Article

Grey Wolf Optimizer in Design Process of the Recurrent Wavelet Neural Controller Applied for Two-Mass System

Mateusz Zychlewicz , Radosław Stanisławski and Marcin Kamiński *

Department of Electrical Machines, Drives and Measurements, Faculty of Electrical Engineering, Wrocław University of Science and Technology, 19 Smoluchowskiego St., 50-372 Wrocław, Poland; mateusz.zychlewicz@pwr.edu.pl (M.Z.); radoslaw.stanislawski@pwr.edu.pl (R.S.)

* Correspondence: marcin.kaminski@pwr.edu.pl

Abstract: In this paper, an adaptive speed controller of the electrical drive is presented. The main part of the control structure is based on the Recurrent Wavelet Neural Network (RWNN). The mechanical part of the plant is considered as an elastic connection of two DC machines. Oscillation damping and robustness against parameter changes are achieved using network parameters updates (online). Moreover, the various combinations of the feedbacks from the state variables are considered. The initial weights of the neural network and the additional gains are tuned using a modified version of the Grey Wolf Optimizer. Convergence of the calculation is forced using a new definition. For theoretical analysis, numerical tests are presented. Then, the RWNN is implemented in a dSPACE card. Finally, the simulation results are verified experimentally.

Keywords: wavelet neural network; grey wolf optimizer; adaptive speed control; design process optimization; two-mass drive

Citation: Zychlewicz, M.; Stanisławski, R.; Kamiński, M. Grey Wolf Optimizer in Design Process of the Recurrent Wavelet Neural Controller Applied for Two-Mass System. *Electronics* **2022**, *11*, 177. <https://doi.org/10.3390/electronics11020177>

Academic Editor: Valentina E. Balas

Received: 6 December 2021

Accepted: 31 December 2021

Published: 7 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The reliability of the applied control methods is one of the main points of scientific work observed in research centers around the world. The advance of these algorithms is possible due to the fact that the computational power of the available programmable devices is now much higher and the tools are cheaper. Adaptive control theory is one of the many fields explored by scientists. Among the techniques applied in this field, neural networks (NNs) are the fastest-growing group—they find use in robotics [1], optimization of complex control schemes, e.g., predictive control [2], or combinations with other intelligent structures such as fuzzy systems [3]. They ensure a model-free design process and recalculation of internal coefficients under changes of the operating point. Due to these advantageous features, NNs are used in almost any engineering field. The real-life implementations also include electrical drives. In the literature, they are applied to control the speed of Permanent Magnet Synchronous Motors (PMSMs) [4], Induction Motors (IMs) [5], and systems with a complex mechanical part [6].

Neural networks theory proposes adaptation methods that can create models for engineering implementations. A significant division concerns the way the weights are recalculated. To implement offline learning, it is necessary to acquire patterns of data that represent the states of the controlled plant [7]. Then, after several pre-processing methods, the collected data are used for training. Through updating the weights online, adaptive structures for control solutions can be created [8]. Another point considered during the application of a neural network is its topology. The most basic arrangement is called Adaline [9]—a neuron with a linear activation function, which has inputs and weights connected to it. As shown in [10,11], when many linear neurons are connected in layers, a multi-layer perceptron is created, which is one of the most often used neural structures in control theory. A more advanced approach is related to recurrent connections in various parts of the network. Elman [12] and Jordan [13] networks are examples of

the most common solutions in this group; for instance, the applicability of recurrent neural structures to electrical drives is presented in [14]. However, the significant point of the topology determination for a given issue is the activation function. The classical approach involves a sigmoidal transfer function [15]. Radial basis function can also be incorporated as the activation function [16], some modifications (recurrent feedbacks in the structure) are also included in the analyzed solutions [17]. The last group consists of wavelet functions [18]. Wavelet is a function whose value decays over a finite time. Every input neuron utilizes a wavelet that is translated and dilated from the mother wavelet function. The greatest advantage of wavelets is the function estimation. Given that, the neural network can approximate any function.

Wavelet neural networks are a combination of wavelets and neural networks. Neural networks have great learning (adaptation) capabilities, while wavelets are used in wavelet transform [19] (similar to Fourier). The wavelet transform changes the time-value system to a time-frequency system [20]. This ability can be used in neural networks to approximate complicated continuous functions. All wavelet functions use translation (μ) and dilatation (σ) coefficients, which facilitate searching the space for an appropriate value. In wavelet neural networks, translation and dilatation can be treated similarly to weights in classical networks, so they can change their value in time to better adapt to the current conditions. In control theory, this concept can be used to synthesize estimators [21], predictors [22], or controllers [23]. Adaptation algorithms can be used to update only the weights between layers [24] or weights as well as dilatation and translation parameters [25,26], which improves the efficiency of the learning algorithm. Recurrent wavelet networks adopt recursive connections in only one layer [27,28] or every layer [29,30].

Authors in [20] replace radial functions with wavelet functions. It has been shown that wavelet networks are better at approximating the desired functions, this comes with less computation power required. It is easier to determine parameters of such a network, e.g., the number of hidden layers or weights. Neural networks that use the wavelet function are also more accurate and faster converging than the well-known multi-layer perceptron (MLP). The authors also show the disadvantages of wavelet neural networks—a large number of input nodes is associated with a greater number of hidden layers, which can increase the complexity of computing. Paper [21] compares different neural networks used as battery state of charge estimators. Decomposing the signal helps to forecast the battery level more accurately. The output signal of classical neural networks is stable most of the time. However, fluctuations in the output values might be observed. Nevertheless, recurrent wavelet neural networks are accurate most of the time, there are no fluctuations in the estimated values. Recurrent wavelet neural networks are also more robust against external disturbances. A wavelet neural network, as well as wavelet decomposition, are applied in [25] to control a permanent magnet motor. Wavelet transform is used to decompose error signal into different frequency components. Meanwhile, a neural network is employed to calculate the gains of the speed controller. The results of this work are then compared to PI and PID controllers. Authors show that the applied algorithm performs much better in many different states of the drive and for different reference speeds.

The optimal selection of the control structure parameters, according to the reduction in the cost function values, can be achieved using methods based on observations of the populations in nature. Simple elementary data processing (without derivatives of the objective function), easy application to various problems, and multiple criteria analysis are the most often listed advantages of the mentioned techniques, which include the Cuckoo Search algorithm [31], Artificial Bee Colony algorithm [32], Particle Swarm Optimization [33] and Flower Pollination algorithm [34]. A general review of nature-inspired algorithms used for electric drive optimization has been provided by Hannan et al. in [35]. Recently, the Grey Wolf Optimizer (GWO) has been successfully used for optimization [36]. Formulas describe the behavior of wolves that try to find, surround, and get closer to the target [37,38]. The following modifications of the basic version of the GWO are analyzed in the literature (e.g., binary and multi-objective) [39,40]. Reduction in the area of searching in the application

of the algorithm in the design process of the electrical machine (PM motor) can lead to faster results. For this purpose, new definitions of variable elements used in calculations are analyzed [41]. This was an inspiration for the new version of the modified Grey Wolf Optimizer (mGWO) described in this paper.

