Using Deep Multi-Agent Reinforcement Learning Approach in Load Frequency Control

Sergio Rozada, Dimitra Apostolopoulou, and Eduardo Alonso
City, University of London
London, UK EC1V 0HB
Email: {Sergio.Rozada, Dimitra.Apostolopoulou, E.Alonso}@city.ac.uk

***Abstract*—The rise of microgrid-based architectures is heavily modifying the energy control landscape in distribution systems. Decentralised control mechanisms are needed to ensure reliable power system operations. We propose using Reinforcement Learning to implement load frequency control without requiring a centralised authority. Specifically, we approximate the optimal solution using Multi-Agent Deep Deterministic Policy Gradient (MADDPG) at all levels: primary, secondary and tertiary. Generation units are characterized as agents that learn how to maximize their long-term performance by acting and interacting with the environment to balance generation and load, thus restoring frequency. We prove numerically that our Reinforcement Learning methodology can be used to implement the load frequency control in a decentralised way, even when more than one balancing authority is considered.**

***Index Terms*—Reinforcement Learning, MADDPG, Droop Control, Automatic Generation Control, Economic Dispatch, Load Frequency Control.**

I. Introduction

The world of electricity is living a revolution. Distributed generation systems are substituting large electromechanical generators driven by heat engines, e.g., microgrids heavily relying on power electronics devices [1]. In the past, large generation units, such as coal or nuclear plants, produced electricity at large scale. Nowadays, every single house can produce and deliver energy to the network at small scale by different means, such as solar panels or windmills.

This paradigm shift is shaping our understanding of energy and bringing us a whole new branch of opportunities and challenges as well. In this context of decentralization, coordination amongst generators to balance generation and load [2] is crucial. A hierarchical control system is used to meet this objective, i.e., primary, secondary and tertiary frequency control. Secondary and tertiary control layers need a centralizing authority to coordinate the generation. However, solving the load frequency control using a centralized authority when dealing with power electronics system is not feasible [3]. Thus, new approaches should be investigated to find out how frequency control can be developed in an economically decentralised optimal way.

Different approaches has attempted to implement in a decentralised manner the traditional hierarchical control (see, e.g., [4], [5], [3]). Regarding primary control, some examples that can be found in the literature try to adapt and mimic droop control strategies [6]. Droop control is by nature a decentralised algorithm that acts upon each individual generator, so the challenge here is to deal with electronic inverters instead of large turbines. The dynamics may change, but the whole philosophy remains the same: each generator individually modulates its power generation with the variations in the frequency of the network accordingly.

There are also several proposed approaches to implement a non-centralised secondary control, e.g., the centralized averaging PI (CAPI) presented in [7] and distributed averaging PI (DAPI) given in [8]. These algorithms use weighted averages of the frequency as the integral feedback. Despite their theoretical appeal, they suffer from lack of robustness, and their communication demands make them difficult to implement in real-life scenarios [9].

Regarding tertiary control, some efforts have attempted to define microgrid architectures where the communication between nodes enable joint global actions [10]. Nevertheless, as with other approaches, communication is intense between nodes and the system may become too complex.

Multi-Agent Reinforcement Learning (MARL) looks like a promising alternative to implement load frequency control in a decentralised way [11]. In MARL, various software agents learn optimal policies by negotiating, cooperating, and/or competing [12]. In this work we formulate the primary, secondary and tertiary control layers as a MARL problem so that the agents, or generation units, learn to keep generation and load balanced by controlling the energy supply whereas minimizing resources and information exchange.

In this paper: i) the frequency control problem is formalised as a MARL problem; ii) Multi-Agent Deep Deterministic Policy Gradient (MADDPG) is used to implement primary, secondary and tertiary control in a multi-agent problem; and iii) numerical studies are presented to demonstrate that the proposed methodology is a valid alternative to solve load frequency control in a decentralised.

The paper is structured as follows: in Section II the control problem is formulated. In Section III we define our approach to solve the problem. In Section IV numerical results are showed. Conclusions are stated in Section V.

II. PRELIMINARIES

The system frequency indicates if supply and demand are properly balanced. When the generated power exceeds the load the system frequency increases and the other way round, the system frequency decreases if generation is not enough. Thus, controlling the frequency of the system is a classical approach to balance demand and supply [13].

A. Secondary Control ignoring network effects

The frequency control is divided in a hierarchy of three layers: primary, secondary and tertiary control. In primary control, generation and demand are rapidly balanced increasing and decreasing the power output to keep frequency around the nominal set point. This is achieved by a decentralised proportional control mechanism called droop control [14]. Then, a secondary control layer implements an integral control that compensates the steady-state error derived from droop control. Automatic Generation Control (AGC) [15] implements the secondary control layer collecting information from all generation units in a centralized way.

The classical representation model considers a power system with n generators. The deviation of the center of inertia speed from the synchronous speed is denoted by $\Delta\omega$. The nominal frequency is denoted by ω_s; the set of n generators $\mathscr{G} = \{G_1, \ldots, G_n\}$; the total electrical power produced $P_G = \sum_{i\in\mathscr{G}} P_i$, where P_i is the output of generator i; the total load P_L; and the total secondary command $Z_G = \sum_{i\in\mathscr{G}} z_i$, where z_i is the participation of each generator i in AGC. The normalized participation factor of bus load changes ΔP_{L_i} with respect to total system load change ΔP_L is denoted by σ_i, then ρ, which denotes the sensitivity of the losses with respect to the system load is

$$\rho = \sum_{i\in\mathscr{G}} \sigma_i \frac{\partial P_{\text{losses}}}{\partial P_i}. \tag{1}$$

The dynamic behaviour of the system is expressed as:

$$M\frac{d\Delta\omega}{dt} = P_G - (1+\rho)P_L - D\Delta\omega, \tag{2}$$

$$T_G\frac{dP_G}{dt} = -P_G + Z_G - \frac{1}{R_D}\Delta\omega, \tag{3}$$

where $M = \frac{2H}{\omega_s}$, with H being the system inertia constant; D is the load damping; R_D is the droop; and T_G is some time constant (see, e.g., [14]).

