applied sciences

MDPI

Article

Deep Reinforcement Learning Agent for Negotiation in Multi-Agent Cooperative Distributed Predictive Control

Oscar Aponte-Rengifo *,†, Pastora Vega † and **Mario Francisco †**

Department of Computer Science and Automatics, Faculty of Sciences, University of Salamanca, Plaza de la Merced, s/n, 37008 Salamanca, Spain
* Correspondence: idu17344@usal.es
† These authors contributed equally to this work.

Abstract: This paper proposes a novel solution for using deep neural networks with reinforcement learning as a valid option in negotiating distributed hierarchical controller agents. The proposed method is implemented in the upper layer of a hierarchical control architecture composed at its lowest levels by distributed control based on local models and negotiation processes with fuzzy logic. The advantage of the proposal is that it does not require the use of models in the negotiation, and it facilitates the minimization of any dynamic behavior index and the specification of constraints. Specifically, it uses a reinforcement learning policy gradient algorithm to achieve a consensus among the agents. The algorithm is successfully applied to a level system composed of eight interconnected tanks that are quite difficult to control due to their non-linear nature and the high interaction among their subsystems.

Keywords: deep reinforcement learning; distributed model predictive control; multi-agent; fuzzy logic

Citation: Aponte-Rengifo, O.; Vega, P.; Francisco, M. Deep Reinforcement Learning Agent for Negotiation in Multi-Agent Cooperative Distributed Predictive Control. *Appl. Sci.* **2023**, *13*, 2432. https://doi.org/10.3390/app13042432

Academic Editors: Eloy Irigoyen, Javier Sanchís and Pedro Cabrera

Received: 31 December 2022
Revised: 30 January 2023
Accepted: 2 February 2023
Published: 14 February 2023

1. Introduction

Model predictive control (MPC) is a successful technique with increasing industrial applications [1]. Nevertheless, the computational requirements that involve a centralized control MPC are, in some cases, unfeasible from a practical point of view due to the large-scale nature of the actual engineering infrastructure. Therefore, distributed model predictive control (DMPC) appears as a possible solution to these problems since a complex problem can always be seen as a set of coupled simpler problems, with a clear structure that represents the global one [2]. In addition, DMPC allows the exploitation of this structure to formulate control laws based on local information and some communication between the agents to achieve global performance and stability. In this case, negotiation among agents is a typical approach. The critical task of solving the negotiation problem in multi-agent DMPC systems comes from designing distributed control protocols based on local information from each agent and its neighbors.

Consequently, the main concern of this negotiation problem is achieving global decisions based on local information. Therefore, strategies that allow agents to change their local decisions online according to changes in the environment or the behavior of other agents to achieve a joint agreement on a global interest is a priority concern. Negotiation to achieve consensus is focused on addressing this problem. An interesting approach is using deep neural networks as a negotiation manager trained in the machine learning method, which offers many advantages and motivates the development of many algorithms for decision making and control, studying all the properties of stability, convergence, and feasibility.

Within artificial intelligence, machine learning techniques have proven to be a powerful tool when using knowledge extracted from data for supervision and the control of

complex processes [3–7]. Technological advances and data availability have boosted this methodology. Deep learning is another field that uses the large amount of data provided by intelligent sensors. This learning is based on neural networks that capture complex behavior, such as [8], where deep learning-based models as soft-sensors to forecast WWTP key features have been developed.

Among them, there is one approach, reinforcement learning (RL) [9] in which an RL agent learns by interacting with the environment in order to determine, based on a policy, what action to take given an environment state, aiming at the maximization of the expected cumulative reward. Ultimately, the RL agent learns the policy to follow when the environment is in a certain state, adapting to variations. In other words, reinforcement learning has as its main objective the training of an RL agent to complete a task in an uncertain environment which it gets to know through the states that the environment assumes due to the actions [10,11].

Valuable surveys about the RL algorithms and their applications can be found in [9,12]. Several online model-free value-function-based reinforcement learning algorithms that use the expected cumulative reward criterion (Q-learning, Sarsa, and actor–critic methods) are described. Further information can also be found in [13–15]. However, several policy search and policy gradient algorithms have been proposed in [16,17].

Many traditional reinforcement learning algorithms have been designed to solve problems in the discrete domain. However, real-world problems are often continuous, making the learning process to select a good or even an optimal policy quite a complex problem. Two RL algorithms that address continuous problems are the deep deterministic policy gradient (DDPG), which uses a deterministic policy, and policy gradient (PG), which assumes a stochastic policy. Despite the significant advances over the last few years, many issues still need to improve the ability of reinforcement learning methods in complex and continuous domains that can be tackled. Furthermore, classical RL algorithms require a large amount of data to obtain suitable policies. Therefore, applying RL to complex problems is not straightforward, as the combination of reinforcement learning algorithms with function approximation is currently an active field of research. A detailed description of RL methodologies in continuous state and action spaces is given in [18].

Recently, in the literature, research works have been published showing that, in particular, RL based on neural network schemes combined with other techniques can be used successfully in the control and monitoring of continuous processes. In [11,19–21] strategies for wind turbine pitch control using RL, lookup tables, and neural networks are present. Specifically, in [11], controlling the direct angle of the wind turbine blades is considered, some hybrid control configurations are proposed. Neuro-estimators improving the controllers, together with the application of some of these techniques in a terrestrial turbine model, are shown. Other control solutions with RL algorithms and without the use of models can be found in [22], among others. In cooperative control, advances have also used RL. Thus, in [7], a cooperative wind farm control with deep reinforcement learning and knowledge-assisted learning is proposed with excellent results. In addition, in [23], a novel proposal is made for high-level control. RL based on RNAs is used for high-level agent negotiation within a distributed model predictive control (DMPC) scheme implemented in the lower layers to control a system level composed of eight interconnected tanks. Negotiation agent approaches based on reinforcement learning have advantages over other approaches, such as genetic algorithms that require multiple tests before arriving at the best strategy [24] or heuristic approaches in which the negotiation agent does not go through a learning process [25].

Negotiation approaches based on reinforcement learning often employ the Q-learning algorithm [26], based on the value function, which predicts the reward that an action will have given a state. This approach is the case in [27], in which a negotiation agent is proposed based on a Q-learning algorithm that computes the value of one or more variables shared between two MPC agents. On the other hand, there are those based on policies, a topic addressed in this work, which directly predict the action itself [28]. One of

these algorithms is the Policy Gradient (PG) [17], which directly parameterizes a policy by tracing the direction of the ascending gradient [16]. In addition, PG employs a deep neural network as a policy approximation function and works in sizeable continuous state and action spaces.

Local MPCs of a DMPC control make decisions based on local objectives. For this reason, consensus between local decisions is necessary to achieve global objectives to improve global index performance. In particular, in this paper, a negotiation agent based on RL manages the negotiation processes among agents of local MPCs to improve a global behavior index of a plant composed of eight interconnected tanks. Given this, the PG-DMPC algorithm within a deep neural network trained by the Policy Gradient (PG) algorithm [29] is proposed and implemented in the upper-level control layer within the control architecture. Therefore, the decision to implement the PG algorithm as a negotiation agent is based on the following advantages: not requiring knowledge of the model, the ability to adapt to handle the uncertainty of the environment, the convergence of the algorithm is assured, and few and easy understanding of tuning parameters.

The remainder of this paper is organized as follows: The problem statement in Section 2. Section 3 presents the RL method, the PG-DMPC algorithm, and DMPC from the low-level control layer. Moreover, Section 4 shows the case study, negotiation framework, the training of the negotiation agent, and performance indexes. Section 5 presents the results. Finally, Section 6 provides conclusions and direction for future work.