The implementation of the control algorithms based on artificial intelligence deals with the problems of mathematical terms and coefficient definitions in several design stages. The first concern is the proper determination of the gains used in the control structure. Their values can affect the precision of control and dynamics of the entire system. Classical methods of the control structure synthesis mostly require parameters and equations of the object to be known. Then, after some recalculations, the formulas are achieved. However, the application of the controllers based on neural networks or fuzzy models can be more complicated. A clear mathematical description of the topology of those controllers might be problematic (the transfer function of the closed control loop). Moreover, for controllers that contain reconfigurable parameters, advanced methods of analysis are used. Another issue refers to the nonlinearities of the plant. It needs to be considered during the design process, even though this is an additional design enhancement. In this paper, different combinations of the additional feedbacks (for better damping of the state variables oscillations observed in the two-mass system) used in the control structure are considered. The feedback loops presented in the literature include using the physically unobtainable signals calculated with state observers [42,43] or variations of the Kalman filter [44,45]. Internal recurrent connections in neural controllers are also tested [46]. Thus, multiple recalculations would be required. For the briefly mentioned above problems, the meta-heuristic algorithm can be a useful and efficient solution.

A separate issue of the neural controller applications is focused on the selection of the initial conditions. The systems often start with the random values of weights. After some time of operation, the adaptive law recalculates those coefficients. However, during the initial phase, high oscillations of the state variables can be observed. Assuming the application of the adaptive neural controller, it can be significant for the mechanical part. Unstable work (even over a short period) may lead to ruptures of the couplings and shafts connecting several parts of the system. This issue was also considered in this work, the GWO algorithm was used for the optimization of the starting point of the adaptive neural speed controller.

This paper contains seven essential sections. It starts with a short presentation of the problems and proposed tools used for solving the issue. Then, a description of the plant is shown. The next part of the manuscript is related to the mathematical details of the adaptive neural controller. The background of the GWO calculations and applied modification is presented in the following section. After the theoretical part, the tests (simulations and laboratory experiments) of the RWNN controller applied for an electrical drive with an elastic connection between the motor and the load machine are analyzed. The final element of the article stands concluding remarks.

2. Mathematical Model of the Controller and the Plant

Control structure described in this paper consists of the Recurrent Wavelet Neural Network applied to a two-mass drive as depicted in Figure 1. Mathematical model of mechanical part used in drive with elastic shaft can be characterized using following equations [6,14]:

$$\begin{cases} \frac{d\omega_1(t)}{dt} = \frac{m_e(t) - m_s(t) - m_{f1}(t)}{T_1} \\ \frac{d\omega_2(t)}{dt} = \frac{m_s(t) - m_L(t) - m_{f2}(t)}{T_2} \\ \frac{dm_s(t)}{dt} = \frac{\omega_1(t) - \omega_2(t)}{T_c} \end{cases} \quad (1)$$

where ω_1 and ω_2 are rotor speeds of motor and load, respectively, T_1 , T_2 , and T_c are mechanical time constants of the motor, the load, and the shaft, respectively, m_e and m_s are

electromagnetic torque and shaft torque, respectively, m_L is load torque, and m_{f1} and m_{f2} are nonlinear functions that describe friction occurring in real electrical drives:

$$m_{fi} = (c \cdot \text{abs}(\omega_i) + d) \text{sgn}(\omega_i) \quad (2)$$

where c and d represent the viscous and the Coulomb friction coefficients, respectively.

Figure 1. Schematic diagram of the adaptive control with the RWNN model.

Inner current loop is assumed to be simplified. It is represented using a first order inertial element with a current time constant T_f :

$$G_f(s) = \frac{1}{T_f s + 1} \quad (3)$$

A schematic diagram of the control structure is presented in Figure 1. The speed controller is based on a recurrent wavelet neural network. It is composed of four layers. The first layer (L1) is linear, it has two inputs—the error signal and its derivative. There is also an additional recurrent connection from the neuron output (z_i^1) to its input (h_i^1). It can be characterized by the following equation:

$$(L1) \begin{cases} h_i^1(k) = x_i(k) + w_i^{in}(k)z_i^1(k-1) \\ z_i^1(k) = h_i^1(k) \end{cases} \quad (4)$$

where x is the input signal; h^1 and z^1 are the neuron input and output, respectively; w^{in} is the connection weight; i is the index; and k is the sample number.

The second layer is the wavelet function layer (L2), in this part of the neural network the output of the input layer is processed according to the mother wavelet equation [47]:

$$(L2) \begin{cases} h_{ij}^2(k) = \frac{z_i^1(k) + \mu_{ij}(k)}{\sigma_{ij}(k)} \\ z_{ij}^2(k) = \lambda(h_{ij}^2(k)) \end{cases} \quad (5)$$

where h^2 and z^2 are the second layer's inputs and outputs, respectively, μ is the translation, σ is the dilatation coefficient, and i and j are indexes.

The mother wavelet used in this paper is the Mexican Hat Wavelet, it can be described using the undermentioned expression [48]:

$$\lambda(x) = (1 - 0.5x^2) \exp(-x^2) \quad (6)$$

where x is the input of the function and $\exp(\cdot)$ represents the exponential function.

Then, the achieved values (from the second layer) are multiplied in the third layer (L3). Nodes in the mentioned layer have four inputs—two of them correspond to outputs from

the second layer, the additional two signals are the recurrent connections from the third and the fourth layers:

$$(L3) \begin{cases} h_j^3(k) = \left(\prod_{j=1}^n z_j^{3r}(k) z_{1j}^2(k) z_{2j}^2(k) \right) z^{4r}(k) \\ z_j^3(k) = h_j^3(k) \end{cases} \quad (7)$$

where h^3 and z^3 are the third layer's inputs and outputs, respectively, and z^{3r} and z^{4r} are the outputs of the recurrent nodes:

$$(L3r) \begin{cases} h_i^{3r}(k) = w_j^{3r}(k) z_j^3(k-1) \\ z_j^3(k) = \theta(h_j^{3r}(k)) \end{cases} \quad (8)$$

$$(L4r) \begin{cases} h^{4r}(k) = w^{4r}(k) y_{nn}(k) \\ z^{4r}(k) = \vartheta(h^{4r}(k)) \end{cases} \quad (9)$$

where h^{3r} , z^{3r} , h^{4r} , and z^{4r} are the inputs and the outputs of the recurrent nodes in L3 and L4, respectively, w^{3r} and w^{4r} are the connection weights, and y_{nn} is the output of the controller.

