2(x_2, U_2, U_1^l) \\ & x_{2,k+1} = A_2 x_{2,k} + B_{22} u_{2,k} + B_{21} u_{1,k} \\ & x_{2,0} = x_2 \\ & x_{2,k} \in \mathscr{X}_2, \quad k = 0, \ldots, N \\ & u_{2,k} \in \mathscr{U}_2, \quad k = 0, \ldots, N-1 \\ & x_{2,N} \in \Omega_2 \end{aligned} \tag{8}$$

with $U_1^0 = U_1^s$ and $U_2^0 = U_2^s$; that is, the initial guess is given by the shifted trajectory. An agreement is reached when the difference between the proposed control vector by each agent at one iteration and its value at the previous iteration is below a threshold. This implies that they do not share information about the utility of each decision, they reach an agreement when neither of them can improve, hence reaching a Nash equilibrium. In the example, we will compare the proposed controller with different controllers based on this distributed scheme, each one carrying out a fixed number of iterations, to demonstrate that the proposed cooperative scheme provides a better performance with a lower number of iterations.

Remark
Although the option chosen by the algorithm is the Pareto optimum of the game that both agents are playing, in general, it is not a Pareto optimum of the multi-objective optimization problem defined by the cost functions J_1 and J_2.

Remark
The proposed scheme can be extended to deal with N agents, however, in order to build a global cost table to take a cooperative decision, the complexity grows exponentially. In order to reduce the complexity, the structure of the system may be exploited taking into account that an input may not affect all the outputs. Also, in general, not all the possible cooperation options are employed with the same frequency, hence, it is possible to reduce further the complexity by not taking into account the less frequent options.

Remark
In the proposed algorithm both agents can operate in parallel; that is, the agents can compute U_i^* and U_i^w simultaneously (steps 2 and 3).

3. STABILITY PROPERTIES

Controlling a system between two independent agents may lead to an unstable system. The resulting closed-loop system is a multiprocess system and studying the stability of this class of systems is in general a difficult task. Following a terminal region/terminal constraint approach [21, 22], in

DOI: 10.1002/oca

this section we provide sufficient conditions that guarantee practical stability of the closed-loop system as well as an optimization-based procedure to design the controller so that these conditions are satisfied. This result is stated in the following theorem:

Theorem 1
Assume that there exist linear feedbacks $u_1 = K_1 x_1$ and $u_2 = K_2 x_2$ such that the following conditions hold:

$$F_1((A_1 + B_{11}K_1)x_1 + B_{12}K_2 x_2) - F_1(x_1) + L_1(x_1, K_1 x_1) - d_1 \leqslant 0 \quad \forall x_2 \in \Omega_2 \tag{9a}$$

$$F_2((A_2 + B_{22}K_2)x_2 + B_{21}K_1 x_1) - F_1(x_2) + L_1(x_2, K_2 x_2) - d_2 \leqslant 0 \quad \forall x_1 \in \Omega_1 \tag{9b}$$

$$x_1 \in \Omega_1 \rightarrow (A_1 + B_{11}K_1)x_1 + B_{12}K_2 x_2 \in \Omega_1 \quad \forall x_2 \in \Omega_2 \tag{9c}$$

$$x_2 \in \Omega_2 \rightarrow (A_2 + B_{22}K_2)x_2 + B_{21}K_1 x_1 \in \Omega_2 \quad \forall x_1 \in \Omega_1 \tag{9d}$$

$$K_1 x_1 \in \mathscr{U}_1 \quad \forall x_1 \in \Omega_1 \tag{9e}$$

$$K_2 x_2 \in \mathscr{U}_2 \quad \forall x_2 \in \Omega_2 \tag{9f}$$

$$\Omega_1 \in \mathscr{X}_1 \tag{9g}$$

$$\Omega_2 \in \mathscr{X}_2 \tag{9h}$$

Then, if at $t = 0$, $U_1^s(0), U_2^s(0)$ are given such that Problems (2) and (3) are feasible for $x_{1,0} = x_1(0), x_{2,0} = x_2(0), U_1 = U_1^s(0)$ and $U_2 = U_2^s(0)$, then the proposed algorithm is feasible for all time steps $t \geqslant 0$ and system (1) in closed-loop with the proposed distributed MPC controller is ultimately bounded in a region that contains the origin in its interior.

Proof
The proof consists of two parts. We first prove recursive feasibility of Problems (2) and (3) if at time t, $U_1^s(t), U_2^s(t)$ are given such that (2) and (3) are feasible for $x_{1,0} = x_1(t), x_{2,0} = x_2(t), U_1 = U_1^s(t)$ and $U_2 = U_2^s(t)$. Then we prove that, under the stated assumptions,

$$J(t) = J_1(x_1(t), U_1^d(t), U_2^d(t)) + J_2(x_2(t), U_2^d(t), U_1^d(t))$$

is a decreasing sequence of values with a lower bound. This implies that system (1) in closed-loop with the proposed distributed MPC controller is ultimately bounded in a region that contains the origin in its interior.

Part 1. We will prove this part by recursion. First, we prove that if the state and input trajectories obtained from $x_1(t-1)$ and $x_2(t-1)$ applying $U_1^d(t-1)$ and $U_2^d(t-1)$ satisfy all the constraints of Problems (2) and (3), then $U_1^d(t)$ and $U_2^d(t)$ also satisfy all the constraints. Recalling step 5 of the proposed algorithm, to prove this statement it is sufficient to prove that there exists at least a pair of input trajectories that satisfy all the constraints. To this end, we will prove that $U_1^s(t), U_2^s(t)$ provide a feasible solution for $x_1(t)$ and $x_2(t)$. Note that, in general, it is not possible to guarantee that any of the other options are feasible.

Taking into account that by definition $U_1^d(t-1)$ and $U_1^d(t-1)$ satisfy the constraints of Problems (2) and (3), the following statements hold:

$$x_{1,k} \in \mathscr{X}_1, \quad k = 0, \ldots, N$$

$$u_{1,k}^d \in \mathscr{U}_1, \quad k = 0, \ldots, N-1$$

$$x_{1,N} \in \Omega_1$$

$$x_{2,k} \in \mathscr{X}_2, \quad k = 0, \ldots, N$$

$$u_{2,k}^d \in \mathscr{U}_2, \quad k = 0, \ldots, N-1$$

$$x_{2,N} \in \Omega_2$$

DOI: 10.1002/oca

where $x_{1,k}, x_{2,k}$ are the k-steps ahead predicted state obtained from $x_1(t-1), x_2(t-1)$, respectively, applying the input trajectories $U_1^d(t-1), U_2^d(t-1)$ defined by $u_{1,k}^d, u_{2,k}^d$ with $k=0, \ldots, N-1$.

At time step $t-1$, the first input of the chosen trajectories $U_1^d(t-1), U_2^d(t-1)$ is applied; that is, $u_1(t-1)=u_{1,0}^d$ and $u_2(t-1)=u_{2,0}^d$. This implies that

$$x_1(t)=A_1x_1(t-1)+B_{11}u_1(t-1)+B_{12}u_2(t-1)=A_1x_{1,0}+B_{11}u_{1,0}^d+B_{12}u_{2,0}^d=x_{1,1}.$$

Taking into account the definitions of $U_1^s(t)$ and $U_2^s(t)$, it can be proved that the k-steps ahead predicted states obtained from $x_1(t), x_2(t)$, respectively, applying the input trajectories $U_1^s(t), U_2^s(t)$ satisfy all the constraints from $k=0$ to $N-1$. Moreover, as

$$x_{1,N} \in \Omega_1, \quad x_{2,N} \in \Omega_2$$

it holds that

$$(A_1+B_{11}K_1)x_{1,N}+B_{12}K_2x_{2,N} \in \Omega_1$$

$$(A_2+B_{22}K_2)x_{2,N}+B_{21}K_1x_{1,N} \in \Omega_2$$

and hence all the constraints of Problems (2) and (3) are satisfied which implies that $U_1^s(t), U_2^s(t)$ and hence $U_1^d(t), U_2^d(t)$ provide a feasible solution for $x_1(t)$ and $x_2(t)$. Taking into account that by assumption, $U_1^s(0), U_2^s(0)$ satisfy all the constraints for $x_1(0)$ and $x_2(0)$ and using the above result recursively, the statement of this part is proved.

