applied sciences

MDPI

Article

Exploring Reward Strategies for Wind Turbine Pitch Control by Reinforcement Learning

Jesús Enrique Sierra-García [1,*] and Matilde Santos [2]

[1] Electromechanical Engineering Department, University of Burgos, 09006 Burgos, Spain
[2] Institute of Technological Knowledge, Complutense University of Madrid, C/Profesor García Santesmases 9, 28040 Madrid, Spain; msantos@ucm.es
* Correspondence: jesierra@ubu.es

Received: 28 September 2020; Accepted: 20 October 2020; Published: 23 October 2020

Featured Application: Wind Turbine Pitch Control.

Abstract: In this work, a pitch controller of a wind turbine (WT) inspired by reinforcement learning (RL) is designed and implemented. The control system consists of a state estimator, a reward strategy, a policy table, and a policy update algorithm. Novel reward strategies related to the energy deviation from the rated power are defined. They are designed to improve the efficiency of the WT. Two new categories of reward strategies are proposed: "only positive" (O-P) and "positive-negative" (P-N) rewards. The relationship of these categories with the exploration-exploitation dilemma, the use of ϵ-greedy methods and the learning convergence are also introduced and linked to the WT control problem. In addition, an extensive analysis of the influence of the different rewards in the controller performance and in the learning speed is carried out. The controller is compared with a proportional-integral-derivative (PID) regulator for the same small wind turbine, obtaining better results. The simulations show how the P-N rewards improve the performance of the controller, stabilize the output power around the rated power, and reduce the error over time.

Keywords: intelligent control; pitch angle; reinforcement learning; reward strategies; wind turbine; renewable energies

1. Introduction

Wind energy gains strength year after year. This renewable energy is becoming one of the most used clean energies worldwide due to its high efficiency, its competitive payback, and the growth in investment in sustainable policies in many countries [1]. Despite its recent great development, there are still many engineering challenges regarding wind turbines (WT) technology that must be addressed [2].

From the control perspective, one of the main goals is to stabilize the output power of the WT around its rated value. This should be achieved while the efficiency is maximized, and vibrations and fatigue are minimized. Even more, safety must be guaranteed under all operation conditions. This may even be more critical for floating offshore wind turbines (FOWT), as it has been proved that the control system can affect the stability of the floating device [3,4]. This general and ambitious control objective is implemented in many different control actions, depending on the type of WT. So, the pitch angle control is normally used to maintain the output power close to its rated value once the wind speed overpasses a certain threshold. The control of the generator speed is intended to track the optimum rotor velocity when the wind speed is below the rated output speed. And finally, the yaw angle control is used to optimize the attitude of the nacelle to follow the wind stream direction.

This work is focused on the pitch control of the wind turbine. Blade pitch control technology alters the pitch angle of a blade to change the angle of wind attack and ultimately to change the aerodynamic forces on the blades of a WT. Therefore, this control system pitches the blades usually a few degrees every time the wind changes in order to keep the rotor blades at the required angle, thus controlling the rotational speed of the turbine [5,6]. This is not a trivial task due to the non-linearity of the equations that describe its dynamics, the coupling between the internal variables, and uncertainty that comes from external loads [7], mainly wind and, in the case of FOWT, also waves and currents, that makes its dynamics changes [8]. The complexity of this system has led to propose the use of advanced and intelligent control techniques such as fuzzy logic, neural networks, and reinforcement learning, among others, to address WT related control problems.

Among control solutions, sliding mode control, and adaptive control have been recently applied to WT with encouraging results. In Liu et al. [9], a PI-type sliding mode control (SMC) strategy for (permanent magnet synchronous generator) PMSG-based wind energy conversion system (WECS) uncertainties is presented. Nasiri et al. propose a super-twisting sliding mode control for a gearless wind turbine by a permanent magnet synchronous generator [10]. A robust SMC approach to control the rotor speed in the presence of uncertainties in the model is also proposed in Colombo et al. [11]. In addition, closed loop convergence of the whole system is proved. In Yin et al. [12], an adaptive robust integral SMC pitch angle controller and a projection-type adaptation law are synthesized to accurately track the desired pitch angle trajectory, while it compensates model uncertainties and disturbances. Bashetty et al. propose an adaptive controller for pitch and torque control of the wind turbines operating under turbulent wind conditions [13].

The WT control problem has also been addressed in the literature using different intelligent control techniques, mainly fuzzy logic and neuro-fuzzy inference systems [14–21]. Reinforcement learning has been an inspiration for the design of control strategies [22,23]. A recent overview of deep reinforcement learning for power system applications can be found in Zhang et al. [24]. In Fernandez-Gauna et al. [25], RL is used for the control of variable speed wind turbines. Particularly, it adapts conventional variable speed WT controllers to changing wind conditions. As a further step, the same authors apply conditioned RL (CRL) to this complex control scenario with large state-action spaces to be explored [26]. In Abouheaf et al. [27], an online controller based on a policy iteration reinforcement learning paradigm along with an adaptive actor-critic technique is applied to the doubly fed induction generator wind turbines. Sedighizadeh proposes an adaptive PID controller tuned by RL [28]. An artificial neural network based on RL for WT yaw control is presented in Saénz-Aguirre et al. [29]. In a more recent paper, the authors propose a performance enhancement of this wind turbine neuro–RL yaw control [30].

The paper by Kuznetsova proposes a RL algorithm to plan the battery scheduling in order to optimize the use of the grid [31]. Tomin et al. propose adaptive control techniques, which first extract the stochastic property of wind speed using a trained RL agent, and then apply their obtained optimal policy to the wind turbine adaptive control design [32]. In Hosseini et al., passive RL solved by particle swarm optimization policy (PSO-P) is used to handle an adaptive neuro-fuzzy inference system type-2 structure with unsupervised clustering for controlling the pitch angle of a real wind turbine [33]. Chen et al. also propose a robust wind turbine controller that adopts adaptive dynamic programming based on RL and system state data [34]. In a related problem, deep reinforcement learning with knowledge assisted learning are applied to deal with the wake effect in a cooperative wind farm control [35].

To summarize, in the literature, the RL approach has been applied to the different control actions of wind turbines or to other related problems with successful results. However, this learning strategy has not been directly applied to the pitch control, neither the reward mechanisms have been analyzed in order to improve the control performance. Thus, in the present work, the RL-inspired pitch control proposed in Sierra-García et al. [36] has been extended to complete the range of reward policies. Novel reward strategies related to the energy deviation from the rated power, namely Mean Squared

Error Reward Strategy (MSE-RS), Mean error Reward Strategy (Mean-RS), and the corresponding increments, ΔMSE-RS and ΔMean-RS, have been proposed, implemented, and combined.

In addition, many of the previous works based on RL execute ϵ-greedy methods in order to increase the exploration level and avoid actions unexplored. The ϵ-greedy methods select an action either randomly or considering the previous experiences, the latter being called greedy selection. The greedy selection is carried out trying to maximize the future expected rewards, based on the previous rewards already received. The probability of selecting a random action is ϵ and the probability of performing a greedy selection is $(1-\epsilon)$. This approach tends to improve the convergence of the learning, but its main drawback is that it introduces a higher randomness in the process and makes the system less deterministic. To avoid the use of ϵ-greedy methods, the concept of Positive-Negative (P-N) rewards, and its relationship with the exploration-exploitation dilemma is here introduced and linked to the WT control problem. An advantage of P-N rewards, observed in this work, is that the behavior is generally more deterministic than ϵ-greedy, allowing more replicable results with fewer iterations.

Moreover, a deep study of the influence of the type of reward in the performance of the system response and in the learning speed has also been carried out. As it will be shown in the simulation experiments, P-N rewards work better than Only-Positive (O-P) rewards for all the policy update algorithms. However, the combination of P-N with O-P rewards helps to soften the variance of the output power.

The rest of the paper is organized as follows. Section 2 describes the model of the small wind turbine used. Section 3 explains the RL-based controller architecture, the policy update algorithms and the reward strategies implemented. The results for different configurations are analyzed and discussed in Section 4. The paper ends with the conclusions and future works.

2. Wind Turbine Model Description

The model of a small turbine of 7 kW is used. The equations of the model are summarized in Equations (1)–(6). The development of these equations can be found in Sierra-García et al. [18].

$$\dot{I}_a = \frac{1}{L_a}\Big(K_g \cdot K_\phi \cdot w - (R_a + R_L) I_a\Big), \tag{1}$$

$$\lambda_i = \left[\left(\frac{1}{\lambda + c_8}\right) - \left(\frac{c_9}{\theta^3 + 1}\right)\right]^{-1}, \tag{2}$$

$$C_p(\lambda_i, \theta) = c_1\left[\frac{C_2}{\lambda_i} - c_3\theta - c_4\theta^{c_5} - c_6\right]e^{-\frac{c_7}{\lambda_i}}, \tag{3}$$

$$\dot{w} = \frac{1}{2 \cdot J \cdot w}\Big(C_p(\lambda_i, \theta) \cdot \rho\pi R^2 \cdot v^3\Big) - \frac{1}{J}\Big(K_g \cdot K_\phi \cdot I_a + K_f w\Big), \tag{4}$$

$$\ddot{\theta} = \frac{1}{T_\theta}\Big[K_\theta\Big(\theta_{ref} - \theta\Big) - \dot{\theta}\Big], \tag{5}$$

$$P_{out} = R_L \cdot {I_a}^2 \tag{6}$$

where L_a is the armature inductance (H), K_g is a dimensionless constant of the generator, K_ϕ is the magnetic flow coupling constant (V·s/rad), R_a is the armature resistance (Ω), R_L is the resistance of the load (Ω), considered in the study as purely resistive, w is the angular rotor speed (rad/s), I_a is the armature current (A), the values of the coefficients c_1 to c_9 depend on the characteristics of the wind turbine, J is the rotational inertia (kg m^2), R is the radius or blade length (m), ρ is the air density (kg/m^3), v is wind speed (m/s), K_f is the friction coefficient (N m/rad/s), θ_{ref} is the reference for the pitch (rad), and θ is the pitch (rad).

The state variables of the control system are the current in the armature and the angular speed of the rotor, $[I_a, w]$. On the other hand, the manipulated or control input variable is the pitch angle,

θ_{ref}, and the controlled variable is the output power, P_{out}, unlike other works where the rotor speed is the controlled variable.

The RL controller proposed in this paper is applied to generate a pitch reference signal, θ_{ref}, in order to stabilize the output power, P_{out}, of the wind turbine around its rated value. Equations (1)–(6) are used to simulate the behavior of the wind turbine and thus allow us to evaluate the performance of the controller.

The values of the parameters used during the simulations (Table 1) are taken from Mikati et al. [37].

Table 1. Parameters of the wind turbine model.