B. Secondary Control with network effects

The net interchange with other balancing authority areas should also be addressed in the power balance problem as well. In order to do so, the primary and secondary problem described in (2) and (3) should be reformulated slightly to account for various areas. The general formulation describes the active and reactive power of the k^{th} bus as:

$$P_k = V_k \sum_{m\in\bar{\Omega}} V_m(G_{km}cos\theta_{km} - B_{km}sin\theta_{km}), \tag{4}$$

$$Q_k = V_k \sum_{m\in\bar{\Omega}} V_m(G_{km}sin\theta_{km} - B_{km}cos\theta_{km}) \tag{5}$$

where $\bar{\Omega}_k = \Omega_k \cup \{k\}$, i.e. the set of buses adjacent to k, including k and the variables G_{km}, B_{km} are the real and imaginary parts of the admittance matrix.

We assume that i) bus voltage magnitudes are $|V| = 1$p.u for all nodes, ii) lines are lossless and characterised by their susceptances $B_{ij} = B_{ji} > 0$, iii) reactive power flows do not affect bus voltage phase angles and frequencies, and iv) coherency between the internal and terminal voltage phase angles of each generator so that these angles tend to "swing together", i.e., $\delta_i = \theta_i$.

Then, for each i^{th} synchronous machine, there are three different states are: the rotor electrical angular position δ_i , the difference of the rotor electrical angular velocity from nominal $\Delta\omega_i$, and the mechanical power P_{G_i}. Then, for each i_{th} balancing authority area the dynamics are

$$\frac{d\delta_i}{dt} = \Delta\omega_i, \tag{6}$$

$$M_i\frac{d\Delta\omega_i}{dt} = P_{G_i} - (1+\rho)P_i - D_i\Delta\omega_i, \tag{7}$$

$$T_{G_i}\frac{dP_{G_i}}{dt} = -P_{G_i} + Z_{G_i} - \frac{1}{R_{D_i}}\Delta\omega_i, \tag{8}$$

Then, we include the network effects with DC power flow by relating P_i , the electrical power output of generator i and the angle δ_i as follows:

$$P_i - P_{L_i} = \sum_{j=i, i\neq j}^{n} B_{ij}(\delta_i - \delta_j) \tag{9}$$

where B_ij is the imaginary part of the (i,k) entry of the network admittance matrix, and P_{L_i} the load at bus i.

C. Tertiary Control

The tertiary control layer has to do with the economic aspect of power system operations. This layer establishes the load sharing between the sources so that the operational costs are minimised [16]. Tertiary control is implemented through the economic dispatch, which calculates the optimal operating point in an offline process.

The economic dispatch process is formulated as an optimisation problem, where the objective function that needs to be minimised is the sum of the individual costs of all generation units, $c_i(P_i)$, for $i \in \mathscr{G}$; this is typically a quadratic function that computes the production cost of each generation unit. Here, the constraint is that the system has to keep generation and load balanced; if generation and load are balanced then

frequency is also nominal. The economic dispatch problem may be formulated as:

$$\begin{aligned} &\underset{P_i}{\text{minimize}} \ \sum_{i\in\mathscr{G}} c_i(P_i) \\ &\text{such that} \ \sum_{i\in\mathscr{G}} P_i = (1+\rho)P_L. \end{aligned} \tag{10}$$

III. Reinforcement Learning For Load Frequency Control

In this section we formulate the MARL problem; design the reward function; and select a proper multi-agent actor-critic algorithm that takes into account the fact that state and action spaces are continuous.

A. Formulating the Markov Decision Process

Reinforcement Learning (RL) is an area of Machine Learning strongly related with the notion of software agent [17]. RL studies how software agents interact in an environment to maximize their long-term performance. We use MARL to train a collection of agents how to implement the load frequency control problem in a decentralized way. RL problems are mathematically formalized through Markov Decision Process (MDP) [18], that are defined as the tuple:

$$MDP = \langle S, A, P, R \rangle, \tag{11}$$

where each term is:

- S *or state space*: all possible states where the agent can be in the environment. There are two continuous states in the load frequency control: $\Delta\omega$, the centre of inertia deviation from the synchronous speed denoted and z_i, the current control action of each generator i. These states provides the agent information about the difference between demand and supply and of how much they are contributing to the total generation.
- A *or action space*: all possible actions that each agent take in every state. Our agents-generators can increase or decrease the control action z_i in order to modify the state of the environment.
- P *or probability state transition function*: it defines the dynamics of the environment, modelling the transition between states. If addressing the problem of a single balancing authority, these dynamics are a set of equations derived from (2) and (3):

$$M\frac{d\Delta\omega_{\text{new}}}{dt} = P_{G_{\text{old}}} - (1+\rho)P_L - D\Delta\omega_{\text{old}}, \tag{12}$$

$$T_G\frac{dP_{G_{\text{new}}}}{dt} = -P_{G_{\text{old}}} + Z_{G_{\text{new}}} - \frac{1}{R_D}\Delta\omega_{\text{old}}, \tag{13}$$

$$Z_{G_{\text{new}}} = \sum_{i\in\mathscr{G}} z_{i_{\text{new}}}, \tag{14}$$

$$z_{i_{\text{new}}} = z_{i_{\text{old}}} + \Delta z_i, \tag{15}$$

$$\Delta\omega_{\text{new}} = \Delta\omega_{\text{old}} + \frac{d\Delta\omega_{\text{new}}}{dt}\Delta t, \tag{16}$$