Notation

$\mathbb{N}_{0+}$ and $\mathbb{R}_{+}$ are, respectively, the sets of non-negative integers and positive real numbers. $\mathbb{R}^n$ refers to an n-dimension Euclidean space. The scalar product of vectors a, b $\in \mathbb{R}^n$ is denoted as $ab^\intercal$ or a $\cdot$ b. Given sets $\mathcal{X}$, † $\subseteq \mathbb{R}^n$, the Cartesian product is $\mathcal{X} \times \mathcal{Y} \triangleq \{(x,y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$. If $\{\mathcal{X}_i\}_{i\in\mathcal{N}}$ is a family of sets indexed by $\mathcal{N}$, then the Cartesian product is $\times_{i\in\mathcal{N}}\mathcal{X}_i \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_N = \{(x_1, \cdots, x_N) : x_1 \in \mathcal{X}_i, \cdots, x_N \in \mathcal{X}_N\}$. Moreover, the Minkowski sum is $\mathcal{X} \oplus \mathcal{Y} \triangleq \{x + y : x \in \mathcal{X}, y \in \mathcal{Y}\}$. The set subtraction operation is symbolized by $\setminus$. The image of a set $\mathcal{X} \subseteq \mathcal{R}^n$ under a linear mapping $A : \mathcal{R}^n \longrightarrow \mathcal{R}^m is \mathcal{A}\mathcal{X} \triangleq \{\mathcal{A}x : x \in \mathcal{X}\}$.

2. Problem Statement

In implementing DMPC methodologies, local decisions must follow a global control objective to maintain the performance and stability of the system. In this paper, we propose reinforcement learning (RL) in the upper-level negotiation layer of a DMPC control system in search of consensus to achieve global decisions. It requires an upper-level agent negotiator to provide a final solution for each DMPC agent of the lower level in which several control actions are available.

A PG-DMPC (policy gradient-DMPC) algorithm is proposed to carry out negotiation. A deep neural network trained by the policy gradient [29] algorithm has as inputs local decisions of a multi-agent FL-DMPC (fuzzy logic DMPC) [30]. Hence, it provides negotiation coefficients applied to the consensus of the global decisions through its pairwise local decisions to achieve the global goal and stability requirements of the FL-DMPC multi-agent control in the following control architecture (Figure 1).

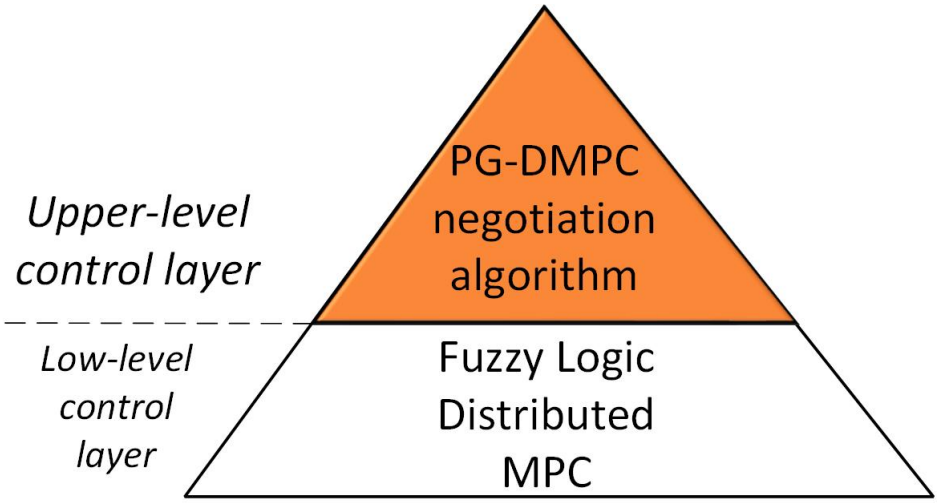


Figure 1. PG-DMPC algorithm proposed in the upper-level PG negotiation control layer within the control architecture.

At each time instant t, a local decision is the control sequence defined as follows:

$$U(t) = \left[u(t), \ldots, u(t+1), \ldots, u\left(t+H_p-1\right)\right]^{\mathsf{T}},$$

where $U(t)$ is the instant control action vector, and H_p is the prediction horizon of the local MPC controllers in the lower layer. The agent i of the MPC with $i \in \{1, 2, \ldots, N\}$ of N agents have local decisions $U_i = \left[U_i^{f_1}(t), U_i^{f_2}(t), \ldots, U_i^{f_{M_i}(t)}\right]$ of M pairwise negotiations in the FL-DMPC lower layer. To achieve a global decision U_i^f, it is proposed to agree as follows:

$$U_i^f(t) = \sum_{m=1}^{M_i} U_i^{f_m}(t) \cdot \alpha_i^m. \tag{1}$$

where α_i^m is the output of the deep neural network considered, like the negotiation coefficients weighing local decisions $U_i^{f_{M_i}}$, to achieve the control objective.

3. Methodology

Given a problem to be solved by the RL method, the environment is modelled as a Markov decision process with state space S, action space A, state transition function $P(s_{t+1}|s_t, a_t)$, with $s \in S$ and $a \in A$, t as the time step, and a reward function $r(s_t, a_t)$. The training of an agent consists of the interaction of the *agent* with the *environment*, for which the *agent* sends an *action* a_t to the *environment*, and it sends back a *state* s_{t+1} and a *reward* r_{t+1} as a qualifier of the a_t according to the *environment* objectives. Consequently, each episode training generates an episode experience composed of the sequence $\{s_t, a_t, r_{t+1}, s_{t+1}, \ldots, s_{T-1}, a_{T-1}, s_T, r_T\}$ with $t \in \{0, 1, \ldots, T\}$ where T is the time horizon.

The PG algorithm selects a_t based on a stochastic policy $\pi_\theta(a_t|s_t)$ that assigns states to actions $(\pi : S \to P(A))$. For this task, the algorithm uses a deep neural network as an approximation function of the stochastic policy $\pi_\theta(a_t, s_t)$. The parameter of the policy optimization θ is performed through the ascent optimization method of the gradient. This optimization aims to maximize for each state in the episode training for $t = 1, 2, \ldots, T-1$, the discounted future reward,

$$G_t = \left\{ \sum_{t=0}^{T} \gamma^t r_t \right\} \tag{2}$$

where γ is a discount factor, and r_t is the reward received at t time step. Therefore, the objective function is

$$G_t \cdot \nabla_\theta \, log \, \pi_\theta(a_t \mid s_t). \tag{3}$$

The score function $\nabla_\theta \, log \, \pi_\theta(a_t \mid s_t)$ allows optimization to be achieved without requiring the dynamic model of the system, optimizing the gradient of the logarithm of $\pi_\theta(a_t|s_t)$ which expresses the probability that the agent selects a_t given a s_t.

The estimate of the discounted future reward G_t made by the Monte Carlo method has a high variance, leading to unstable learning updates, slow convergence, and thus slow learning of the optimal policy. The baseline method takes from G_t a value given from a baseline function $b(s; \varphi)$ to address the variance in the estimate,

$$\hat{A}_t = G_t - b_\varphi(s), \tag{4}$$

where $\hat{A}_t$ is called the advantage function. Consequently, the objective function becomes,

$$\hat{A}_t \cdot \nabla_\theta \, log \, \pi_\theta(a_t \mid s_t). \tag{5}$$

The exploration policy used by PG is based on a categorical distribution, where $\pi_\theta(a \mid s) = \mathbb{P}(a \mid s; \theta)$ is a discrete probability distribution that describes the probability $\pi(s, a\,; \theta)$ that the policy can take action a of a set k actions, with $i \in \{1, \ldots, k\}$, given a state s, with the probability of each action listed separately and is the sum of the probabilities of all the actions equal to zero, $\sum_{i=1}^{k} p_i = 1$.

Figure 2 shows the policy gradient algorithm for the negotiating agent training. The optimization of deep neural networks is based on the input data collected at training by a categorical distribution exploration policy. Model knowledge is not necessary, only local decisions as the *state* and control objectives as the *reward*, a consequence of the *action* of the previous t time step employed to agree among the local decisions and obtain the global decisions of the agents.