Parameter	Description	Value/Units
L_a	Inductance of the armature	13.5 mH
K_g	Constant of the generator	23.31
K_ϕ	Magnetic flow coupling constant	0.264 V/rad/s
R_a	Resistance of the armature	0.275 Ω
R_L	Resistance of the load	8 Ω
J	Inertia	6.53 kg m^2
R	Radio of the blade	3.2 m
ρ	Density of the air	1.223 kg/m^3
K_f	Friction coefficient	0.025 N m/rad/s
$[c_1, c_2, c_3]$	C_p constants	[0.73, 151, 0.58]
$[c_4, c_5, c_6]$	C_p constants	[0.002, 2.14, 13.2, 18.4]
$[c_7, c_8, c_9]$	C_p constants	[18.4, −0.02, −0.003]
$[K_\theta, T_\theta]$	Pitch actuator constants	[0.15, 2]

3. RL-Inspired Controller

The reinforcement learning approach consists of an environment, an agent and an interpreter. The agent, based on the state perceived by the interpreter s_t and the previous rewards provided by the interpreter $r_{1...t}$, selects the best action to be carried out. This action, a_t, produces an effect on the environment. This fact is observed by the interpreter who provides information to the agent about the new state, s_{t+1}, and the reward of the previous action, r_{t+1}, closing the loop [38,39]. Some authors consider that the interpreter is embedded in either the environment or the agent; in any case, the function of the interpreter is always present.

Discrete reinforcement learning can be expressed as follows [40]:

- S is a finite set of states perceived by the interpreter. This set is made with variables of the environment, which must be observable by the interpreter and may be different to the state variables of the environment.
- A is a finite set of actions to be conducted by the agent.
- s_t is the state at t
- a_t is the action performed by the agent when the interpreter perceives the state s_t
- r_{t+1} is the reward received after action a_t is carried out
- s_{t+1} is the state after action a_t is carried out
- The environment or world is a Markov process: $MDP = \langle s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2 \ldots \rangle$
- $\pi : S \times A \rightarrow [0, 1]$ is the policy; this function provides the probability of selection of an action a for every pair (s, a)
- $p^a_{ss'} = \Pr\{s_{t+1} = s' \big| s_t = s \wedge a_t = a\}$ is the probability that a state changes from s to s' with action a
- $p^\pi(s', a')$ is the probability of selecting action a' at state s' under policy π
- $r^a_s = E\{r_{t+1} \big| s_t = s \wedge a_t = a\}$ is the expected one-step reward
- $Q^\pi_{(s,a)} = r^a_s + \gamma \sum_{s'} p^a_{ss'} \sum_{a'} p^\pi(s', a') Q^\pi_{(s',a')}$ is the expected sum of discounted rewards

The scheme of the designed controller inspired by this RL approach is presented in Figure 1. It is composed by a state estimator, a reward calculator, a policy table, an actuator, and a method to update

the policy. The state estimator receives the output power error (P_{err},), defined as the difference between the rated power P_{ref} and the current output power P_{out}, and its derivative, $\dot{P}_{err}$. These signals are discretized and define the state $s_t \in S$, where t is the current time. The interpreter is implemented by the state estimator and the reward calculator. The agent includes the policy, the policy update algorithm and the actuator. Both interpreter and agent form the controller (Figure 1).

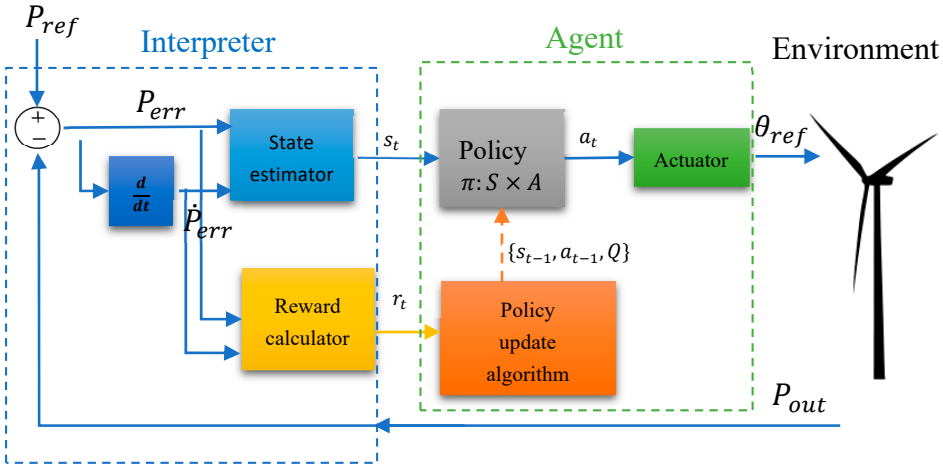

Figure 1. Scheme of the reinforcement learning (RL) controller.

The policy is defined as a function $\pi : S \rightarrow A$ which assigns an action $a_t \in A$ to each state in S. This action a_t is selected in a way that maximizes the long-term expected reward. The actuator transforms the discrete action a_t into a control signal for the pitch θ_{ref} in the range $[0, \pi/2]$. Each time an action is executed, in the next iteration, the reward calculator observes the new P_{err} and $\dot{P}_{err}$ and calculates a reward/penalty r_t for action a_{t-1}. The policy update algorithm uses this reward to modify the policy for the state s_{t-1}.

The policy has been implemented as a table $T^{\pi}_{(s,a)} : S \times A \rightarrow \mathrm{R}$ together with a function $f_a : S \rightarrow A$. The table relates each pair $(s, a) \in S \times A$ with a real number that represents an estimation of the long-term expected reward, that is, the one that will be received when action a is executed in the state s, also known as Q. The estimate depends on the policy update algorithm. The table has s rows (states) and a columns (actions). Given a state s, the function f_a searches for the action with the maximum value of Q in the table.

3.1. Policy Update Algorithm

The policy update algorithm calculates the estimate corresponding to the previous pair $(s, a) \in S \times A$ of the $T^{\pi}_{(s,a)}$ each control cycle. At t_i, the last state and action, that is, (s_{t-1}, a_{t-1}), are updated by the policy function f_π, using the previous estimation of the long-term expected reward $T^{\pi}_{(s_{t-1},a_{t-1})}$ and the current reward r_t, Equation (7).

$$T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) := f_\pi\left(T^{\pi}_{(s_{t-1},a_{t-1})}(t_{i-1}), r_t, \ldots\right) \tag{7}$$

Once the table T^{π} is updated, the table is searched for the action that maximizes the reward, Equation (8):

$$f_a(s_t) := \underset{a}{\mathrm{arg}MAX}\left(T^{\pi}_{(s_t,a)}(t_i)\right) \tag{8}$$

The different policy update algorithms f_π that have been implemented and compared in the experiments are the following

i. One-reward (OR), the last one received. As it only takes into account the last reward (smallest memory), it may be very useful when the system to be controlled changes frequently, Equation (9)

$$OR : T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) := r_t \tag{9}$$

ii. Summation of all previous rewards (SAR). It may cause an overflow in the long term, that could be solved if the values are saturated to be maintained within some limits, Equation (10)

$$SAR : T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) := T^{\pi}_{(s_{t-1},a_{t-1})}(t_{i-1}) + r_t \tag{10}$$

iii. Mean of all previous rewards (MAR). This policy gives more opportunities to not yet selected actions than SAR, especially when there are many rewards with the same sign, Equation (11)

$$MAR : T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) := \frac{1}{i}\Big[T^{\pi}_{(s_{t-1},a_{t-1})}(t_{i-1}) + r_t\Big] \tag{11}$$

iv. Only learning with learning rate (OL-LR). It considers a percentage of all previous rewards, Equation (12), given by the learning rate parameter $\alpha \in \mathbb{R}\ [0,1]$.

$$OL-LR : T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) := T^{\pi}_{(s_{t-1},a_{t-1})}(t_{i-1}) + \alpha{\cdot}r_t \tag{12}$$

v. Learning and forgetting with learning rate (LF-LR). The previous methods do not forget any previous reward; this may be effective for steady systems but for changing models it might be advantageous to forget some previous rewards, Equation (13). The forgetting factor is modelled as the complementary leaning rate $(1-\alpha)$.

$$LF-LR : T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) := (1-\alpha)T^{\pi}_{(s_{t-1},a_{t-1})}(t_{i-1}) + \alpha{\cdot}r_t \tag{13}$$

vi. Q-learning (QL). The discount factor, $\gamma \in \mathbb{R}\ [0,1]$ is included in the policy function, Equations (14) and (15).

$$a_{max} = \underset{a}{\arg MAX}\Big(T^{\pi}_{(s_t,a)}(t_{i-1})\Big), \tag{14}$$

$$Q_L : T^{\pi}_{(s_{t-1},a_{t-1})}(t_i) = (1-\alpha){\cdot}T^{\pi}_{(s_{t-1},a_{t-1})}(t_{i-1}) + \alpha\Big[r_t - \gamma{\cdot}T^{\pi}_{(s_{t-1},a_{max})}(t_{i-1})\Big] \tag{15}$$

3.2. Exploring Reward Strategies

Once the table T^{π} and the function f_a are implemented, and a policy update algorithm f_{π} is selected, it is necessary to define the reward strategy of the reinforcement learning procedure. Although so far the definition of the policy update algorithm is general, the design of the rewards and punishments requires expert knowledge about the specific system.

In this work the target is to stabilize the output power of the WT around it nominal value thus reducing the error between the output power and the rated power. The error will then be the key to define the reward.

3.2.1. Only Positive (O-P) Reward Strategies

The most intuitive approach seems to be rewarding the relative position of the system output to the rated (reference) value. The closer the output to the desired value, the bigger the reward. However, considering the distance (absolute value of the error), the reward grows with the error. To avoid this

problem, a maximum error is defined, $P_{err_{MAX}}$, and the absolute error is subtracted from it. This is called "Position Reward Strategy" (PRS), Equation (16).

$$PRS : r_{t_i} = P_{err_{MAX}} - \left|P_{err}(t_i)\right| \tag{16}$$

This strategy only provides positive rewards and no punishments. Thus it belongs to the category only positive (O-P) reinforcement. As it will be seen in the results section, this is the cause of its lack of convergence when individually applied. The main drawback of O-P rewards is that the same actions are selected repeatedly, and many others are not explored. This means that the optimal actions are rarely visited. To solve it, exploration can be externally forced by greedy-methods [40] or O-P rewards can be combined with positive-negative reinforcement (P-N rewards).

To illustrate the problem, let $T^{\pi}_{(s,a)}$ be initialized to 0 for all the states and actions, $T^{\pi}_{(s,a)}(t_0) = 0$, $\forall(s,a)$. At t_0 the system is in the state $s_0 = s$. Since all actions have the same value in the table, any action $a_0 = a_{s_0}$ is at random selected. At the next control time t_1 the state is s_1, which can be different or equal to s_0. The reward received is $r_1 > 0$. The policy update algorithm modifies the value of the table associated to the previous pair (state, action) $T^{\pi}_{(s_0,a_0)}(t_1) = f_{\pi}(r_1) > 0$. Now a new action a_1 associated with state s_1 must be selected. If $s_1 \neq s_0$ the action is randomly selected again because all the actions in the row have the value 0. However, if the state is the same as the previous one, the selected action is the same as in t_0, $a_1 = a_0 = a_{s_0}$ because $f_{\pi}(r_1) > 0$ is the maximum value of the row. In that case, at the next control time t_2 the table is updated $T^{\pi}_{(s_0,a_0)}(t_2) = f_{\pi}(r_1, r_2)$. With this O-P rewards this value always tends to be greater than 0, forcing the same actions to be selected. Only some specific QL configurations may give negative values. This process is repeated every control period. If the state action is different from all previous states, a new cell in the table is populated. Otherwise, the selected action will be the first action selected in that state.

