Hindawi
Complexity
Volume 2020, Article ID 4681767, 15 pages
https://doi.org/10.1155/2020/4681767

Research Article

Performance Analysis of a Wind Turbine Pitch Neurocontroller with Unsupervised Learning

J. Enrique Sierra-García[1] **and Matilde Santos**[2]

[1]*Department of Electromechanical Engineering, University of Burgos, Burgos 09006, Spain*
[2]*Technological Knowledge Institute, Complutense University of Madrid, Madrid 28040, Spain*

Correspondence should be addressed to J. Enrique Sierra-García; jesierra@ubu.es

Received 16 July 2020; Revised 18 August 2020; Accepted 29 August 2020; Published 15 September 2020

Academic Editor: Ramon Costa-Castelló

In this work, a neural controller for wind turbine pitch control is presented. The controller is based on a radial basis function (RBF) network with unsupervised learning algorithm. The RBF network uses the error between the output power and the rated power and its derivative as inputs, while the integral of the error feeds the learning algorithm. A performance analysis of this neurocontrol strategy is carried out, showing the influence of the RBF parameters, wind speed, learning parameters, and control period, on the system response. The neurocontroller has been compared with a proportional-integral-derivative (PID) regulator for the same small wind turbine, obtaining better results. Simulation results show how the learning algorithm allows the neural network to adjust the proper control law to stabilize the output power around the rated power and reduce the mean squared error (MSE) over time.

1. Introduction

Green directives in many countries promote the use of renewable energies to improve the sustainability of worldwide energy systems. Indeed, the number of terawatts produced by clean energies is growing each year [1]. Among clean energies, wind is the second most used natural resource after hydropower, due to its high efficiency. Although a mature technology, there are still many engineering challenges related to wind turbines (WTs) that must be addressed [2].

Depending on the type of wind turbine, different control actions can be applied, namely: the pitch angle of the blades or rotor control, which is used as a brake to maintain the rated power of the turbine once the wind surpasses certain threshold; the yaw angle, which is used to change the attitude of the nacelle to match the wind stream direction; and finally, the generator speed control that seeks to reach the optimal rotor velocity when the wind is below the rated-output speed. The WT controller is in charge of managing all of these mechanisms to optimize the efficiency of the system while the safety must be guaranteed under all possible wind conditions. This fact may be even more critical for floating offshore wind turbines (FOWTs) as it has been proved that the control system can affect the stability of the floating device [3, 4].

The pitch control of a wind turbine is a complex task itself due to the highly nonlinear behaviour of these devices, the coupling between the internal variables, and because they are subjected to uncertain and varying parameters due to external loads, mainly wind, and in the case of FOWT, waves and currents also. These reasons have led to explore intelligent control techniques to tackle these challenges [5]. Among traditional control solutions, sliding mode control has been recently applied with successful results, such as in [6], where 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. [7] proposed a supertwisting sliding mode control for a gearless wind turbine by a permanent magnet synchronous generator. A robust SMC approach is also proposed in [8], where authors use the blade pitch as control input, in order to regulate the rotor speed to a fixed rated value. In [9], 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.

Regarding intelligent control, fuzzy logic has been widely applied to the wind turbine pitch control. For example, in [10], pitch angle fuzzy control is proposed and compared to a PI controller for real weather characteristics and load variations. Rocha et al. [11] applied a fuzzy controller to a variable speed wind turbine and compared the results with a classical proportional controller in terms of system response characteristics. Rubio et al. [12] presented a fuzzy logic-based control system for the control of a wind turbine installed on a semisubmersible platform. But application of neural networks to turbine pitch control is scarcer, maybe due to the lack of real data to train the network [13]. However, Asghar and Liu [14] designed a neurofuzzy algorithm for optimal rotor speed of a wind turbine. In [15], artificial neural network-based reinforcement learning for WT yaw control is presented. In [5], a passive reinforcement learning algorithm solved by particle swarm optimization is used to handle an adaptive neurofuzzy type-2 inference system for controlling the pitch angle of a real wind turbine. In [16], a robust H∞ observer-based fuzzy controller is designed to control the turbine using the estimated wind speed. Two artificial neural networks are used to accurately model the aerodynamic curves. From a different point of view, in [17], the authors proposed an information management system based on mixed integer linear programming (MILP) for a wind power producer having an energy storage system and participating in a day-ahead electricity market.

In this work, we have focused on the pitch control of a small wind turbine. Based on the neural control strategy proposed in [18], we have extended it to deal with the dynamics of the pitch actuator. Besides, the derivative and the integration of the power error have been added as inputs to the learning algorithm. This way the error variation and the past error values are considered and used to update the weights of the neural network; this helps to accelerate the learning process. The main contribution of this paper is twofold. On the one hand, a radial basis network (RBF) wind turbine pitch controller is designed and implemented. This controller uses the output power to update the weights of the neural network in an unsupervised way. On the other hand, a detailed analysis has been carried out on how the configuration of the neural network, the learning algorithm, and the controller parameters affect the control performance and the evolution of the error. Another advantage of the approach here presented is that, in contrast to traditional controllers which have different control schemes for different wind speed regions, only one controller is used for all operational regions of the wind turbine.

The rest of the paper is organized as follows. Section 2 describes the model of the small wind turbine used. Section 3 explains the neural controller architecture and the unsupervised learning strategy. The results for different neural network configuration and learning parameters are analysed and discussed in Section 4. The paper ends with the conclusions and future works.

2. Wind Turbine Model Description

The model of a small 7 kW wind turbine is developed. The ratio of the gear box is set to 1, so the rotor torque is the same as the mechanical torque of the generator, T_m (Nm), given by the following equation [19]:

$$T_m = \frac{C_p(\lambda, \theta)\rho A v^3}{2w}, \tag{1}$$

where C_p is the power coefficient; ρ is the air density (kg/m^3); A is the area swept by the turbine blades (m^2); v is wind speed (m/s); and w is the angular rotor speed (rad/s). The blade swept area can be approximated by $A = \pi R^2$, where R is the radius or blade length.

The power coefficient is usually determined experimentally for each turbine. There are different expressions to approximate C_p; in this case, it has been calculated as a function of the tip speed ratio λ and the blade pitch angle θ (rad):

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

$$C_p(\lambda_i, \theta) = c_1\left[\frac{c_2}{\lambda_i} - c_3\theta - c_4\theta^{c_5} - c_6\right]e^{-(c_7/\lambda_i)}, \tag{3}$$

where the values of the coefficients c_1 to c_9 depend on the characteristics of the wind turbine. The pitch angle θ is defined as the angle between the rotation plane and the blade cross section chord, and the tip speed ratio is given by the following equation:

$$\lambda = \frac{w \cdot R}{v}. \tag{4}$$

From equation (3), it is possible to observe how C_p decreases with the pitch angle. Indeed, when $\theta = 0$ (rad), the blades are pitched so the blade is all out and producing at its full potential, but with $\theta = (\pi/2)$ (rad), the blades are out of the wind.

The pitch actuator is modelled as a second-order system. This assumption is widely used to model pitch systems in wind turbines and other mechanical actuators [20]. In this case, θ_{ref} is the input of the pitch actuator and θ is its output:

$$\frac{\theta(s)}{\theta_{\text{ref}}(s)} = \frac{K_\theta}{T_\theta s^2 + s + K_\theta}. \tag{5}$$

Thus far, the model is focused on the mechanical aspects of the system. But dynamics of the generator combine the mechanical and electrical domains. The relation between the rotor angular speed w and the mechanical torque T_m in a continuous current generator is given by the following expressions [21]:

$$J\frac{dw}{dt} = T_m - T_{em} - K_f w, \tag{6}$$

$$T_{\text{em}} = K_g \cdot K_\phi \cdot I_a, \tag{7}$$

where T_{em} is the electromagnetic torque (Nm), J is the rotational inertia (kg·m^2), K_f is the friction coefficient (N·m·s/rad), K_g is a dimensionless constant of the generator, K_ϕ is the magnetic flow coupling constant (V s/rad), and I_a is the armature current (A).

The armature current of the generator is then given by the following equations:

$$L_a \frac{dI_a}{dt} = E_a - V - R_a I_a, \tag{8}$$

$$E_a = K_g \cdot K_\phi \cdot w, \tag{9}$$

where L_a is the armature inductance (H), E_a is the induced electromotive force (V), V is the generator output voltage (V), and R_a is the armature resistance (Ω). For simplicity, it is commonly assumed that the load is purely resistive, given by R_a. Thus, $V = R_L I_a$, and the output power (W) is $P_{out} = R_L I_a^2$.

By the combination of the previous equations (1)–(9), the following expressions summarize the dynamics of the wind turbine.

$$\dot{I}_a = \frac{1}{L_a}\left(K_g \cdot K_\phi \cdot w - (R_a + R_L) I_a\right), \tag{10}$$

$$\lambda_i = \left[\left(\frac{1}{\lambda + c_8}\right) - \left(\frac{c_9}{\theta^3 + 1}\right)\right]^{-1}, \tag{11}$$

$$C_p(\lambda_i, \theta) = c_1\left[\frac{c_2}{\lambda_i} - c_3\theta - c_4\theta^{c_5} - c_6\right] e^{-\left(c_7/\lambda_i\right)}, \tag{12}$$

$$\dot{w} = \frac{1}{2 \cdot J \cdot w}\left(C_p(\lambda_i, \theta) \cdot \rho\pi R^2 \cdot v^3\right) - \frac{1}{J}\left(K_g \cdot K_\phi \cdot I_a + K_f w\right), \tag{13}$$

$$\ddot{\theta} = \frac{1}{T_\theta}\left[K_\theta\left(\boldsymbol{\theta}_{ref} - \theta\right) - \dot{\theta}\right], \tag{14}$$

$$\mathbf{P}_{out} = R_L \cdot I_a^2. \tag{15}$$

In this work, we have focused on controlling the output power by means of the pitch angle, so the input control variable is θ_{ref} and the controlled output variable is P_{out} (boldfaced in equations (10)–(15). The state variables are I_a, w, θ, and $\dot{\theta}$.