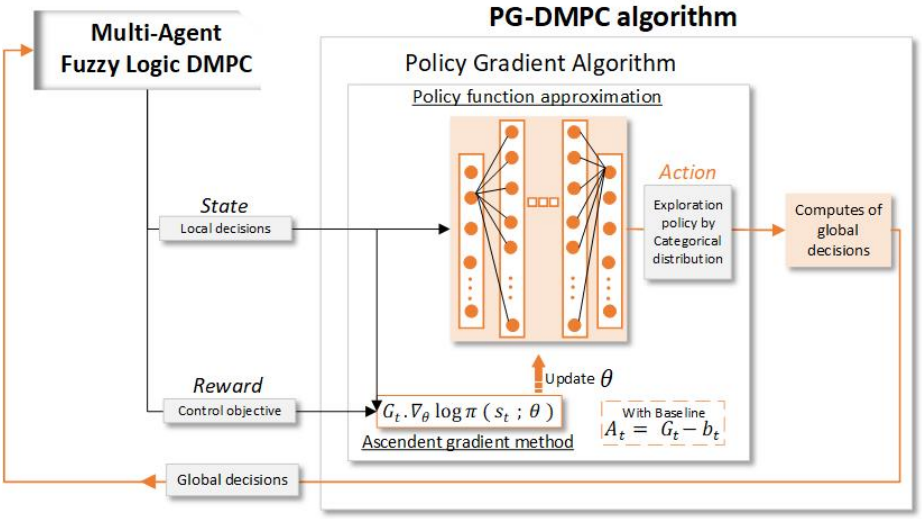


Figure 2. PG optimization scheme for the required negotiation agent in the upper-level control layer in the distributed MPC-based control system.

3.1. PG-DMPC Algorithm

This section details the PG-DMPC algorithm implemented as a negotiating agent required for the upper layer of a distributed MPC control system. The aim of the upper layer PG-DMPC algorithm is to achieve a consensus among all candidate control sequences for each agent to obtain the final control action implemented.

In order to reach consensus, the deep neural network receives the available control sequences for each agent as *state* s_t,

$$s_t = [U_1(t), U_2(t), \ldots, U_N(t)],$$

where $U_i = \left[U_i^{f_1}(t), U_i^{f_2}(t), \ldots, U_i^{f_{M_i}}(t)\right]$, N is the number of agents, $i \in \{1, 2, \ldots, N\}$, and M_i is the total number of pairwise fuzzy negotiations in the lower layer performed by agent i. The total number of control sequences received by the PG-DMPC upper level is defined as $M = \sum_{i=1}^{N} M_i$. The states $s_t \in S$, where S is a compact subspace $S \subseteq \mathbb{R}^{N \cdot M \cdot H_p}$.

The action $a_t \in A$ where $A \subseteq \mathbb{R}^{N \cdot M}$ is a compact set, a_t are some coefficients weighting the control sequences to compute a suitable final control sequence U_i^f to be applied in the plant for agent i, this procedure being the way to achieve consensus among agents in the upper layer. Particularly, the action vector is defined as $a_t = \{\phi_i, \ldots, \phi_N\}$, where $\phi_i = \left[\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{M_{i-1}}\right\}$, $i \in \{1, \ldots, N\}$. Note that for agent i, the number of coefficients provided is M_{i-1} because the last one is obtained as the complement to 1 in order to reduce the computational load, and provided that $\sum_{m=1}^{M} \alpha_i^m < 1$ due to normalization when training the deep neural network,

$$\alpha_i^{M_i} = 1 - \sum_{m=1}^{M_i - 1} \alpha_i^m. \tag{6}$$

Finally, the reward provided by the environment at each time step is defined in Section 4.2.1 because it depends on the particular case study and its global control objective.

Considering states and actions defined above, Algorithm 1 developed negotiation in this paper is described:

Algorithm 1 Proposed PG-DMPC negotiation algorithm for multiples agents

At each time step t for each agent i:

1. The states s_t are taken from the environment (low-level control layer) by the RL upper layer (deep neural network) defined by non-linear function ζ.
2. The deep neural network takes the states as inputs in order to provide the actions a_t which are the negotiation coefficients α_i^m, for $i \in \{1, \ldots, N\}$, $m \in \{1, \ldots, M_{i-1}\}$, as outputs,

$$a_t = \zeta(s_t).$$

   The deep neural network ζ outputs satisfy $\sum_{m=1}^{M_{i-1}} \leq 1$ because it is a constraint considered at training for normalization.
3. The final control sequence U_i^f is obtained, considering the actions provided by ζ, $U_i^f(t) = \sum_{m=1}^{M_i} U_i^{f_m}(t) \cdot \alpha_i^m$.
4. The control sequences for each agent are aggregated to $U_N = \left[U_1^f, U_2^f, \ldots, U_N^f\right]$. $U_N^f(t)$ is computed and compared with the global cost $J_N^f(x_N(t))$ from the previous time t. Otherwise, the previous t control sequences that check stability, $U_N^s(t+1)$, are applied. Hence, the global cost function is

$$\begin{aligned} J_N^f\left(x_N(t), U_N^f(t)\right) = \sum_{k=1}^{H_p - 1} \Big(\|x_N(t+k) - x_{r_N}(t+k)\|_{Q_N}^2 + \\ \|u_N(t+k) - u_{r_N}(t+k)\|_{R_N}^2 \Big) + \|x_N(t+H_P) - x_{r_N}(t+H_P)\|_{P_N}^2, \end{aligned} \tag{7}$$

   with the weighting matrices $Q_N = [Q_i]_{i \in N}$ and $R_N = [R_i]_{i \in N}$ and the terminal cost matrix $P_N = [P_i]_{i \in N}$, x_{r_N} and u_{r_N} the global state and input references is calculated by a procedure to remove offset based on [30].

The proposed PG-DMPC negotiation algorithm is performed in the upper-level control layer from the control architecture, (Figure 1), where the low-level control layer is based on fuzzy logic DMPC and detailed in the next section.

3.2. Distributed MPC

The low-level control layer is a DMPC in which agent $i \in \{1,2,3,4\}$ negotiates with its neighbours in a pairwise manner. Local models are defined as:

$$x_i(t+1) = A_i x_i(t) + B_{ii} u_i(t) + B_{d_i} d_i(t) + w_i(t), \tag{8}$$

where $t \in \mathbb{N}_0+$ denotes the time instant; $x_i \in \mathbb{R}^{q_i}$, $u_i \in \mathbb{R}^{r_i}$, and $d_i \in \mathbb{R}^{d_i}$ are, respectively, the state, inputl and disturbance vectors of each subsystem $i \in \mathcal{N}$, constrained in the convex sets containing the origin in their interior $\mathcal{X}_i = \{x_i \in \mathbb{R}_{q_i} : A_{x,i} x_i \leq b_{x,i}\}$, $\mathcal{U}_i = \{u_i \in \mathbb{R}_{r_i} : A_{u,i} u_i \leq b_{u,i}\}$, respectively; and $A_i \in \mathbb{R}^{q_i x q_i}$, $B_{ii} \in \mathbb{R}^{q_i x r_i}$ and B_{d_i} are matrices of proper dimensions and can be seen in [30].

The vector $w_i \in \mathbb{R}^{q_i}$ represents the coupling with other subsystems j belonging to the set of neighbors $\mathcal{N}_i = \{j \in \mathcal{N} \setminus \{i\} : B_{ij} \neq 0\}$, i.e,

$$w_i(t) = \sum_{j \in \mathcal{N}_i} B_{ij} u_j(t), \tag{9}$$

where $u_j \in \mathbb{R}^{r_j}$ is the input vector of subsystem $j \in \mathcal{N}_i$, and matrix $B_{ij} \in \mathbb{R}^{q_i \times r_j}$ models the input coupling between i y j. Moreover, w_i is bounded in a convex set $\mathcal{W}_i \triangleq \oplus_{j \in \mathcal{N}} B_{ij} \mathrm{U}_j$ due to the system constraints.

The linear discrete-time state-space model is

$$\widetilde{x}_{\mathcal{N}}(t+1) = A_{\mathcal{N}} \widetilde{x}_{\mathcal{N}}(t) + B_{\mathcal{N}} \widetilde{u}_{\mathcal{N}}(t) + B_{\widetilde{d}_{\mathcal{N}}} \widetilde{d}_{\mathcal{N}}(t), \tag{10}$$

where $\widetilde{x}_{\mathcal{N}}$ and $\widetilde{u}_{\mathcal{N}}$ are the state vector and the input vector respectively, $\widetilde{d}_{\mathcal{N}}$ is the disturbance vector, and $A_{\mathcal{N}}$, $B_{\mathcal{N}}$, and $B_{\widetilde{d}_{\mathcal{N}}}$ are the corresponding matrices of the global system.