If the initialization of the table $T^{\pi}_{(s,a)}$ is high enough (regarding the rewards), we can ensure that all actions will be visited at least once for OR, MAR and QL update policies. This can be a solution if the system is stationary, because the best action for each state does not change, so that once all the actions have been tested, the optimum one has necessarily been found. However, if the system is changing this method is not enough. In these cases, ϵ-greedy methods have shown successful results [40]. In each control period, the new action is randomly selected with a probability ϵ (forced exploration) or selected from the table $T^{\pi}_{(s,a)}$ with a probability $(1-\epsilon)$ (exploitation).

Another possible measure that can be used with the O-P strategy to calculate the reward is the previous MSE. In this case, the reward is calculated by applying a time window to capture the errors prior to the current moment. We have called this strategy "MSE reward strategy" (MSE-RS) and it is calculated by Equation (17):

$$MSE - RS : r_{t_i} = \frac{1}{\sqrt{\frac{1}{T_w}\sum_{j=i-k}^{i}\left[P_{err}\left(t_j\right)^2 T_{sj}\right]}} \tag{17}$$

where T_w is the time length window; T_{sj} is the variable step size at t_j, and k is calculated so $t_i - t_{i-k} = T_w$. As it may be observed, greater errors will produce smaller rewards.

In a similar way it is possible to use the mean value, "Mean reward strategy" (Mean-RS), that is defined as Equation (18),

$$Mean - RS : r_{t_i} = \frac{1}{\left|\frac{1}{T_w}\sum_{j=i-k}^{i}\left[P_{err}\left(t_j\right)T_{sj}\right]\right|} \tag{18}$$

In the results section it will be shown how the Mean-RS strategy reduces the mean of the output error and cuts down the error when it is combined with a P-N reward

3.2.2. Positive Negative (P-N) Reward Strategies

Unlike O-P reinforcement, P-N reward strategies encourage the natural exploration of actions that enable convergence of learning. The positive and negative rewards compensate the values of the table $T^{\pi}_{(s,a)}$, which makes it easier to carry out different actions even if the states are repeated. An advantage of natural exploration over ϵ-greedy methods is that their behavior is more deterministic. This provides more repeatable results with fewer iterations. However, the disadvantage is that if rewards are not well balanced, the exploration may be insufficient.

To ensure that the rewards are well balanced, it is helpful to calculate the rewards with some measure of the error variance. PRS, MSE-RS, Mean-RS perform an error measurement in a specific period of time. They do provide neither a measure of the variation nor how quickly its value changes. A natural evolution of PRS is to use speed rather than position to measure whether we are getting closer to or away from rated power and how fast we are doing so. We call it "velocity reward strategy" (VRS), Equations (19) and (20).

$$r_v = \begin{cases} -\dot{P}_{err}(t_i) & P_{err}(t_i) > 0 \\ \dot{P}_{err}(t_i) & P_{err}(t_i) \leq 0 \end{cases}, \tag{19}$$

$$VRS: r_{t_i} = \begin{cases} -r_v & sgn\left(\dot{P}_{err}(t_i)\right) \neq sgn\left(\dot{P}_{err}(t_i - 1)\right) \wedge \left|\dot{P}_{err}(t_i)\right| < \left|\dot{P}_{err}(t_i - 1)\right| \\ r_v & otherwise \end{cases} \tag{20}$$

The calculation is divided into two parts. First, r_v is calculated to indicate whether we are getting closer to or away from the nominal power, Equation (19). If the error is positive and decreases, we are getting closer to the reference. The second part is to detect when the error changes sign and the new absolute error is less than the previous one. It would be a punished action according to Equation (19) but when this case is detected, the punishment becomes a reward, Equation (20).

A change in the MSE-RS can also be measured as Equation (17). This produces a new P-N reward, the ΔMSE-RS, which is calculated by Equation (21):

$$\Delta MSE - RS: r_{t_i} = \sqrt{\frac{1}{T_w}\sum_{j=i-1-k}^{i-1}\left[P_{err}\left(t_j\right)^2 T_{sj}\right]} - \sqrt{\frac{1}{T_w}\sum_{j=i-k}^{i}\left[P_{err}\left(t_j\right)^2 T_{sj}\right]} \tag{21}$$

Comparing Equation (17) and Equation (21) it is possible to observe how the reward is calculated by subtracting the inverse of MSE-RS at t_i from the inverse of MSE-RS at t_{i-1}. In this way, if the MSE is reduced, the reward is positive; otherwise it is negative.

Similar to MSE, a P-N reward strategy can be obtained based on the mean value of the error. It is calculated with Equation (22) and is called $\Delta Mean - RS$.

$$\Delta Mean - RS: r_{t_i} = \left|\frac{1}{T_w}\sum_{j=i-1-k}^{i-1}\left[P_{err}\left(t_j\right)T_{sj}\right]\right| - \left|\frac{1}{T_w}\sum_{j=i-k}^{i}\left[P_{err}\left(t_j\right)T_{sj}\right]\right| \tag{22}$$

As will be shown in the results, the $\Delta Mean - RS$ improve the mean value of the error compared to other reward strategies.

In addition to these P-N rewards, it is possible to define new rewards strategies by combining O-P with P-N rewards, such as PRS·VRS. However, the combination of P-N rewards between them does not generally provide better results; even in the same cases this combination can produce an effective O-P reward if they always have the same sign.

4. Simulation Results and Discussion

An in-depth analysis of the performance of the RL controller under different configurations and reward strategies has been carried out. The algorithm has been coded by the authors using Matlab/Simulink software. The duration of each simulation is 100 s. To reduce the discretization error, a variable step size has been used for simulations, with a maximum step size of 10 ms. The control sampling period T_c has been set to 100 ms. In all the experiments, the wind speed is randomly generated between 11.5 m/s and 14 m/s.

For comparison purposes, a PID is also designed with the same goal of stabilizing the output power around the rated value of the wind turbine. Thus, the input of the PID regulator is the output power and its output is the pitch angle reference. In order to make a fair comparison, the PID output has been scaled to adjust its range to [0, π/2] rad and it has been also biased by the term π/4. The output of the PID is saturated for values below 0° and above 90°. The parameters of the PID have been tuned by trial and error, and they have been set to KP = 0.9, KD = 0.2 and KI = 0.5.

Figure 2 compares the power output, the generator torque and the pitch signal obtained with different control strategies. The blue line represents the output power when the angle of the blades is 0°, that is, the wind turbine collects the maximum energy from the wind. As you would expect, this action provides maximum power output. The red line represents the opposite case, the pitch angle is set to 90° (feather position). In this position, the blades offer minimal resistance to the wind, so the energy extracted is also minimal. The pitch angle reference values are fixed for the open loop system in both cases, without using any external controller. In a real wind turbine, there is a controller to regulate the current of the blade rotor in order to adjust the pitch angle. In our work this is simulated by Equation (5). The yellow line is the output obtained with the PID, the purple line when the RL controller is used, and the green is the rated power. In this experiment, the policy update algorithm is SAR and the reward strategy is VRS. It is observed how the response of the RL controller is much better than that of the PID, with smaller error and less variation. As expected, the pitch signal is smoother with the PID regulator than with the RL controller. However, as a counterpart, the PID reacts slower, producing bigger overshoot and longer stabilization time of the power output.

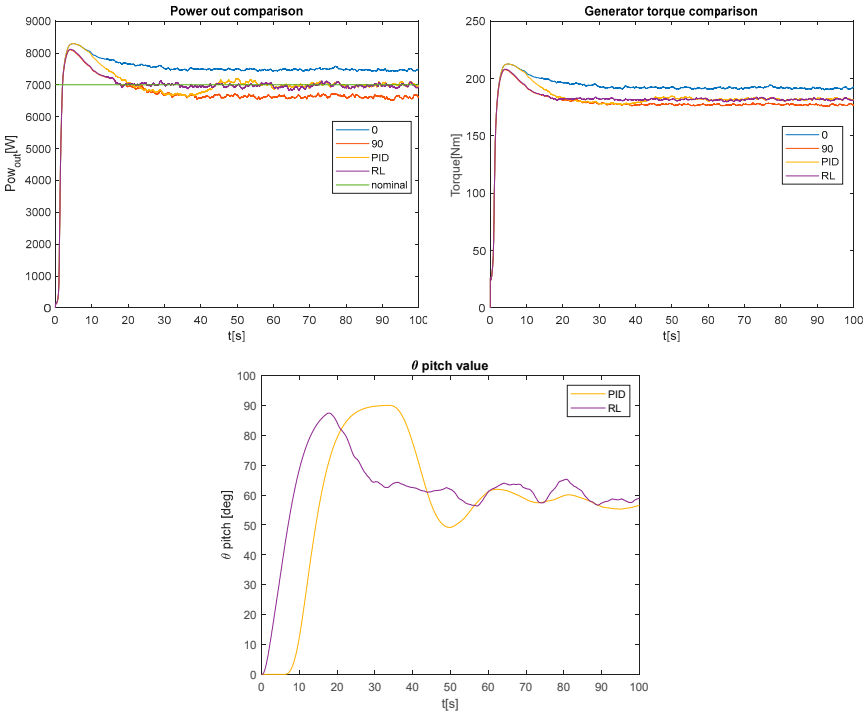

Figure 2. Comparison of power output (**up-left**), generator torque (**up-right**) and pitch angle, for proportional-integral-derivative (PID) and different reinforcement learning (RL) control strategies.

Several experiments have been carried out comparing the performance of the RL controller when using different policy update algorithms and reward strategies. The quantitative results are presented in Tables 2–4 and confirm the graphical results of Figure 2. These data were extracted at the end of iteration 25. The reward window was set to 100 ms. In these tables, the best results per column (policy) have been boldfaced and the best results per row (reward) have been underlined.

Table 2. MSE [W] for different policy update algorithms and reward strategies.

	Policy Update					
Reward	OR	SAR	MAR	OL-LR	LF-LR	QL
PRS	402.86	407.59	403.22	404.11	402.28	406.53
VRS	**306.71**	**265.62**	**271.56**	**266.37**	**281.39**	287.83
MSE-RS	405.24	407.81	403.03	403.22	405.88	405.43
MEAN-RS	401.07	402.86	404.10	405,73	401.63	405.32
ΔMSE-RS	308.49	270.21	273.08	274.71	282.84	**274.33**
ΔMEAN-RS	307.50	272.93	274.47	277.17	287.18	284.73
PID	394.09					

Table 3. Output power mean [kW] for different policy update algorithms and reward strategies.

	Policy Update					
Reward	OR	SAR	MAR	OL-LR	LF-LR	QL
PRS	**6.80**	6.79	6.80	6.80	6.80	6.79
VRS	7.22	7.06	7.13	7.09	7.17	7.18
MSE-RS	6.79	6.79	6.80	6.80	6.79	6.79
MEAN-RS	**6.80**	6.80	6.80	6.79	6.80	6.79
ΔMSE-RS	7.21	7.06	**7.07**	7.04	**7.16**	**7.14**
ΔMEAN-RS	7.21	**7.05**	7.12	**7.03**	7.17	7.16
PID	7.25					

Table 4. Variance [kW] for different policy update algorithms and reward strategies.