The wind turbine parameters used during the simulations are shown in Table 1 [19].

3. Neural Pitch Control Strategy

3.1. Neural Controller Architecture. The architecture of the proposed wind turbine neural controller is shown in Figure 1. The error P_{err} is the difference between the power reference signal P_{ref} (rated power) and the power output. The nominal power of this wind turbine is 7 kW. Power error, P_{err}, and its derivative, $\dot{P}_{err}$, are saturated to maintain their values within a suitable range; the saturated signals are P_{err_S} and $\dot{P}_{err_S}$, respectively. They are the inputs of the radial basis neural network that is used to implement the controller. The output of the neural network RBF_o is biased for $(\pi/4)$ and goes through a saturation block to adapt it to the range $[0, (\pi/2)]$ (rad). The result of this process is the signal θ_{ref} that will be used as pitch reference of the wind turbine control.

The neural network must learn the control law f_c: $\mathbb{R}^2 \longrightarrow \mathbb{R}$, which will be able to stabilize the wind turbine output power around its nominal value. This function is not known beforehand. In other control schemes, the weights of the RBF network are updated using supervised learning. That requires a known input/output dataset to train the neural network. This way it generates the expected output when it receives a similar input to the ones used for the training. However, in our case, there are no labelled output data to train the network.

If we knew the correct pitch control signal for each P_{err} and $\dot{P}_{err}$, we would know the appropriate control law, and we would not need a neural network to learn it. For this reason, it is not possible to use supervised learning. That is why in this approach the learning algorithm receives the error signal P_{err}, its derivative, and its integral and combines them to generate the new weights of the neural network.

The equations of this neurocontrol strategy are the following:

$$P_{err}(t_i) = P_{ref}(t_{i-1}) - P_{out}(t_{i-1}), \tag{16}$$

$$\dot{P}_{err}(t_i) = \frac{1}{T_c}\left(P_{err}(t_i) - P_{err}(t_i - T_c)\right), \tag{17}$$

$$P_{err_S}(t_i) = \text{MIN}\left(P_{err_{MAX}}, \text{MAX}\left(P_{err_{MIN}}, P_{err}(t_i)\right)\right), \tag{18}$$

$$\dot{P}_{err_S}(t_i) = \text{MIN}\left(\dot{P}_{err_{MAX}}, \text{MAX}\left(\dot{P}_{err_{MIN}}, \dot{P}_{err}(t_i)\right)\right), \tag{19}$$

$$RBF_o(t_i) = f_{RBF}\left(P_{err_S}(t_i), \dot{P}_{err_S}(t_i), \overline{W}(t_{i-1})\right), \tag{20}$$

$$\overline{W}(t_i) = f_{learn}\left(P_{err_S}(t_i), \dot{P}_{err_S}(t_i), \overline{W}(t_{i-1})\right), \tag{21}$$

$$\theta_{ref}(t_i) = \text{MIN}\left(\frac{\pi}{2}, \text{MAX}\left(0, \frac{\pi}{4} - RBF_o(t_i)\right)\right), \tag{22}$$

where T_c is the control period (s); the maximum and minimum values of the variables, $[P_{err_{MIN}}, P_{err_{MAX}}, \dot{P}_{err_{MIN}}, \dot{P}_{err_{MAX}}] \in \mathscr{R}^4$, are the constants that will allow to adjust the range of the controller, with the constraints $P_{err_{MIN}} < P_{err_{MAX}}$ and $\dot{P}_{err_{MIN}} < \dot{P}_{err_{MAX}}$; f_{RBF} is the RBF function; and f_{learn} denotes the function of the learning algorithm.

The MIN and MAX operators in equations (18), (19), and (22) are applied to maintain the signal value within boundary conditions. The expression $\text{MIN}(v_1, \text{MAX}(v_2, v_3))$ sets v_1 as the upper boundary; v_2 as the lower limit; and v_3 as the signal to be saturated. The MAX operator holds v_3 beyond the lower bound. The output of the MAX operator is kept below the upper limit by the MIN operator.

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]

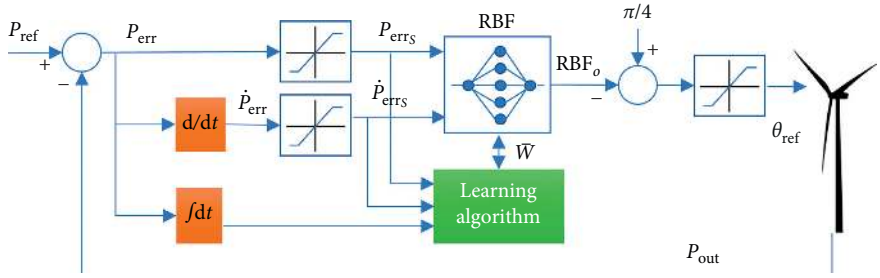


Figure 1: Architecture of the neural controller.

All the variables in equations (16)–(22) are updated each T_c second; otherwise, their values remain constant.

3.2. Setting Up RBF. The aim of the RBF is to compute the bidimensional function $f_c(P_{err}, \dot{P}_{err}) \longrightarrow RBF_o$ which implements the control law that it is able to stabilize P_{out} around P_{ref}. As it is well known, any derivable continuous function can be approximated by the sum of exponential functions. In this work, we take advantage of this property to approximate the control law by the RBF neural network. In order to map the input space to the output space, we discretize the bidimensional input space of the neural network (P_{err}, $\dot{P}_{err}$) applying a gridding. Figure 2 shows the $\Delta X \times \Delta Y$ grid. The centres of the neurons are initialized to the intersection points of the grid lines. This will set the precision of the error.

The number of rows and columns, the horizontal and vertical length of the cells, ΔX and ΔY, respectively, and the number of neurons M are related by the following expressions:

$$N_x = \left(\frac{1}{\Delta X}\left(P_{err_{MAX}} - P_{err_{MIN}}\right) + 1\right),$$
$$N_y = \left(\frac{1}{\Delta Y}\left(\dot{P}_{err_{MAX}} - \dot{P}_{err_{MIN}}\right) + 1\right), \quad (23)$$
$$M = N_x \cdot N_y.$$

where the number of horizontal lines is N_x, i.e., $N_x - 1$ rows, and the number of vertical lines is N_y. In order to ensure

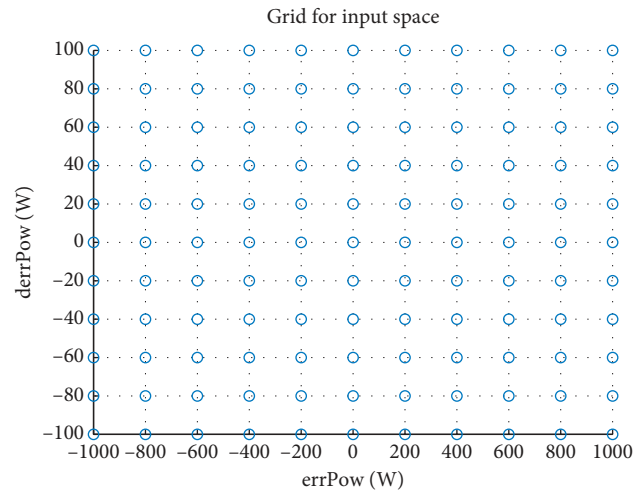


Figure 2: Gridding of the input space of the RBF.

that a horizontal line and a vertical line intersects the point (0, 0), N_x and N_y must be odd and bigger than 1.

Once ΔX and ΔY are determined, the centre of the neurons is obtained by the following equation:

$$c_{i1} = \left(i \,\mathrm{DIV}\, N_y\right) \cdot \Delta X + P_{err_{MIN}} \quad \forall i \in \mathcal{N} \bigcup 0 \,|\, i < M,$$
$$c_{i2} = \left(i \,\mathrm{MOD}\, N_y\right) \cdot \Delta Y + \dot{P}_{err_{MIN}} \quad \forall i \in \mathcal{N} \bigcup 0 \,|\, i < M, \quad (24)$$

where (c_{i1}, c_{i2}) is the centre of the i neuron.