Part 2. In this part we will prove that

$$J_1(x_1(t), U_1^s(t), U_2^s(t))+J_2(x_2(t), U_2^s(t), U_1^s(t)) \leqslant J(t-1)+d_1+d_2$$

where

$$J(t-1)=J_1(x_1(t-1), U_1^d(t-1), U_2^d(t-1))+J_2(x_2(t-1), U_2^d(t-1), U_1^d(t-1)).$$

Taking into account the definitions of $U_1^d(t-1)$ and $U_1^s(t)$ it follows that

$$J_1(x_1(t), U_1^s(t), U_2^s(t))-J_1(x_1(t-1), U_1^d(t-1), U_2^d(t-1))$$

is equal to

$$F_1((A_1+B_{11}K_1)x_{1,N}+B_{12}K_2x_{2,N})-F_1(x_{1,N})+L_1(x_{1,N}, K_1x_{1,N})-L_1(x_{1,0}, K_1x_{1,0})$$

As $L_1(x_{1,0}, K_1x_{1,0}) \geqslant 0$ and taking into account (9a) and (9b), that $x_{1,N} \in \Omega_1$ and that $x_{2,N} \in \Omega_2$ it follows that

$$J_1(x_1(t), U_1^s(t), U_2^s(t))-J_1(x_1(t-1), U_1^d(t-1), U_2^d(t-1))-d_1 \leqslant 0$$

Following the same steps for J_2 we obtain that

$$J_2(x_2(t), U_2^s(t), U_1^s(t))-J_2(x_2(t-1), U_2^d(t-1), U_1^d(t-1))-d_2 \leqslant 0$$

and hence

$$J_1(x_1(t), U_1^s(t), U_2^s(t))+J_2(x_2(t), U_2^s(t), U_1^s(t)) \leqslant J(t-1)+d_1+d_2$$

As the proposed algorithm chooses $U_1^d(t), U_2^d(t)$ as the pair of input trajectories that yield the minimum cost, it is easy to see that

$$J(t) \leqslant J(t-1)+d_1+d_2$$

Following standard Lyapunov arguments and taking into account that recursive feasibility is guaranteed (see the first part of the proof), it is proved that system (1) in closed-loop with the proposed controller is ultimately bounded in a region that contains the origin in its interior. □

DOI: 10.1002/oca

3.1. Design procedure

In the previous section, we have provided sufficient conditions to guarantee that the system in closed-loop with the proposed distributed MPC scheme is practically stable. In general, designing the controller parameters so that these conditions are satisfied is a hard problem because the design constraints are coupled; for example, the constraints that define the invariant set Ω_1 depend on the set Ω_2 and viceversa. For centralized MPC controllers, there are various methods described in the literature on how to design a stabilizing local controller, terminal cost function and terminal region [21–23] (for example, the local controller and the terminal cost can be obtained solving a LQR problem). These results however cannot be applied to the distributed case. In this section, we present an optimization-based procedure to find local controllers K_1, K_2, matrices P_1, P_2 and regions Ω_1, Ω_2 such that (9a) holds for a given system.

The procedure determines first matrices K_1, K_2, P_1 and P_2 such that (9a) and (9b) hold for any given sets Ω_1 and Ω_2 solving a linear matrix inequality (LMI) optimization problem. Once the local feedbacks K_1 and K_2 are fixed, the invariant sets Ω_1 and Ω_2 are obtained. Note that constants d_1 and d_2 are determined *a posteriori*, once the local feedbacks, terminal costs and terminal regions are fixed.

From the point of view of each agent, its neighbor's input can be viewed as a disturbance. This allows us to use well-known tools from control of linear uncertain systems in order to determine a local controller such that a given degree of robustness is guaranteed. In [24–26] several methods to solve this class of problems are presented. In particular, constraint (9a) can be transformed into a LMI and solved using standard techniques, moreover, is equivalent to designing an $\mathscr{H}$-infinity controller for subsystem 1 assuming that u_2 is the disturbance [26]. The same technique can be followed to design K_2 and P_2. The following theorem defines a LMI constraint that only depends on the system matrices that guarantee that there exist K_1, K_2, P_1 and P_2 such that (9a) and (9b) hold.

Theorem 2
Consider system (1). If there exist matrices W_i, Y_i and a constant γ_i such that the following inequality holds:

$$\begin{bmatrix} \gamma_i I & 0 & B_{i,ni}^{\mathrm{T}} & 0 & 0 \\ * & W_i & W_i A_{ii}^{\mathrm{T}} + Y_i^{\mathrm{T}} B_{ii}^{\mathrm{T}} & W_i Q_i^{1/2} & Y_i^{\mathrm{T}} R_i^{1/2} \\ * & * & W_i & 0 & 0 \\ * & * & * & I & 0 \\ * & * & * & * & I \end{bmatrix} > 0 \tag{10}$$

then (9a) (or (9b), depending on the agent) is satisfied for $P_i = W_i^{-1}$, $K_i = Y_i W_i^{-1}$, and

$$d_i = \gamma_i \max_{x \in \Omega_{ni}} (K_{ni} x)^{\mathrm{T}} K_{ni} x$$

Proof
In [26] it is proved that if (10) holds, then the following constraint is satisfied:

$$F_i((A_i + B_{ii} K_i) x_i + B_{i,ni} v) - F_i(x_i) + L_i(x_i, K_i x_i) - \gamma_i v^{\mathrm{T}} v \leqslant 0 \quad \forall v \tag{11}$$

It follows that (9a) (or (9b), depending on the agent) holds for

$$d_i = \gamma_i \max_{x \in \Omega_{ni}} (K_{ni} x)^{\mathrm{T}} K_{ni} x$$

□

Once the local controllers and the terminal cost functions are fixed, in order to design a distributed MPC scheme that satisfies the assumptions of Theorem 1, one needs to find sets Ω_1, Ω_2 such that (9c)–(9g) hold. In general this is a difficult problem because each of the sets depends on the other.

DOI: 10.1002/oca

The size of the terminal region for agent 1 is determined by the magnitude of the disturbances induced by its neighbor agent 2 and viceversa. We provide next an optimization-based procedure to solve this problem. In order to present the algorithm we need the following definitions.

Definition 1
Given the following discrete-time linear system subject to bounded additive uncertainties:

$$x^+ = Ax + Bu + Dw$$

with $w \in \mathscr{W}$ subject to constraints in the state and the input $x \in \mathscr{X}, u \in \mathscr{U}$ and a linear feedback $u = Kx$; a set Ω is said to be a robust positive invariant set for the system if the following constraints hold:

$$x \in \Omega \rightarrow (A + BK)x + BKx \in \Omega \quad \forall w \in \mathscr{W}$$

$$Kx \in \mathscr{U}$$

$$\Omega \in \mathscr{X}$$

Given system matrices A, B, D, K and the sets $\mathscr{X}, \mathscr{U}, \mathscr{W}$, there exist several methods to find a set Ω that satisfies these constraints, see, for example, [27] for a procedure to find the maximal robust positive invariant and [28] for a procedure to find an approximation of the minimal robust positive invariant. We denote $\Omega(A, B, D, K, \mathscr{X}, \mathscr{U}, \mathscr{W})$ the corresponding maximal robust positive invariant set.

Taking into account that the input of the neighbor agent can be considered as an unknown bounded disturbance, in order to decouple the computation of the sets Ω_1 and Ω_2, we use the following result based on finding a robust positive invariant set for each subsystem:

Theorem 2
Given constants $\lambda_1 \in (0, 1]$ and $\lambda_2 \in (0, 1]$, if the sets defined as

$$\Omega_1 = \Omega(A_1, B_{11}, B_{12}, \mathscr{X}_1, K_1, \lambda_1 \mathscr{U}_1, \lambda_2 \mathscr{U}_2)$$

$$\Omega_2 = \Omega(A_2, B_{22}, B_{21}, \mathscr{X}_2, K_2, \lambda_2 \mathscr{U}_2, \lambda_1 \mathscr{U}_1)$$

are not empty, then constraints (9c) to (9g) are satisfied.