$$P_{G_{\text{new}}} = P_{G_{\text{old}}} + \frac{dP_{G_{\text{new}}}}{dt}\Delta t, \tag{17}$$

where Δz_i is the increase or decrease in power generation by each unit i in $\mathscr{G}$ estimated by each agent. Multi-Agent Deep Deterministic Policy Gradient is used to estimate Δz_i, as described in Section III-B. On the other hand, we can include the network effects by slightly modifying the transition equations based on the (6),(7), (8) and (9) to define the dynamics of the i^{th} synchronous generator as follows:

$$P_i - P_{L_i} = \sum_{j=i, i\neq j}^{n} B_{ij}(\delta_i^{\text{new}} - \delta_j^{\text{new}}), \tag{18}$$

$$M_i\frac{d\Delta\omega_i^{\text{new}}}{dt} = P_{G_i}^{\text{old}} - (1+\rho)P_i - D_i\Delta\omega_i^{\text{old}}, \tag{19}$$

$$T_{G_i}\frac{dP_i^{\text{new}}}{dt} = -P_i^{\text{old}} + Z_{G_i}^{\text{new}} - \frac{1}{R_{D_i}}\Delta\omega_i^{\text{old}}, \tag{20}$$

$$Z_{G_i}^{\text{new}} = \sum_{k\in i} z_k^{\text{new}}, \tag{21}$$

$$z_k^{\text{new}} = z_k^{\text{old}} + \Delta z_k, \tag{22}$$

$$\Delta\omega_i^{\text{new}} = \Delta\omega_i^{\text{old}} + \frac{d\Delta\omega_i^{\text{new}}}{dt}\Delta t, \tag{23}$$

$$\delta_i^{\text{new}} = \delta_i^{\text{old}} + \Delta\omega_i^{\text{new}}, \tag{24}$$

$$P_{G_i}^{\text{new}} = P_{G_i}^{\text{old}} + \frac{dP_{G_i}^{\text{new}}}{dt}\Delta t, \tag{25}$$

- R *or reward function*: it defines a numerical signal or reward expressing the goodness of being in a state and performing an action. The reward function considers two different dimensions in our case: on the one hand, deviation of centre of inertia speed from synchronous $\Delta\omega$ should be as close to zero as possible; on the other hand, operational costs have to be as low as possible.

MARL attempts to learn an optimal policy $\pi : S \mapsto A$ that maximises the cumulative reward, or return. However, the reward is instantaneous and does not address the global nature of the task: i.e., one bad action can lead to an extremely good position from which the agent can obtain a good reward. Thus, action-value functions Q^π are used in RL to express the expected long-term reward achievable from being in an state, taking an action and following a the policy π:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi\left[R_t | s_t, a_t\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} | s_t, a_t\right], \tag{26}$$

where $\mathbb{E}[\cdot]$ is the expectation operator, γ is the discount factor, which expresses the trustness in long-term predictions of Q^π, the cumulative reward achievable in the long run R_t, and the reward r_t at time t. Most algorithms strongly support their learning process in value functions. The most famous and paradigmatic is Q-learning [19].

The action-value function associates a value Q^π to each state-action pair. However, when the number of states and actions is too large, it becomes computationally challenging to estimate them efficiently. Recent work has merged the field of RL with Deep Learning, giving birth to a powerful algorithm called Deep Q-learning (DQN) [20]. This algorithm uses

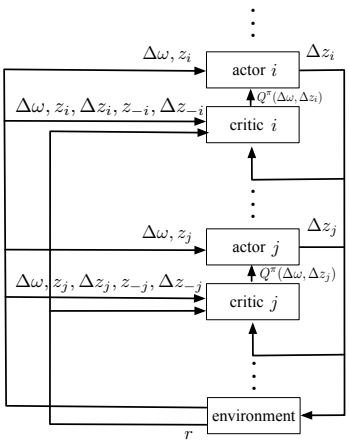

Fig. 1: MADDPG schema in a frequency control scenario.

deep neural networks as parametric function approximators to estimate the action-value function of each state-action pair.

The spectrum of existent algorithms to solve MARL problems is wide. Most of them use game-theoretic approaches to augment Q-learning: i.e., Nash Q-learning or minimax Q-learning [21]. In our problem, state and action spaces are continuous and the interaction of various agents are required. This limits the range of algorithms available in the literature. Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [22] addresses both problems at the same time.

B. Multi-Agent Deep Deterministic Policy Gradient

MADDPG is an actor-critic algorithm. This means that the architecture of each agent, or generation unit, is split into two. First, the actor directly estimates an action while secondly, the critic assess whether the action was good or not by estimating the action-value function of the state-action pair. The Q^π estimated by the critic is used by the own critic and the actor to learn how to behave in the environment. In MADDPG, the critics use central information to teach each actor the dynamics of the environment and the behavior of the rest of the agents as well. In operation, actors only use local information because they learnt how other actors will do.

More specifically, each actor i estimates Δz_i given the state of the environment $\Delta\omega$ and its current z_i. Each critic assess each state-action pair defined by the environment and the actions of all the actors. The critic estimates each state-action action-value that is used during actor's training, as it can be seen in Fig. 1. We denote by Δz_{-i} (Δz_{-j}) the action predicted by all other actors besides i (j) and z_{-i} (z_{-j}) the control action state of all other actors besides i (j).