The low-level control layer performs the FL-DMPC algorithm for multiple agents. Specifically, fuzzy-based negotiations are made in pairs considering the couplings with their neighboring subsystems, which are assumed to hold their current trajectories. To this end, it a shifted sequence of agent i is used, which is defined by adding $K_i x_i(t+N_\mathrm{p})$ on the sequence chosen at the previous time step $U_i(t-1)$:

$$U_i^\mathrm{s}(t) = \begin{bmatrix} u_i(t+1|t-1) \\ u_i(t+2|t-1) \\ \vdots \\ u_i(t+N_\mathrm{p}-1|t-1) \\ K_i x_i(t+N_\mathrm{p}|t-1) \end{bmatrix} = \begin{bmatrix} u_i^\mathrm{s}(t) \\ u_i^\mathrm{s}(t+1) \\ \vdots \\ u_i^\mathrm{s}(t+N_\mathrm{p}-2) \\ u_i^\mathrm{s}(t+N_\mathrm{p}-1) \end{bmatrix}. \tag{11}$$

To make the paper self-contained, a brief description of the FL-DMPC algorithm [30] is presented here (Algorithm 2):

Algorithm 2 Multi-agent FL-DMPC algorithm

At each time step t for each agent i

1. Firstly, agent i measures its local state $\tilde{x}_i(t)$ and disturbance $\tilde{d}_i$.
2. Agent i calculates its shifted trajectory $U_i^s(t)$ and sends it to its neighbors.
3. Agent i minimizes its cost function considering that neighbor $j \in \mathcal{N}_i$ applies its shifted trajectory $U_j^\mathrm{s}(t)$. It is assumed that the rest of the neighboring subsystems $l \in \mathcal{N}_i \setminus j$ follows their current control trajectories $U_l^\mathrm{s}(t)$. Specifically, agent i solves

Algorithm 2 *Cont.*

$$U_i^*(t) = \arg\min_{U_i(t)} J_i\big(x_i(t), U_i(t), U_j^{\mathrm{s}}(t), U_l^{\mathrm{s}}(t)\big), \tag{12}$$

subject to

$$\begin{aligned} &x_i(t+k+1) = A_i x_i(t+k) + B_{ii} u_i(t+k) + \\ &B_{ij} u_j(t+k) + B_{d_i} d_i(t+k) + \sum_{l \in \mathcal{N}_i \setminus \{j\}} B_{il} u_l(t+k), \end{aligned} \tag{13}$$

Constraints

$$\begin{aligned} &x_i(t) = \tilde{x}_i(t),\ i \in \mathcal{N}, \\ &x_i(t+k) \in \mathcal{X}_i, && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &x_i(t+N_{\mathrm{p}}) \in \Omega_i, \\ &u_i(t+k) \in \mathcal{U}_i, && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &u_j(t+k) = u_j^{\mathrm{s}}(t+k), && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &u_l(t+k) = u_l^{\mathrm{s}}(t+k), && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &d_i(t+1) = d_i(t), \\ &d_i(0) = \tilde{d}_i(t), \end{aligned} \tag{14}$$

where set Ω_i is imposed as the terminal state constraint of agent i. Details regarding the calculation Ω_i are given in [30].

4. Agent i again optimizes its cost $J_i(\cdot)$ maintaining its optimal input sequence $U_i^*(t)$ to send to find the input sequence wished for its neighbors $U_j^{\mathrm{w}_i}(t)$. Here, it is also assumed that subsystems l follow their current trajectories. To this end, agent i solves

$$U_j^{\mathrm{w}_i}(t) = \arg\min_{U_j(t)} J_i\big(x_i(t), U_i^*(t), U_j(t), U_l^{\mathrm{s}}(t)\big), \tag{15}$$

subject to

$$\begin{aligned} &x_i(t+k+1) = A_i x_i(t+k) + B_{ii} u_i(t+k) + \\ &B_{ij} u_j(t+k) + B_{d_i} d_i(t+k) + \sum_{l \in \mathcal{N}_i \setminus \{j\}} B_{il} u_l(t+k), \end{aligned} \tag{16}$$

Constraints

$$\begin{aligned} &x_i(t) = \tilde{x}_i(t),\ i \in \mathcal{N}, \\ &x_i(t+k) \in \mathcal{X}_i, && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &x_i(t+N_{\mathrm{p}}) \in \Omega_i, \\ &u_i(t+k) = u_i^*(t+k), && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &u_j(t+k) \in \mathcal{U}_j, j \in \mathcal{N}_i, && k = 0\ \ldots, N_{\mathrm{p}} - 1, \\ &u_l(t+k) = u_l^{\mathrm{s}}(t+k), && k = 0,\ \ldots, N_{\mathrm{p}} - 1, \\ &d_i(t+1) = d_i(t), \\ &d_i(0) = \tilde{d}_i(t), \end{aligned} \tag{17}$$

5. Agent i sends $U_j^{\mathrm{w}_i}(t)$ to agent j and receives $U_i^{\mathrm{w}_j}(t)$.
6. For each agent $i \in \mathcal{N}$, the triple of possible inputs is $\left\{U_i^s(t), U_i^{w_j}(t), U_i^*(t)\right\}$. Since the wished control sequence $U_i^{w_j}$ is computed by neighbor j without considering state constraints of agent i, it is needed to check whether state constraint satisfaction of agent i holds after applying $U_i^{w_j}$. Otherwise, it is excluded from the fuzzification process. Afterwards, fuzzy negotiation is applied to compute the final sequence $U_i^{f_m}$. Similarly, $U_j^{f_m}$ is computed.
7. A resulting pairwise fuzzy negotiation sequence $U_{ij}^{f_m}(t) = \left\{U_i^{f_m}(t), U_j^{f_m}(t), U_l^s(t)\right\}$ is defined based on $U_i^{f_m}$ and $U_j^{f_m}$, assuming that the rest of subsystem $l \in \mathcal{N} \setminus \{i, j\}$ follows their pre-defined trajectories.

Algorithm 2 *Cont.*

8. Agent i sends its cost for the fuzzy and stabilizing control inputs to its neighbours and vice versa. Let us define $U_{ij}^{s} = \left\{ U_i^s(t), U_j^s(t), U_l^s(t) \right\}$; if the condition

$$\sum_{l \in \mathcal{M}_i \mathcal{M}_j \cup \{i,j\}} J_l\left(x_l(t), U_{ij}^{f_m}\right) \leq \sum_{l \in \mathcal{M}_i \mathcal{M}_j \cup \{i,j\}} J_l\left(x_l(t), U_{ij}^{s}\right) \tag{18}$$

holds, then stability is guaranteed, and thus, $U_i^{f_m}(t)$ is sent to the upper control layer. Otherwise, $U_i^s(t)$ is sent.

Hence, the cost function $J_i(\cdot)$ is

$$\begin{aligned} J_i\left(x_i(t), U_i(t), U_j(t), U_l(t)\right) = \\ \sum_{k=1}^{H_p-1} \Big(\|x_i(t+k) - x_{r_i}(t+k)\|_{Q_i}^2 + \\ \|u_i(t+k) - u_{r_i}(t+k)\|_{R_i}^2 + \\ \left\|u_j(t+k) - u_{r_j}(t+k)\right\|_{R_i}^2 + \\ \left\|u_l(t+k) - u_{r_l}(t+k)\right\|_{R_i}^2 \Big) + \\ \|x_i(t+H_P) - x_{r_i}(t+H_P)\|_{P_i}^2, \end{aligned} \tag{19}$$

with Q_i being a semi-positive definite matrix, R_i, P_i being positive-definite matrices, and x_{r_i} and u_{r_i} being the state and input references calculated by a procedure to remove offset based on [30].