The output of the RBF neural network (20) is then given by the following expressions (where the variable t_i has been omitted for sake of clarity):

$$f_{\mathrm{RBF}}\left(P_{\mathrm{err}_S}, \dot{P}_{\mathrm{err}_S}, \overline{W}\right) = \sum_{i=1}^{M} W_i \cdot e^{-\left(\mathrm{dist}\left(P_{err_S}, \dot{P}_{err_S}, c_{i1}, c_{i2}\right)/\sigma_i\right)}, \tag{25}$$

$$\mathrm{dist}\left(P_{\mathrm{err}_S}, \dot{P}_{\mathrm{err}_S}, c_{i1}, c_{i2}\right) = \sqrt{\frac{\left(c_{i1} - P_{err_S}\right)^2}{P^2_{\mathrm{err}_{\mathrm{MAX}}}} + \frac{\left(c_{i2} - \dot{P}_{\mathrm{err}_S}\right)^2}{\dot{P}^2_{\mathrm{err}_{\mathrm{MAX}}}}}, \tag{26}$$

where dist is a normalized distance measure, M is number of neurons in the hidden layer, W_i is the weight of the i-neuron, and σ_i is the width of the i-neuron activation function, which is normally the same for all neurons. The width of the neuron is also related to the error accuracy. The normalized distance (29) is calculated by the 2-D Euclidean distance once each 1-D distance has been normalized to the range [0, 1]. The range of $(c_{i1} - P_{\mathrm{err}_S})$ is $[-P_{\mathrm{err}_{\mathrm{MAX}}}, P_{\mathrm{err}_{\mathrm{MAX}}}]$, so division by $P_{\mathrm{err}_{\mathrm{MAX}}}$ normalizes it to [−1, 1], whereas the range of $(c_{i2} - \dot{P}_{\mathrm{err}_S})$ is $[-\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}, \dot{P}_{\mathrm{err}_{\mathrm{MAX}}}]$, and thus division by $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$ normalizes it to [−1, 1]. This way the output range of (29) is $[0, \sqrt{2}]$.

3.3. Unsupervised Learning Algorithm. The parameters to be updated by a learning algorithm in an RBF neural network are the centres of the RBF neurons, the σ_i parameters, and the output weights. As explained before, the centres of the neurons are equally distributed in all the input space. In addition, in this work, as is common, it is assumed that the entire input space is equally important when obtaining the output of the network; therefore, the σ_i parameters are set in advance to the same value for all the neurons. Thus, the learning algorithm only has to update the weights.

As said before, many control schemes with RBF neural networks use supervised learning to update the weights but this is not the case. There are no labelled output data to train the network, so the neural network must learn a control law previously unknown in an unsupervised way. This learning procedure is as follows.

The input space has been pseudodiscretized, placing the RBF neurons at the centres of the grid. Given a network input pair $(P_{\mathrm{err}}, \dot{P}_{\mathrm{err}})$, the neuron closest to this pair will have the biggest contribution to the output value of the mapping. Although it will not be the only neuron that influences the output, the contribution decreases with the distance and increases with the width of the activation function.

If the centres of the neurons are separated enough and the width of the activation function is correctly selected, the contribution of the surrounding neurons may be neglected and all points in the input space are discretized to the centre of its closest neuron. Therefore, by updating the weight W_i of the i-neuron, it is possible to adjust the output value of the input pair (c_{i1}, c_{i2}), due to the fact that in these points the value of the exponential function is 1. Thus, the closer the input pair is to the centre of some neuron, the better is the approximation of the RBF function f_c. The learning algorithm will be in charge of updating the weights of the network based on the output power errors, adjusting the mapping of the f_c function. In the output layer of the neural network, all the partial contributions of the neurons are linearly combined to obtain the output value (28).

In order to illustrate this unsupervised learning procedure, Figure 3(a) shows an example of the initial surface of the weights of the neural network with all set to 1. Figure 3(b) shows the corresponding pitch control law at the output of the $(\pi/4) - \mathrm{bias}$ before learning, and Figure 4(a) presents the control surface after applying the learning strategy, with the final values of the weights. The pitch control law, before and after the learning, is shown in Figures 3 and 4(b), respectively. It is possible to see, as expected, that positive errors increase the weights, bending upwards the surface and thus incrementing the output value of the neural network. This means reducing the pitch angle reference, θ_{ref}, and enlarging the output power.

In this work, we take as starting point the typical supervised learning strategy of an RBF to reduce the error at each iteration, given by equation (27), where T is the expected output value and RBF_o is the current output value.

$$W_j(t_i) = W_j(t_{i-1}) + \mu \cdot \left(T - \mathrm{RBF}_o\right) \cdot e^{-\left(\mathrm{dist}\left(in, c_j\right)/\sigma_j\right)} \quad \forall j \in \mathcal{N} \bigcup 0 \,|\, j < M. \tag{27}$$

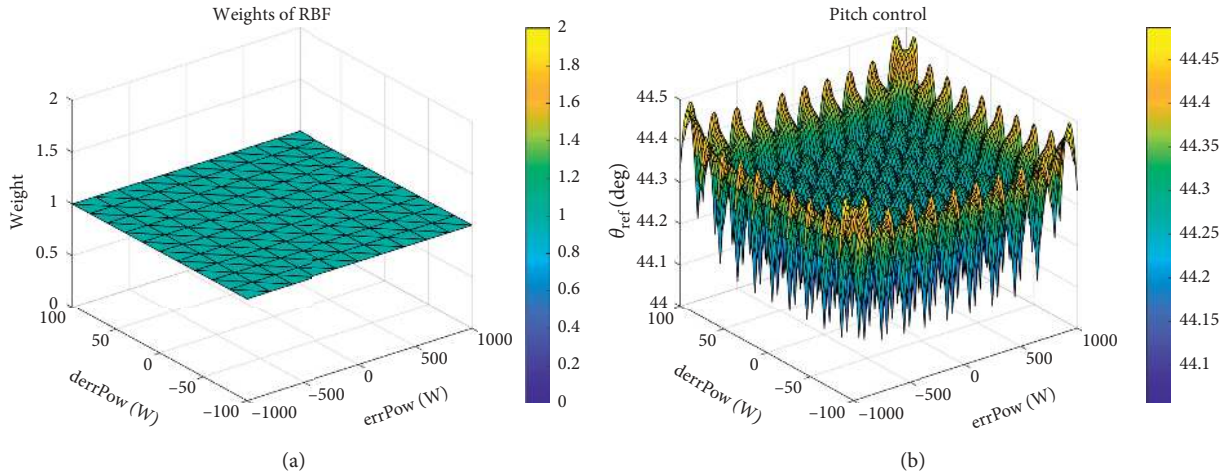
As T is not available, in order to make the power error zero, the term $(T - \mathrm{RBF}_o)$ is replaced by P_{err_S}. Equation (28) details how the function f_{learn} of equation (21) is then calculated, that is, how the weights of the RBF neural network are modified.

$$W_j(t_i) = W_j(t_{i-1}) + \mu \cdot \left(K_{pL} \cdot P_{\mathrm{err}_S}(t_i) + K_{\mathrm{dL}} \cdot \dot{P}_{\mathrm{err}_S}(t_i) + K_{iL} \cdot \int P_{\mathrm{err}}(t_i)\right) \cdot e^{-\left(\left(\mathrm{dist}\left(P_{\mathrm{err}_S}(t_i), \dot{P}_{\mathrm{err}_S}(t_i), c_{j1}, c_{j2}\right)\right)/\sigma_j\right)} \quad \forall j \in \mathcal{N} \bigcup 0 \,|\, j < M, \tag{28}$$

where μ is the learning rate and $[K_{pL}, K_{dL}, K_{iL}]$ are positive constants. As it may be observed, the exponential term is the same as in equations (25) and (26).

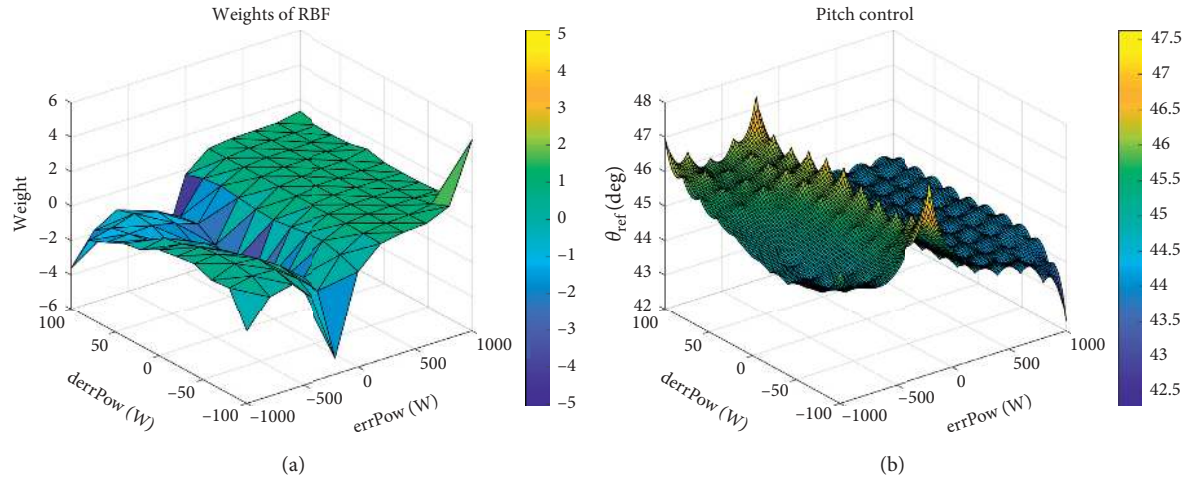
The following pseudocode details the unsupervised algorithm which updates the weights of the RBF network (Algorithm 1):

Here, M is the number of neurons, W is an array with the weights, Nx is the number of neurons in the x-axis of the grid input space, and Ny is the number of neurons in the y-axis of the grid input space. A learning threshold, minErr, is defined so errors below that value are discarded. The centres of the neurons are represented in the array cNet. The tuning parameters of the learning rate are mu, KP, KD, and KI; the control sampling time is Tc, F is an array with the output of



(a) (b)

Figure 3: Weight surface (a) and pitch control law (b) before learning.



(a) (b)

Figure 4: Weight surface (a) and pitch control law (b) after learning.

each exponential function of the RBF before the addition of all the neurons (28), and Fold is an array that saves the previous value of F.