Deep Recurrent Neural Networks, particularly Long Short-Term Memory Network (LSTM) [23], are used to model each actor and critic. LSTMs implement memory so that previous history is stored and acted upon [24]. The Markov assumption rules MDPs, according to which the current state comprises all information needed to choose an action. However, in the frequency control problem the dynamics are quite complex and the Markov assumption may not hold. Thus, LSTMs help us correcting the violation of the Markov assumption.

The actor network, see Fig. 2, inputs $\Delta\omega$ and z_i and computes Δz_i. The critic network, see Fig. 3, inputs the frequency state of the network $\Delta\omega$, the secondary control action z_i and the change in the action predicted by the actor associated to that critic Δz_i, and the secondary control action z_{-i} the change in the action predicted by all other actors Δz_{-i}, and computes the $Q^\pi(\cdot)$ value of the state-action pair estimated by the actor associated to that critic. Then, both networks have an 100-neuron LSTM that implements memory, and three more 1000, 100 and 50 fully-connected hidden layers.

The design of the reward funcion is critical, since it determines the behavior that agent will learn. Ideally, the reward function incorporates two different components: (i) on the one hand, the reward function must be based in the frequency state of the environment to solve the primary and secondary problem; and (ii) on the ohter hand, the reward function should also address the operational cost associated with the system to solve the tertiary control problem. Taking into account the frequency component in the reward function is straightforward since we set a higher reward for smaller frequency deviations $\Delta\omega$. Next, we need to determine how to define the reward function in order to take into account the cost component. In this regard, we study the case where the cost functions of generators are of the form $c_i(P_i) = a_i P_i^2 + \beta P_i + \gamma_i$ for $i \in \mathscr{G}$ [**?**]. The cost minimisation is part of the tertiary control in the hierarchical control setting; the formulation of which may be found in (10). For quadratic cost functions under no generation limits we may find the optimal solution in an analytical way [13]. The Lagrangian may be written as

$$\mathcal{L}(P_i, \lambda) = \sum_{i\in\mathscr{G}} c_i(P_i) + \lambda \left((1+\rho)P_L - \sum_{i\in\mathscr{G}} P_i \right),$$

were λ is the dual variable of the power balance constraint. The necessary conditions for a minimum are

$$\frac{\partial \mathcal{L}}{\partial P_i} = 0 \Rightarrow \frac{dc_i}{dP_i} - \lambda = 0 \Rightarrow 2a_i P_i + \beta_i = \lambda, \forall i \in \mathscr{G}. \quad (27)$$

The solution to the problem above defines the base point operation of tertiary control. We now define with the aid of

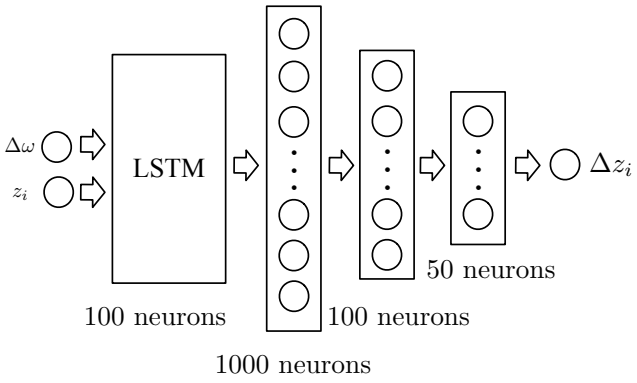

Fig. 2: Architecture of the MADDPG actor.

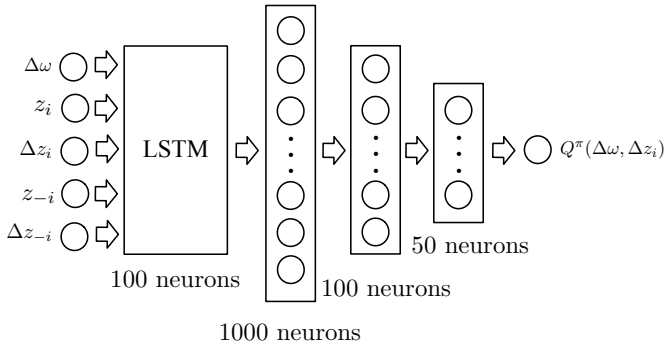

Fig. 3: Architecture of the MADDPG critic.

participation factors how would a generator participate in a load change in order that the new load be served at the most economic operating point. We start from a given base point λ_0 as found from (27). Assume the change in load is ΔP_L; the system incremental cost moves from λ^0 to $\lambda^0 + \Delta\lambda$. For a small change in power output on unit i ΔP_i we have

$$\Delta\lambda \approx \frac{d^2c_i}{dP_i^2}\Delta P_i \Rightarrow \Delta P_i = \frac{\Delta\lambda}{\frac{d^2c_i}{dP_i^2}}, \forall i \in \mathscr{G}. \tag{28}$$

Thus we wish that each generator i changes its output so the following holds

$$\Delta\lambda = \frac{d^2c_i}{dP_i^2}\Delta P_i = \frac{d^2c_j}{dP_j^2}\Delta P_j, \forall i, j \in \mathscr{G}, \tag{29}$$

i.e., for each generator the change in the action Δz_i, $i \in \mathscr{G}$ we wish that

$$\left|\Delta z_i \frac{d^2c_i}{dP_i^2} - \Delta z_j \frac{d^2c_j}{dP_j^2}\right| = 0, \forall i, j \in \mathscr{G}. \tag{30}$$

We construct two conditions that will be used in the formulation of the reward function. The first condition is:

$$C1 : \Delta\omega < \epsilon_1,$$

where ϵ_1 is some selected tolerance; this condition ensures that r will reward actions that help in frequency restoration. The second condition is:

$$C2 : \frac{\sum_{i\in\mathscr{G}}\sum_{j\in\mathscr{G}, j>i}\left|z_i\frac{d^2c_i}{dP_i^2} - z_j\frac{d^2c_j}{dP_j^2}\right|}{(n-1)!} < \epsilon_2,$$

where ϵ_2 is some selected tolerance; this condition ensure that r will reward actions that follow the cost efficient path.