4. Case Study

This section describes the coupled eight-tank plant based on the quadruple tank process in which the proposed PG-DMPC algorithm was implemented together with the previously detailed low-level control layer.

4.1. Plant Description and Control Objective

The eight-coupled tanks plant comprises eight interconnected tanks (Figure 3) with four upper tanks (3, 4, 7, and 8) that discharge into the lower ones (1, 2, 5, and 6), which in turn, discharge into sinking tanks. The plant is controlled by four pumps whose flows are divided through six three-way valves γ_v with $v \in \{1, 2, \ldots, 6\}$ manually operated.

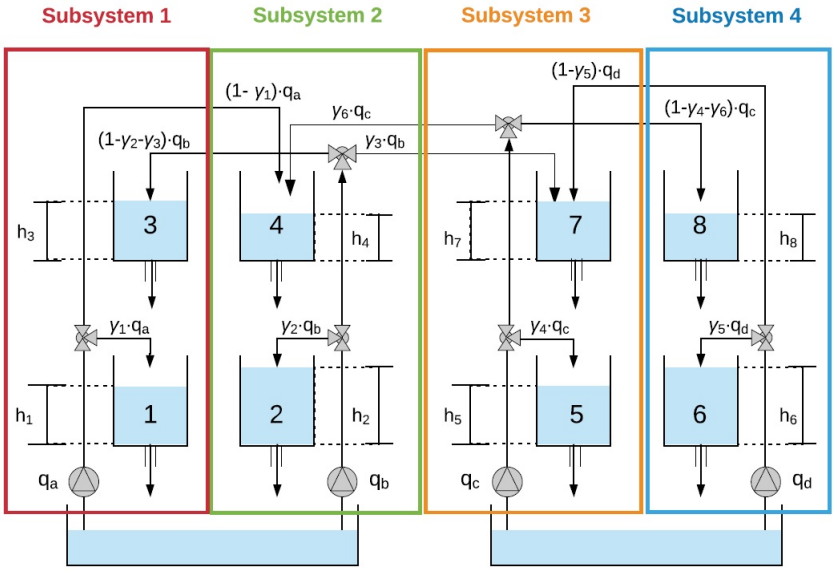


Figure 3. Schematic diagram of the eight-coupled tanks plant with the proposed subsystems.

The system is divided into s subsystems with $s = \{1, 2, 3, 4\}$: Tanks 1 and 3 are part of Subsystem 1; Tanks 2 and 4 form Subsystem 2; Tanks 5 and 7 belong to Subsystem 3; and the rest of the tanks form Subsystem 4. The level of Tank h_n with $n \in \{1, 2, \dots, 8\}$, being the controlled variables the levels h_1, h_2, h_5, and h_6 renamed h_s^c as, $h_1^c = h_1$, $h_2^c = h_2$, $h_3^c = h_5$, and $h_4^c = h_6$ for each agent. The manipulated variable q_s is the flow given by the pump of each Subsystem s. The tank level operating point is established, h_n^0 for $n \in \{1, 2, \dots, 8\}$ as $h_1^0 = 0.10$, $h_2^0 = 0.15$, $h_3^0 = 0.07$, $h_4^0 = 0.03$, $h_5^0 = 0.10$, $h_6^0 = 0.15$, $h_7^0 = 0.025$, $h_8^0 = 0.10$ (units: meters). In addition, the operating point of the flow of the pumps q_s^0 as $q_1^0 = 0.142$, $q_2^0 = 0.421$, $q_3^0 = 0.421$, $q_4^0 = 0.140$ (units : m^3/h).

The aggregated state vector and input state vector are defined as

$$\widetilde{x}_{\mathcal{N}} = \left[h_1^c(t) - h_1^0, h_2^c(t) - h_2^0, h_3(t) - h_3^0, h_4(t) - h_4^0, h_3^c(t) - h_5^0, h_4^c(t) - h_6^0, h_7(t) - h_7^0, h_8(t) - h_8^0\right]^\intercal,$$

$$\widetilde{u}_{\mathcal{N}} = \left[q_1(t) - q_1^0, \dots, q_4(t) - q_4^0\right]^\intercal,$$

The disturbance state vector is defined as $\widetilde{d}_{\mathcal{N}} = \left[d_1(t) - d_1^0, \dots, d_4(t) - d_4^0\right]^\intercal$.

The control objective is to solve a tracking problem to reach the reference $\widetilde{\mathcal{T}}_{\mathcal{N}}$,

$$\widetilde{\mathcal{T}}_{\mathcal{N}} = \left[h_1^{\mathcal{T}}(t) - h_1^0, h_2^{\mathcal{T}}(t) - h_2^0, 0, 0, h_3^{\mathcal{T}}(t) - h_5^0, h_4^{\mathcal{T}}(t) - h_6^0, 0, 0\right],$$

where $h_s^{\mathcal{T}}$ is the target level of the controlled variable h_s^c.

For the upper tank levels, the objective is to keep the operating point despite the disturbance $\widetilde{d}_{\mathcal{N}}$ affecting the system. Additionally, the state and input vectors are constrained by

$$-h_n^0 < \tilde{x}_n(t) \leq 0.08, -q_i^0 < \tilde{u}_i(t) \leq 0.04, \quad \forall n \in \{1, 2, \dots, 8\}, \forall i \in \{1, 2, 3, 4\}.$$

The linear state space models of the global system and subsystems are detailed in [30].

4.2. Negotiation Framework

In this case study, there are four agents. Subsystems 1 and 4 have only one neighbour, Subsystems 2 and 3, respectively, because coupling only exists with them. On the other hand, Subsystems 2 and 3 have two neighbours each. Then, the low-level control layer FL-DMPC provides agent i the following control sequences $U_1 = \left[U_1^{f_1}(t)\right]$, $U_2 = \left[U_2^{f_1}(t), U_2^{f_2}(t)\right]$, $U_3 = \left[U_3^{f_1}(t), U_3^{f_2}(t)\right]$, and $U_4 = \left[U_4^{f_1}(t)\right]$, obtained by performing

pairwise negotiation among agents. In particular, $M_1 = 1, M_2 = 2, M_3 = 2$, and $M_4 = 1$ in (Equation (1)).

Due to the particular configuration of the eight coupled tanks system, negotiation is not required in the upper layer for Subsystems 1 and 4, and therefore $U_1^f(t) = U_1^{f_1}$ and $U_4^f(t) = U_4^{f_1}$. Negotiation is only needed for Agents 2 and 3, and the state is defined as:

$$s_t \equiv [U_2(t), U_3(t)].$$

The actions a_t provided by the deep neural network are:

$$a_t \equiv [\phi_2, \phi_3] = \zeta(s_t; \theta)$$

where $\phi_2 = [\alpha_2^1]$ and $\phi_3 = [\alpha_3^1]$. Note that only one weighting coefficient is obtained for each subsystem because the other ones are directly obtained from:

$$\alpha_2^2 = 1 - \alpha_2^1; \alpha_3^2 = 1 - \alpha_3^1$$

Then,

$$U_2^f(t) = U_2^{f_1}(t) \cdot \alpha_2^1 + U_2^{f_2}(t) \cdot \alpha_2^2 \tag{20}$$

$$U_3^f(t) = U_3^{f_1}(t) \cdot \alpha_3^1 + U_3^{f_2}(t) \cdot \alpha_3^2 \tag{21}$$

Therefore, considering that the final sequence for each agent is $U_i^f = [u_i^f(t), u_i^f(t+1), \ldots, u_i^f(t+H_{p-1})]$, with $H_p = 5$ the prediction horizon for each local MPC controller and manipulated variable, q_s, to be applied to the system is selected as the first value of the sequences, usually an MPC framework:

$$q_1 = u_1^f(1), q_2 = u_2^f(1), q_3 = u_3^f(1), q_4 = u_4^f(1)$$

Figure 4 displays the implemented PG-DMPC algorithm within the control architecture performed in the case study, where PG-DMPC receives local decisions from a FL-DMPC and gives global decisions according to the control objective.