Proof
The theorem is proved taking into account the definition of the operator Ω and that $\lambda_1 \mathscr{U}_1 \subseteq \mathscr{U}_1$ and $\lambda_2 \mathscr{U}_2 \subseteq \mathscr{U}_2$. □

The main idea is that to determine the invariant sets both agents limit their inputs by a factor $\lambda_i \in (0, 1]$ with $i = 1, 2$, hence, the other agent can find the maximal robust positive invariant set with respect to a known bounded disturbance. For example, agent 1 finds the maximal robust positive invariant with respect to a disturbance bounded in $\lambda_2 \mathscr{U}_2$ assuming that its input is bounded in $\lambda_1 \mathscr{U}_1$. Agent 2 does the same. Note that these sets may be empty depending on the value of λ_1 and λ_2. If both sets exists, then they satisfy the stability constraints. In general, an infinite number of possible values of λ_1 and λ_2 such that both sets are non-empty may exist. In order to choose one, we propose to solve the following optimization problem to maximize the feasibility region of the distributed MPC controller:

$$\begin{aligned} \max_{\lambda_1 \in (0,1], \lambda_2 \in (0,1]} \quad & f(\Omega_1 \times \Omega_2) \\ & \Omega_1 = \Omega(A_1, B_{11}, B_{12}, K_1, \mathscr{X}_1, \lambda_1 \mathscr{U}_1, \lambda_2 \mathscr{U}_2) \\ & \Omega_2 = \Omega(A_2, B_{22}, B_{21}, K_2, \mathscr{X}_2, \lambda_2 \mathscr{U}_2, \lambda_1 \mathscr{U}_1) \end{aligned} \tag{12}$$

where $f(\cdot)$ is a measure of the size of a polyhedron (for example, its Chebyshev radius).

DOI: 10.1002/oca

Once matrices K_1, K_2, P_1, P_2 and the sets Ω_1 and Ω_2 are determined, constants d_1 and d_2 can be calculated in order to obtain an estimation of the set in which the closed-loop system is ultimately bounded.

Remark
Theorem 1 guarantees that the closed-loop system is ultimately bounded in a closed region that contains the origin, however, taking into account that the local controller guarantees that each subsystem is input-to-state stable [29] with respect to the input of the neighbor subsystem, asymptotic stability can be proved under some additional assumptions regarding the decrease rate of the norm of the input with respect to the norm of the state as it approaches the origin.

4. EXAMPLES

In this section the theoretical results and the design procedure are illustrated using two different examples. The first example is focused on the controller design procedure. The second controller shows the application of the proposed approach to a supply chain problem. The simulations presented in this paper were performed using Matlab in a computer equipped with a 2.2 GHz Core 2 duo processor and 3 GB of RAM memory.

4.1. Two double integrators with coupled inputs

The system considered is composed of two double integrators with coupled inputs. The first subsystem is defined by the following matrices:

$$A_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B_{11} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad B_{12} = \begin{bmatrix} 0 \\ 0.4 \end{bmatrix}$$

and the second subsystems is defined by:

$$A_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B_{22} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad B_{21} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The state and the input must satisfy the following constraints:

$$\|x_1\|_\infty \leqslant 1, \quad \|x_2\|_\infty \leqslant 2, \quad |u_1| \leqslant 1, \quad |u_2| \leqslant 1$$

The stage cost functions of each agent are defined by $Q_i = I$ and $R_i = 1$ with $i = 1, 2$.

In order to determine the local controllers K_i and the corresponding weight matrices P_i that define the terminal cost function, an LMI problem based on (10) that minimizes the constant γ_i is solved for each agent. The following matrices are obtained:

$$K_i = [-0.2023 \quad -0.9254], \quad i = 1, 2$$

$$P_1 = \begin{bmatrix} 32.6719 & -17.5149 \\ -17.5149 & 54.6366 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 38.4509 & -5.6447 \\ -5.6447 & 50.1686 \end{bmatrix}$$

The last step necessary to apply the proposed algorithm is to determine an invariant region for the two agents, Ω_1 and Ω_2. Different approaches can be used to determine the values of λ_i that maximize the size of the terminal regions. In this example, the terminal region was calculated for a grid with different values of λ_i. The criterion to select the maximum invariant region was the Chebyshev radius of the maximum ball inside the region. The results were $\lambda_1^* = 0.3$ and $\lambda_2^* = 0.5$. Figure 4 shows a 3D plot of the Chebyshev radius as a function of λ_1 and λ_2.

The constants λ_1 and λ_2 define a trade-off between the degree of freedom that the agents have in order to stabilize the system, and the size of the terminal region that determines the size of

DOI: 10.1002/oca

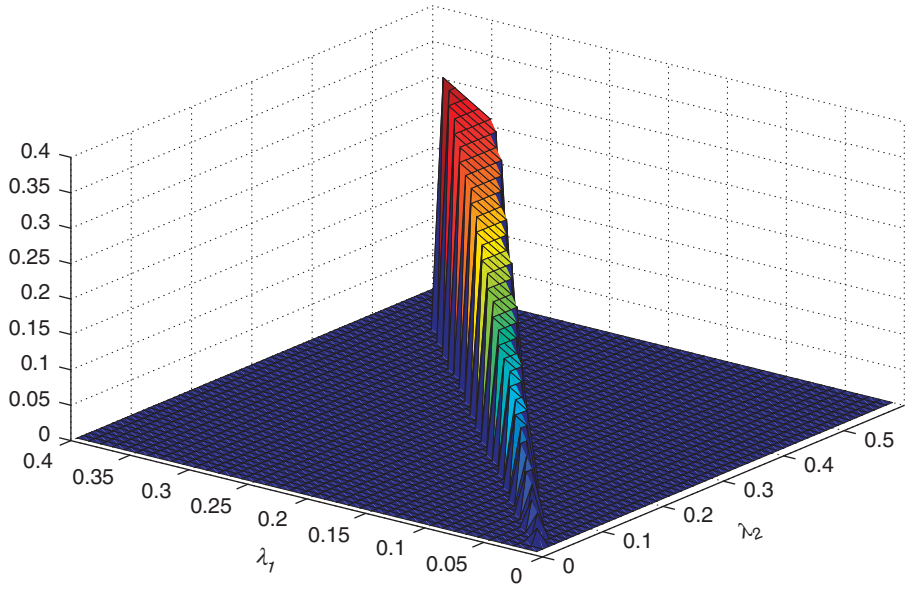

Figure 4. Chebyshev radius of the set $\Omega_1 \times \Omega_2$ for different values of λ_1 and λ_2.

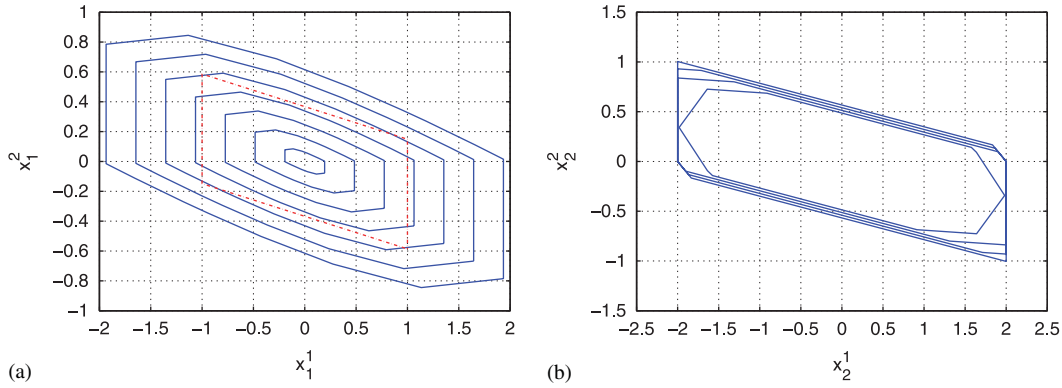

Figure 5. (a) Minimum robust invariant set for agent 1 as a function of λ_2 with λ_1 fixed and (b) maximum robust invariant set for agent 2 as a function of λ_2 with λ_1 fixed.