In the above equations, θ and ϑ are nonlinear functions that help to propagate the signals back. These functions are defined as follows:

$$\theta(x) = \frac{1}{1 + \exp(-x)} \quad (10)$$

$$\vartheta(x) = \exp(-x^2) \quad (11)$$

The last neuron (L4) combines (sums) the output signals from the previous layer:

$$(L4) \begin{cases} h^4(k) = \sum_{j=1}^n w_j^{out}(k) z_j^3(k) \\ z^4(k) = h^4(k) = y_{nn}(k) \end{cases} \quad (12)$$

where h^4 and z^4 are the inputs and the outputs of the last layer, respectively, and w^{out} is the weight connection between neurons.

To achieve a proper operation of the controller, it is necessary to provide a learning process to update the weights of the neural network. This process consists of three parts—the forward pass, the backward pass, and the weights update. The forward pass of signals was explained at the beginning of this section. Details of the following stages are described below.

Adaptation is performed to reduce the cost function values— E . In this paper, for the RWNN, following formula was assumed [14]:

$$E(k) = 0.5 e_m^2(k) \quad (13)$$

$$e_m(k) = \omega_{ref}^m(k) - \omega_1(k) \quad (14)$$

where e_m is the difference between the reference model speed ω_{ref}^m and the actual motor speed ω_1 .

To calculate new values of the parameters (weights), the gradient values with respect to all the layers need to be obtained. For the last layer, the gradient is calculated according to the following expression:

$$\delta^4(k) = \frac{\partial E(k)}{\partial y_{nn}(k)} \rightarrow e_m(k) \quad (15)$$

Then, the calculations use the chain rule applied for partial derivatives [27,30]. Values for the third layer are determined as:

$$\delta_j^3(k) = \frac{\partial E(k)}{\partial z_j^3(k)} = \frac{\partial E(k)}{\partial y_{nn}} \frac{\partial y_{nn}(k)}{\partial h^4(k)} \frac{\partial h^4(k)}{\partial z_j^3(k)} \rightarrow \delta^4 w_j^{out}(k) \quad (16)$$

Similarly, data processing is realized for the second layer:

$$\delta_{1j}^2(k) = \frac{\partial E(k)}{\partial z_{1j}^2(k)} = \frac{\partial E(k)}{\partial z_j^3(k)} \frac{\partial z_j^3(k)}{\partial h_j^3(k)} \frac{\partial h_j^3(k)}{\partial z_{1j}^2(k)} \rightarrow \delta_j^3 z^{4r}(k) z_j^{3r}(k) z_{1j}^2(k) \quad (17)$$

$$\delta_{2j}^2(k) = \frac{\partial E(k)}{\partial z_{2j}^2(k)} = \frac{\partial E(k)}{\partial z_j^3(k)} \frac{\partial z_j^3(k)}{\partial h_j^3(k)} \frac{\partial h_j^3(k)}{\partial z_{2j}^2(k)} \rightarrow \delta_j^3 z^{4r}(k) z_j^{3r}(k) z_{2j}^2(k) \quad (18)$$

Finally, for the input layer, gradient values are achieved using the formula:

$$\delta_{ij}^1(k) = \frac{\partial E(k)}{\partial z_i^1(k)} = \frac{\partial E(k)}{\partial z_{ij}^2(k)} \frac{\partial z_{ij}^2(k)}{\partial h_{ij}^2(k)} \frac{\partial h_{ij}^2(k)}{\partial z_i^1(k)} \rightarrow \delta_{ij}^2 \frac{\phi \vartheta(\phi)(1 - 0.5\phi^2) - \phi \vartheta(\phi)}{\sigma_{ij}(k)} \quad (19)$$

where:

$$\phi(k) = \frac{z_i^1(k) - \mu_{ij}(k)}{\sigma_{ij}(k)} \quad (20)$$

After the calculation of all the gradients, all variable parameters are updated in each iteration. In this work, all the weights, as well as the parameters of the wavelet function—translation (μ) and dilatation (σ)—are updated. Weights in the last recurrent layer are adjusted using the following expression:

$$\Delta w^{4r}(k) = \frac{\partial E(k)}{\partial w^{4r}(k)} = \frac{\partial E(k)}{\partial y_{nn}(k)} \frac{\partial y_{nn}(k)}{\partial h^{4r}(k)} \frac{\partial h^{4r}(k)}{\partial w^{4r}(k)} \rightarrow \delta^4 y_{nn}(k) \quad (21)$$

They are then updated according to the delta rule with the learning factor η :

$$w^{4r}(k+1) = w^{4r}(k) - \eta^{4r} \Delta w^{4r}(k) \quad (22)$$

Weights between the third and the last layer are recalculated using the following expression:

$$\Delta w_j^{out}(k) = \frac{\partial E(k)}{\partial w_j^{out}(k)} = \frac{\partial E(k)}{\partial y_{nn}(k)} \frac{\partial y_{nn}(k)}{\partial h^4(k)} \frac{\partial h^4(k)}{\partial w_j^{out}(k)} \rightarrow \delta^4 z_j^3(k) \quad (23)$$

Output weights are updated according to the adaptation law described using the expression presented:

$$w^{out}(k+1) = w^{out}(k) - \eta^{out} \Delta w_j^{out}(k) \quad (24)$$

Parameters in the recurrent nodes of the third layer are changed similarly to the previous ones:

$$\Delta w_j^{3r}(k) = \frac{\partial E(k)}{\partial w_j^{3r}(k)} = \frac{\partial E(k)}{\partial z_j^{3r}(k)} \frac{\partial z_j^{3r}(k)}{\partial h_j^{3r}(k)} \frac{\partial h_j^{3r}(k)}{\partial w_j^{3r}(k)} \rightarrow \delta_j^3 z^3(k-1) \quad (25)$$