	Policy Update					
Reward	OR	SAR	MAR	OL-LR	LF-LR	QL
PRS	123	126	**123**	123	123	125
VRS	193	125	152	137	156	174
MSE_RS	124	124	**123**	123	124	**124**
MEAN_RS	**122**	**123**	124	124	**122**	**124**
ΔMSE-RS	183	127	130	122	159	150
ΔMEAN-RS	180	124	149	**121**	168	165
PID	273					

The smallest MSE error is obtained by combining SAR and VRS. Overall, SAR provides the best results, closely followed by MAR and OL-LR. As expected, the worst results are produced by O-R, this can be explained because it only considers the last reward, which limits the learning capacity. For almost all policy update algorithms, the MSE is lower when VRS is applied; the only exception is QL, which performs better with ΔMSE-RS.

Another interesting result is that the performance of O-P rewards is much worse than for P-N rewards. The reason may be that exploration with O-P rewards is very low, and the best actions for many of the states are not exploited. The exploration can be increased by changing the start of the Qtable. Finally, the P-N reward provides better performance than the PID even with OR.

Table 3 shows the mean value of the power output obtained by these experiments. The best value is obtained by combining OL-LR and ΔMEAN-RS. OL-LR is the best policy update followed by SAR. O-R again provides the worst results. In this case, the best reward strategies are ΔMEAN-RS and ΔMSE-RS. This may be because the mean value measurement is intrinsically considered in the reward calculation.

Table 4 presents the variance of the power output in the previous experiments. Unlike Table 2, in general, N-P rewards produce worse results than O-P rewards. This is logical because N-P rewards produce more change in the selected actions and a more varying output, therefore more variation. However, it is notable that the combination of VRS and SAR provides a good balance between MSE, mean value, and variance.

Figure 3 represents the evolution of the saturated error and its derivative, iteration by iteration. In this experiment, the combination of SAR and ΔMSE-RS is used. In Figure 3 it is possible to observe an initial peak of −1000 W in the error (horizontal axis). This error corresponds to an output power of 8 kW (the rate power is 7 kW). It has not been possible to avoid it at the initial stage with any of the tested control strategies, even forcing the pitch to feather. A remarkable result is that, in each iteration, the errors are merged and centered around a cluster. This explains how the mean value of the output power approaches the nominal power over time.

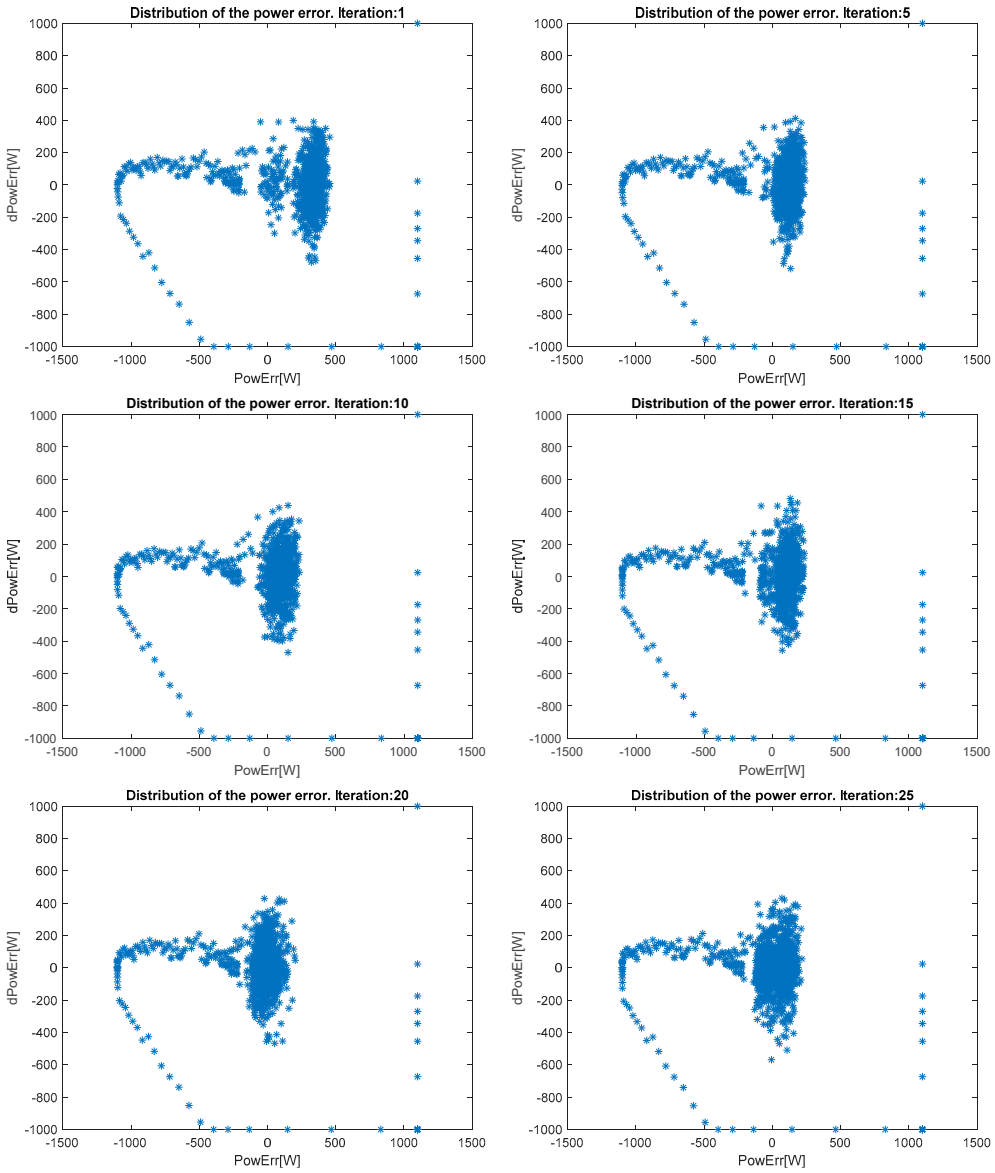

Figure 3. Distribution of the error when ΔMean Squared Error Reward Strategy (ΔMSE-RS) and summation of all previous rewards (SAR) are applied for the first 25 iterations.

One way to measure the center of this cluster is to use the radius of the centroid of the error calculated by the Equation (23):

$$R_c = \sqrt{\left(\sum\left[P_{out_i} - P_{ref}\right]\right)^2 + \left(\sum\left[\dot{P}_{out_i} - \dot{P}_{ref}\right]\right)^2} \tag{23}$$

Figures 4–9 show the evolution of the MSE (left) and the error centroid radius (right) for different combinations of reward strategy (Section 3.2) and policy update algorithm (Section 3.1). In each figure, the policy update algorithm is the same and a sweep of different reward strategies is performed. Each reward strategy is represented by a different color: PRS in dark blue, VRS in red, MSE-RS in yellow, Mean-RS in purple, ΔMSE-RS in green and ΔMEAN-RS in light blue. It is possible to observe how, in general, the MSE and the radius decrease with time, although the ratio is quite different depending on the combination of policy update algorithm and reward strategy that is used.

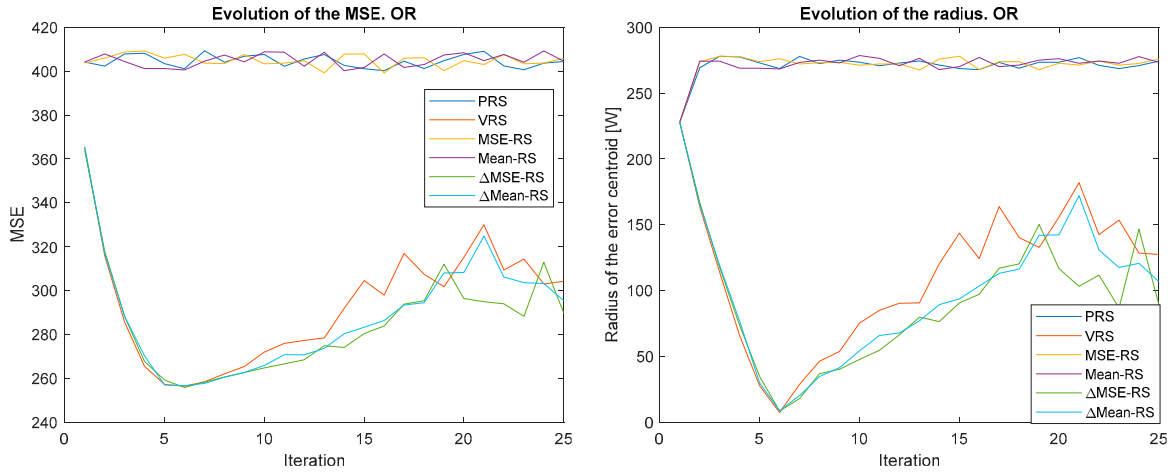

Figure 4. Evolution of the MSE (**left**) and error centroid radius (**right**) for different reward strategies and update policy One-reward (OR).

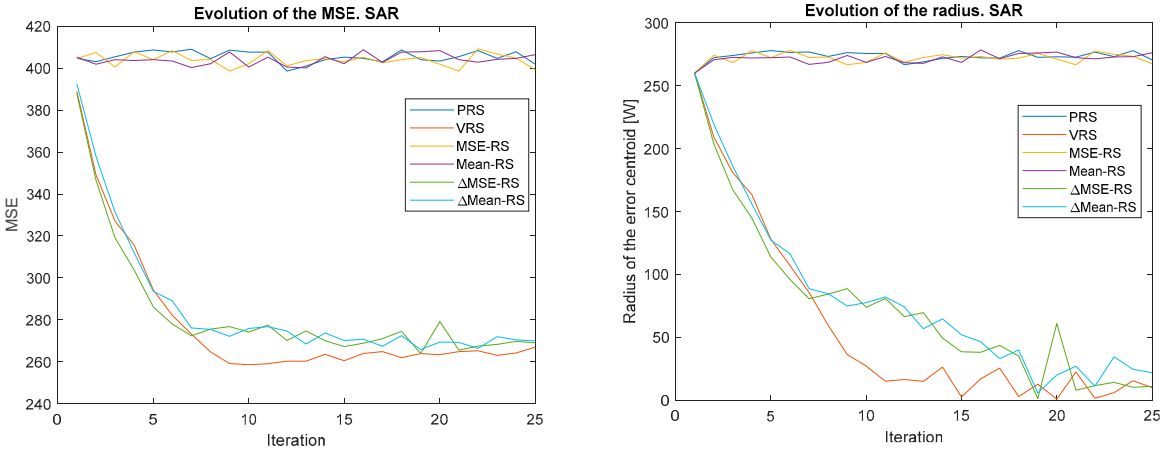

Figure 5. Evolution of the MSE (**left**) and error centroid radius (**right**) for different reward strategies and update policy SAR.