the disturbance. As λ_2 increases, the set defined by $K_2 x \in \lambda_2 \mathscr{U}_2$ increases. This implies that the set $\Omega_2 = \Omega(A_2, B_{22}, B_{21}, K_2, \mathscr{X}_2, \lambda_2 \mathscr{U}_2, \lambda_1 \mathscr{U}_1)$ becomes larger because the feasibility region of the input is larger, while the set $\Omega_1 = \Omega(A_1, B_{11}, B_{12}, K_1, \mathscr{X}_1, \lambda_1 \mathscr{U}_1, \lambda_2 \mathscr{U}_2)$ has to take into account bigger disturbances and may even cease to be defined (i.e. is empty). This happens when the minimum positive robust invariant set for an uncertainty bounded in $\lambda_2 \mathscr{U}_2$ is not included in the feasibility region defined by $\mathscr{X}_1$ and $K_1 x_1 \in \lambda_1 \mathscr{U}_1$. In Figure 5(a), inner approximations of the minimum positive invariant sets of subsystem 1 for different values of λ_2 and a fixed value of λ_1 are shown. It can be seen that for large values of λ_2, the inner approximation is not contained in the feasibility region of agent 2 (shown in dashed line), and hence, it is empty. In Figure 5(b) the maximum positive invariant set for the same values of λ_1 and a fixed value of λ_2 are shown. It can be seen how the size of the set always increases with λ_2.

In the first time step, a feasible solution for the centralized problem is used as the shifted trajectories. Simulations results are shown in the next figures for the following initial conditions:

$$x_1(0) = \begin{bmatrix} 0.7 \\ -1 \end{bmatrix}, \quad x_2(0) = \begin{bmatrix} 1 \\ 0.8 \end{bmatrix}$$

DOI: 10.1002/oca

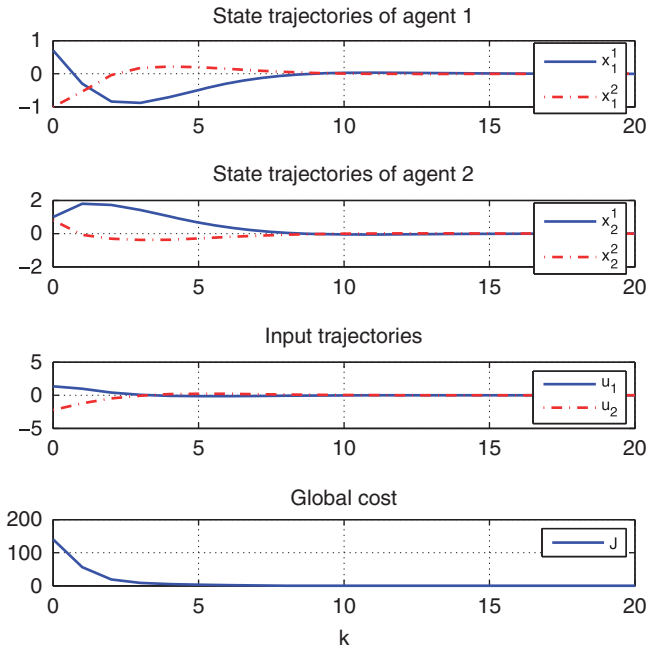

Figure 6. State, input and global cost $J(t)$ trajectories of the double integrators in closed-loop with the proposed controller.

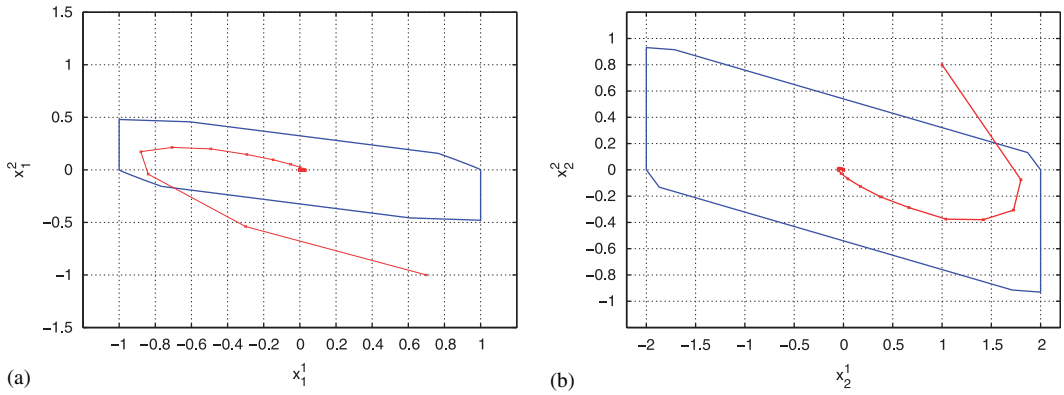

Figure 7. (a) Agent 1 state evolution and (b) agent 2 state evolution.

Figure 6 shows the trajectories of the states of each agent, the inputs and the cost index. Figures 7(a) and 7(b) show the state trajectories of each agent along with its corresponding invariant set.

4.2. Application to a supply chain problem

In this section, we apply the proposed controller to a reduced version of the MIT beer game and compare the performance with other control schemes. The MIT beer game was developed by Jay Forrester in the late 1950s to show his management students how oscillations arise in a supply chain, see, for example [30]. A supply chain is the set of structures and processes used by an organization to provide a service or a good to a consumer. Typically three phenomena take place in supply chains flows: oscillation, amplification and phase lag. Owing to material or informational delays in the flows of the supply chain, they are prone to oscillation; that is, production and inventories overshoot and undershoot the optimal levels. The magnitude of the fluctuations increases as they

DOI: 10.1002/oca

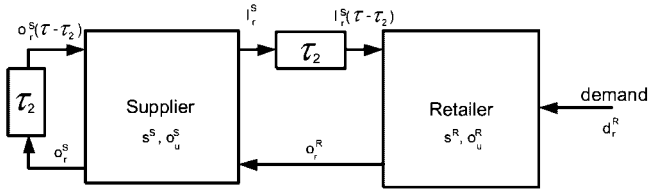

Figure 8. Reduced beer game.

propagate from the customer to the factory, with each upstream stage tending to lag behind the downstream one in what is commonly known as the bullwhip effect.

The original MIT beer game is composed of four agents: retailer, wholesaler, distributor and factory. Customers demand beer from the first one, who orders beer from the wholesaler, who orders and receives beer from the distributor, who finally orders and receives orders from the factory. There are shipping and processing delays at each stage. In [30], the original model and all the difficulties of the corresponding stock management problem are explained in detail. This problem has been widely used in the literature. In particular, in [31] it has been used as application example for a DMPC scheme. The main difference between the proposed scheme and the DMPC proposed in [31] is that in [31] the agents only communicated once and the only information shared was the future input trajectories (a strategy similar to Iter 1).

In this paper, a reduced version of the problem with two agents is considered: the retailer and its supplier, see Figure 8. There is no loss of generality since the structure of the game is regular: there is a cascade of firms, each maintaining and controlling its stock. The continuous time equations for the supplier are [30]:

$$\begin{aligned}
\dot{s}^{\mathrm{S}}(\tau) &= o_r^{\mathrm{S}}(\tau-\tau_2)-o_r^{\mathrm{R}}(\tau-\tau_1)-b^{\mathrm{S}}(\tau)/t_b \\
\dot{o}_u^{\mathrm{S}}(\tau) &= o_r^{\mathrm{S}}(\tau)-o_r^{\mathrm{S}}(\tau-\tau_2) \\
\dot{b}^{\mathrm{S}}(\tau) &= o_r^{\mathrm{R}}(\tau)-o_r^{\mathrm{R}}(\tau-\tau_1)-b^{\mathrm{S}}(\tau)/t_b
\end{aligned} \tag{13}$$