The model of the WT, MODEL(), receives as input the pitch control reference. The function RBF() calculates (28), where DIV() is the integer division, MOD() is the module operator, ABS() is the absolute value operator, and MIN() and MAX() are the minimum and maximum functions. Therefore, the external parameters of the algorithm are PerrMin, PerrMax, dotPerrMin, dotPerrMax, Nx, Ny, mu, KP, KD, KI, and minErr.

At the beginning of the procedure, all variables are initialized, and the centres of the RBF are calculated. Then, the simulation is run each Ts second. The controller is updated each Tc second; therefore, Tc must be larger than Ts. Each control sample time, Tc, the output of the RBF, RBFout, and the WT pith reference, pitchCon, are obtained. If the error is above the threshold, a combination of the error, its derivative, and its integration is calculated (variable errM). Then, the array with the increments of the weights, Winc, is obtained from the previous F array, Fold, and the current error measurement, errM.

4. Performance Analysis of the Neurocontrol Strategy

A performance analysis of this unsupervised neurocontrol strategy has been carried out under different network configurations and varying some parameters of the learning algorithm and of the pitch control law. The software Matlab/Simulink has been used. The duration of each simulation is 100 s. In order to reduce the discretization error, a variable step size has been used for the simulation experiments, with maximum step size set to 10 ms. The control sample time T_c has been fixed to 100 ms.

The neurocontroller performance is compared with a PID regulator. In order to make a fair comparison, the PID output has been scaled to adjust its range to $[0, (\pi/2)]$ and it has been also biased by $(\pi/4)$. The equation of the biased PID controller is expressed as follows:

```
% Initialization
Xmin ⟵ PerrMin
Xmax ⟵ PerrMax
Ymin ⟵ dotPerrMin
Ymax ⟵ dotPerrMax
IncX ⟵ (Xmax − Xmin)/(Nx − 1)
IncY ⟵ (Ymax − Ymin)/(Ny − 1)
M ⟵ Nx ∗ Ny
for i = 0 to M − 1
    cNetX ⟵ (i DIV Ny) ∗ IncX + XMin
    cNetY ⟵ (i MOD Ny) ∗ IncY + YMin
    cNet (i) ⟵ (cNetX, cNetY)
    W (i) ⟵ 1
    Fold (i) ⟵ 0
end for
F ⟵ Fold
tconOld ⟵ 0
pitchCon ⟵ 0
% Execute algorithm
(errPow, derrPow, errPowSum) ⟵ MODEL (0)
for t = 0 to tEnd
    if t ≥ tconOld + Tc then
        errPowSat ⟵ MIN (Xmax, MAX (Xmin, errPow))
        derrPowSat = MIN (Ymax, MAX (Ymin, derrPow))
        [RBFout, F] = RBF(cNet, W, errPowSat, derrPowSat)
        Fold ⟵ F
        pitchCon ⟵ (pi/4) − RBFout
        if ABS (errPowSat) < minErr then
            Winc ⟵ 0
        else
            errM ⟵ errPowSat ∗ KP + derrPowSat ∗ KD + errPowSum ∗ KI
            Winc ⟵ Fold ∗ errM ∗ mu
        end if
        W ⟵ W + Winc
        tconOld ⟵ t
    end if
    (errPow, derrPow, errPowSum) ⟵ MODEL (pitchCon)
end for
```

Algorithm 1: Proposed learning algorithm.

$$\theta = \frac{\pi}{4} - \frac{\pi}{4P_{\mathrm{err}_{\mathrm{MAX}}}}\left[K_P \cdot P_{\mathrm{err}} + K_D \cdot \frac{\mathrm{d}}{\mathrm{d}t}P_{\mathrm{err}} + K_I \cdot \int P_{\mathrm{err}}\mathrm{d}t\right]. \tag{29}$$

The wind turbine nominal power is 7 kW, and thus the reference $P_{\mathrm{ref}} = 7000$ W. The tuning parameters $[K_P, K_D, K_I]$ have been determined by trial and error, and their values are [1, 0.2, 0.9], respectively. The parameter minErr of the learning algorithm is set to 15.

The performance of the controllers has been evaluated with the MSE, the mean value, and the variance, calculated as

$$\begin{aligned}
\mathrm{MSE} &= \sqrt{\frac{1}{T_{\mathrm{sim}}}\sum_i\left[\left(P_{\mathrm{out}_i} - P_{\mathrm{ref}}\right)^2 T_{s_i}\right]},\\
\mathrm{Mean} &= \frac{1}{T_{\mathrm{sim}}}\sum_i\left[P_{\mathrm{out}_i} \cdot T_{s_i}\right],\\
\mathrm{Var} &= \frac{1}{T_{\mathrm{sim}}}\sum_i\left[\left(P_{\mathrm{out}_i} - \mathrm{Mean}\right)^2 T_{s_i}\right],
\end{aligned} \tag{30}$$

where T_{sim} is the simulation time and T_{s_i} is the sampling time i that is necessary due to the variable step size that has been used.

Figure 5(a) shows the output power when different strategies are applied. The blue line represents the output when the pitch is permanently set to zero, the red one represents the output when pitch angle is set to feather position (90°), the yellow line is the response with the PID, the purple one is the response with the neural controller, and finally the green line represents the rated power. Figure 5(b) shows a zoom of the previous figure to see better the variations of the signals. In this experiment, the wind is randomly generated with a speed between 11.5 and 14 m/s, the RBF has 25 neurons in the hidden layer, σ is set to 0.1, the maximum and minimum values of the parameters $[P_{\mathrm{err}_{\mathrm{MIN}}}, P_{\mathrm{err}_{\mathrm{MAX}}}, \dot{P}_{\mathrm{err}_{\mathrm{MIN}}}, \dot{P}_{\mathrm{err}_{\mathrm{MAX}}}]$ are set to [−1000, 1000, −400, 400], and learning rate μ is 0.0001 ∗ 1.5.

As shown in Figure 5, when the pitch is set to zero, the output power is always bigger than the rated power because the blades harness the maximum power of the wind. As expected, when the pitch is fixed to feather, the opposite

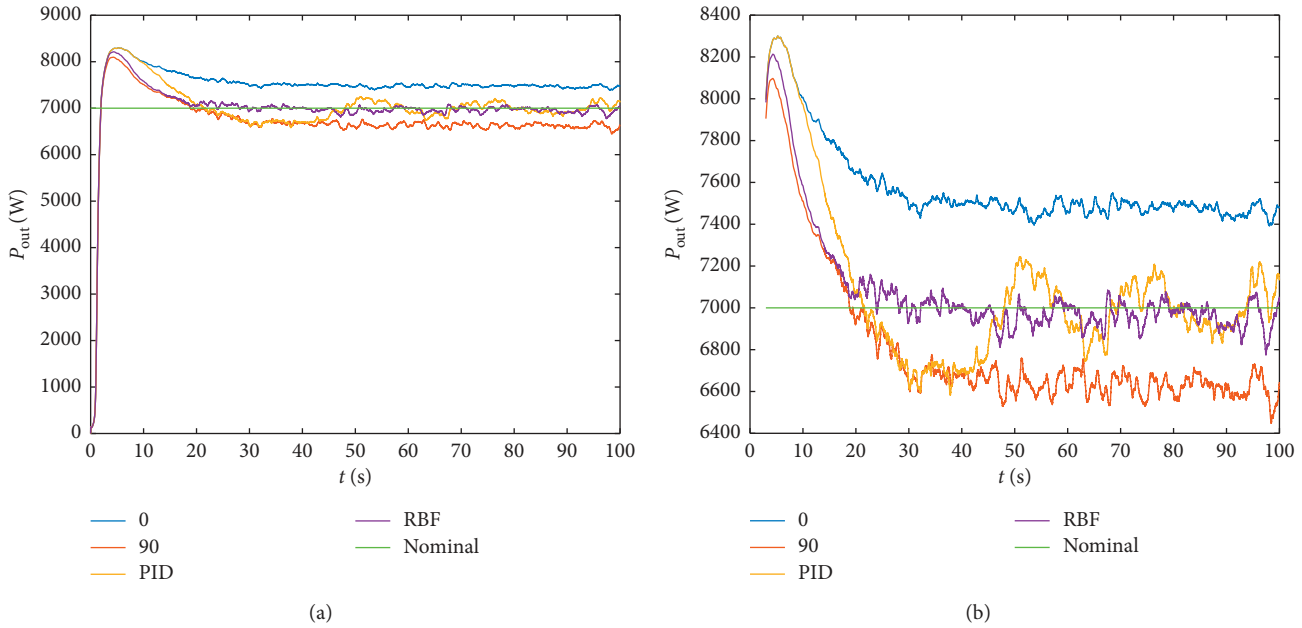


Figure 5: Output power with different strategies (a) and output power zoom (b).

happens, as the surface which faces the wind is minimum. Another interesting outcome is that the proposed neurocontroller is not only able to stabilize the output power around the nominal value, but its performance is better than the PID, particularly up to 50 s, and it is less oscillatory.

As the output power depends on the wind, different simulations were carried out varying the wind speed. The configuration of the neural network and the learning algorithm is the same as in the previous experiment. Figure 6 shows the influence of the wind speed regarding the mean square power error (MSE). The red bar is the MSE with the neural controller, and the blue one is the MSE with the PID. As expected, the higher the wind, the larger the error. For all the ranges of wind speed, the neurocontrol strategy has been proved to be better than the PID.