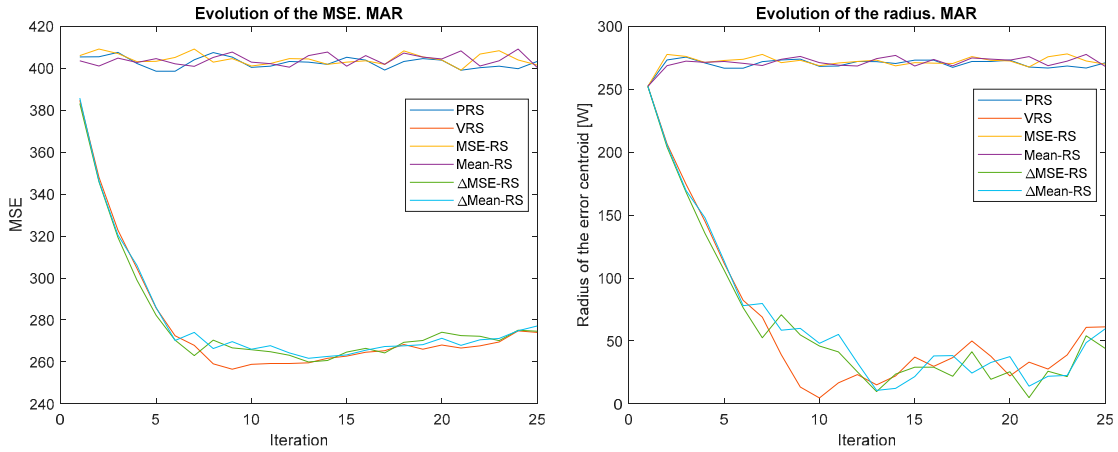

Figure 6. Evolution of the MSE (**left**) and error centroid radius (**right**) for different reward strategies and update policy means of all previous rewards (MAR).

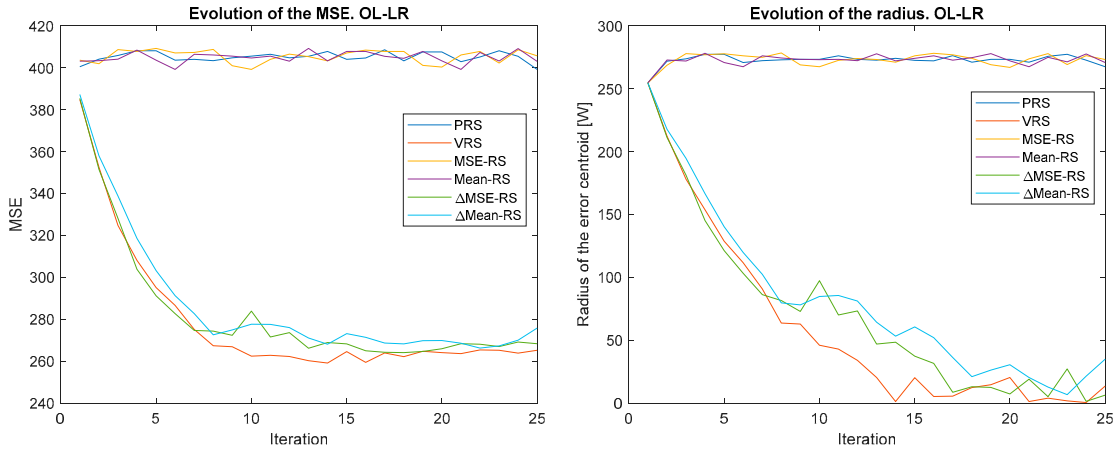

Figure 7. Evolution of the MSE (**left**) and error centroid radius (**right**) for different reward strategies and update policy only learning with learning rate (OL-LR).

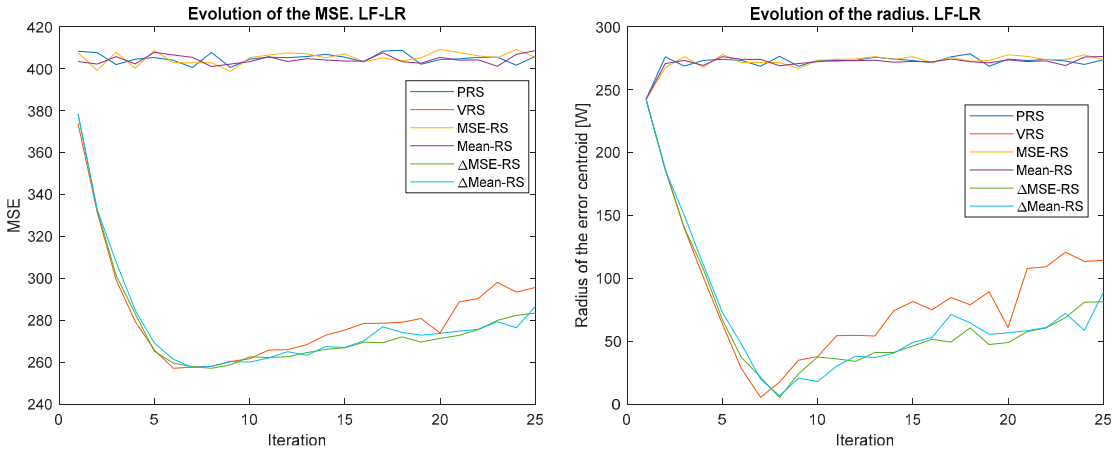

Figure 8. Evolution of the MSE (**left**) and error centroid radius (**right**) for different reward strategies and update policy Learning and forgetting with learning rate (LF-LR).

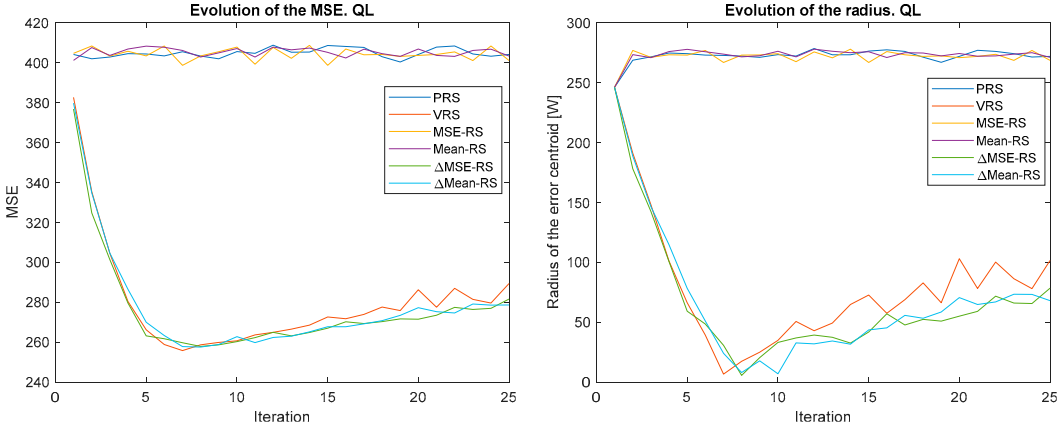

Figure 9. Evolution of the MSE (**left**) and error centroid radius (**right**) for different reward strategies and update policy Q-learning (QL).

The MSE and radius decrease to a minimum, which is typically reached between iterations 5 and 10. The MSE minimum is greater than 260 and is 0 for the radius. The MSE minimum is high because, as explained, the first peak in power output cannot be avoided, it cannot be improved by learning. As expected, the smallest MSE errors correspond to the smallest values of the radius.

Another remarkable result is that iteration by iteration learning is not observed when O-P rewards are applied. This is because, as stated, these strategies do not promote exploration and optimal actions are not discovered. As will be shown in Section 4.3, this problem is solved when O-P rewards are combined with P-N rewards. It can also be highlighted how all the P-N rewards converge at approximately the same speed up to the minimum value, but this speed is different for each policy update algorithm. From this point on, there are major differences between the P-N reward strategies. For some policy update algorithms (OR, LF-LR, and QL), these differences increase over time, while for the rest, they decrease.

The OR strategy is the one that converges the fastest to the minimum, but from this point, the MSE grows and becomes more unstable. Therefore, it is not recommended in the long term. However, SAR provides a good balance between convergence speed and stability. When it is used, the MSE for the three P-N reward strategies converges to the same value.

As expected, OL-LR and SAR produce very similar results because the only difference between them is that in the former, the rewards are multiplied by a constant. As the rewards are higher, the actions are reinforced more and there are fewer jumps between actions. This can be seen in Figures 5 and 7.

4.1. Influence of the Reward Window

Several of the reward strategies calculate the reward applying a time window, that is, considering N previous samples of the error signal, specifically: MSE-RS, MEAN-RS, ΔMSE-RS, ΔMEAN-RS. To evaluate the influence of the size of this window, several experiments have been carried out varying this parameter. In all of them, the policy update algorithm has been SAR.

Figure 10 (left) shows the results when ΔMSE-RS is applied and Figure 10 (right) for ΔMEAN-RS. Each line is associated with a different window size and is represented in a different color. The reward is a dimensionless parameter as table $T^{\pi}_{(s_t,\, a_t)}$ is dimensionless. The legend shows the size of the window in seconds. The value −1 indicates that all the previous values, from instant 0 of the simulation, have been taken into account to obtain the reward. That is, the size of the window is variable and increases in each control period, and covers from the start of the simulation to the current moment. MSE-RS and MEAN-RS have not been included as, as explained, they are O-P reward strategies and do not converge without forced exploration.

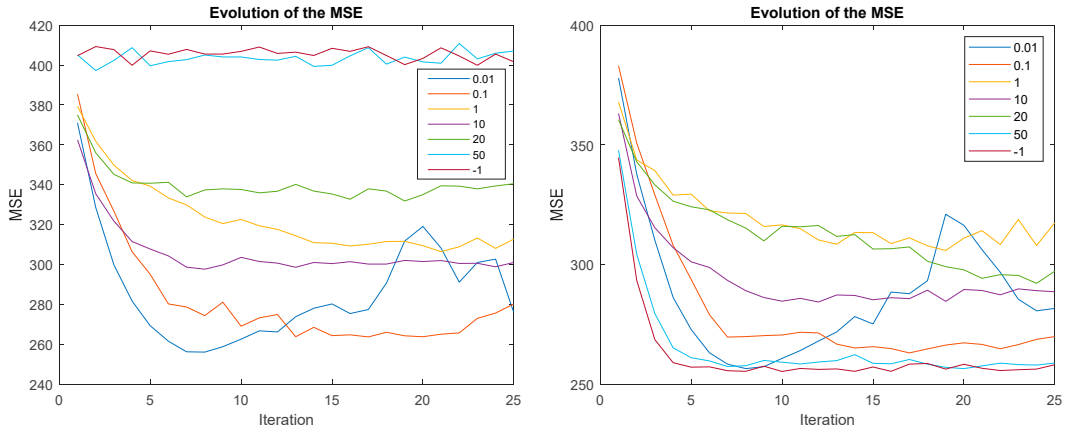

Figure 10. Evolution of the MSE for ΔMSE-RS (**left**) and ΔMEAN-RS (**right**) and different temporal window.