The equations for the retailer are:

$$\begin{aligned}
\dot{s}^{\mathrm{R}}(\tau) &= o_r^{\mathrm{R}}(\tau-\tau_1-\tau_2)+b^{\mathrm{S}}(\tau-\tau_2)/t_b-d_r(\tau-\tau_1)-b^{\mathrm{R}}(\tau)/t_b \\
\dot{o}_u^{\mathrm{R}}(\tau) &= o_r^{\mathrm{R}}(\tau)-o_r^{\mathrm{R}}(\tau-\tau_1-\tau_2)-b^{\mathrm{S}}(\tau-\tau_2)/t_b \\
\dot{b}^{\mathrm{R}}(\tau) &= d_r^{\mathrm{R}}(\tau)-d_r^{\mathrm{R}}(\tau-\tau_1)-b^{\mathrm{R}}(\tau)/t_b
\end{aligned} \tag{14}$$

The super-scripts R, S denote variables from the retailer and the supplier, respectively. Variable $s^i(\tau)$ is the stock level; that is, the number of items available in that stage for shipment downstream. The unfulfilled order of stock $o_u^i(\tau)$ stands for the number of ordered items that the agent is waiting to receive from the upstream stage. The backlog of unfulfilled orders $b^i(\tau)$ accounts for the number of committed items that have to be shipped to the downstream stage. The parameter t_b stands for the average backlog clearance time and introduces a first-order dynamic in the process. The customer demand $d_r^{\mathrm{R}}(\tau)$ represents how many items are demanded by the customers. From a control point of view, it can be seen as a measurable perturbation that has to be rejected in order to maintain the stock and the production at the desired levels. The information flows are assumed to have no time delays and the material flows have a delay modeled by τ_2. A delay for processing the received orders is introduced by means of the parameter τ_1. The manipulated variable at each stage is the order rate o_r^i; that is, the number of items demanded upstream. The supplier demands directly to the factory, which is modeled here as a pure delay. This model is different from other supply chain models, in which each agent has to decide not only what to order downstream, but what to send upstream. In this model of the MIT beer game, items sent to the upstream agent are not a decision variable. They are fixed by the orders received. In particular, items sent and the orders received are related through a first-order system with a delay; that is, the shipment rate $l_r^i(\tau)$

DOI: 10.1002/oca

is defined by the following equations:

$$l_r^{\mathrm{S}}(\tau) = o_r^{\mathrm{R}}(\tau-\tau_1)+b^{\mathrm{S}}(\tau)/t_b$$

$$l_r^{\mathrm{R}}(\tau) = d_r^{\mathrm{R}}(\tau-\tau_1)+b^{\mathrm{R}}(\tau)/t_b$$

These relations have already been taken into account in the model.

The model of the system is defined by the parameters τ_1, τ_2 and t_b. In the simulations done in this paper, we use $\tau_1 = 2d$, $\tau_2 = 1d$ and $t_b = 4d$. In order to obtain a discrete time model of the system, the continuous time model of Equations (14)–(13) is discretized with a sampling time $\Delta = 1d$. Auxiliary states are introduced to take into account the delays. The resulting discrete time linear model is the one used in all the simulations carried out in this section.

In addition, an integrator is added to the controller; that is, the MPC controller decides the increment on the orders made downstream. This implies that the controller evaluates Δo_r^{S} and Δo_r^{R} defined as follows:

$$\Delta o_r^{\mathrm{S}}(t) = o_r^{\mathrm{S}}(t) - o_r^{\mathrm{S}}(t-1)$$

$$\Delta o_r^{\mathrm{R}}(t) = o_r^{\mathrm{R}}(t) - o_r^{\mathrm{R}}(t-1)$$

The state of the model of the first subsystem (the retailer) is given by:

$$x_1(t) = \begin{bmatrix} s^{\mathrm{R}}(t) \\ o_u^{\mathrm{R}}(t) \\ b^{\mathrm{R}}(t) \\ o_r^{\mathrm{R}}(t-1) \\ o_r^{\mathrm{R}}(t-2) \\ o_r^{\mathrm{R}}(t-3) \\ b^{\mathrm{S}}(t) \\ b^{\mathrm{S}}(t-1) \\ d_r(t-1) \\ d_r(t-2) \end{bmatrix}$$

The state of the model of the second subsystem (the supplier) is given by:

$$x_2(t) = \begin{bmatrix} s^{\mathrm{S}}(t) \\ o_u^{\mathrm{S}}(t) \\ b^{\mathrm{S}}(t) \\ o_r^{\mathrm{S}}(t-1) \\ o_r^{\mathrm{R}}(t-1) \\ o_r^{\mathrm{R}}(t-2) \end{bmatrix}$$

It can be seen that both models share some information. In particular, the retailer model needs to keep track of the unfulfilled orders of the supplier, while the supplier model needs to keep track of the orders received of the retailer. The input of the first agent is $u_1 = \Delta o_r^{\mathrm{R}}(t)$ and the input of the second agent is $u_2 = \Delta o_r^{\mathrm{S}}(t)$.

The control objective is to regulate the stock levels and the orders placed by both agents to a desired value. The orders received by the retailer from the external demand forces him to send items

DOI: 10.1002/oca

upstream, and hence to lose stock. These orders can be seen as external disturbances that have to be rejected. To this end, the retailer sends and order for more items downstream. These orders can be seen as an external disturbance for the supplier, which in order to reject this disturbances generates new items. The retailer's stock has to be regulated to a reference value of $r_s^{\mathrm{R}}(t)$. Analogously, the supplier's stock is regulated to a value $r_s^{\mathrm{S}}(t)$. The reference signals for the orders are given by $r_o^{\mathrm{R}}(t)$ for the retailer and by $r_o^{\mathrm{S}}(t)$ for the supplier. Note that in general, the orders references signal should be chosen accordingly with the predicted demand.

To this end, we consider different MPC controllers based on the following cost functions:

$$J_1 = \sum_{k=0}^{N-1} (r_{s,k}^{\mathrm{R}} - s_k^{\mathrm{R}})^2 W_s^{\mathrm{R}} + (r_{o,k}^{R} - o_k^{\mathrm{R}})^2 W_o^{\mathrm{R}} + (\Delta o_k^{\mathrm{R}})^2 W_\Delta^{\mathrm{R}}$$

$$J_2 = \sum_{k=0}^{N-1} (r_{s,k}^{\mathrm{S}} - s_k^{\mathrm{S}})^2 W_s^{\mathrm{S}} + (r_{o,k}^{S} - o_k^{\mathrm{S}})^2 W_o^{\mathrm{S}} + (\Delta o_k^{\mathrm{S}})^2 W_\Delta^{\mathrm{S}}$$

where N is the prediction horizon, the subindex k denotes the k-steps predicted value of a signal and $W_s^{\mathrm{R}}, W_o^{\mathrm{R}}, W_\Delta^{\mathrm{R}}, W_s^{\mathrm{S}}, W_o^{\mathrm{S}}, W_\Delta^{\mathrm{S}}$ are constant weight matrices that define the stage cost. It is important to remark that no terminal cost function is considered in this example.

Note that in order to obtain predictions for the states of the retailer, an estimation of the future demand is needed. We denote the estimated demand as $\hat{d}_r^{\mathrm{R}}(t)$. This signal may be different from the actual demand $d_r^{\mathrm{R}}(t)$ in a given simulation.

The following values were used for the controller parameters:

$$N = 6, \quad W_s^{\mathrm{R}} = 30, \quad W_o^{\mathrm{R}} = 30, \quad W_\Delta^{\mathrm{R}} = 1, \quad W_s^{\mathrm{S}} = 30, \quad W_o^{\mathrm{S}} = 30, \quad W_\Delta^{\mathrm{S}} = 1 \tag{15}$$

For these simulations we have considered that the stocks and orders must be non-negative.