When only the primary and secondary control problems need to be solved, the reward function may be formulated using $C1$ as

$$r = \begin{cases} d, & \text{if } C1 \\ 0, & \text{otherwise} \end{cases}, \tag{31}$$

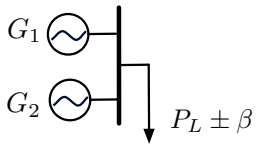

Fig. 4: Test case: two generators and a load; training case: random load variation.

Nominal frequency	$f^{\text{nom}} = 50$ Hz
Initial operating point	$P_1 = 1.5$pu, $P_2 = 1.5$pu
Inertia parameter	$M = 0.1$pu
Droop	$R_D = 0.1$pu
Load damping	$D = 0.0160$pu
Generator dynamics time constant	$T_G = 30$s
Cost function generator 1	$c_1 = 2P_1^2$ [£/pu]
Cost function generator 2	$c_2 = P_2^2$ [£/pu]

TABLE I: Test case 1 data; pu refers to 100 MVA base power reference.

where d is a constant. On the other hand, by taking these two conditions into account we may formulate a general form reward function to solve the tertiary control problem as

$$r = \begin{cases} d_1, & \text{if } C1 \text{ and } C2 \\ d_2, & \text{if } C1 \text{ or } C2 \\ 0, & \text{otherwise} \end{cases}, \tag{32}$$

where d_1, d_2 are constants with $d_2 < d_1$. This reward function both helps in frequency restoration as well as performs it a cost efficient way is rewarded higher that if either purposes are met.

IV. Numerical Results

We validate the MARL methodology using two test systems. On the one hand, we formulate the reward function and present the results of the tertiary control problem for one single balancing authority area. We demonstrate that the generators are able to restore the system frequency back to nominal and operate at a point close to optimal when a change of load occurs in a decentralised way. On the other hand, we formulate the reward function and present the results of the primary and secondary control problem for two balancing authority areas exchanging generation between them.

A. Tertiary control in a single balancing authority area

The test case designed to test the performance of the tertiary control in a single balancing authority area comprises of two generation units or agents that interact with a load. The configuration of the system during training can be found in Fig. 4. The parameters of the environment can be found in Table I. In each episode, or training simulation, the load varies randomly around a nominal set point. The load varies as $P_L \pm \Delta P_L = 3 \pm \beta$ pu, where β follows a uniform distribution.

The reward function has been derived following (32). We set $\epsilon_1 = 0.05$pu, $\epsilon_2 = 0.2$pu, $d_1 = 200$, and $d_2 = 100$. Thus we have the two conditions:

$$C1 : \Delta\omega < 0.05,$$

and

$$C2 : |2z_1 - z_2| < 0.2.$$

Taking these two conditions into account we may formulate the reward function as

$$r = \begin{cases} 200, & \text{if } C1 \text{ and } C2 \\ 100, & \text{if } C1 \text{ or } C2 \\ 0, & \text{otherwise} \end{cases} . \tag{33}$$

The reward function is used only during the training period. In the operation phase, the actors interact with the environment without experiencing any reward. Agents only observe the frequency of the system and its own control action z. They have learnt during training how behave according to the evolution of the environment to balance supply and demand while optimizing operational costs. In the test, we change the load by by 0.15pu and then, we observe how the agents restore the system frequency.

We can observe in Fig. 5 the cumulative reward obtained by the agents. The agents can obtain $20,000$ at maximum per episode: i.e., the maximum reward per step is 200 and the number of steps per episode is 100. The agents learn how to obtain higher rewards as episodes goes on. If that was not the case the cumulative reward function would oscillate around small values near zero.

In Figs. 6, 7 we see how the agents restore the frequency back around the nominal set point, thus balancing supply and demand. Actors learn how to balance generation and demand without exchanging information. The agents have learnt that keeping $\Delta\omega$ close to 0 is the key to obtain high rewards. They have learnt how to perform the primary and secondary control.

In order to test the optimality of the solution provided by the proposed approach in terms of cost, we need to calculate the optimal point when the load in the system, as described in Fig. 4, is $P_L = 3.15$pu. By solving the economic dispatch problem as given in (10) we have $P_1 = 1.05$pu and $P_2 = 2.10$pu. In Fig. 8, 9, the behaviour of each generator output and the associated is depicted. We may see how they both converge to a set value. It can be observed that the agents intuit how much they have to produce to be around the optimal solution. As seen in Fig. 8, the control action of agent one stabilises around a set point that is approximately half of the control action of agent two. It is not exactly the optimal solution (slightly above half the production: 60%), but they get an intuition of what to do while keeping load and supply balanced in a fully decentralised way. The performance of the agents is determined by what actions they learn during training that lead to high rewards. Thus, the reward function is the main tool to show each agent what they have to do. The reward function defined in (34) builds a reward combining two different dimensions: cost and frequency. This means

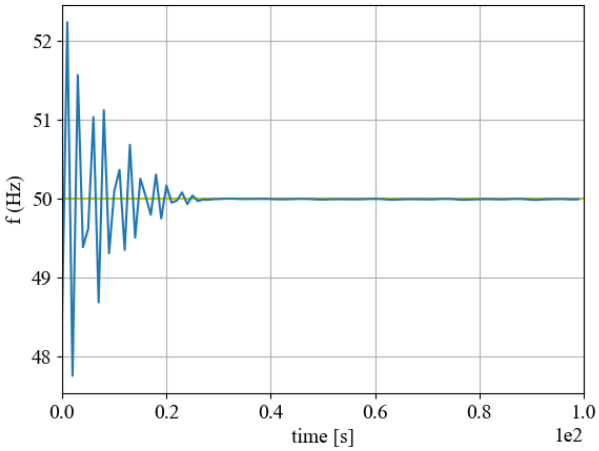

Fig. 6: System frequency after change in load by 0.15pu.