In general, a small window size results in a faster convergence to the minimum, but if the size is too small it can cause oscillations after the absolute minimum. This happens with a window of 0.01s, the MSE oscillates and is even less stable for ΔMEAN-RS. A small window size produces noisy rewards. This parameter seems to be related to the control period; a size smaller than the control period produces oscillations.

For the ΔMSE-RS strategy, the convergence speed decreases with the size of the window up to 1 s. For smaller window sizes it does not converge. This can be explained as if the window is longer than the control period, the window can be divided into two parts: the value of a control period preceding the end of the Tw_2 window, and the remaining part from the beginning of the Tw_1 window (Figure 11). An action performed at t_{i-1} produces an effect that is evaluated when the reward is calculated at t_i. When the size of the window grows during the control period, the Tw_1 part also grows, but Tw_2 remains invariant. To produce positive rewards, it is necessary to reduce the MSE, therefore, during Tw_2, the increases in Tw_1 should be compensated. A larger Tw_1 would give a larger accumulated error in this part, which would be more difficult to compensate during Tw_2 since only the squared error can be positive. It can then be concluded that the optimal window size for ΔMSE-RS is the control period, in this case, 100 ms.

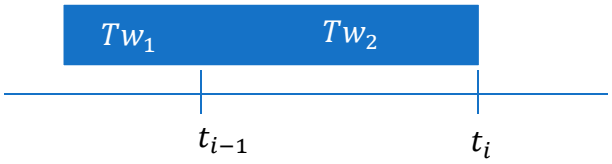

Figure 11. Reward time window regarding the control period.

The behavior of ΔMEAN-RS with respect to the size of the window is similar up to a size of around 20 s; from this value increasing the window size accelerates the convergence and decreases the MSE. This is because a larger window size implies a longer Tw_1 part (Figure 11). However, unlike ΔMSE-RS, a longer Tw_1 produces less accumulated error in Tw_1 since, in this case, the positive errors compensate for the negative ones, and the accumulated error tends to 0. Therefore, Tw_2 has a greater influence on the window, and learning is faster.

Figure 12 shows the variation of the MSE with the size of the reward window, at iteration 5. It is possible to observe how the MSE grows until the size of the window is 1 s, decreases until a size around 10 s, and then it grows again until around 25 s, which continues to grow for ΔMSE-RS and decreases for ΔMEAN-RS. The numerical values of these local minima and maxima are related to the duration of

the initial peak (Figure 2). The O-P rewards have also been represented with different reward windows. It is possible to observe how for long windows, the ΔMSE-RS tends to behave like the O-P reward strategies and reaches the same values.

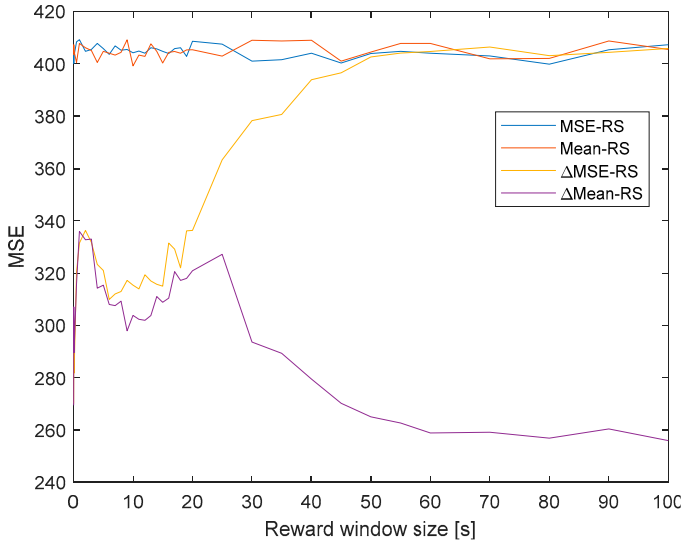

Figure 12. Variation of the MSE with the size of the reward window for different reward strategies.

4.2. Influence of the Size of the Reward

Up to this subsection, the reward mechanism provides a variable size reward/punishment depending on how good the previous action was. Better/worse actions give greater positive/negative rewards. In this section the case of all rewards and punishments having the same size is analyzed. To do so, the P-N reward strategies are binarized, that is, the value +r is assigned if the reward is positive and -r if it is negative. Several experiments have been carried out varying this parameter r to check its influence. In all experiments the policy update algorithm is SAR and the window size is 100 ms.

The results are shown in Figures 13–15, on the left the evolution of the MSE and on the right the evolution of the variance. Each line represents a different size of reward with a color code. The legend indicates the size of the reward. "Var" indicates that the reward strategy is not binarized and therefore the size of the reward is variable.

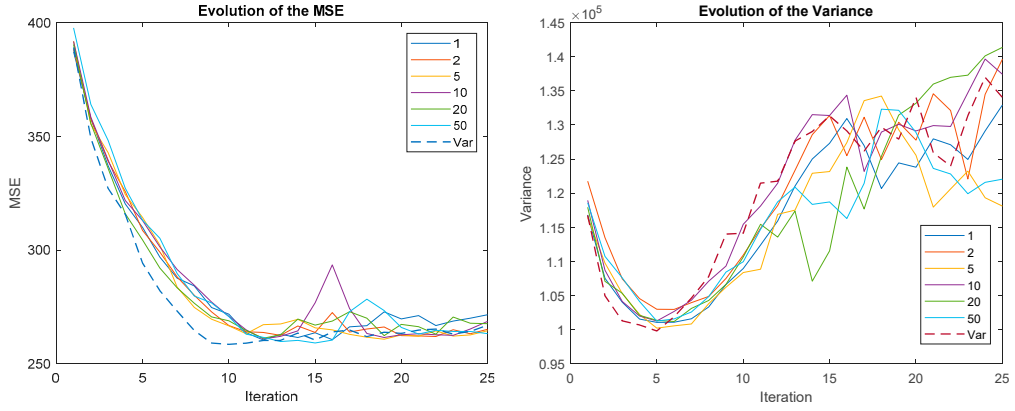

Figure 13. Evolution of the MSE (**left**) and variance (**right**) for different reward sizes when velocity reward strategy (VRS) is applied.

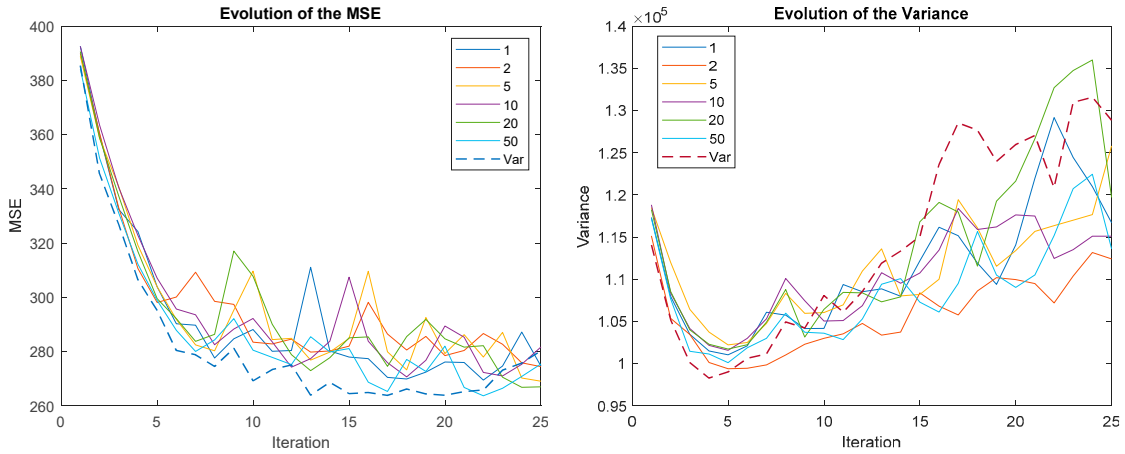

Figure 14. Evolution of the MSE (**left**) and variance (**right**) for different reward sizes when ΔMSE-RS is applied.

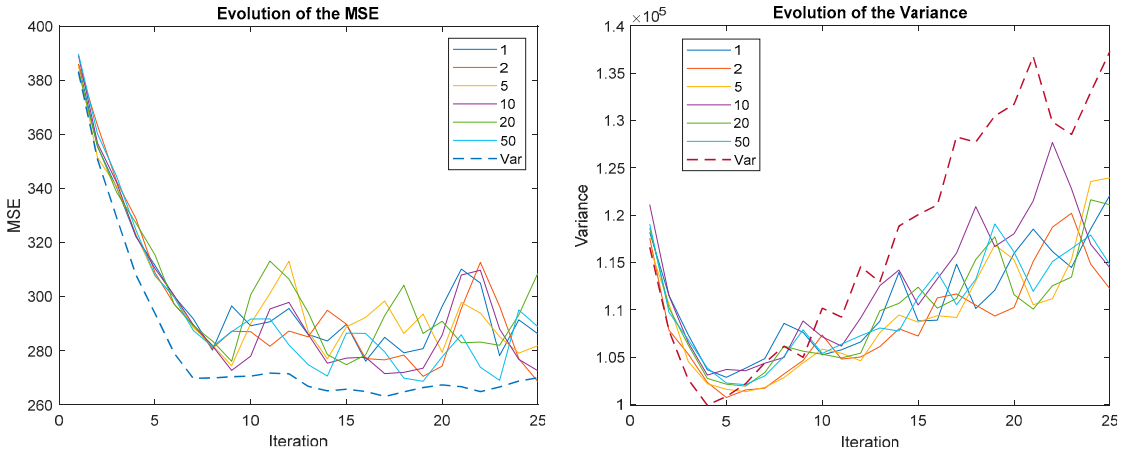

Figure 15. Evolution of the MSE (**left**) and variance (**right**) for different reward sizes when ΔMean-RS is applied.

It can be seen how, in all cases, the MSE is much better when the size of the reward is not binarized. However, the variation is similar or even worse. It has already been explained how better MSE performance typically leads to greater variance. Another interesting result is that the speed of convergence does not depend on the size of the reward, all the curves provide similar results up to the absolute minimum. However, from this point on, oscillations appear in the MSE and vary with size.

The VSE reward strategy is the least susceptible to variations in reward size. For ΔMSE-RS, the oscillations seem to depend on the size of the reward: the larger the size, the greater the oscillations; however, its amplitude decreases with time. For ΔMEAN-RS this relationship is not so clear and, what is worse, the amplitude of the oscillations seems to increase with time. Therefore, it is not recommended to use a fixed reward size with ΔMSE-RS and ΔMEAN-RS, it is preferable to use a variable reward size.

4.3. Combination of Individual Reward Strategies

As discussed above, O-P reward strategies do not converge due to the lack of exploration of the entire space of possible actions. To solve this, ϵ-greedy methods can be applied [40] or they can be combined with P-N reward strategies. This last option is explored in this section. Different experiments have been carried out combining reward strategies O-P with P-N and their performance is studied. In all experiments, the policy update algorithm is SAR and the window size is 100 ms.