Table 2 summarizes the detailed results of the simulation experiments with different wind speeds between 12.2 and 12.8 m/s. At a wind speed below 12.2 m/s, the stabilized output power is always lower than 7 kW, even with pitch angle set to 0. With a wind speed over 12.8 m/s, the steady output power is always higher than 7 kW even when the pitch is set to 90°. In all the cases, the error is smaller with the neural controller than with the PID but for 12.5, 12.7, and 12.8 m/s, the mean obtained with the PID is slightly smaller.

A sinusoidal wind signal has been also tested. The average wind speed is 12.5 m/s with an amplitude of 0.6 m/s and a period of 50 s. The result of the experiment is shown in Figure 7. The output power is represented with the same colour code as Figure 5. To show how the RBF learns, Figure 7(a) shows the response for iterations from 1 to 175. In Figure 7(b), the output power for different control strategies described before when the system has already learned is represented.

The learning capability of the neurocontroller is shown in Figure 8. The MSE quickly converges in a few iterations. It

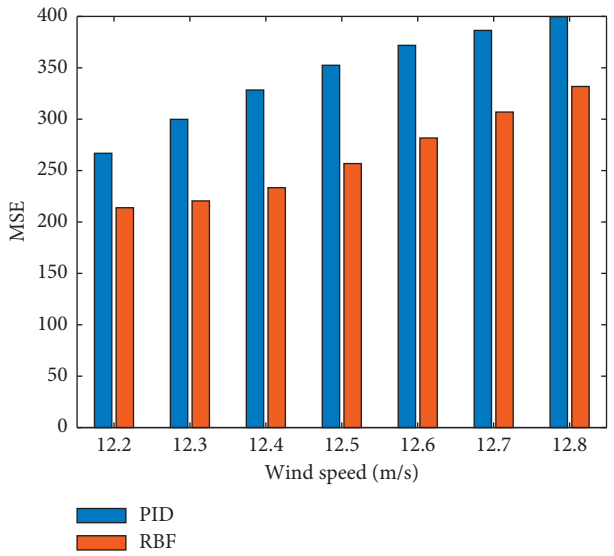


Figure 6: Evolution of MSE with wind speed.

is also possible to observe an inflection point around iteration 30. From this point on, the learning speed decreases. Indeed, from them, the MSE hardly varies.

The frequency of the sinusoidal wind speed signal also influences the results. Figure 9 shows the results for different periods (blue, PID; red, neurocontrol). The minimum MSE is reached for the minimum period; from this value, the MSE grows. At a period of 20 s, a local maximum for the neural controller appears, and the same happens at 35 s for the PID. From then on, the error decreases for both controllers. In all the cases, the error is much smaller with the neurocontroller than with the PID.

Table 3 summarizes the results obtained with this experiment. In all cases, the MSE is much smaller with the

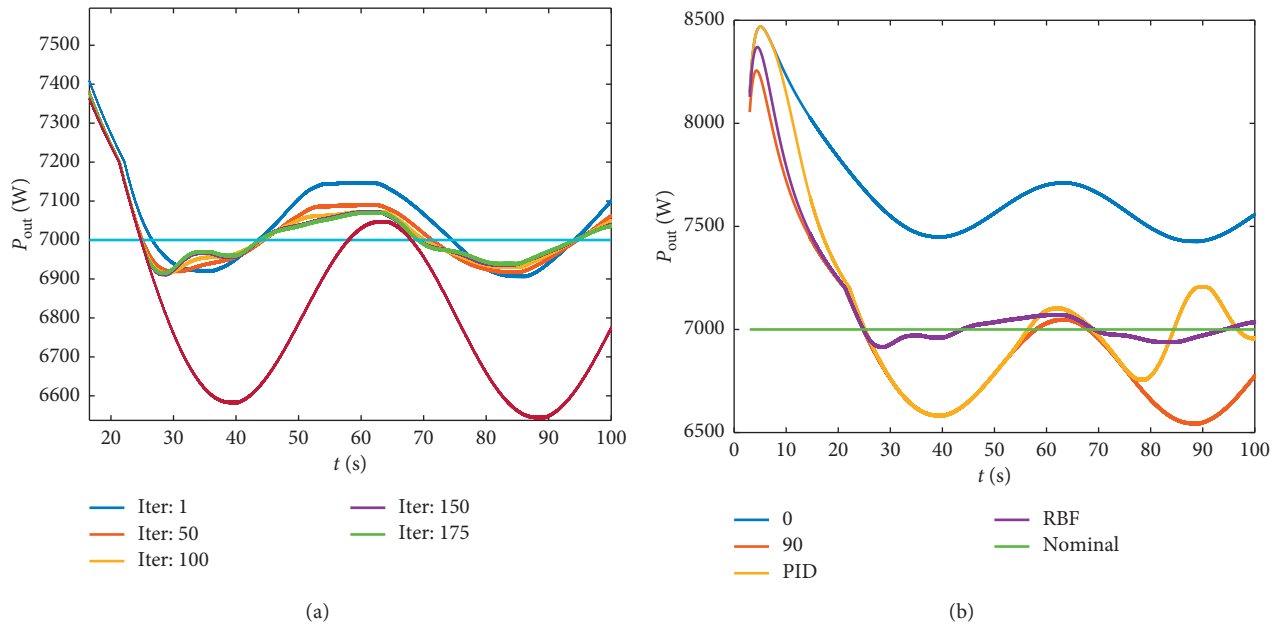


Figure 7: Output power for learning iterations from 1 to 175 with sinusoidal wind speed (a). Output power for different control strategies when the neural controller has learned (b).

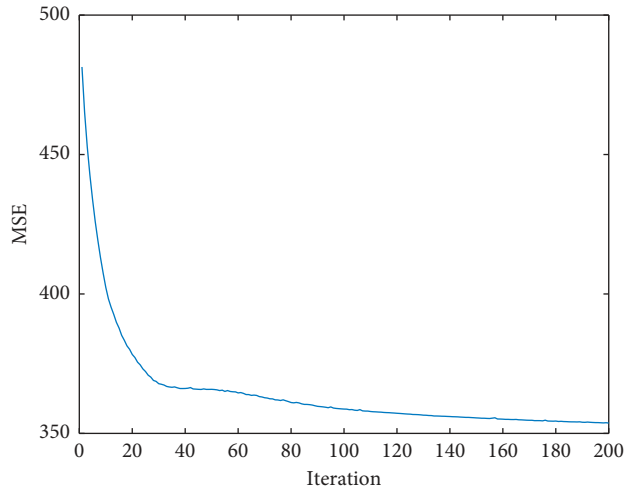


Figure 8: MSE evolution of the neurocontroller with a sinusoidal wind speed.

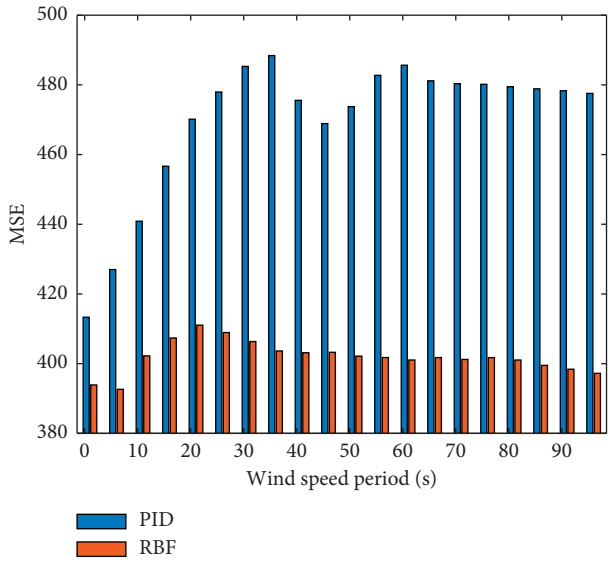


Figure 9: Evolution of error for different periods of the sinusoidal wind speed.

neural controller. Moreover, it is possible to observe how the response with the neural controller slightly improves when the period is larger than 20 s: the MSE and the variance decrease, and the mean value remains almost unchanged. Meanwhile, for the PID, there are several local minimums and maximums in the MSE and the variance. These upward and downward trends also appear in the MSE and the variance evolution. Nevertheless, the influence of the wind period is not so relevant.

4.1. Influence of the RBF. The influence of the configuration of the RBF neural network in the performance of the controller has also been evaluated. Different number of neurons, values of the σ parameter, and several limits have been tested. The wind turbine is subjected to a random wind with mean speed between 11.5 and 14 m/s; lambda is $0.0001 * 1.5$, σ is set to 0.1 in this experiment, the lower and upper limits $[P_{\text{err}_{\text{MIN}}}, P_{\text{err}_{\text{MAX}}}, \dot{P}_{\text{err}_{\text{MIN}}}, \dot{P}_{\text{err}_{\text{MAX}}}]$ are set to [−1000, 1000, −400, 400], and the number of neurons varies.

Figure 10 shows the influence of the number of neurons, M, in the evolution of the MSE. The colour is associated to the number of neurons (see the legends). All the curves of this figure have a similar shape, and the main difference is the slope before the inflection point. It is possible to see that the more the neurons, the higher the slopes. In general, the

Table 2: MSE, output power mean, and variance for different wind speeds.