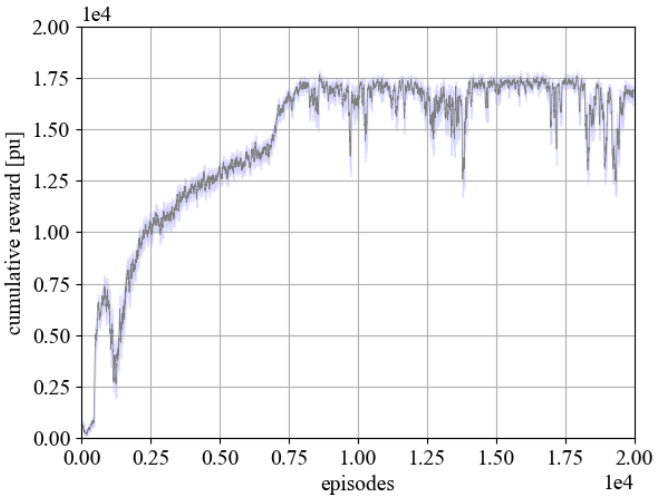

Fig. 5: Smoothed cumulative reward per episode with 95% confidence levels.

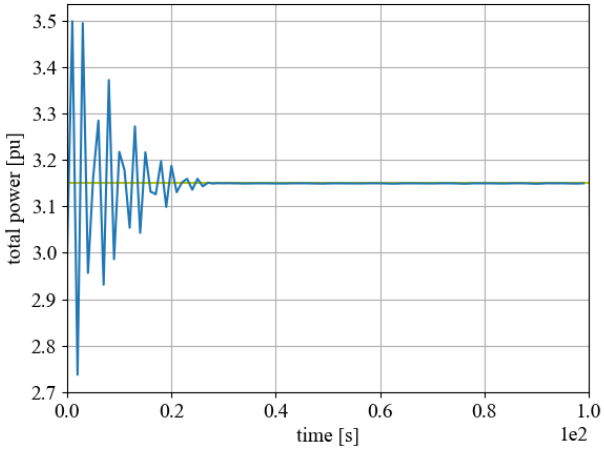

Fig. 7: Total power after change in load by 0.15pu.

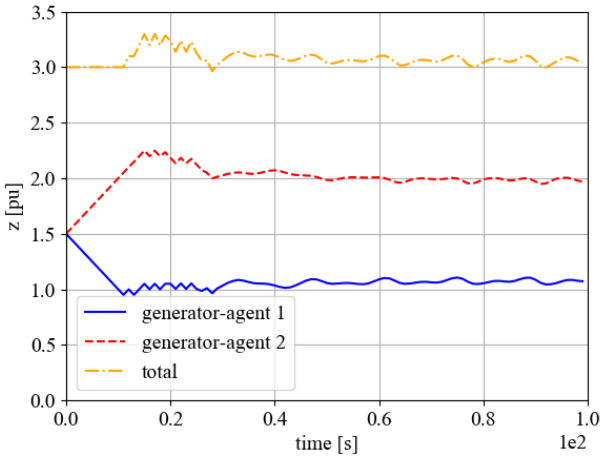

Fig. 8: Generators' output after change in load by 0.15pu.

that the reward function can show various maxima depending on the combination of both reward dimensions. The agents learn by trial and error a behavioural heuristic to obtain high rewards, but they may converge to a local optimum that may be different from the global one. An improvement of the reward function (34) could help the agents to improve the results showed here and to get closer to the optimum solution.

Here, instead of solving the economic dispatch to obtain the optimal operating point, we are proposing a MARL framework to infer the production costs and the necessity of balancing demand and supply from the reward function and enclose this information in the behaviour of the actors. The benefit of the proposed approach is that these agents can act in real time in a decentralised way. Once trained, they do not need to centralise information at all. Dynamics are embedded in the agents that only use local information.

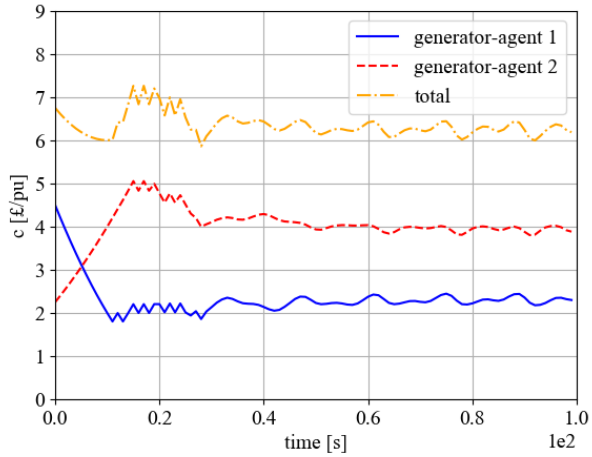

Fig. 9: Generators' cost after change in load by 0.15pu.

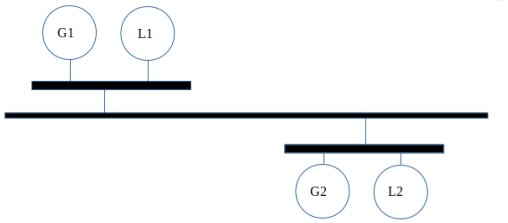

Fig. 10: Test case: two balancing authorities: each one has a generator and a load.