Figure 16 shows the results of applying PRS (blue) and VRS (red), PRS·VRS (yellow) and PRS+K·VRS (purple), with K = 2. It is possible to see how the multiplication makes PRS converge. However, this is not true with addition, as this operator cannot convert PRS to a P-N reward strategy in all cases. It depends on the size of each individual reward and on the value of K. Therefore, the addition operator will not be used from now on to combine the rewards. Another interesting result is that the PRS·VRS combination smoothens the VRS curve, the result converges at a slightly slower speed but is more stable for iterations over 15. This combination presents less variance than VRS in general.

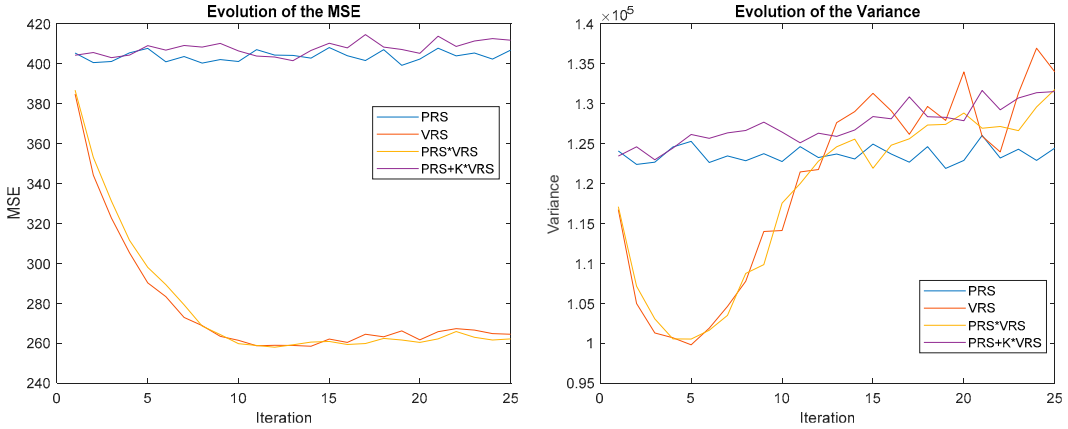

Figure 16. Evolution of the MSE (**left**) and variance (**right**) for different combinations of position reward strategy (PRS) and velocity reward strategy (VRS).

Figure 17 shows the results of MSE-RS (blue), MEAN-RS (red), MSE-RS·VRS (yellow), Mean-RS ·VRS (purple), and VRS (green). All strategies combined converge at roughly the same speed, slightly slower than VRS. During iteration 10, the performance of VRS and MSE-RS·VRS is similar; however, Mean-RS·VRS worsens its performance over VRS.

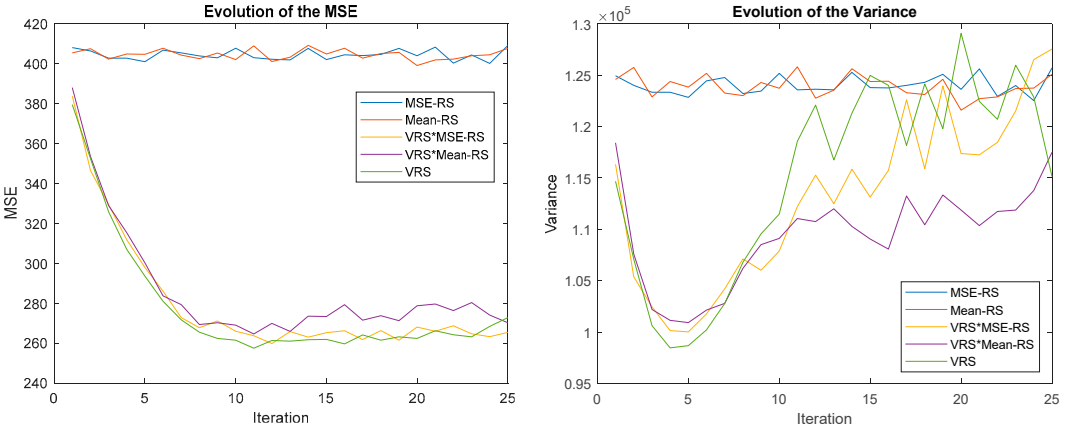

Figure 17. Evolution of the MSE (**left**) and variance (**right**) for combinations of MSE-RS, Mean-RS, and VRS.

In the following experiment, the P-N ΔMSE-RS reward strategy is combined with each O-P reward strategy. Figure 18 shows the results, PRS (dark blue), MSE-RS (red), ΔMSE-RS (purple), PRS·ΔMSE-RS (green), MSE-RS·ΔMSE-RS (light blue) and Mean-RS·ΔMSE-RS (magenta). Again it is possible to observe how all the combined strategies and ΔMSE-RS converge at the same speed until iteration 5. From this point on ΔMSE-RS and PRS·ΔMSE-RS give smaller error MSE. On the other hand, the variance decreases until iteration 5, after which it grows, although less for MSE-RS·ΔMSE-RS and Mean-RS·ΔMSE-RS. Furthermore, PRS·ΔMSE-RS tends to be slightly smaller than ΔMSE-RS.

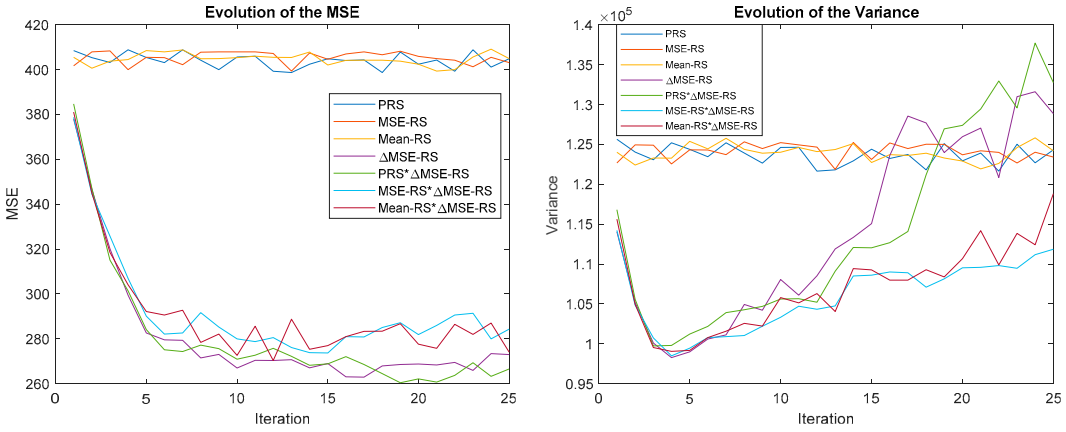

Figure 18. Evolution of the MSE (**left**) and variance (**right**) for combinations of PRS, MSE-RS, Mean-RS, and ΔMSE-RS.

In this last experiment, the P-N ΔMean-RS strategy is combined with each O-P strategy. Figure 19 shows the results, with PRS (dark blue), MSE-RS (red), ΔMean-RS (purple), PRS·ΔMean-RS (green), Mean-RS·ΔMSE-RS (light blue) and Mean-RS·ΔMean-RS (magenta). Again it is possible to observe how all the combined strategies and ΔMean-RS converge at the same speed until approximately iteration 5, where PRS·ΔMean-RS improves the MSE. The combination of these strategies provides a better result than their individual application. Furthermore, the variance also decreases with iterations. The combination of ΔMean-RS with Mean-RS and MSE-RS only offers an appreciable improvement in variance.

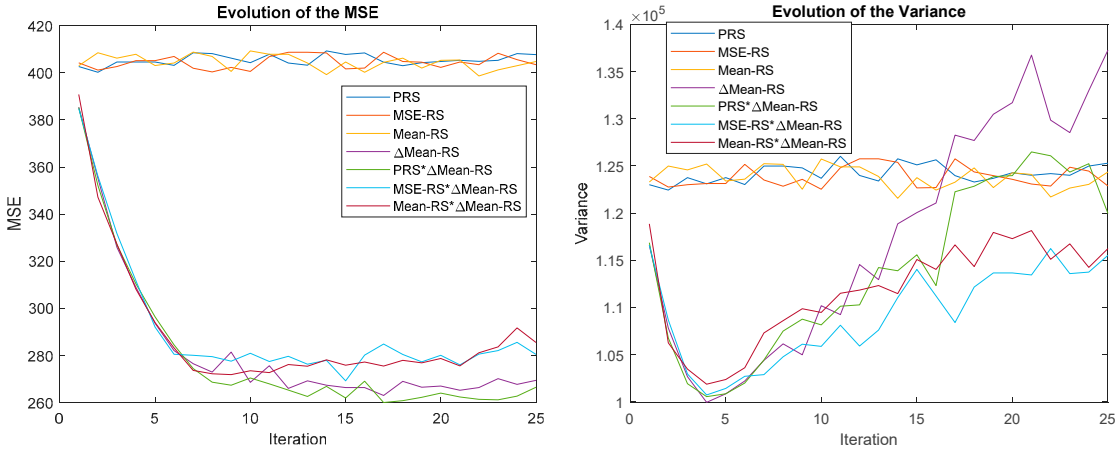

Figure 19. Evolution of the MSE (**left**) and variance (**right**) for combinations of PRS, MSE-RS, Mean-RS and ΔMean-RS.

Table 5 compiles the numerical results of the previous experiments. The KPIs have been measured at iteration 25. The best MSE is obtained by the combination of PRS·VRS and the best mean value and variance by MSE-RS·ΔMSE-RS. In general, it is possible to observe how the combination with PRS decreases the MSE and the combination with MEAN-RS and MSE-RS improves the mean value and the variance.

In view of these results, it is possible to conclude that the combination of individual rewards of O-P and P-N is beneficial since it converges learning by O-P rewards, without worsening the speed of convergence and learning is more stable.

Table 5. MSE, mean value and variance for different combinations of reward strategies.

Reward	MSE [W]	KPI Mean [kW]	KPI Var [kW]
PRS	404.10	6.80	124
MSE-RS	403.03	6.80	123
MEAN_RS	404.81	6.80	124
VRS	272.90	7.08	115
ΔMSE-RS	267.91	7.07	135
ΔMEAN-RS	269.61	7.08	137
PRS·VRS	262.20	7.07	131
VRS·MSE-RS	265.50	7.05	127
VRS·MEAN-RS	270.30	7.04	117
PRS·ΔMSE-RS	264.99	7.80	125
MSE-RS·ΔMSE-RS	284.39	7.00	111
MEAN-RS·ΔMSE-RS	274.10	7.03	118
PRS·ΔMEAN-RS	266.61	7.04	119
MSE-RS·ΔMEAN-RS	280.54	7.01	115
MEAN-RS·ΔMEAN-RS	285.60	7.00	116

5. Conclusions and Future Works

In this work, a RL-inspired pitch control strategy of a wind turbine is presented. The controller is composed by a state estimator, a policy update algorithm, a reward strategy, and an actuator. The reward strategies are specifically designed to consider the energy deviation from the rated power aiming to improve the efficiency of the WT.

The performance of the controller has been tested in simulation on a 7 kW wind turbine model with varying different configuration parameters, especially those related to rewards. The RL-inspired controller performance is compared to a tuned PID giving better results in terms of system response.