Wind speed (m/s)	RBF			PID		
	MSE	Mean	Variance	MSE	Mean	Variance
12.2	**213.87**	**7.07e + 3**	**1.03e + 5**	**266.83**	**7.20e + 3**	**1.24e + 5**
12.3	220.51	7.13e + 3	1.14e + 5	299.86	7.21e + 3	1.51e + 5
12.4	233.36	7.20e + 3	1.19e + 5	328.40	7.22e + 03	1.80e + 5
12.5	256.85	7.24e + 3	1.36e + 5	352.47	7.22e + 03	2.07e + 5
12.6	281.72	7.24e + 3	1.47e + 5	371.84	7.24e + 03	2.35e + 5
12.7	306.96	7.27e + 3	1.69e + 5	386.39	7.26e + 3	2.62e + 5
12.8	331.85	7.32e + 3	1.89e + 5	399.76	7.28e + 3	2.92e + 5

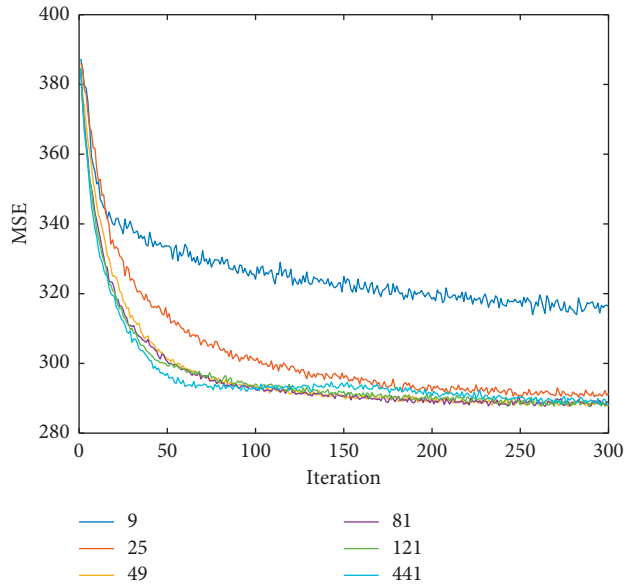


Figure 10: Evolution of the MSE with different number of neurons.

error decreases with the number of neurons until the number is so large that the network does not learn. For example, the MSE with 441 neurons is bigger than that with 121.

To evaluate the influence of the width σ of the activation function, the configuration of the RBF network is set to the previous values, with the number of neurons $M = 9$, 25, and 121 (Figure 11, from left to right). The σ parameter varies from 0.05 to 0.75. In all the cases, the MSE tends to decrease as σ increases. There is a sharp drop in MSE during the first iterations for values of σ greater than 0.25. The descent rate also grows with the number of neurons.

Figure 12 shows another perspective of the influence of σ in the error. It represents the MSE at iteration 200 for different σ values and different number of neurons. In this figure, it is also possible to see how the MSE decreases with σ until this parameter is around 0.25, where it starts to grow. This inflexion point does not depend on the number of neurons but the fall before that minimum does depend on the number of neurons (the bigger the number of neurons, the larger the descent rate).

The performance of the neurocontroller can be also adjusted by the modification of the saturation limits of the input space. Different sets of values of $[P_{\mathrm{err}_{\mathrm{MIN}}}, P_{\mathrm{err}_{\mathrm{MAX}}}, \dot{P}_{\mathrm{err}_{\mathrm{MIN}}}, \dot{P}_{\mathrm{err}_{\mathrm{MAX}}}]$ have been tested, one varying $P_{\mathrm{err}_{\mathrm{MAX}}}$ and another changing $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$. In both experiments, the number of neurons M is set to 121, σ is 0.25, and 5 iterations are run. When $P_{\mathrm{err}_{\mathrm{MAX}}}$. is changed, the value of $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$ is kept constant to 400, and when $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$ varies, the limit $P_{\mathrm{err}_{\mathrm{MAX}}}$ is fixed to 1000. The corresponding negative boundaries have the same absolute value.

Table 4 shows the variation of MSE, the output power mean, and its variance when $P_{\mathrm{err}_{\mathrm{MAX}}}$ is modified from 100 to 1500. The MSE and the mean value decrease with $P_{\mathrm{err}_{\mathrm{MAX}}}$; however, the variance grows. This may be due to the fact that bigger values of $P_{\mathrm{err}_{\mathrm{MAX}}}$ mean bigger variations in the output power and thus larger variance. The influence in the MSE is explained since wider boundaries produce less saturated values and more available information for the learning process. But if the saturation is not reached, too high value of $P_{\mathrm{err}_{\mathrm{MAX}}}$ may be counterproductive because the spatial distribution of the neurons makes more neurons to be useless.

Table 5 summarizes the variation of the MSE, the output power mean, and its variance when $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$ changes from 50 to 7050. Similar to Table 4, the MSE and the mean value are reduced when $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$ increases until a local minimum is reached. It may be also explained for the reduction of the saturated values. However, in this case, the variance also decreases with $\dot{P}_{\mathrm{err}_{\mathrm{MAX}}}$.

4.2. Influence of the Learning Parameters. Several experiments have been carried out to show the influence of the learning parameters μ, K_{pL}, K_{dL}, and K_{iL}. The configuration of the RBF network is $M = 121$, $\sigma = 0.1$, and $[P_{\mathrm{err}_{\mathrm{MIN}}}, P_{\mathrm{err}_{\mathrm{MAX}}}, \dot{P}_{\mathrm{err}_{\mathrm{MIN}}}, \dot{P}_{\mathrm{err}_{\mathrm{MAX}}}] = [-1000, 1000, -400, 400]$. In the first experiment, 200 iterations have been simulated and the tuple $[K_{pL}, K_{dL}, K_{iL}]$ is set to [1, 0.1, 0]. Figure 13 shows the results for different learning rate μ. Again, the MSE decreases at each iteration. As expected, the descent rate grows with the learning rate. These results may also be seen in Table 6 (at iteration 5). The output power mean value also decreases with the learning rate. However, the variance grows, and larger values of μ produce bigger increments in the weights of the neural network (28) and thus bigger variations in the pitch reference and also greater changes in the output power.

In the next experiment, K_{dL} and K_{iL} are set to 0 and K_{pL} is varied. The effect of varying K_{pL} is the same than modifying μ due to the fact that both are constants that multiply P_{err_S} (although the results are different because $K_{dL} = 0.1$ in the previous experiment). The results are shown in Table 7.

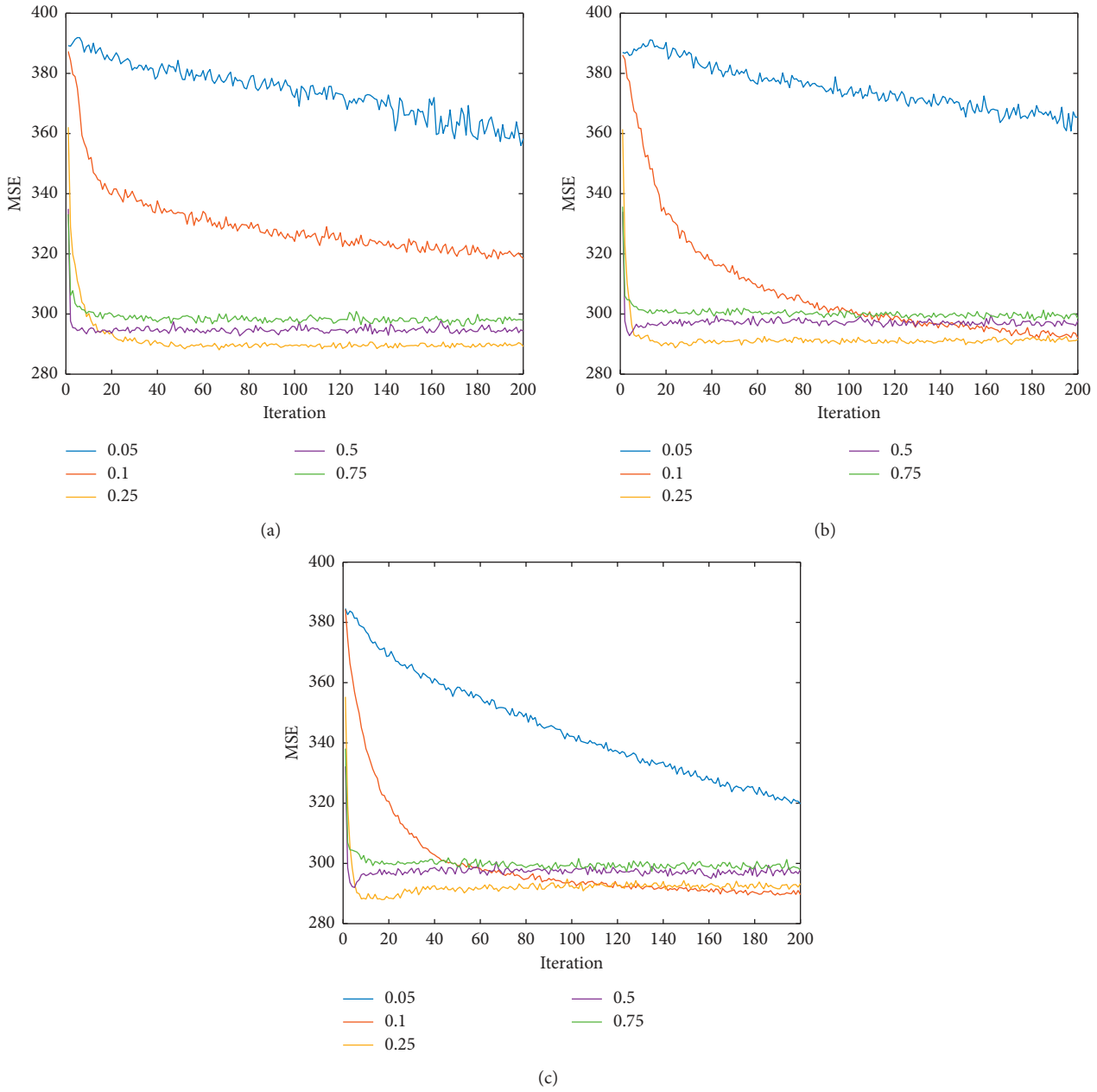


Figure 11: Evolution of the MSE for different values of σ and number of neurons M. (a) Influence of σ in MSE evolution, $M = 9$. (b) Influence of σ in MSE evolution, $M = 25$. (c) Influence of σ in MSE evolution, $M = 121$.