The objective of this section is to compare the performance of different MPC schemes. To this end, we have carried out a set of simulations in four different scenarios for each controller. The first controller considered is the centralized MPC scheme defined by the optimization problem (6). This controller decides both inputs with a single optimization problem based on the full model of the system and the global cost function $J = J_1 + J_2$. In general, the centralized MPC provides the best performance and has the higher computational burden. The second control scheme considered is the proposed distributed MPC controller in which two different agents communicate to find a cooperative solution. In addition, we have considered several controllers based on the iterative controller defined by the optimization problems (7), (8). To avoid the case in which the agents do not reach an agreement, a maximum number of iterations is fixed. Different controllers with a maximum number of iterations of 1, 2, 5 and 10 have been considered. We denote these controllers as Iter1, Iter2, Iter5 and Iter10, respectively. In the case of convergence in this bargaining process, the agents reach a Nash solution from a game theory point of view. None of them consider the cost function of the other agent. For any given input trajectory proposed by its neighbor, the agent evaluated the best possible input for his performance index. By definition, in a situation of equilibrium, this situation constitutes a Nash equilibrium. It is important to remark that in the controller defined by a single iteration (Iter1), the agents do not reach an agreement. They just advice each other about their predicted inputs. Each agent uses this information to estimate the future behavior of the other one. This is not a cooperative scheme because agents do not have a chance to bargain. From the point of view of each agent the other's actions are simply measurable disturbances.

In order to compare the performance of the controllers, four different scenarios have been taken into account. Each scenario is defined by a different initial state, a different retailer demand, and a different demand forecast. All the simulations are done with the discrete time model presented before.

Scenario 1: Both agents begin with 250 items in stock. The demand of the system $d_r^{\mathrm{R}}(t)$ is defined in the following way: during the first 15 days its value is 70. After that, it is set to 130 during 10 days and finally it returns to its initial value for 70 days. The estimated demand $\hat{d}_r^{\mathrm{R}}(t)$ is equal to the real demand.

DOI: 10.1002/oca

Table II. Results for scenario 1.

	J	T_{sim}
Centralized	3.6179e+006	1.4187
DMPC	4.9827e+006	0.8806
Iter1	2.1866e+007	0.5362
Iter2	5.6999e+006	0.7200
Iter5	5.8449e+006	1.7651
Iter10	4.1679e+006	2.2721

Table III. Results for scenario 2.

	J	T_{sim}
Centralized	4.3228e+006	1.44
DMPC	5.3558e+006	0.8890
Iter1	2.1921e+007	0.5008
Iter2	6.0941e+006	0.8771
Iter5	6.2743e+006	1.8354
Iter10	4.7223e+006	2.2581

Table IV. Results for scenario 3.

	J	T_{sim}
Centralized	5.9327e+006	1.427
DMPC	9.6698e+006	0.8265
Iter1	2.2047e+007	0.6887
Iter2	9.0370e+006	0.8287
Iter5	1.0595e+007	1.8209
Iter10	6.2798e+006	1.2073

Scenario 2: Same initial state and estimated demand of the first scenario. In this case, the real demand differs from the estimated demand. At each time step, the real demand is obtained adding a random variable with mean 0 and standard deviation of 15 to the estimated demand.

Scenario 3: Same initial state and real demand of the first scenario. In this case, the estimated demand is supposed to be constant and equal to the latest demand received; that is, the instant demand is extended in time as a forecast.

Scenario 4: Same real and estimated demands of the first scenario. The initial state is below the reference. The retailer has an initial stock of 100 items while the supplier is out of stock.

The results obtained are shown in Tables II–V. In these tables, the total accumulated cost and the total CPU time of each simulation is shown. The total CPU time includes not only the time of solving the different optimization problems but also all the additional computations such as evaluating the system model. In the simulations, the distributed schemes have not been implemented in parallel, and hence the centralized and the distributed controllers have the same computational power. The total simulation time provides an estimate of the computational burden of each of the controllers, in particular, it shows that for this particular example the centralized problem has a low computational burden and that the computational burden of the iterative controllers increases as the maximum number of iterations increases. In addition, for scenario 1 figures are shown for all the different controllers considered.

Some conclusions can be obtained from the preceding experiments. In general, the proposed algorithm provides a performance of the same order of magnitude than the one provided by the centralized MPC which, as expected, has the best results. Regarding the simulation time, it can be seen that for this particular case, the CPU time needed to solve in parallel the sequence of low-order optimization problems is very similar to the time needed to solve the large-scale problem. With respect to the non-cooperative distributed MPC controllers, the proposed distributed scheme

DOI: 10.1002/oca

Table V. Results for scenario 4.

	J	T_{sim}
Centralized	7.2608e+006	1.55
DMPC	8.1302e+006	0.9521
Iter1	2.9982e+007	0.6116
Iter2	1.0397e+007	0.8542
Iter5	1.0444e+007	1.8621
Iter10	9.4679e+006	2.4107

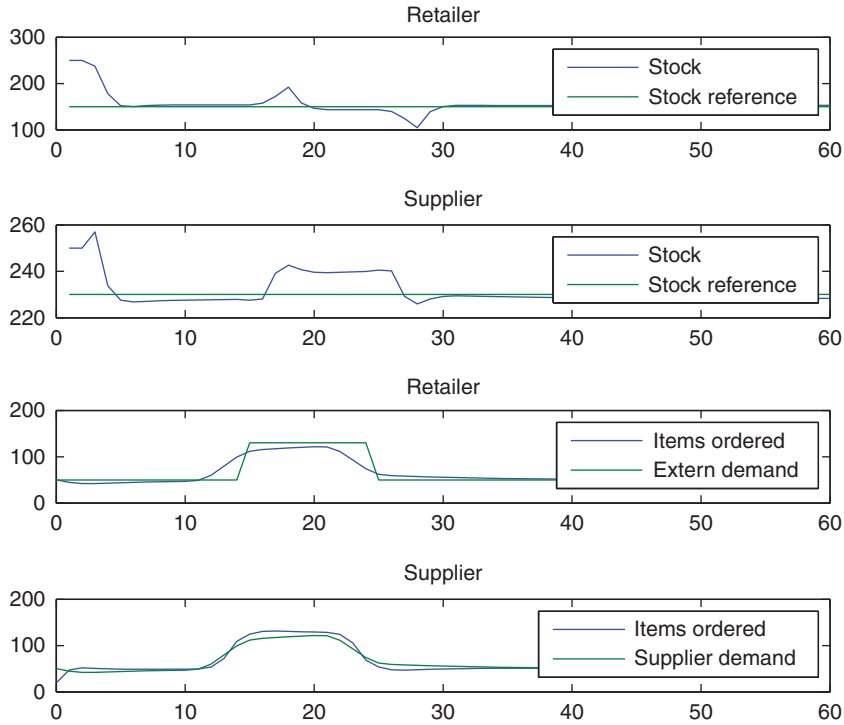

Figure 9. Centralized MPC closed-loop trajectories for scenario 1.

provides a better performance than Iter1, Iter2 and Iter5. The controller Iter10 provides a better performance but needs more communication cycles in order to achieve an agreement. Even in this case, the solution is still a Nash equilibrium, hence, there is no guarantee that it will provide a good overall performance. Note that the iterative controllers' results show that increasing the number of iterations of the bargaining process does not guarantee an improvement in the performance. It can be seen that Iter5 sometimes is worse than Iter2 from the performance point of view.

The simulations demonstrate that the proposed distributed scheme provides a good performance with only two communication cycles because it obtains a cooperative solution; that is, the decision is taken in order to optimize a global performance index. The iterative controllers do not take a cooperative decision and this implies that, in general, the solutions provided are worst. This can be clearly seen in Figures 9–14, which show the closed-loop trajectories of each of the controllers considered for scenario 1. In this scenario, the centralized MPC is able to react in advance to the demand peak, maintaining the stocks close to the references. The trajectories of the proposed distributed MPC scheme show a larger deviation of the stocks from the references, however, these trajectories do not present oscillations as the trajectories corresponding to Iter1, Iter2 and Iter5. Oscillations are a common result of non-cooperative bargaining processes. In this scenario, Iter10 however provides a better response than the proposed DMPC, at the cost of a high computational burden and a large number of communication steps.