The new values are obtained by the following expression:

$$w^{3r}(k+1) = w^{3r}(k) - \eta^{3r} \Delta w_j^{3r}(k) \quad (26)$$

Weights of the second and the third layer are fixed as ones, thus there is no need to update those values. The parameters of the wavelet function are determined according to equations:

$$\Delta\mu_{ij}(k) = \frac{\partial E(k)}{\partial \mu_{ij}(k)} = \frac{\partial E(k)}{\partial z_{ij}^2(k)} \frac{\partial z_{ij}^2(k)}{\partial h_{ij}^2(k)} \frac{\partial h_{ij}^2(k)}{\partial \mu_{ij}(k)} \rightarrow \delta_{ij}^2 \frac{\phi\vartheta(\phi)(1-0.5\phi^2) - \phi\vartheta(\phi)}{\sigma_{ij}(k)} \quad (27)$$

$$\Delta\sigma_{ij}(k) = \frac{\partial E(k)}{\partial \sigma_{ij}(k)} = \frac{\partial E(k)}{\partial z_{ij}^2(k)} \frac{\partial z_{ij}^2(k)}{\partial h_{ij}^2(k)} \frac{\partial h_{ij}^2(k)}{\partial \sigma_{ij}(k)} \rightarrow \delta_{ij}^2 \frac{\phi\vartheta(\phi)(1-0.5\phi^2) - \phi\vartheta(\phi)}{\sigma_{ij}^2(k)} \quad (28)$$

Abovementioned values are applied in expressions:

$$\mu_{ij}(k+1) = \mu_{ij}(k) - \eta^\mu \Delta\mu_{ij}(k) \quad (29)$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \eta^\sigma \Delta\sigma_{ij}(k) \quad (30)$$

The last parameters that need to be updated are recurrent weights that are present in the input neurons:

$$\Delta w_i^{in}(k) = \frac{\partial E(k)}{\partial w_i^{in}(k)} = \frac{\partial E(k)}{\partial z_i^1(k)} \frac{\partial z_i^1(k)}{\partial h_i^1(k)} \frac{\partial h_i^1(k)}{\partial w_i^{in}(k)} \rightarrow \delta_i^1 z_i^1(k-1) \quad (31)$$

$$w_i^{in}(k+1) = w_i^{in}(k) - \eta^{3r} \Delta w_i^{in}(k) \quad (32)$$

Constants η^{out} , η^{4r} , η^{3r} , η^μ , and η^σ correspond to learning rates used in Equations (24), (26), (29), (30) and (32), respectively. Based on the above mathematical description, the adaptive neural controller was designed. The details of the topology are presented in Figure 2.

Figure 2. The recurrent wavelet neural network.

3. The Design Process of the Adaptive Neural Controller Using Grey Wolf Optimizer

The most basic process of establishing the initial values of weights and learning rates is to assign all the values using a pseudo-random number generator. Though it is easy, it comes with some disadvantages. The first few seconds of the operation of the system heavily depend on the initial values of these parameters. If they are too large, oscillations and overshoot may occur. On the other hand, when smaller values are applied, the network needs more time to adapt to the current state of the drive. To cope with this phenomenon, the initial values of the weights, the learning rates, and the gains in the control structure (an in-depth analysis of this topic is presented in Section 5) are chosen using a modified Grey Wolf Optimizer [36]. It should be noted that the optimization process is an offline operation performed before the start of the system, while backpropagation is constantly updating the weights of the controller during the system's operation.

Grey Wolf Optimizer is a nature-inspired metaheuristic algorithm. The most basic concept of the algorithm comes from observations of packs of wolves. When a pack tries to attack prey, they form a few smaller groups of wolves. All of them are led by an alpha wolf. Finally, the wolves approach their prey until they reach it.

In the algorithm, wolves are the points on the optimization plane, the prey is the investigated minimum of the function and the distance between the wolves and the prey is the value of the fitness function. In addition, some additional parameters are required to be established.

The current solution and the next point can be written as equations:

$$D = |C \cdot X_p(k_{iter}) - X(k_{iter})| \quad (33)$$

$$X(k_{iter} + 1) = X_p(k_{iter}) - A \cdot D \quad (34)$$

where X_p are the optimal values from the previous iteration, X is the current iteration solution, and k_{iter} is the number of the current iteration.

Parameters A and C are adjusted in every iteration of the algorithm, according to the equations below:

$$C = 2 \cdot r_1 \quad (35)$$

$$A = 2 \cdot a \cdot r_2 \quad (36)$$

Values of r_1 and r_2 are random in the range of $[0, 1]$. The value of the parameter a is changed from 2 to 0, descending over the course of iterations [49].

$$a = 2 \left(1 - \frac{k_{iter}}{k_{max}} \right) \quad (37)$$

When a is in the range of $[1, 2]$, the algorithm is in the exploration state. The exploitation state lasts when a is in the range of $[0, 1]$. In the exploration state, the algorithm seeks possible solutions in the search space while the exploitation results in narrowing the search plane. The modified formula for a allows for a longer exploration time so the final solution can be found more easily than in the classical GWO.

$$a = 2 \left(1 - \frac{k_{iter}^2}{k_{max}^2} \right) \quad (38)$$

Figure 3 shows how different formulae for a influence the GWO algorithm. The horizontal axis shows the iteration count (T is the maximum value set), whilst the vertical axis shows the change of a over the course of iterations. The area covered in blue marks the exploitation phase and the red area denotes the exploration state. In the basic version of the GWO, the time for both stages is equal, while the modified version emphasizes searching for the new values for the algorithm. If a better solution is needed, the exploration stage must be extended by altering Equation (35).

Figure 3. Changes of the a parameter in (a) the classical GWO, (b) the modified GWO.

The distance between the three best points (X_α , X_β , and X_δ) and the solution candidate can be calculated using Equations (39)–(41).

$$D_\alpha = |C_1 \cdot X_\alpha - X| \quad (39)$$

$$D_\beta = |C_2 \cdot X_\beta - X| \quad (40)$$

$$D_\delta = |C_3 \cdot X_\delta - X| \quad (41)$$

Later, the whole population is updated:

$$X_1 = X_1 - A_1 \cdot D_\alpha \quad (42)$$

$$X_2 = X_2 - A_2 \cdot D_\beta \quad (43)$$

$$X_3 = X_3 - A_3 \cdot D_\delta \quad (44)$$

$$X(k+1) = \frac{1}{3} \sum_{i=1}^3 X_i \quad (45)$$

The algorithm uses a fitness function that can be written as follows:

$$J_{GWO}(k_{iter}) = \frac{1}{t_{prob}} \sum_{k=1}^{t_{prob}} |\omega_{ref}^m(k) - \omega_1(k)| \quad (46)$$

where ω_{ref}^m output of reference model.

All the steps of the GWO algorithm are pictured below in a block diagram in Figure 4.

Figure 5 shows examples of different transients of motor speed gathered during the optimization process. The transient from the third iteration has the greatest amplitude of oscillations and the settling time is the slowest amongst all the transients shown. With each consecutive iteration, the achieved results are improving—after 20 iterations, the speed transient oscillations are almost fully dampened and the overshoot issue is solved. The fitness function values for each of the presented transients are attached in Table 1.