The MSE and the output power mean value decrease with K_{pL} and the variance grows.

Now, K_{pL} and K_{iL} are set to 0 and K_{dL} is varied. The results are shown in Table 8. Initially, the MSE and the output power mean decrease but from $K_{dL} = 5$, these values grow continuously. An increment of K_{dL} makes the system learn faster. Also, it reacts faster to changes so that MSE can be reduced. However, a very high value amplifies the first ramp that moves the pitch reference to 0. After this point, the system takes a long time to learn as it would need a big downward ramp to recover the initial weight values of the neural network. This also explains why the variance decreases with K_{dL}. After an initial ramp, the values are almost stable producing only small variations in the weights and thus a small variance.

Finally, K_{pL} and K_{dL} are set to 0 and K_{iL} is varied to test its influence. The results are shown in Table 9. Initially, the MSE and the output power mean decrease with K_{iL} until K_{iL} is equal to 0.1; from this value on, they grow continuously. K_{iL} helps the controller to learn how to reduce the steady-state error, so the MSE decreases when K_{iL} increases. However, if this parameter is too high, the controller becomes sluggish and the MSE grows. The variance also increases with K_{iL} since it makes the controller slower, so higher output values are reached. Keeping these high outputs longer generates a greater variance.

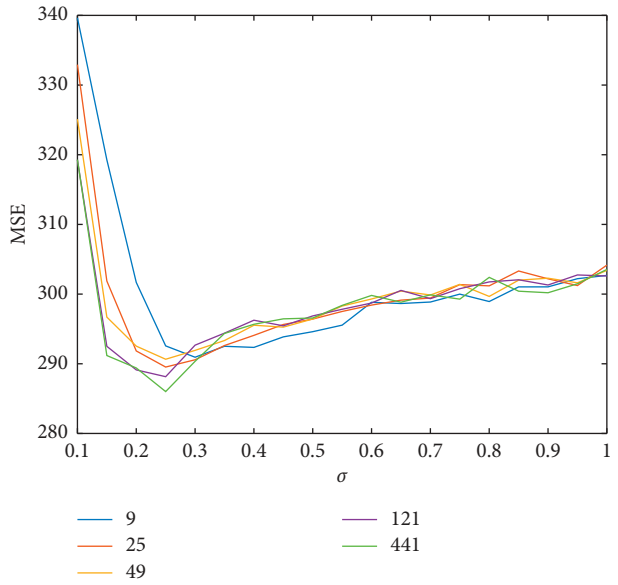


Figure 12: Influence of σ in the MSE for different number of neurons.

Table 3: MSE, output power mean, and variance for different periods of sinusoidal wind speed.

Wind speed period	RBF			PID		
	MSE	Mean	Variance	MSE	Mean	Variance
1	**393.89**	**7.43e + 3**	**1.58e + 5**	**413.33**	**7.29e + 3**	**3.22e + 5**
11	402.22	7.40e + 3	1.97e + 5	440.87	7.28e + 3	3.56e + 5
21	411.05	7.39e + 3	2.29e + 5	470.15	7.29e + 3	3.94e + 5
31	406.35	7.40e + 3	2.31e + 5	485.30	7.29e + 3	4.00e + 5
41	403.14	7.40e + 3	2.25e + 5	475.56	7.31e + 3	3.85e + 5
51	402.16	7.40e + 3	2.25e + 5	473.75	7.30e + 3	3.90e + 5
61	401.05	7.41e + 3	2.25e + 5	485.64	7.33e + 3	4.04e + 5
71	401.21	7.42e + 3	2.20e + 5	480.16	7.32e + 3	3.83e + 5
81	401.04	7.41e + 3	2.14e + 5	478.86	7.32e + 3	3.82e + 5
91	398.40	7.41e + 3	2.09e + 5	477.56	7.31e + 3	3.85e + 5

Table 4: Variation of MSE, output power mean, and variance with $P_{\text{err}_{\text{MAX}}}$.

$P_{\text{err}_{\text{MAX}}}$	RBF		
	MSE	Mean	Variance
100	312.02	7.28e + 3	1.60e + 5
300	298.22	7.24e + 3	1.67e + 5
500	293.76	7.23e + 3	1.71e + 5
700	**291.90**	7.21e + 3	1.74e + 5
900	291.91	7.21e + 3	1.76e + 5
1100	292.71	7.20e + 3	1.78e + 5
1300	293.18	**7.19e + 3**	1.83e + 5
1500	293.21	**7.19e + 3**	1.82e + 5
PID	397.95	7.25e + 3	2.78e + 5

4.3. Influence of the Control Period. Once the influence of the neural network configuration and the learning algorithm parameters has been analysed, the last experiment evaluates how the control period affects the performance of the neurocontroller. The number of neurons is set to 11, $\sigma = 0.1$, $[P_{\text{err}_{\text{MIN}}}, P_{\text{err}_{\text{MAX}}},$

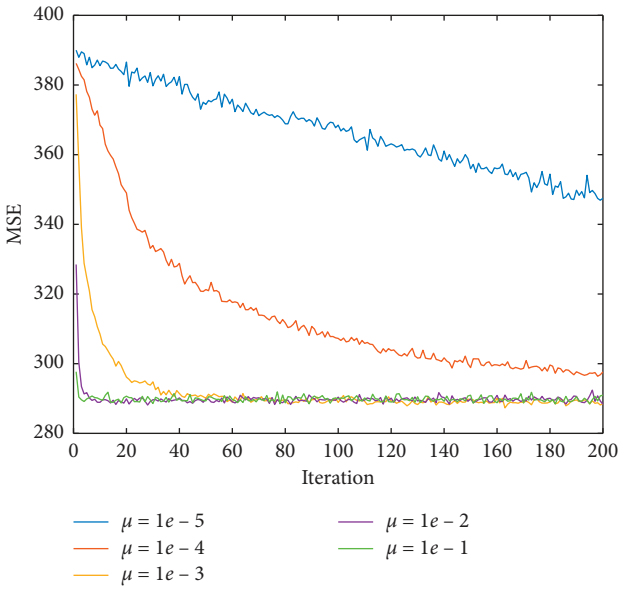


Figure 13: Evolution of MSE for different learning parameters μ.

Table 5: Variation of MSE, output power mean, and variance with $\dot{P}_{\mathrm{err}_{MAX}}$.

$\dot{P}_{\mathrm{err}_{MAX}}$	RBF		
	MSE	Mean	Variance
50	301.34	$7.23e+3$	$1.76e+5$
150	299.82	$7.22e+3$	$1.79e+5$
250	297.11	$7.21e+3$	$1.79e+5$
350	293.56	$7.21e+3$	$1.78e+5$
450	293.23	$7.21e+3$	$1.78e+5$
550	290.34	$7.21e+3$	$1.78e+5$
1050	285.16	$7.20e+3$	$1.71e+5$
3050	280.55	$7.20e+3$	$1.66e+5$
5050	**278.96**	**7.20e + 3**	**1.64e + 5**
7050	279.11	$7.20e+3$	$1.65e+5$
PID	397.95	$7.25e+3$	$2.78e+5$

Table 6: Variation of MSE, output power mean, and variance with the learning rate μ.

Learning rate μ	RBF		
	MSE	Mean	Variance
0.0001	379.45	$7.40e+3$	1.26
0.001	324.60	$7.28e+3$	1.76
0.01	290.37	$7.22e+3$	1.72
0.1	**289.27**	**7.23e + 3**	**1.68**
1	290.41	$7.23e+3$	1.68

Table 7: Variation of MSE output power mean and variance with K_{pL}.

K_{pL}	RBF		
	MSE	Mean	Variance
0.1	386.54	$7.41e+3$	$1.23e+5$
0.2	384.43	$7.41e+3$	$1.23e+5$
0.5	380.96	$7.41e+3$	$1.25e+5$
1	374.78	$7.40e+3$	$1.30e+5$
2	361.46	$7.37e+3$	$1.40e+5$
5	331.12	$7.33e+3$	$1.69e+5$
10	314.50	$7.25e+3$	$1.80e+5$
20	302.65	$7.24e+3$	$1.75e+5$
50	292.63	$7.23e+3$	$1.70e+5$
100	290.34	**7.22e + 3**	$1.69e+5$
200	**289.34**	$7.23e+3$	**1.69e + 5**
500	290.32	$7.23e+3$	$1.70e+5$
PID	397.95	$7.25e+3$	$2.78e+5$

$\dot{P}_{\mathrm{err}_{MIN}}, \dot{P}_{\mathrm{err}_{MAX}}] = [-1000, 1000, -1000, 1000]$, and $[K_{pL}, K_{dL}, K_{iL}] = [1, 0.1, 0]$. Table 10 shows the results at iteration 5 when the control period varies from 10 to 100 ms. If the control sample time is too small, the neural controller reacts to the noisy component of the wind, and this increases the MSE and the variance. On other hand, a very big control period makes the system too slow and also increases the MSE. Therefore, an intermediate value would be the best option. In any case, the performance of the neural controller is much better than the PID response for all the control periods tested.

Table 8: Variation of MSE, output power mean, and variance with K_{dL}.