DOI: 10.1002/oca

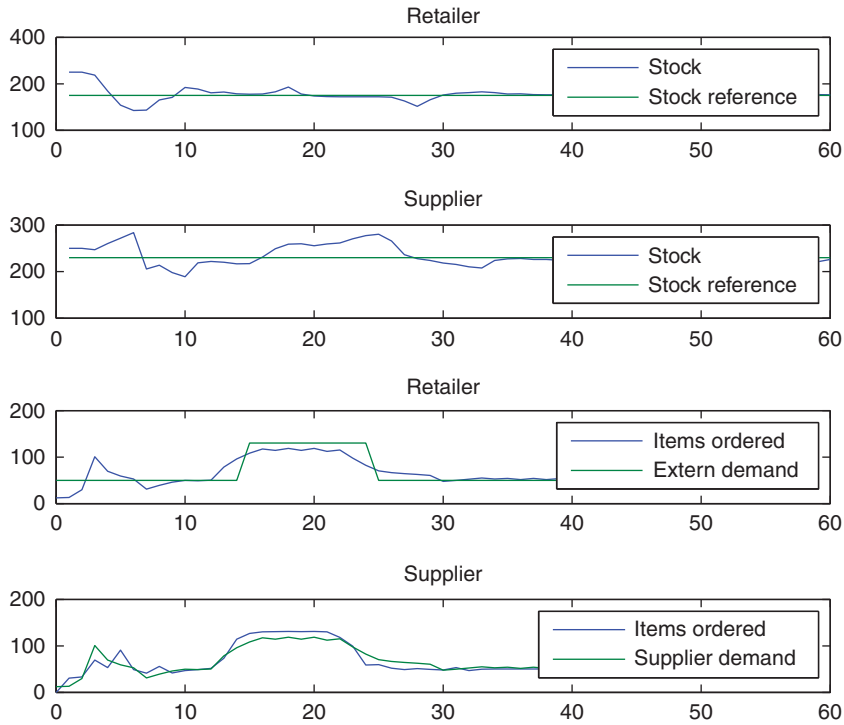

Figure 10. Proposed DMPC closed-loop trajectories for scenario 1.

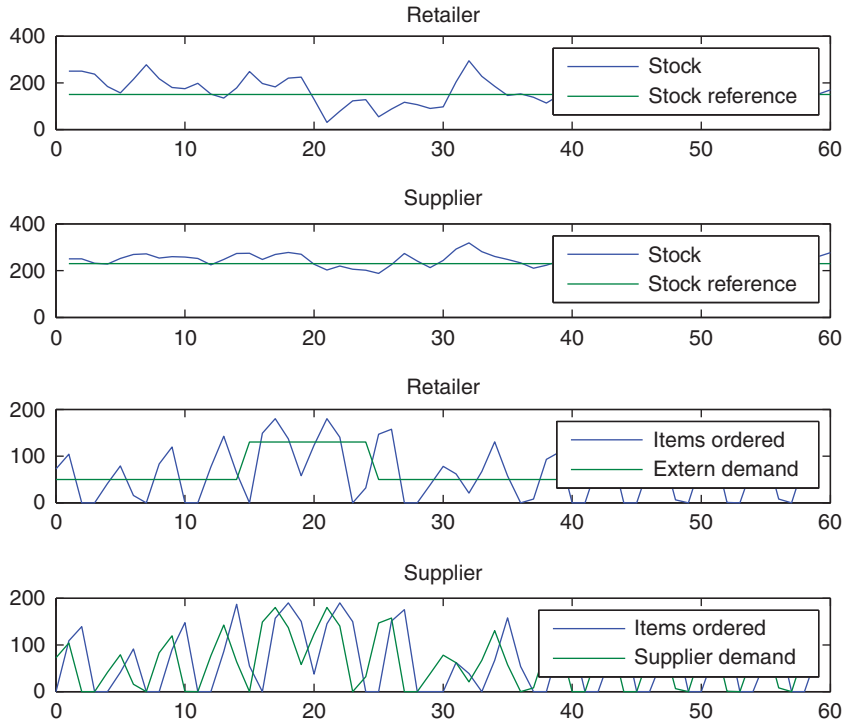

Figure 11. Iter1 closed-loop trajectories for scenario 1.

DOI: 10.1002/oca

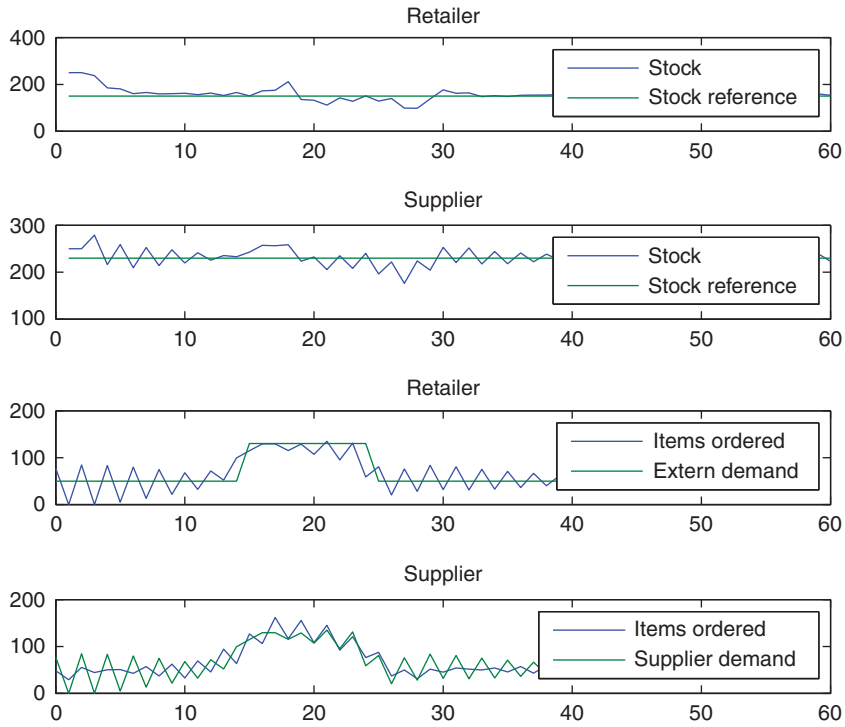

Figure 12. Iter2 closed-loop trajectories for scenario 1.

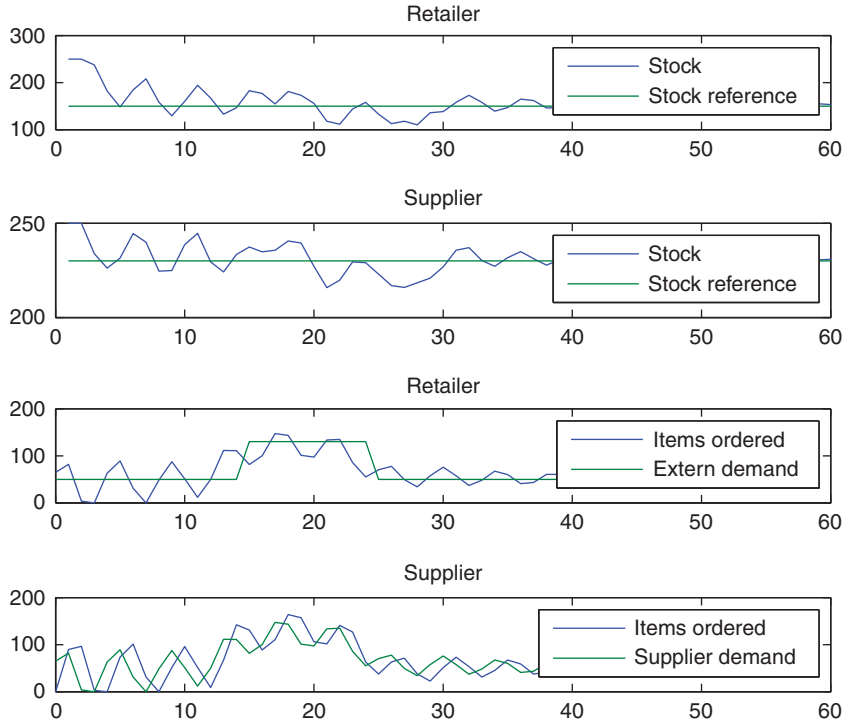

Figure 13. Iter5 closed-loop trajectories for scenario 1.