Changes in the fitness function for both versions of the algorithm are depicted below in Figure 6. The crucial parameters of both algorithms are presented in Table 2. The only change that influences the results is the additional multiplications in the rate of change of a . After the first few iterations, the fitness function was lowering much faster for the modified version compared to the classical. Additionally, it can be seen that when the value of a is lower than 1, the convergence is faster. A value higher than 1 causes the algorithm to

find a better value. After a few optimization runs, it was noticed that the modified Grey Wolf Optimizer can achieve similar values of the fitness function in less than half of the iterations required by the classic Grey Wolf Optimizer.

Figure 4. The Grey Wolf Optimizer.

Figure 5. Speed transients' changes during optimization.

Table 1. The values of the fitness function.

Iteration	Fitness Function Value (*10 ⁻⁴)
3	0.1661
5	0.1052
10	0.0683
20	0.0164

Figure 6. Comparison of fitness function for the GWO and the mGWO.

Table 2. Parameters of GWO and mGWO.

	GWO	mGWO
Number of iterations		20
Population		40
Rate of change of a parameter	$2\left(1 - \frac{k_{iter}}{k_{max}}\right)$	$2\left(1 - \frac{k_{iter}^2}{k_{max}^2}\right)$
Fitness function	$\frac{1}{t_{prob}} \sum_{k=1}^{t_{prob}} \left \omega_{ref}^m(k) - \omega_1(k) \right $	

The modified version of the algorithm can spend more time in the exploration state (red section of Figure 4) which means it is looking for the global extremum (minimum) longer. If one is found, in the exploitation phase the algorithm seeks the minimum value close to the extremum found. The longer the exploration phase lasts, the more accurate value should have been found, so the exploitation time (blue part of Figure 4) can be shortened.

The Grey Wolf Optimizer was implemented in Matlab software. For calculations, the machine with Intel Core 7-7700 CPU (3.60 GHz) with 16 GB of RAM and 64 bit Windows 10 was used. It took 12pprox.. 25.78 min to complete the optimization process for the modified version of the algorithm. The standard version is processed in about 17.52 min. During calculations an identical number of search agents and iterations were assumed—the exact parameters are presented in Table 2. An extended time needed for the optimization process is observed due to additional tasks (additional multiplications in the equation used for the value of a) in the code. However, these calculations are performed offline, therefore they do not affect the work of the whole system.

4. Simulation Tests

This section of the manuscript presents the numerical tests of the adaptive speed control structure based on the RWNN model. For all simulations, the sampling time equal to 100 μ s is assumed, calculations take 20 s. Parameters of the two-mass system obtained through the identification of the real drive used in the experiment are as follows: $T_1 = T_2 = 0.203$ s and $T_c = 0.0012$ s.

Results acquired for the nominal parameters of the drive are presented in Figure 7. The observation of the operation of the control structure with the recurrent wavelet neural network controller makes it possible to state that the drive is working properly. There is no apparent overshoot or oscillations. Electromagnetic torque is produced rapidly.

Such a result can be achieved through the continuous adaptation of parameters of the neural controller.

Figure 7. Speeds (ω_{ref} , ω_1 , and ω_2) and torques (m_e and m_s)—transients for nominal parameters of drive.

Next, a set of simulations is carried out to verify how the changes in the drive's mechanical properties (time constants) affect the drive's performance. Figure 8 shows gathered results for an increased value of the mechanical time constant of the load machine ($T_2 = 2 T_{2n}$). In this case, there is a slight overshoot at the first phase, but it decays as simulation time passes. Slight oscillations can also be observed in electromagnetic and shaft torques, this is the effect of an increased time constant in a drive with elastic joints.

In Figure 9 the transients of the rotational speeds and torques of the drive with an increased value of the time constant of the shaft are presented. The increased time constant T_c introduces negligible oscillations in torques and speeds of the motor and the load. It is apparent that before and after switching the load torque the drive achieves the reference speed with high dynamics.

Figures 10 and 11 show the results achieved for decreased time constants of the load motor and the shaft, respectively. Transients of both speeds for the reduced time constants are similar to the results obtained for the nominal parameters of the drive, except the speed drop occurring directly after the change of the load ($t_{L1} = 9$ s, $t_{L2} = 11$ s) is smaller for the reduced value of the shaft stiffness time constant T_c . The difference can also be noticed in torques transients. The torque changes are as dynamic as with the nominal values, but oscillations can be seen in both instances, especially when the load torque is switched.

Figure 8. Speeds (ω_{ref} , ω_1 , and ω_2) and torques (m_e and m_s)—transients for increased load time constant ($T_2 = 2 T_{2n}$).

Figure 9. Speeds (ω_{ref} , ω_1 , and ω_2) and torques (m_e and m_s)—transients for the increased stiffness time constant ($T_c = 2 T_{cn}$).

Figure 10. Speeds (ω_{ref} , ω_1 , and ω_2) and torques (m_e and m_s)—transients for the decreased load time constant ($T_2 = 0.5 T_{2n}$).

Figure 11. Speeds (ω_{ref} , ω_1 , and ω_2) and torques (m_e and m_s)—transients for the decreased shaft time constant ($T_c = 0.5 T_{cn}$).

To verify the performance of the proposed control structure, RWNN was compared to the classical PI controller. Both systems were tested after changing the mechanical time constant of the load drive— $T_{2n} = 2 T_{2n}$. Comparison is depicted in Figure 12. The

PI controller was tuned according to the pole placement method—parameters were set according to the predefined design parameters—the damping coefficient and the reference resonant frequency. It should be added that both gains of the controller are depended on the proper identification of the plant. In comparison to the PI controller, structure with the RWNN ensures adaptive properties. As a result, rapid response to the parametric changes of the drive can be observed. It is due to the weights being constantly updated. Moreover, high overshoot can be noticed in the drive with the classical solution, which is not present in the adaptive structure.

Figure 12. Results for RWNN controller and PI controller with increased load time constant— $T_{2n} = 2 T_{2n}$.

The impact of the learning rate value was also analyzed and shown in Figure 12. The results were gathered for the nominal parameters of the drive, but the outcome is similar when the parameters of the drive are changed.

Figure 13 shows how the drive behaves when different learning rates are applied to the controller where η_{mGWO} is the learning rate obtained from the mGWO optimization process ($\eta_{mGWO} = 0.0982$) and η_1 and η_2 are values 5 times lower and higher than the optimized value. Tests were performed for reduced and increased values of the learning rate. The speed of the drive is shaped in a similar way for the optimized and the greater learning rates, but oscillations are greater for the higher value. On the other hand, for the decreased value, it takes more time to reach the reference speed value. It was also observed that the bigger the value of the learning coefficient, the higher the maximum value of the overshoot, therefore results for any higher values are not shown in order not to blur the image.