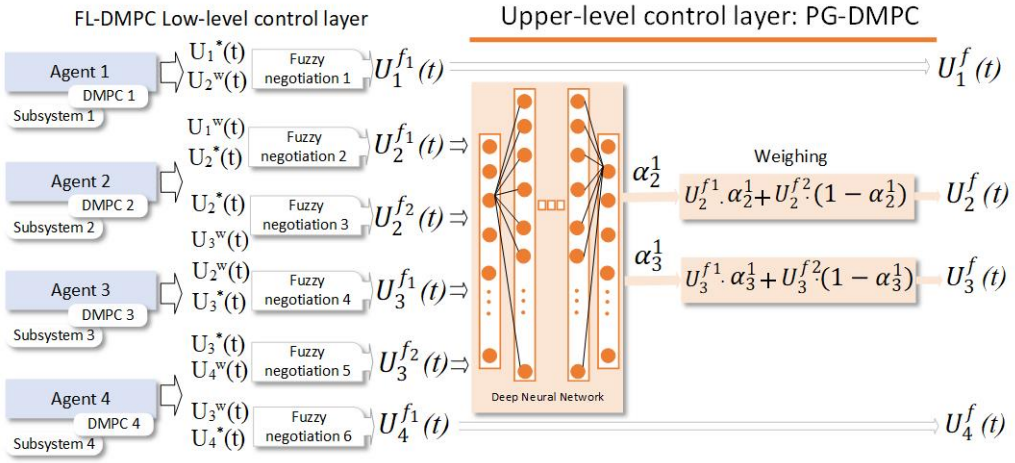


Figure 4. PG-DMPC negotiation algorithm implemented.

4.2.1. Reward

The PG algorithm optimizes weights θ of the deep neural network $\zeta(s_t; \theta)$ used as an approximation function of the stochastic policy $\pi_\theta(s_t, a_t)$ in order to maximize the discounted future reward G_t (Equation (2)). The *reward* function is critical for the proper working of the RL, and it is defined heuristically as:

$$r_t(s_t, a_t) = \begin{cases} z_t(a_t), & if\ \alpha_2^1\ or\ \alpha_3^1 \notin [0,1] \\ 2000, & if\ e_s(t) < e_{b_s}(t)\ \forall s \\ -1000, & Otherwise \end{cases}, \quad if\ \alpha_2^1\ and\ \alpha_3^1 \in [0,1] \tag{22}$$

where $e_s(t) = \left(h_s^{\mathcal{T}}(t) - h_s^c(t)\right)^2$ is the tracking square error, and e_s^b is the error threshold for each subsystem. Note that a positive reward corresponds to a desired situation and negative rewards penalties for the RL.

The maximum reward is given when the tracking error does not exceed the threshold value for any subsystem and the weighting coefficient provided by the deep neural network belongs to $[0, 1]$. A small penalty is given when any error exceeds the threshold, but still, α_2^1 and $\alpha_3^1 \in [0, 1]$. Finally, a large penalty defined by z_t is added if α_2^1 or $\alpha_3^1 \notin [0, 1]$ in order to obtain normalized values for the coefficients:

$$z_t(a_t) = -3000 - \left(\left(\left|\alpha_2^1\right| + \left|\alpha_3^1\right|\right) \times 1000\right). \tag{23}$$

4.2.2. Discrete Action Framework

Discretization is the process of transferring continuous functions, models, variables, and equations to discrete counterparts. In our case, the actions of the trading agent are discrete values, which in Equations (20) and (21) give a discrete weight to the continuous sequences for consensus. More specifically, the discretization is carried out as $\phi_i = 0.05 \cdot x$, whit $x \in \{1, 2, ...20\}$. A discrete action framework is employed to obtain a shorter training time, simpler calculations, a lower computational cost, and faster convergence. According to discrete stochastic policy, each output of the deep neural network is the probability of assigning a combination of discrete values of ϕ_2 and ϕ_3 given a specific state in the inputs.

4.3. Training

This section compares the influence of the baseline method with two approaches, the convergence velocity and stability, during the search for the optimal policy. Table 1 shows the two PG configurations compared and trained.

Table 1. Policy gradient configurations implemented.

Named	Gradient Estimation	Exploration Policy
Conf. 1	Non-baseline and	Categorical distribution
Conf. 2	Baseline and	Categorical distribution

Policy and baseline networks employed as approximation functions are set up to a 0.01 fixed learning rate using the Adam optimizer. The policy deep neural network consists of three fully connected hidden layers, whereas the baseline network uses one fully connected hidden layer.

Training and results are obtained using the MATLAB reinforcement learning toolbox. The two configurations trained for 1000 episodes with the time horizon $T = 120$ and the simulation of the environment with sample time $T_s = 5$. Consequently, each T_s is an RL training time step. The initial randomization of levels taken from the interval $\left[h_n^0 - 0.01,\, h_n^0 + 0.01\right]$ is required and performed. As a remark, although the agents trained with the exploratory policy based on categorical distribution, their policies during validation follow greedy exploration, which selects the action with maximum likelihood.

Figure 5 displays the non-use and use of the baseline method under the categorical distribution exploration policy. Baseline and non-baseline in policy convergence have quite a similar approach time to the high rewards zone. Despite this, the baseline case sets more stability and a faster approach to the high-reward zone.

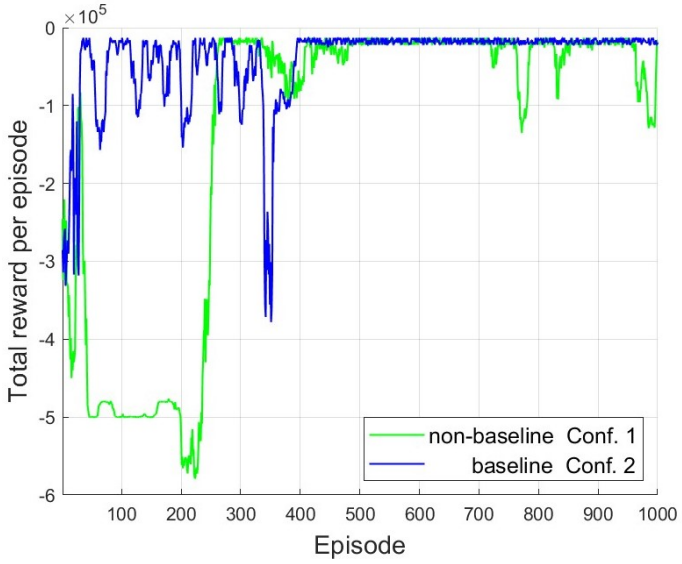


Figure 5. Gradient estimation variance approach by exploration policy based on categorical distribution.

4.4. Performance Indexes

In order to evaluate the proposal, the following performance indexes were established. MPC global cost function,

$$P_e = \sum_{t=0}^{T} J_N(\cdot), \tag{24}$$

where J_N is detailed in Equation (7).

Sum of the integrals of squared errors of controlled levels,

$$ISE = \sum_{s} ISE_s, \tag{25}$$

with

$$ISE_s = \int_0^T \left(h_s^c(t) - h_s^{\mathcal{T}}(t)\right)^2 dt. \tag{26}$$

The sum of the squared differences of the controlled levels $h_{v_{sb}}$ is,

$$e(t) = \sum \left(h_s^c(t) - h_s^{\mathcal{T}}(t)\right)^2 \tag{27}$$

The sum of the pumping energies PE_s for each pump s as the sum of the average of pumping energy over the prediction horizon is proportional to water flows provided by the pumps:

$$PE(t) = \sum_{s} PE_s(t), \tag{28}$$

with

$$PE_s(t) = \frac{0.04}{3600 H_p} \sum_{k=1}^{H_p} q_s(t+k) \tag{29}$$

with $s \in \{1, 2, 3, 4\}$ for all performance indexes.