DOI: 10.1002/oca

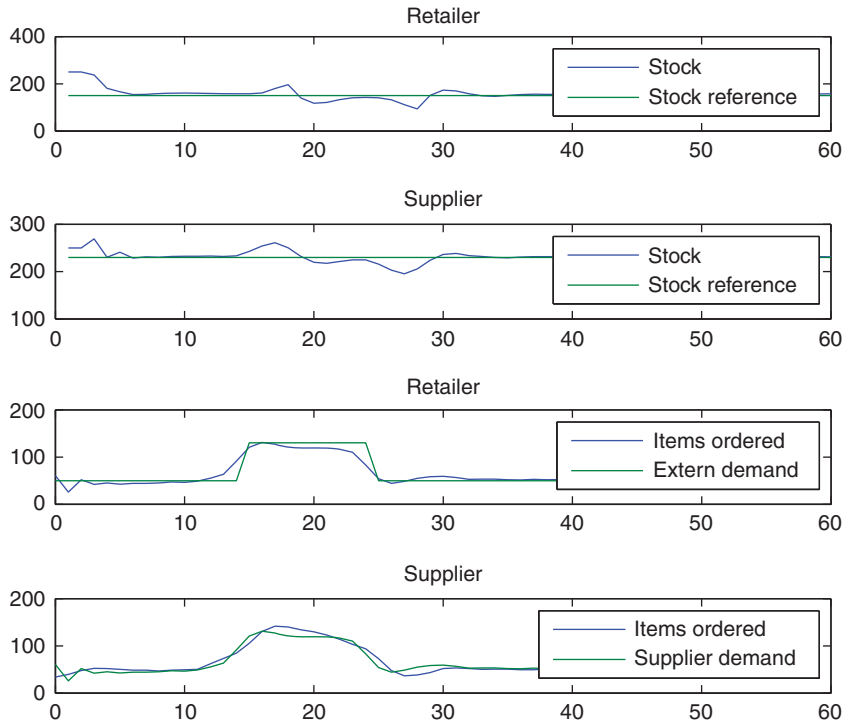

Figure 14. Iter10 closed-loop trajectories for scenario 1.

5. CONCLUSIONS

In this work we have proposed a novel distributed MPC algorithm based on game theory for a class of systems controlled by two agents. The proposed controller only needs two communication steps in order to obtain a cooperative solution to the centralized optimization problem. Each agent solves an optimization problem that only depends on its local model and partial state information. After sharing information about the local cost, the agents choose the solution that yields the best global performance among a set of suboptimal possibilities. The options are suboptimal because each agent has an incomplete view of the system and they propose the best solutions from their point of view. The proposed algorithm has low communication and computational burdens and provides a feasible solution to the centralized problem. In addition, we provide sufficient conditions that guarantee practical stability of the closed-loop system as well as an optimization-based procedure to design the controller so that these conditions are satisfied.

ACKNOWLEDGEMENTS

Financial support from the European Commission, INFSO-ICT-223854 and MEC-Spain, DPI2008-05818, is gratefully acknowledged.

REFERENCES

1. Camacho EF, Bordons C. *Model Predictive Control in the Process Industry* (2nd edn). Springer: London, England, 2004.
2. Negenborn RR, De Schutter B, Hellendoorn H. Multi-agent model predictive control of transportation networks. *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control* (*ICNSC 2006*), Ft. Lauderdale, FL, 2006; 296–301.
3. Yang TC. Networked control systems: a brief survey. *IEE Procedings—Control Theory and Applications* 2006; **152**:403–412.

DOI: 10.1002/oca

4. Neumann P. Communication in industrial automation—what is going on? *Control Engineering Practice* 2007; **15**: 1332–1347.
5. Magni L, Scattolini R. Stabilizing decentralized model predictive control of nonlinear systems. *Automatica* 2006; **42**:1231–1236.
6. Raimondo DM, Magni L, Scattolini R. Decentralized MPC of nonlinear system: an input-to-state stability approach. *International Journal of Robust and Nonlinear Control* 2007; **17**:1651–1667.
7. Camponogara E, Jia D, Krogh BH, Talukdar S. Distributed model predictive control. *IEEE Control Systems Magazine* 2002; **22**:44–52.
8. Rawlings JB, Stewart BT. Coordinating multiple optimization based controllers: new opportunities and challenges. *Journal of Process Control* 2008; **18**:839–845.
9. Li S, Zhang Y, Zhu Q. Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem. *Information Sciences—Informatics and Computer Science* 2005; **170**:329–349.
10. Camponogara E. Controlling networks with collaborative nets. *Ph.D. Thesis*, Carnegie Mellon University, September 2000.
11. Dunbar WB. Distributed receding horizon control of dynamically coupled nonlinear systems. *IEEE Transactions on Automatic Control* 2007; **52**:1249–1263.
12. Richards A, How JP. Robust distributed model predictive control. *International Journal of Control* 2007; **80**: 1517–1531.
13. Jia D, Krogh B. Min–max feedback model predictive control for distributed control with communication. *Proceedings of the American Control Conference*, Anchorage, 2002; 4507–4512.
14. Keviczky T, Borrelli F, Balas GJ. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica* 2006; **42**:2105–2115.
15. Venkat AN, Rawlings JB, Wright SJ. Stability and optimality of distributed model predictive control. *Proceedings of the 44th IEEE Conference on Decision and Control*, *and the European Control Conference ECC 2005*, Seville, Spain, 2005; 6680–6685.
16. Liu J, Muñoz de la Peña D, Ohran B, Christofides PD, Davis JF. A two-tier architecture for networked process control. *Chemical Engineering Science* 2008; **63**:5349–5409.
17. Liu J, Muñoz de la Peña D, Christofides PD, Davis JF. A two-tier control architecture for nonlinear process systems with continuous/asynchronous feedback. *Proceedings of 2009 American Control Conference*, St. Louis, MO, U.S.A., 2009; 133–139.
18. Liu J, Muñoz de la Peña D, Christofides PD. Distributed model predictive control of nonlinear process systems. *AIChE Journal* 2009; **55**:1171–1184.
19. Basar T, Olsder GJ. *Dynamic Noncooperative Game Theory*. SIAM: Philadelphia, 1999.
20. Myerson RB. *Game Theory*, *Analysis of Conflict*. Hardvard University Press: Cambridge, MA, 1997.
21. Mayne DQ, Rawlings JB, Rao CV, Scokaert POM. Constrained model predictive control: stability and optimality. *Automatica* 2000; **36**:789–814.
22. Gilbert EG, Tan KT. Linear systems with state and control constraints: the theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control* 1991; **36**:1008–1020.
23. Kerrigan EC. Robust constraint satisfaction: invariant sets and predictive control. *Ph.D. Thesis*, Department of Engineering, University of Cambridge, U.K., November 2000.
24. Magni L, De Nicolao G, Scattolini R, Allgower F. Robust mpc for nonlinear discrete-time systems. *International Journal of Robust and Nonlinear Control* 2003; **13**:229–246.
25. Alamo T, Muñoz de la Peña D, Limon D, Camacho EF. Constrained minimax predictive control: modifications of the objective function leading to polynomial complexity. *IEEE Transactions on Automatic Control* 2005; **50**(5):710–714.
26. Lazar M, Heemels WPMHD, Muñoz de la Peña D, Alamo T. Further results on robust mpc using linear matrix inequalities. *Proceedings of International Workshop on Assessment and Future Directions of NMPC*, Pavia, Italy, 2008.
27. Kolmanovsky I, Gilbert EG. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, *Theory*, *Methods and Applications* 1998; **4**:317–367.
28. Rakovic S, Kerrigan E, Kouramas K, Mayne D. Robust mpc for nonlinear discrete-time systems. *IEEE Transactions on Automatic Control* 2005; **50**:406–410.
29. Sontag ED. Smooth stabilization implies coprime factorization. *IEEE Transactions on Automatic Control* 1989; **34**:435–443.
30. Sterman JD. *Business Dynamics—Systems Thinking and Modelling in a Complex World*. McGraw-Hill: New York, 2000.
31. Dunbar WB, Desa S. Distributed mpc for dynamic supply chain management. *International Workshop on Assessment and Future Directions of NMPC*, Freudenstadt-Lauterbad, Germany, 26–30 August, 2005.

DOI: 10.1002/oca