Similar tests were carried out for the changes in the initial values of the weights of the controller. Figure 14 shows the changes in the speed of the motor speed when different initial values of weights were applied (red line indicates the load speed transient with the initial values optimized by the mGWO algorithm, the green line shows the situation, where the values of the weights were halved, and the yellow line shows the situation of a two-times increase in weights' value compared to the nominal ones).

As can be seen in Figure 14b, higher initial values cause greater overshoot in the starting process, and it takes more time to adapt the controller to the reference speed. On the other hand, when small values are applied the opposite is happening—there is an undershoot that needs to be corrected during the operation of the drive. The advantage of using the metaheuristic algorithm is the compromise between the two values with a long rise time and an overshoot that can be neglected as it is present only over a short period during the initial phase of the drive's operation.

Figure 13. Influence of the learning rate parameter on the adaptation process (a), the initial part of the simulation (b), applying load torque (c).

Figure 14. The impact of different initial values of weights in the adaptive speed controller based on the RWNN (a), the initial part of the simulation (b), applying load torque (c), the final part of the simulation (d).

In addition to transients, parameters of the step response were calculated for data presented in Figures 13 and 14 and presented in Table 3. The rise time of the signal can be calculated as the time it takes for the response to rise from 10% to 90% of the reference value. Settling time can be defined as the time it takes for the error to stay below 2% of the difference between reference speed and speed of the drive. All parameters were calculated for the first reversion from $\omega_{init} = -0.25$ p.u. to $\omega_{final} = 0.25$ p.u.

Table 3. Settling and rise times for different states of the drive.

Case	t_{rise} (s)	$t_{settling}$ (s)
$\eta_{mGWO}, w_{initGWO}$ (nominal)	0.186	0.25
$\eta_1 = 2 \eta_{mGWO}$	0.189	0.69
$\eta_2 = 0.5 \eta_{mGWO}$	0.191	0.27
$w_{init1} = 2 w_{initGWO}$	0.190	0.34
$w_{init2} = 0.5 w_{initGWO}$	0.263	0.40

The rise time for the nominal parameters, as well as for different learning rates applied, is almost equal. Lower initial values of the weights affect the time needed for adaptation, which results in lower rise time and higher overshoot in the first reversion of the drive. Even minimal changes of the analyzed parameters can influence the quality of the response (the settling time). The fastest response of the system can be observed for the neural network with parameters optimized with mGWO.

Initial weights of the controller also have an influence on the parameters of the step response. Rise time for increased initial values stays the same, but lowering them increases the value. Lower values of initial weights introduce overshoot which increases the rise time and settling time of the speed of the drive. Higher initial values of weights cause oscillations which increase the settling time.

5. The Influence of Additional Feedbacks in the Speed Loop

All previous tests were conducted for a structure with a classical negative speed feedback loop, where the speed of the motor was subtracted from the reference speed. In addition to the motor speed feedback, auxiliary feedbacks were applied with no additional gains present in these feedback loops. Additional loops can improve the characteristics of the drive in dynamical states, e.g., reversions. Supplementary feedback in the torque control loop provides better damping of torsional vibrations [50]. A signal from the difference in speeds inserted in a speed loop ensures good dynamical characteristics [51]. There are many possible feedback loop modifications, but for this study, the simplest were selected. The derivatives of the state variables could also be used, but they cause damping of high-frequency vibrations and a decline in the system's dynamical properties.

Results achieved for different feedbacks depicted in Figure 15 are presented in this section. By taking a look at Figure 16, only a small difference can be observed. Close-ups of different parts of the simulation are given in Figure 16b,c. First reversion shows that there are slight oscillations when additional feedback from the speed difference is applied (yellow graph). Green and red transients, indicating additional shaft torque and speed difference feedbacks and shaft torque feedback, respectively, are the closest to reference speed. The structure with no additional feedbacks has the highest error between the reference speed and the load speed. On the contrary, after switching the load at $t_{L1} = 9$ s, the red and green transients are the most distant from the set speed transient.

Oscillations are damped the most when the additional shaft torque feedback is applied (red and green transients in Figure 16b). During the simulations, all parameters of the recurrent wavelet neural network controller are updated, so the difference in speed transients is less visible at the end of the test. After applying the load, the structure with no additional feedbacks and with the feedback loop from the difference between the speeds perform the best. Near the end of the simulation tests in Figure 16c, it can be seen that all structures perform similarly. Drive with both loops (shaft torque and difference between speeds) achieves the best results overall.

To achieve the best results of the drive, it is suggested to implement the structure incorporating both the shaft torque feedback loop as well as the speed difference feedback loop. It is assumed that in the simulations all the state variables are accessible, and therefore they do not need to be estimated.

Figure 15. Combinations of feedbacks applied in the control structure, a classical feedback loop (a), an additional speed difference feedback (b), an additional shaft torque feedback (c) and a combination of speed and torque feedbacks (d).

Figure 16. Comparison of speeds (ω_{ref} and ω_1) transients obtained for different connections of feedbacks (a), the initial part of the simulation (b), the final of the simulation (c).

6. Experimental Results

To confirm the theoretical tests, experimental studies on the laboratory stand were carried out. The laboratory system presented in Figure 17 consists of two DC motors coupled by a long steel shaft. To change the time constant of the load, motor flywheels can be added. Drive is powered through an H-bridge. Current measurements are conducted by LEM sensors and incremental encoders are mounted on both machines to measure the speed of the drive. The control algorithm is compiled on a PC and uploaded to a DSpace 1103 card with a digital signal processor. Everything is connected through a control panel. To simplify the process of compiling and uploading the algorithm to the processor, the controller was implemented as a Matlab Embedded Function.

Figure 17. The laboratory setup.

The code is built using Matlab's built-in compiler. These data are then loaded to a ControlDesk virtual panel (which is a part of the dSPACE software). ControlDesk is used to capture signals from the dSPACE card, display the results and then save them in a *.mat file which can be read by Matlab. The nominal parameters of the experimental setup are presented in Table 4.

Table 4. Parameters of the experimental setup.

Motor Nominal Power	500 W
Load nominal power	500 W
Shaft length	600 mm
Shaft diameter	5 mm
Encoder impulse	36,000 pulses/rev

The experimental verification of the simulation tests can be found in Figure 18, presented transients match the results obtained in Section 4. The upper part of the graph shows the speeds of the motor and the load for nominal time constants of the drive. Reference speed of 25% of the nominal speed value was used with cyclic reversions occurring every 5 s.