K_{dL}	RBF		
	MSE	Mean	Variance
0.1	389.37	$7.42e+3$	$1.21e+5$
0.2	386.27	$7.41e+3$	$1.23e+5$
0.5	386.87	$7.41e+3$	$1.21e+5$
1	389.15	$7.42e+3$	$1.19e+5$
2	385.34	$7.41e+3$	$1.20e+5$
5	**384.66**	**7.41e + 3**	**1.14e + 5**
10	386.09	$7.41e+3$	$1.09e+5$
20	393.09	$7.43e+3$	$1.05e+5$
50	431.41	$7.47e+3$	$0.93e+5$
100	511.32	$7.56e+3$	$0.60e+5$
200	550.49	$7.60e+3$	$0.54e+5$
500	547.87	$7.60e+3$	$0.56e+5$
PID	397.95	$7.25e+3$	$2.78e+5$

Table 9: Variation of MSE, output power mean, and variance with K_{iL}.

K_{iL}	RBF		
	MSE	Mean	Variance
0.001	387.91	$7.42e+3$	$1.22e+5$
0.002	388.70	$7.42e+3$	$1.21e+5$
0.005	386.95	$7.41e+3$	$1.23e+5$
0.01	381.03	$7.40e+3$	$1.23e+5$
0.02	373.85	$7.39e+3$	$1.36e+5$
0.05	357.37	$7.32e+3$	$1.80e+5$
0.1	**356.62**	**7.25e + 3**	**2.17e + 5**
0.2	368.96	$7.23e+3$	$2.41e+5$
0.5	388.23	$7.20e+3$	$2.73e+5$
1	408.56	$7.19e+3$	$2.99e+5$
2	435.42	$7.18e+3$	$3.30e+5$
5	462.72	$7.17e+3$	$3.61e+5$
PID	397.95	$7.25e+3$	$2.78e+5$

Table 10: Variation of MSE, output power mean, and variance with the control period.

Control period (ms)	RBF			PID		
	MSE	Mean	Variance	MSE	Mean	Variance
10	292.86	$7.20e+3$	$1.82e+5$	401.24	$7.26e+3$	$2.82e+5$
20	288.39	$7.20e+3$	$1.76e+5$	401.21	$7.26e+3$	$2.79e+5$
30	285.66	$7.20e+3$	$1.72e+5$	401.40	$7.26e+3$	$2.81e+5$
40	284.63	$7.20e+3$	$1.71e+5$	398.58	$7.26e+3$	$2.79e+5$
50	284.28	$7.20e+3$	$1.70e+5$	402.38	$7.26e+3$	$2.80e+5$
60	**282.29**	**7.20e + 3**	**1.69e + 5**	402.12	$7.26e+3$	$2.80e+5$
70	285.12	$7.19e+3$	$1.72e+5$	**400.91**	**7.26e + 3**	$2.80e+5$
80	284.77	$7.20e+3$	$1.71e+5$	399.21	$7.26e+3$	$2.80e+5$
90	285.70	$7.20e+3$	$1.72e+5$	399.32	$7.26e+3$	$2.79e+5$
100	285.85	$7.20e+3$	$1.72e+5$	397.95	$7.26e+3$	**2.78e + 5**

5. Conclusions and Future Works

In this work, an intelligent wind turbine pitch control strategy is presented, and the influence of the parameters of the neurocontrol systems is analysed. The pitch controller is

based on an RBF neural network that learns in an unsupervised way. The control goal is to maintain the output power around its rated value, obtaining the appropriate pitch angle reference. The output power errors are introduced both in the neurocontroller and in the learning algorithm.

Extensive simulation tests have been carried out on a 7 kW wind turbine, varying different network configuration parameters as well as the wind speed. The performance of the neurocontroller is compared with a tuned PID obtaining better results in all the cases.

These experiments have led to draw some interesting conclusions. Among them, we can highlight the small influence of the wind frequency. However, the learning rate grows significantly with the number of neurons. There exists an optimum sigma value different for each number of neurons, between 0.2 and 0.4. Another interesting result is how the gains K_{pL} and K_{dL} as well as K_{iL} accelerate the learning, and, in general, low values of these tuning parameters improve the stability. The control sample time has a clear effect on the system response, making it slower or faster.

In future, it would be desirable to test the proposal on a real prototype of a wind turbine. In addition, it would be interesting to apply this control strategy to a bigger turbine and to see if this control action affects the stability of a floating offshore wind turbine.

Data Availability

The findings of this study have been generated by the equations and parameters cited in the article.

Disclosure

An earlier version of this paper was presented in 15th Int. Conf. on Soft Computing Models in Industrial and Environmental Applications, 2020 [18].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was partially supported by the Spanish Ministry of Science, Innovation and Universities, under MCI/AEI/FEDER Project no. RTI2018-094902-B-C21.

References

[1] Our World in Data, 2020, https://ourworldindata.org/renewable-energy.

[2] M. Mikati, M. Santos, and C. Armenta, "Electric grid dependence on the configuration of a small-scale wind and solar power hybrid system," *Renewable Energy*, vol. 57, pp. 587–593, 2013.

[3] M. Tomás-Rodríguez and M. Santos, "Modelado y control de turbinas eólicas marinas flotantes," *Revista Iberoamericana de Automática e Informática industrial*, vol. 16, no. 4, pp. 381–390, 2019.

[4] C. Kim, E. Muljadi, and C. C. Chung, "Coordinated control of wind turbine and energy storage system for reducing wind power fluctuation," *Energies*, vol. 11, no. 1, p. 52, 2017.

[5] E. Hosseini, E. Aghadavoodi, G. Shahgolian, and H. Mahdavi-Nasab, "Intelligent pitch angle control based on gain-scheduled recurrent ANFIS," *Journal of Renewable Energy and Environment*, vol. 6, no. 1, pp. 36–45, 2019.

[6] J. Liu, F. Zhou, C. Zhao, and Z. Wang, "A PI-type sliding mode controller design for PMSG-based wind turbine," *Complexity*, vol. 2019, Article ID 2538206, 12 pages, 2019.

[7] M. Nasiri, S. Mobayen, and Q. M. Zhu, "Super-twisting sliding mode control for gearless PMSG-based wind turbine," *Complexity*, vol. 2019, Article ID 6141607, 15 pages, 2019.

[8] L. Colombo, M. L. Corradini, G. Ippoliti, and G. Orlando, "Pitch angle control of a wind turbine operating above the rated wind speed: a sliding mode control approach," *ISA Transactions*, vol. 96, pp. 95–102, 2020.

[9] X. Yin, W. Zhang, Z. Jiang, and L. Pan, "Adaptive robust integral sliding mode pitch angle control of an electro-hydraulic servo pitch system for wind turbine," *Mechanical Systems and Signal Processing*, vol. 133, Article ID 105704, 2019.

[10] S. Z. Hassan, H. Li, T. Kamal, M. Q. Abbas, M. A. Khan, and G. M. Mufti, "An intelligent pitch angle control of wind turbine," in *Proceedings of the 2017 International Symposium on Recent Advances in Electrical Engineering (RAEE)*, IEEE, Islamabad, Pakistan, pp. 1–6, 2017 October.

[11] M. M. Rocha, J. P. da Silva, and F. D. C. B. De Sena, "Simulation of a fuzzy control applied to a variable speed wind system connected to the electrical network," *IEEE Latin America Transactions*, vol. 16, no. 2, pp. 521–526, 2018.

[12] P. M. Rubio, J. F. Quijano, P. Z. López et al., "Intelligent control for improving the efficiency of a hybrid semi- submersible platform with wind turbine and wave energy converters," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 16, no. 4, pp. 480–491, 2019.

[13] A. P. Marugán, F. P. G. Márquez, J. M. P. Perez, and D. Ruiz-Hernández, "A survey of artificial neural network in wind energy systems," *Applied Energy*, vol. 228, pp. 1822–1836, 2018.

[14] A. B. Asghar and X. Liu, "Adaptive neuro-fuzzy algorithm to estimate effective wind speed and optimal rotor speed for variable-speed wind turbine," *Neurocomputing*, vol. 272, pp. 495–504, 2018.

[15] A. Saénz-Aguirre, E. Zulueta, U. Fernández-Gamiz, J. Lozano, and J. Lopez-Guede, "Artificial neural network based reinforcement learning for wind turbine yaw control," *Energies*, vol. 12, no. 3, p. 436, 2019.

[16] H. Moodi and D. Bustan, "Wind turbine control using T-S systems with nonlinear consequent parts," *Energy*, vol. 172, pp. 922–931, 2019.

[17] I. L. R. Gomes, R. Melicio, V. M. F. Mendes, and H. M. I. Pousinho, "Wind power with energy storage arbitrage in day-ahead market by a stochastic MILP approach," *Logic Journal of the IGPL*, vol. 28, no. 4, pp. 570–582, 2020.

[18] J. E. Sierra-Garcia and M. Santos, "Wind turbine pitch control with an RBF neural network," in *Proceedings of the 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)*, Seville, Spain, May 2020.

[19] M. Mikati, M. Santos, and C. Armenta, "Modelado y Simulación de un Sistema Conjunto de Energía Solar y Eólica para

Analizar su Dependencia de la Red Eléctrica," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 9, no. 3, pp. 267–281, 2012.

[20] L. Acho, "A proportional plus a hysteretic term control design: a throttle experimental emulation to wind turbines pitch control," *Energies*, vol. 12, no. 10, p. 1961, 2019.

[21] T. Ackermann, *Wind Power in Power Systems*, John Wiley & Sons, Hoboken, NJ, USA, 2005.