Nominal frequency	$f^{\text{nom}}_{area_1}, f^{\text{nom}}_{area_2} = 50$ Hz
Initial operating point	$P_1 = 1.5$pu, $P_2 = 1.5$pu
Inertia parameter	$M_1 = 0.1$pu, $M_2 = 0.15$pu
Droop	$R_{D_1} = 0.1$pu, $R_{D_2} = 0.08$pu
Load damping	$D_1 = 0.0160$pu, $D_2 = 0.0180$pu
Generator dynamics time constant	$T_{G_1}, T_{G_2} = 30$s

TABLE II: Test case 2 data; pu refers to 100 MVA base power reference.

B. Primary and Secondary accounting for network effects

Analogously, we have designed a the test case to test the performance of the proposed solution on the primary and secondary control problem when multiple balancing authority areas are present. There are two balancing authority areas, each one composed by one generation unit or agents that interact with a single load. The configuration of the system can be found in Fig. 10. The parameters of the environment can be found in Table II. In each training episode, the load varies around a nominal set point randomly. The modification is indicated by $P_L \pm \Delta P_L = 3 \pm \beta$ pu, where β follows a uniform distribution.

In this case, as long as we are not concerned about tertiary control here, the reward function has been derived following (31). Setting $\epsilon_1 = 0.05$pu and $d = 200$. Thus we can use the condition $C1 : \Delta\omega_i < 0.05$ to formulate the reward function as

$$r = \begin{cases} 200, & \text{if } C1 \text{ for both areas} \\ 100, & \text{if } C1 \text{ for one area} \\ 0, & \text{otherwise} \end{cases} . \tag{34}$$

In Fig. 11 the cumulative reward obtained by the agents during training can be seen. Again, it can be clearly seen that the agents are learning and have discovered how to obtain higher rewards in this second scenario. In this case, the agents learn how to jointly balance generation and demand in both areas.

Following the same schema, we change the load by 0.15pu in both areas and the, we observe how the frequency changes and the system power in both areas too. The frequency is restored and demand and generation are rapidly balanced, as can be seen in Figs. 12, 13. This confirm that the described methodology can be applied to solve primary and secondary control problems when more than one balancing area are

present. Whithout centralising any kind of information, the agents learn how to exchange generation. Now, the agents learn that keeping $\Delta\omega$ close to 0 in all areas is associated with high rewards.

It can be seen in Fig. 14 the secondary control action of each generator. Here, as long as we are not addressing how to solve the tertiary control problem, the agents can freely decide the control actions that they estimate. Their only concern is to keep $\Delta\omega_i$ close to 0 for all i. Extending the tertiary control solution methodology is not straightforward when considering multiple balancing authority areas.

V. Concluding Remarks

In this paper, we propose a MARL alternative to implement load frequency control. We showed how to formalize the load frequency control as a MARL problem and how to design a reward function that is based on insights on the economic dispatch problem. Furthermore, we proposed an algorithm, MADDPG, to solve the RL problem. Then, we showed numerical results in two scale test systems. They showed that MADDPG performed efficiently at implementing primary and secondary control even when more than one balancing authority is present and looked promising when applied to tertiary control, i.e., restored frequency with a closed-to-optimal solution. Our key contribution is that the three layers of control are implemented in a fully decentralised way, since traditional solutions centralise information to solve the problem and new technological paradigms make these approaches no longer useful.

For future work, we plan working on different elements of the MARL paradigm that could be enhanced, i.e., the reward function, the LSTM architecture and the introduction of domain knowledge could be further analysed to come up with agents that are able to improve their performance. The applicability and scalability of these techniques in more complex scenarios also needs to be investigated. It would be interesting as well to check the validity of MADDPG to deal with different types of generation plants and to estimate to what extent MARL could be applied in this kind of contexts.

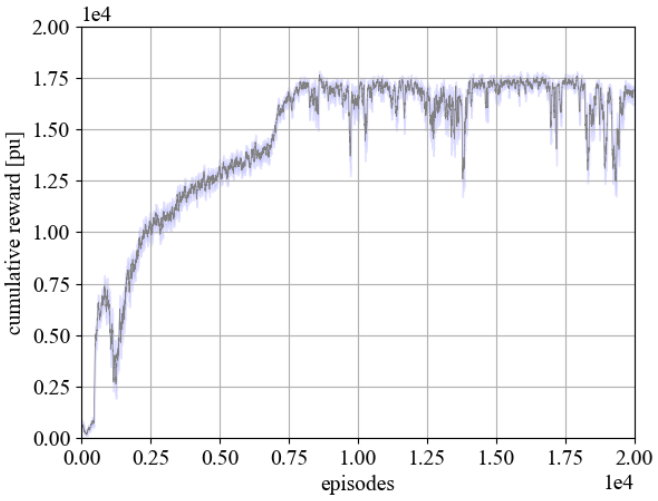

Fig. 11: Smoothed cumulative reward per episode with 95% confidence levels.

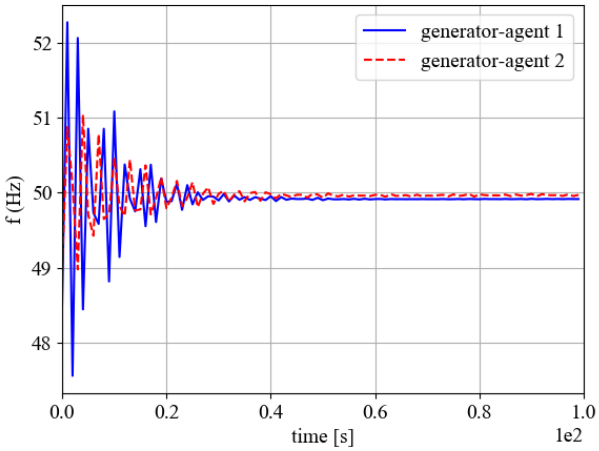

Fig. 12: System frequency in both areas after change in each load by 0.15pu.