The lower part of the graph shows the influence of an increased load time constant on the control structure. A small overshoot is observed, but it is gradually reduced with the adaptation process. The same can be noticed for the oscillations of speeds close to the setpoint, the frequency of oscillations decreases over the time of the experiment. The reaction to the load torque is comparable to the results achieved with the nominal parameters.

Settling time and rise time for nominal parameters in the experimental bench were also calculated. The calculated value of the rise time was equal to $t_{rise} = 0.25$ s, while the settling time was $t_{settling} = 0.38$ s which is comparable to simulation results.

Another test was also carried out to see how an increased learning rate impacts the dynamics of the drive. As presented in Figure 19, when the higher value of learning rate is applied, the overshoot on the first reversion is substantially higher. After the controller updates all parameters, the overshoot is reduced to zero. The test proves the control structure's adaptive qualities. Other than that, the performance of the drive is exactly the same as in previous tests.

Figure 18. Transients of experimental results for nominal parameters of the drive (a) and increased load time constant (b).

Figure 19. Results from the experimental setup obtained after increasing the value of the learning rate.

7. Concluding Remarks

In this paper, a recurrent wavelet neural network designed to control the electrical drive with an elastic shaft is investigated. Various combinations of state feedback signals from the plant are considered. It starts with only a basic connection from the motor speed, then additional solutions are analyzed. It can be important not only in theoretical assumptions, but also in real applications (economic aspects and reliability). One of the main points of the study is the application of the modified Grey Wolf Optimizer in the design process of the control structure. The efficiency of the proposed solutions was tested in simulations and experiments. Based on the obtained results, the main remarks presented below can be formulated.

- Recurrent wavelet neural network can form the basis of an adaptive speed controller designed for an electrical drive with a compound mechanical part.
- Presented results show that the adaptation is performed properly. As a result, overshoots are reduced, and oscillations are damped.
- Correct work of the control system is observed even when the parameters are changed.
- The Grey Wolf Optimizer can be used as a universal tool for solving issues observed in the design process of adaptive neural speed controllers for the drive.
- Proposed improvements (achieved through processing the data using the mGWO) expedite the process of the neural controller optimization.
- To effectively damp the oscillations of the state variables, implementation of an additional feedback loop is necessary.

Author Contributions: Conceptualization, M.K., R.S. and M.Z.; methodology, M.Z.; software, M.Z.; data curation, M.Z.; data, M.Z.; writing—original draft preparation, M.K., R.S. and M.Z.; writing—review and editing, M.K., R.S. and M.Z.; visualization, M.Z.; supervision, M.K.; funding acquisition, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Pedraza, L.F.; Hernández, H.A.; Hernández, C.A. Artificial Neural Network Controller for a Modular Robot Using a Software Defined Radio Communication System. *Electronics* **2020**, *9*, 1626. [\[CrossRef\]](#)
2. Yan, J.; Jin, L.; Yuan, Z.; Liu, Z. RNN for Receding Horizon Control of Redundant Robot Manipulators. *IEEE Trans. Ind. Electron.* **2022**, *69*, 1608–1619. [\[CrossRef\]](#)
3. Muthusamy, P.K.; Garratt, M.; Pota, H.; Muthusamy, R. Real-Time Adaptive Intelligent Control System for Quadcopter Unmanned Aerial Vehicles with Payload Uncertainties. *IEEE Trans. Ind. Electron.* **2022**, *69*, 1641–1653. [\[CrossRef\]](#)
4. Pajchrowski, T.; Zawirski, K. Application of artificial neural network to robust speed control of servodrive. *IEEE Trans. Ind. Electron.* **2007**, *54*, 200–207. [\[CrossRef\]](#)
5. Kaminski, M. Nature-Inspired Algorithm Implemented for Stable Radial Basis Function Neural Controller of Electric Drive with Induction Motor. *Energies* **2020**, *13*, 6541. [\[CrossRef\]](#)
6. Brock, S.; Łuczak, D.; Nowopolski, K.; Pajchrowski, T.; Zawirski, K. Two approaches to speed control for multi-mass system with variable mechanical parameters. *IEEE Trans. Ind. Electron.* **2017**, *64*, 3338–3347. [\[CrossRef\]](#)
7. Liu, G.; Xu, X.; Yu, X.; Wang, F. Graphite Classification Based on Improved Convolution Neural Network. *Processes* **2021**, *9*, 1995. [\[CrossRef\]](#)
8. Ruan, W.; Dong, Q.; Zhang, X.; Li, Z. Friction Compensation Control of Electromechanical Actuator Based on Neural Network Adaptive Sliding Mode. *Sensors* **2021**, *21*, 1508. [\[CrossRef\]](#)
9. Mohamed, Y.A.I.M. A Novel Direct Instantaneous Torque and Flux Control With an ADALINE-Based Motor Model for a High Performance DD-PMSM. *IEEE Trans. Power Electron.* **2007**, *22*, 2042–2049. [\[CrossRef\]](#)
10. Soufleri, E.; Roy, K. Network Compression via Mixed Precision Quantization Using a Multi-Layer Perceptron for the Bit-Width Allocation. *IEEE Access* **2021**, *9*, 135059–135068. [\[CrossRef\]](#)
11. Ke, K.-C.; Huang, M.-S. Quality Prediction for Injection Molding by Using a Multilayer Perceptron Neural Network. *Polymers* **2020**, *12*, 1812. [\[CrossRef\]](#)
12. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [\[CrossRef\]](#)
13. Jordan, M.I. Generic constraints on underspecified target trajectories. In Proceedings of the International Joint Conference on Neural Networks, Washington, DC, USA, 18–22 June 1989; Volume 1, pp. 217–225. [\[CrossRef\]](#)
14. Kamiński, M. Recurrent neural controller applied for two-mass system. In Proceedings of the International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland, 29 August–1 September 2016; pp. 128–133. [\[CrossRef\]](#)
15. Chien, T.; Chen, C.; Huang, Y.; Lin, W. Stability and Almost Disturbance Decoupling Analysis of Nonlinear System Subject to Feedback Linearization and Feedforward Neural Network Controller. *IEEE Neural Netw.* **2008**, *19*, 1220–1230. [\[CrossRef\]](#)