The relationship of the rewards with the exploration-exploitation dilemma and the ϵ-greedy methods is studied. On this basis, two novel categories of reward strategies are proposed, O-P (Only-Positive) and P-N (Positive-Negative) rewards. The performance of the controller has been analyzed for different reward strategies and different policy update algorithms. The individual behavior of these methods and their combination have also been studied. It has been shown that the P-N rewards improve the learning convergence and the performance of the controller.

The influence of the control parameters and RL configuration on the turbine response has been throughout analyzed and different conclusions regarding learning speed and convergence have been drawn. It is worth noting the relationship between the size of the reward and the need for forced exploration for the convergence of learning.

Some potential challenges may include to extend this proposal to design model-free general purpose tracking controllers. Another research line would be to incorporate risk detection in the P-N reward mechanisms to perform safe non-forced exploration for systems, which must fulfill safety requirements during the learning process.

As other future works, it would be desirable to test the proposal in a real prototype of a wind turbine. Also, it would be interesting to apply this control strategy to a larger turbine, and see if this control action affects the stability of a floating offshore wind turbine.

Author Contributions: Conceptualization, J.E.S.-G. and M.S.; methodology, J.E.S.-G. and M.S.; software, J.E.S.-G.; validation, J.E.S.-G.; formal analysis, J.E.S.-G.; investigation, J.E.S.-G. and M.S.; resources, J.E.S.-G. and M.S.; data curation, J.E.S.-G.; writing—original draft preparation, J.E.S.-G. and M.S.; writing—review and editing, J.E.S.-G. and M.S.; visualization, J.E.S.-G. and M.S.; supervision, J.E.S.-G. and M.S.; project administration, M.S.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Spanish Ministry of Science, Innovation and Universities under MCI/AEI/FEDER Project number RTI2018-094902-B-C21.

Acknowledgments: An earlier version of this paper was presented at (21st International Conference on Intelligent Data Engineering and Automated Learning—IDEAL 2020) [36].

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

References

1. Our World in Data. 2020. Available online: https://ourworldindata.org/renewable-energy (accessed on 8 September 2020).
2. Mikati, M.; Santos, M.; Armenta, C. Electric grid dependence on the configuration of a small-scale wind and solar power hybrid system. *Renew. Energy* **2013**, *57*, 587–593. [CrossRef]
3. Tomás-Rodríguez, M.; Santos, M. Modelling and control of floating offshore wind turbines. *Rev. Iberoam. Autom. Inf. Ind.* **2019**, *16*, 381–390. [CrossRef]
4. Kim, D.; Lee, D. Hierarchical fault-tolerant control using model predictive control for wind turbine pitch actuator faults. *Energies* **2019**, *12*, 3097. [CrossRef]
5. Bianchi, F.D.; De Battista, H.; Mantz, R.J. *Wind Turbine Control Systems: Principles, Modelling and Gain Scheduling Design*; Springer: London, UK, 2006.
6. Salle, S.D.L.; Reardon, D.; Leithead, W.E.; Grilmble, M.J. Review of wind turbine control. *Int. J. Control* **1990**, *52*, 1295–1310. [CrossRef]
7. Acho, L. A proportional plus a hysteretic term control design: A throttle experimental emulation to wind turbines pitch control. *Energies* **2019**, *12*, 1961. [CrossRef]
8. Astolfi, D.; Castellani, F.; Berno, F.; Terzi, L. Numerical and experimental methods for the assessment of wind turbine control upgrades. *Appl. Sci.* **2018**, *8*, 2639. [CrossRef]
9. Liu, J.; Zhou, F.; Zhao, C.; Wang, Z. A PI-type sliding mode controller design for PMSG-based wind turbine. *Complexity* **2019**, *2019*, 2538206. [CrossRef]
10. Nasiri, M.; Mobayen, S.; Zhu, Q.M. Super-twisting sliding mode control for gearless PMSG-based wind turbine. *Complexity* **2019**, *2019*, 6141607. [CrossRef]
11. Colombo, L.; Corradini, M.L.; Ippoliti, G.; Orlando, G. Pitch angle control of a wind turbine operating above the rated wind speed: A sliding mode control approach. *ISA Trans.* **2020**, *96*, 95–102. [CrossRef]
12. Yin, X.; Zhang, W.; Jiang, Z.; Pan, L. Adaptive robust integral sliding mode pitch angle control of an electro-hydraulic servo pitch system for wind turbine. *Mech. Syst. Signal Process.* **2019**, *133*, 105704. [CrossRef]
13. Bashetty, S.; Guillamon, J.I.; Mutnuri, S.S.; Ozcelik, S. Design of a Robust Adaptive Controller for the Pitch and Torque Control of Wind Turbines. *Energies* **2020**, *13*, 1195. [CrossRef]
14. Rocha, M.M.; da Silva, J.P.; De Sena, F.D.C.B. Simulation of a fuzzy control applied to a variable speed wind system connected to the electrical network. *IEEE Latin Am. Trans.* **2018**, *16*, 521–526. [CrossRef]
15. Rubio, P.M.; Quijano, J.F.; López, P.Z. Intelligent control for improving the efficiency of a hybrid semi- submersible platform with wind turbine and wave energy converters. *Rev. Iberoam. Autom. Inf. Ind.* **2019**, *16*, 480–491.
16. Marugán, A.P.; Márquez, F.P.G.; Perez, J.M.P.; Ruiz-Hernández, D. A survey of artificial neural network in wind energy systems. *Appl. Energy* **2018**, *228*, 1822–1836. [CrossRef]
17. Asghar, A.B.; Liu, X. Adaptive neuro-fuzzy algorithm to estimate effective wind speed and optimal rotor speed for variable-speed wind turbine. *Neurocomputing* **2018**, *272*, 495–504. [CrossRef]
18. Sierra-García, J.E.; Santos, M. Performance Analysis of a Wind Turbine Pitch Neurocontroller with Unsupervised Learning. *Complexity* **2020**, *2020*, 4681767. [CrossRef]
19. Chavero-Navarrete, E.; Trejo-Perea, M.; Jáuregui-Correa, J.C.; Carrillo-Serrano, R.V.; Ronquillo-Lomeli, G.; Ríos-Moreno, J.G. Hierarchical Pitch Control for Small Wind Turbines Based on Fuzzy Logic and Anticipated Wind Speed Measurement. *Appl. Sci.* **2020**, *10*, 4592. [CrossRef]
20. Lotfy, M.E.; Senjyu, T.; Farahat, M.A.F.; Abdel-Gawad, A.F.; Lei, L.; Datta, M. Hybrid genetic algorithm fuzzy-based control schemes for small power system with high-penetration wind farms. *Appl. Sci.* **2018**, *8*, 373. [CrossRef]
21. Li, M.; Wang, S.; Fang, S.; Zhao, J. Anomaly Detection of Wind Turbines Based on Deep Small-World Neural Network. *Appl. Sci.* **2020**, *10*, 1243. [CrossRef]

22. Wang, Z.; Hong, T. Reinforcement learning for building controls: The opportunities and challenges. *Appl. Energy* **2020**, *269*, 115036. [CrossRef]
23. Khamparia, A.; Singh, K.M. A systematic review on deep learning architectures and applications. *Expert Syst.* **2019**, *36*, e12400. [CrossRef]
24. Zhang, Z.; Zhang, D.; Qiu, R.C. Deep reinforcement learning for power system applications: An overview. *CSEE J. Power Energy Syst.* **2019**, *6*, 213–225.
25. Fernandez-Gauna, B.; Fernandez-Gamiz, U.; Grana, M. Variable speed wind turbine controller adaptation by reinforcement learning. *Integr. Comput.-Aided Eng.* **2017**, *24*, 27–39. [CrossRef]
26. Fernandez-Gauna, B.; Osa, J.L.; Graña, M. Experiments of conditioned reinforcement learning in continuous space control tasks. *Neurocomputing* **2018**, *271*, 38–47. [CrossRef]
27. Abouheaf, M.; Gueaieb, W.; Sharaf, A. Model-free adaptive learning control scheme for wind turbines with doubly fed induction generators. *IET Renew. Power Gener.* **2018**, *12*, 1675–1686. [CrossRef]
28. Sedighizadeh, M.; Rezazadeh, A. Adaptive PID controller based on reinforcement learning for wind turbine control. *Proc. World Acad. Sci. Eng. Technol.* **2008**, *27*, 257–262.
29. Saénz-Aguirre, A.; Zulueta, E.; Fernández-Gamiz, U.; Lozano, J.; Lopez-Guede, J.M. Artificial neural network based reinforcement learning for wind turbine yaw control. *Energies* **2019**, *12*, 436. [CrossRef]
30. Saenz-Aguirre, A.; Zulueta, E.; Fernandez-Gamiz, U.; Ulazia, A.; Teso-Fz-Betono, D. Performance enhancement of the artificial neural network–based reinforcement learning for wind turbine yaw control. *Wind Energy* **2020**, *23*, 676–690. [CrossRef]
31. Kuznetsova, E.; Li, Y.F.; Ruiz, C.; Zio, E.; Ault, G.; Bell, K. Reinforcement learning for microgrid energy management. *Energy* **2013**, *59*, 133–146. [CrossRef]
32. Tomin, N.; Kurbatsky, V.; Guliyev, H. Intelligent control of a wind turbine based on reinforcement learning. In Proceedings of the 2019 16th Conference on Electrical Machines, Drives and Power Systems ELMA, Varna, Bulgaria, 6–8 June 2019; pp. 1–6.
33. Hosseini, E.; Aghadavoodi, E.; Ramírez, L.M.F. Improving response of wind turbines by pitch angle controller based on gain-scheduled recurrent ANFIS type 2 with passive reinforcement learning. *Renew. Energy* **2020**, *157*, 897–910. [CrossRef]
34. Chen, P.; Han, D.; Tan, F.; Wang, J. Reinforcement-based robust variable pitch control of wind turbines. *IEEE Access* **2020**, *8*, 20493–20502. [CrossRef]
35. 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*, 6912–6921. [CrossRef]
36. Sierra-García, J.E.; Santos, M. Wind Turbine Pitch Control First Approach based on Reinforcement Learning. In Proceedings of the 21st International Conference on Intelligent Data Engineering and Automated Learning—IDEAL Guimarães, Guimarães, Portugal, 4–6 November 2020.
37. Mikati, M.; Santos, M.; Armenta, C. Modeling and Simulation of a Hybrid Wind and Solar Power System for the Analysis of Electricity Grid Dependency. *Rev. Iberoam. Autom. Inf. Ind.* **2012**, *9*, 267–281. [CrossRef]
38. Jiang, M.; Hai, T.; Pan, Z.; Wang, H.; Jia, Y.; Deng, C. Multi-agent deep reinforcement learning for multi-object tracker. *IEEE Access* **2019**, *7*, 32400–32407. [CrossRef]
39. Santos, M.; López, V.; Botella, G. Dyna-H: A heuristic planning reinforcement learning algorithm applied to role-playing game strategy decision systems. *Knowl.-Based Syst.* **2012**, *32*, 28–36. [CrossRef]
40. Sutton, R.S.; Barto, A.G. *Reinforcement Learning an Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2015; in progress.