5. Results

Results of the proposed PG-DMPC are presented in this section. The objective of the PG-DMPC algorithm is to provide consensus among local decisions from an FL-DMPC algorithm to achieve control objectives. The proposal PG-DMPC is compared with fuzzy

DMPC (FL) [30], DMPC using a cooperative game [31], and the centralized MPC. FL and Coop. The game was also implemented in the upper layer. Furthermore, the influence of the baseline method is evaluated. Three validation cases (Table 2) were employed to demonstrate the proper performance of the proposal under the different initial, reference state vectors and disturbance states. The sampling time used in simulations is $T_s = 5$ s.

Table 2. Validation cases.

	Case 1	**Case 2**	**Case 3**
Control problem	regulation	regulation	tracking
Disturbance	none	from $t > 245$ to $t < 305$	none
Reference	steady state	steady state	step from one to another steady state

Case 1: The reference vector is the same as the one used for training, and the state vector of the operating point was considered as the initial state vector, $\widetilde{x}_{\mathcal{N}} = \widetilde{x}^0_{\mathcal{N}}$.

Case 2: The reference vector and the initial state vector are the same as those used in Case 1, but for this case, the disturbances $\widetilde{d}_{\mathcal{N}} = [0.02; 0.02; 0.02; 0.02]$ were included.

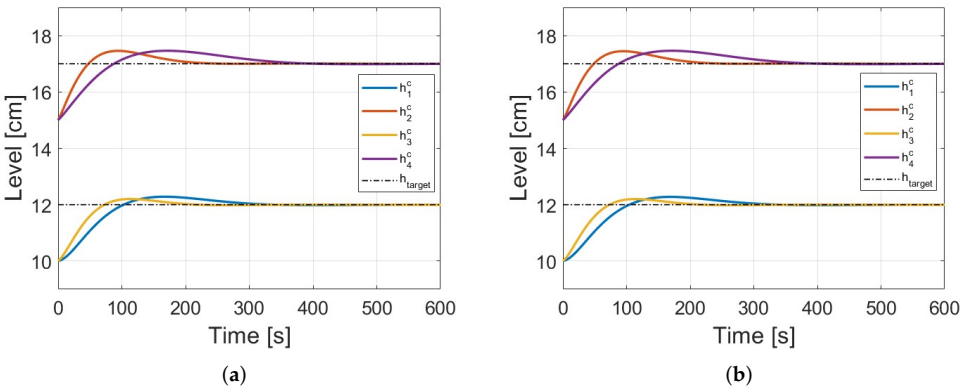
Case 3: The reference vector changes at $t > 200$ from the operation point as initial reference vector to $\widetilde{T}^2_{\mathcal{N}}$ with $h_{\mathcal{T}_1} = h^0_1 + 0.04$, $h_{\mathcal{T}_2} = h^0_2 + 0.03$, $h_{\mathcal{T}_3} = h^0_5 + 0.04$, $h_{\mathcal{T}_4} = h^0_6 + 0.03$. The initial state vector was taken from $\left[h^0_n - 0.03, h^0_n + 0.03\right]$. Note that the initial state vector is taken out of the interval for initial state vectors during training.

Tables 3–5 show the performance of the proposal and the techniques considered in Cases 1, 2, and 3, respectively. In the same order as the validation cases, Figures 6–8 portray the evolution of the state vectors until the reference vector is reached by Conf. 1 and Conf. 2.

Tables 3–5 show that the centralized control has the best results because it has the availability of full plant information for prediction. For validation Case 1, Coop game offers better results in Pe, ISE, and e. Both configurations of the PG-DMPC show an advantage in P_e over the FL, while *ISE* and e display very close values to FL. (Table 3).

Table 3. Case 1: Configurations performance through performance indices values.

Index	**Conf. 1**	**Conf. 2**	**FL-DMPC**	**Coop. Game**	**Centralized MPC**
P_e	0.03594	0.03591	0.03605	0.03040	0.01958
ISE	0.04272	0.04297	0.04285	0.04130	0.03012
e	0.00934	0.00939	0.00937	0.00906	0.00682
$PE \times 100$	0.77505	0.77501	0.77503	0.77505	0.77463



(**a**) (**b**)

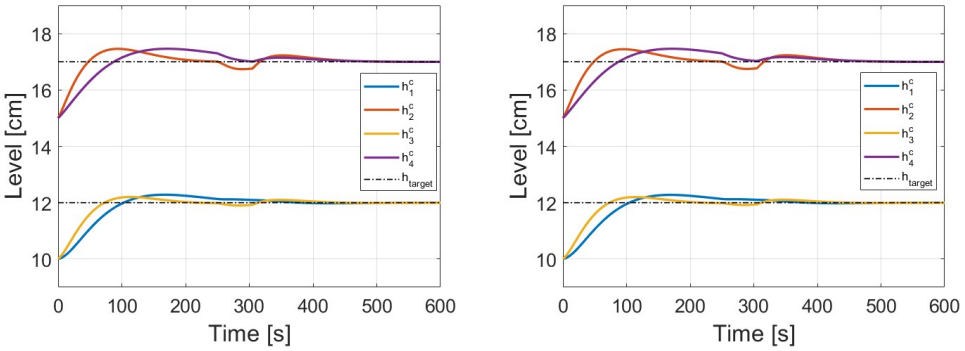
Figure 6. Validation Case 1: Evolution of the levels of the controlled variables: (**a**) Conf. 1 performance; (**b**) Conf. 2 performance.

In validation Case 2, Coop. game shows better P_e than PG-DMPC and FL techniques. In this case, the worst P_e are given by PG-DMPC. On the other hand, these three techniques

show *ISE* and e are very close to each other. Although the measure disturbance was added, an acceptable response was raised by PG-DMPC (Table 4).

Table 4. Case 2: Configurations performance through performance indices value.

Index	Conf. 1	Conf. 2	FL	Coop. Game	Centralized MPC
P_e	0.03745	0.03778	0.03733	0.03132	0.02029
ISE	0.04349	0.04369	0.04351	0.04227	0.03080
e	0.00947	0.00953	0.00950	0.00925	0.00696
$PE * 100$	0.77038	0.77040	0.77028	0.77016	0.76993



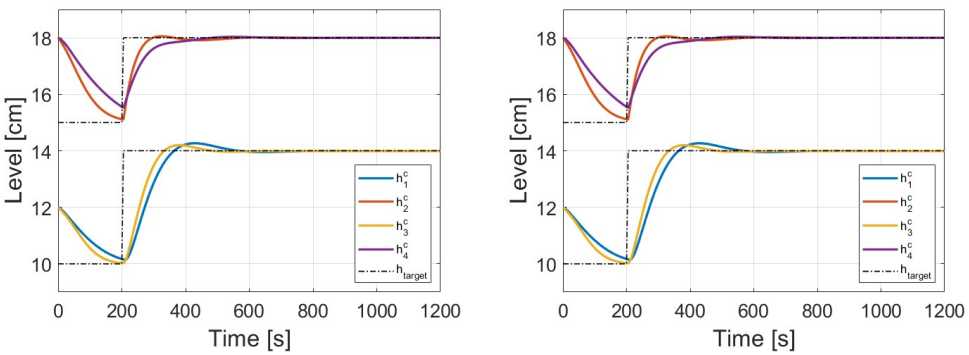
(**a**) (**b**)

Figure 7. Validation Case 2: Evolution of the levels of the controlled variables involving disturbance in the state vector; (**a**) Conf. 1 performance; (**b**) Conf. 2 performance.

Finally, in Case 3, the Coop game and FL show better P_e values. At the same time, the worst of *ISE* and e are given by FL. The two PG-DMPC configurations display similarly in the four indices. The PG-DMPC agent has an acceptable response and is comparable to the other techniques even though it was not trained for reference tracking (Table 5).

Table 5. Case 3: Configurations performance through performance indices values.

Index	Conf. 1	Conf. 2	FL	Coop. Game	Centralized MPC
P_e	0.26797	0.26716	0.27060	0.22242	0.12143
ISE	0.33429	0.33409	0.33465	0.33341	0.20224
e	0.06815	0.06811	0.06823	0.06798	0.04175
$PE \times 100$	1.59417	1.59418	1.59410	1.59450	1.59313



(**a**) (**b**)

Figure 8. Validation Case 3: Evolution of the levels of the controlled variables involving steps: (**a**) Conf. 1 performance; (**b**) Conf. 2 performance.