References

[1] A. Singh and B. S. Surjan, "Microgrid: A review," *IJRET: International Journal of Research in Engineering and Technology*, vol. 3, no. 2, pp. 185–198, Feb. 2014.

[2] X. Wang, J. M. Guerrero, F. Blaabjerg, and Z. Chen, "A review of power electronics based microgrids," *Journal of Power Electronics*, vol. 12, no. 1, pp. 181–192, Jan. 2012.

[3] S. T. Cady, M. Zholbaryssov, A. D. Domínguez-García, and C. N. Hadjicostis, "A distributed frequency regulation architecture for islanded inertialess ac microgrids," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 1961–1977, Nov. 2017.

[4] D. Apostolopoulou, P. W. Sauer, and A. D. Domínguez-García, "Distributed optimal load frequency control and balancing authority area coordination," in *2015 North American Power Symposium (NAPS)*, Oct. 2015, pp. 1–5.

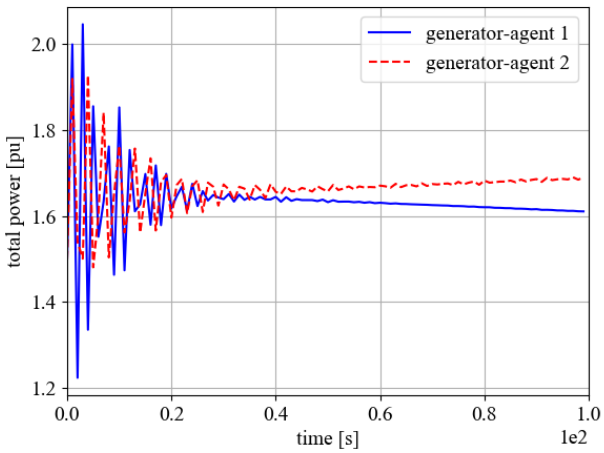

Fig. 13: Total power after change in each load by 0.15pu.

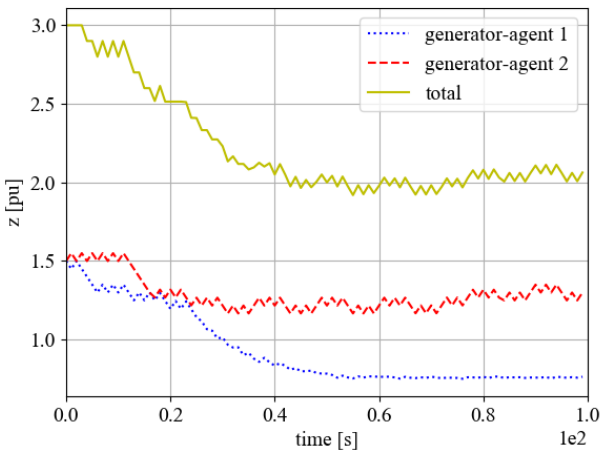

Fig. 14: Generators' output after change in each load by 0.15pu.

[5] D. Apostolopoulou, P. W. Sauer, and A. D. Domínguez-García, "Balancing authority area coordination with limited exchange of information," in *2015 IEEE Power & Energy Society General Meeting*. IEEE, 2015, pp. 1–5.

[6] J. M. Guerrero, J. C. Vasquez, J. Matas, L. G. De Vicuña, and M. Castilla, "Hierarchical control of droop-controlled ac and dc microgrids—a general approach toward standardization," *IEEE Transactions on industrial electronics*, vol. 58, no. 1, pp. 158–172, 2011.

[7] Q. Shafiee, J. M. Guerrero, and J. C. Vasquez, "Distributed secondary control for islanded microgrids—a novel approach," *IEEE Transactions on power electronics*, vol. 29, no. 2, pp. 1018–1031, 2014.

[8] J. W. Simpson-Porco, Q. Shafiee, F. Dörfler, J. C. Vasquez, J. M. Guerrero, and F. Bullo, "Secondary frequency and voltage control of islanded microgrids via distributed averaging," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 11, pp. 7025–7038, 2015.

[9] F. Dörfler, J. W. Simpson-Porco, and F. Bullo, "Breaking the hierarchy: Distributed control and economic optimality in microgrids," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 3, pp. 241–253, 2016.

[10] S. Moayedi and A. Davoudi, "Distributed tertiary control of dc microgrid clusters," *IEEE Transactions on Power Electronics*, vol. 31, no. 2, pp. 1717–1733, Feb. 2016.

[11] S. Rozada, D. Apostolopoulou, and E. Alonso, "Load frequency control: A deep multi-agent reinforcement learning approach," 2020.

[12] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 16, pp. 285–286, 1988.

[13] A. Wood and B. Wollenberg, *Power Generation, Operation and Control*. New York: Wiley, 1996.

[14] P. W. Sauer and M. A. Pai, *Power System Dynamics and Stability*. Upper Saddle River, NJ: Prentice Hall, 1998.

[15] H. Glavitsch and J. Stoffel, "Automatic generation control," *International Journal of Electrical Power & Energy Systems*, vol. 2, no. 1, pp. 21 – 28, 1980.

[16] D. Kirschen and G. Strbac, *Fundamentals of Power System Economics*. Wiley, 2004.

[17] H. S. Nwana, "Software agents: an overview," *The Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–244, 1996.

[18] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994, pp. 157–163.

[19] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[21] L. Bosoniu, R. Babuŝka, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[22] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6379–6390.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," 2017.