In addition, Table 6 shows the compliance of the stability evaluation according to Step 4 of Algorithm 1 (see Section 2). For each configuration and validation case, the percentage of total time that cost J_N decreases without using the backup sequence U^s is shown in the table. Both configurations show similar stability, highlighting the stability in Case 2 under Conf. 2.

Table 6. Compliance with stability requirement: % of total time.

Validation Case	Conf. 1	Conf. 2
Case 1	84.2%	83.33%
Case 2	80.84%	82.5%
Case 3	70.41%	70.41%

6. Conclusions

In this paper, the PG-DMPC algorithm was implemented as a negotiating agent in a distributed MPC control system. The proposal generates negotiation coefficients to achieve final control sequences according to the control objectives of DMPC agents. The key of the proposed PG-DMPC algorithm is the deep neural network trained by the RL method. The results obtained are satisfactory, obtaining a successful consensus between the sequences for negotiation in the evaluation cases. It shows similar results to the other techniques with which it was compared even though no prior knowledge of the negotiation problem was required for its training. Therefore, we demonstrated that PG-DMPC is a powerful technique for negotiation problems in multi-agent DMPC control.

PG-DMPC configuration results are similar due to optimal policies converging to the maximum sum of rewards during training. As an advantage, it is reaffirmed that the baseline configurations present a faster convergence to the zone of high reward values. Therefore, the stopping criterion linked to the speed of arrival at the high reward zone would present significant benefits in computational cost and training time compared to the other configurations. Despite using discrete values, the proposed algorithm shows an acceptable tracking of the controlled levels. In addition, it allows any other criteria in possible future rewards to adapt to any control objectives. In fact, PG-DMPC could be implemented as a single control layer of the present architecture without knowledge of the local models of the system, which is necessary for a centralized control that involves higher computational costs.

Author Contributions: Conceptualization, P.V. and M.F.; formal analysis, O.A.-R.; funding acquisition, P.V. and M.F.; investigation, O.A.-R.; methodology, O.A.-R. and P.V.; project administration, P.V. and M.F.; resources, P.V. and M.F.; software, O.A.-R. and M.F.; supervision, P.V. and M.F.; validation, O.A.-R.; visualization, O.A.-R.; writing—original draft, O.A.-R.; writing—review and editing, P.V. and M.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by projects PID2019-105434RB-C31 and TED2021-129201B-I00 of the Spanish Government and Samuel Solórzano Foundation Project FS/11-2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qin, S.; Badgwell, T.A. A survey of industrial model predictive control technology. *Control. Eng. Pract.* **2003**, *11*, 733–764. [CrossRef]
2. Christofides, P.D.; Scattolini, R.; de la Peña, D.M.; Liu, J. Distributed model predictive control: A tutorial review and future research directions. *Comput. Chem. Eng.* **2013**, *51*, 21–41. [CrossRef]
3. Zamarreno, J.M.; Vega, P. Neural predictive control. Application to a highly non-linear system. *Eng. Appl. Artif. Intell.* **1999**, *12*, 149–158. [CrossRef]

4. Huang, J.-Q.; Lewis, F. Neural-network predictive control for nonlinear dynamic systems with time-delay. *IEEE Trans. Neural Netw.* **2003**, *14*, 377–389. [CrossRef] [PubMed]
5. Fernandez-Gauna, B.; Osa, J.L.; Graña, M. Experiments of conditioned reinforcement learning in continuous space control tasks. *Neurocomputing* **2018**, *271*, 38–47. [CrossRef]
6. Sierra, J.E.; Santos, M. Modelling engineering systems using analytical and neural techniques: Hybridization. *Neurocomputing* **2018**, *271*, 70–83. [CrossRef]
7. Zhao, H.; Zhao, J.; Qiu, J.; Liang, G.; Dong, Z.Y. Cooperative Wind Farm Control With Deep Reinforcement Learning and Knowledge-Assisted Learning. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6912. [CrossRef]
8. Cheng, T.; Harrou, F.; Kadri, F.; Sun, Y.; Leiknes, T. Forecasting of Wastewater Treatment Plant Key Features Using Deep Learning-Based Models: A Case Study. *IEEE Access* **2020**, *8*, 184475–184485. [CrossRef]
9. Sutton, R.S.; Barto, A.G. *Reinforcement Learning, Second Edition: An Introduction*; MIT Press: Cambridge, MA, USA , 2018.
10. Neftci, E.O.; Averbeck, B.B. Reinforcement learning in artificial and biological systems. *Nat. Mach. Intell.* **2019**, *1*, 133–143. [CrossRef]
11. Sierra-García, J.; Santos, M. Lookup Table and Neural Network Hybrid Strategy for Wind Turbine Pitch Control. *Sustainability* **2021**, *13*, 3235. [CrossRef]
12. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement Learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
13. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, Cambridge, UK, 1989.
14. Rummery, G.A.; Niranjan, M. *Online Q-Learning Using Connectionist Systems*; Citeseer: Princeton, NJ, USA, 1994; Volume 37.
15. Sutton, L.K.R. Model-based reinforcement learning with an approximate, learned model. In Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems, New Haven, CT, USA, 10–12 June 1996; Volume 1996, pp. 101–105.
16. Baxter, J.; Bartlett, P.L. Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res.* **2001**, *15*, 319–350. [CrossRef]
17. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1999; Volume 12.
18. Hasselt, H.V. Reinforcement learning in continuous state and action spaces. In *Reinforcement Learning*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 207–251.
19. Sierra-García, J.E.; Santos, M. Performance Analysis of a Wind Turbine Pitch Neurocontroller with Unsupervised Learning. *Complexity* **2020**, *2020*, e4681767. [CrossRef]
20. Sierra-García, J.E.; Santos, M. Improving Wind Turbine Pitch Control by Effective Wind Neuro-Estimators. *IEEE Access* **2021**, *9*, 10413–10425. [CrossRef]
21. Sierra-Garcia, J.E.; Santos, M. Deep learning and fuzzy logic to implement a hybrid wind turbine pitch control. *Neural Comput. Appl.* **2022**, *34*, 10503–10517. [CrossRef]
22. Recht, B. A Tour of Reinforcement Learning: The View from Continuous Control. *Annu. Rev. Control. Robot. Auton. Syst.* **2019**, *2*, 253–279. [CrossRef]
23. Aponte, O.; Vega, P.; Francisco, M. Avances en Informática y Automática. Master's Thesis, University of Salamanca, Salamanca, Spain, 2022.
24. Oliver, J.R. A Machine-Learning Approach to Automated Negotiation and Prospects for Electronic Commerce. *J. Manag. Inf. Syst.* **1996**, *13*, 83. [CrossRef]
25. Nguyen, T.D.; Jennings, N.R. Coordinating multiple concurrent negotiations. In Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems, New York, NY, USA, 19–23 July 2004; pp. 1064–1071.
26. Bakker, J.; Hammond, A.; Bloembergen, D.; Baarslag, T. RLBOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, Montreal, QC, Canada, 13–17 May 2019; p. 9.
27. Javalera, V.; Morcego, B.; Puig, V. Negotiation and Learning in distributed MPC of Large Scale Systems. In Proceedings of the 2010 American Control Conference, Baltimore, MD, USA, 30 June–2 July 2010; pp. 3168–3173. ISSN 2378-5861. [CrossRef]
28. Kakade, S.M. A Natural Policy Gradient. In *Proceedings of the Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2001; Volume 14.
29. Williams, R.J. *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*; Kluwer Academic Publishers: Amsterdam, The Netherlands, 1992; p. 28.
30. Masero, E.; Francisco, M.; Maestre, J.M.; Revollar, S.; Vega, P. Hierarchical distributed model predictive control based on fuzzy negotiation. *Expert Syst. Appl.* **2021**, *176*, 114836. [CrossRef]
31. Maestre, J.M.; Muñoz De La Peña, D.; Camacho, E.F. Distributed model predictive control based on a cooperative game. *Optim. Control. Appl. Methods* **2010**, *32*, 153–176. [CrossRef]