16. Yipeng, L.; Wenkang, Y.; Ming, Z. Design of Adaptive Neural Network Backstepping Controller for Linear Motor Magnetic Levitation System. In Proceedings of the IEEE Industry Applications Society Annual Meeting, Baltimore, MD, USA, 29 September–3 October 2019; pp. 1–6. [\[CrossRef\]](#)
17. Han, H.; Zhang, L.; Hou, Y.; Qiao, J. Nonlinear Model Predictive Control Based on a Self-Organizing Recurrent Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 2. [\[CrossRef\]](#)
18. Lin, F.; Wai, R.; Chen, M. Wavelet neural network control for linear ultrasonic motor drive via adaptive sliding-mode technique. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2003**, *50*, 686–698. [\[CrossRef\]](#)
19. Zhang, J.; Walter, G.G.; Miao, Y.; Lee, W.N.W. Wavelet neural networks for function learning. *IEEE Trans. Signal Process.* **1995**, *43*, 1485–1496. [\[CrossRef\]](#)
20. Chen, C.F.; Hsiao, C.H. Wavelet approach to optimizing dynamic systems. *IEEE Trans. Signal Process.* **1995**, *43*, 1485–1496. [\[CrossRef\]](#)
21. Gao, A.; Zhang, F.; Fu, Z.; Zhang, Z.; Li, H. The SOC estimation and simulation of power battery based on self-recurrent wavelet neural network. *Chin. Autom. Congr.* **2017**, 4247–4252. [\[CrossRef\]](#)
22. Rafiei, M.; Niknam, T.; Aghaei, J.; Shafie-khah, M.; Catalão, J.P.S. Probabilistic Load Forecasting using an Improved Wavelet Neural Network Trained by Generalized Extreme Learning Machine. *IEEE Trans. Smart Grid* **2018**, *9*, 6961–6971. [\[CrossRef\]](#)
23. Sheng, L.; Xiaojie, G.; Lanyong, Z. Robust Adaptive Backstepping Sliding Mode Control for Six-Phase Permanent Magnet Synchronous Motor Using Recurrent Wavelet Fuzzy Neural Network. *IEEE Access* **2017**, *5*, 14502–14515. [\[CrossRef\]](#)
24. Xiao, Q.; Ge, G.; Wang, J. The Neural Network Adaptive Filter Model Based on Wavelet Transform. In Proceedings of the 2009 Ninth International Conference on Hybrid Intelligent Systems, Shenyang, China, 12–14 August 2009; pp. 529–534. [\[CrossRef\]](#)
25. Khan, M.A.; Uddin, M.N.; Rahman, M.A. A Novel Wavelet-Neural-Network-Based Robust Controller for IPM Motor Drives. *IEEE Trans. Appl.* **2013**, *49*, 2341–2351. [\[CrossRef\]](#)
26. Liang, J.; Chen, H. Development of a Piezoelectric-Actuated Drop-On Demand Droplet Generator using Adaptive Wavelet Neural Network Control Scheme. In Proceedings of the 2013 IEEE International Conference on Mechatronics and Automation, Takamatsu, Japan, 4–7 August 2013; pp. 382–388. [\[CrossRef\]](#)
27. Li, Z.; Ruan, Y. A novel control method based on wavelet neural networks for vector control of induction motor drives. In Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition, Hong Kong, China, 30–31 August 2008; pp. 1–6. [\[CrossRef\]](#)
28. Sharma, M.; Verma, A. Adaptive Observer Based Tracking Control for a Class of Uncertain Nonlinear Systems with Delayed States and Input Using Self Recurrent Wavelet Neural Network. In Proceedings of the 2nd International Conference on Advances in Computing, Control, and Telecommunication Technologies, Jakarta, Indonesia, 203 December 2010; pp. 27–31. [\[CrossRef\]](#)
29. El-Sousy, F.F.M.; Abuhasel, K.A. Self-organizing recurrent fuzzy wavelet neural network-based mixed H₂/H_∞ adaptive tracking control for uncertain two-axis motion control system. *IEEE Ind. Appl. Soc. Ann. Meet.* **2015**, 1–14. [\[CrossRef\]](#)
30. Gao, W.; Guo, Z. Research of recurrent wavelet neural network speed controller based on chaotic series adaptive PSO. In Proceedings of the 3rd International Conference on Information Management (ICIM), Chengdu, China, 21–23 April 2017; pp. 470–475. [\[CrossRef\]](#)
31. Knypiński, Ł.; Kuroczycki, S.; Márquez, F.P.G. Minimization of Torque Ripple in the Brushless DC Motor Using Constrained Cuckoo Search Algorithm. *Electronics* **2021**, *10*, 2299. [\[CrossRef\]](#)
32. Tarczewski, T.; Grzesiak, L.M. Artificial bee colony based auto-tuning of PMSM state feedback speed controller. In Proceedings of the 2016 IEEE International Power Electronics and Motion Control Conference, Varna, Bulgaria, 5–28 September 2016; pp. 1155–1160. [\[CrossRef\]](#)
33. Zhao, J.; Lin, M.; Xu, D.; Hao, L.; Zhang, W. Vector control of a hybrid axial field flux-switching permanent magnet machine based on particle swarm optimization. *IEEE Trans. Magn.* **2015**, *51*, 1–4. [\[CrossRef\]](#)
34. Tarczewski, T.; Grzesiak, L.M. An application of novel nature-inspired optimization algorithms to auto-tuning state feedback speed controller for PMSM. *IEEE Trans. Ind. Appl.* **2018**, *54*, 2913–2925. [\[CrossRef\]](#)
35. Hannan, M.A.; Ali, J.A.; Mohamed, A.; Hussain, A. Optimization techniques to enhance the performance of induction motor drives: A review. *Renew. Sustain. Energy Rev.* **2018**, *81*, 1611–1626. [\[CrossRef\]](#)
36. Xu, L.; Wang, H.; Lin, W.; Gulliver, T.A.; Le, K.N. GWO-BP Neural Network Based OP Performance Prediction for Mobile Multiuser Communication Networks. *IEEE Access* **2019**, *7*, 152690–152700. [\[CrossRef\]](#)
37. Jaiswal, K.; Mittal, H.; Kukreja, S. Randomized grey wolf optimizer (RGWO) with randomly weighted coefficients. In Proceedings of the 2017 Tenth International Conference on Contemporary Computing, Noida, India, 10–12 August 2017; pp. 1–3. [\[CrossRef\]](#)
38. GU, W.; Zhou, B. Improved grey wolf optimization based on the Quantum-behaved mechanism. In Proceedings of the 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference, Chengdu, China, 20–22 December 2019; pp. 1537–1540. [\[CrossRef\]](#)
39. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [\[CrossRef\]](#)
40. Mirjalili, S.; Saremi, S.; Mirjalili, S.M.; Coelho, L.S. Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Syst. Appl.* **2016**, *47*, 106–119. [\[CrossRef\]](#)
41. Knypiński, Ł. Modified grey wolf method for optimization of PM motors. *ITM Web Conf.* **2019**, *28*. [\[CrossRef\]](#)

42. Kamiński, M.; Szabat, K. Adaptive Control Structure with Neural Data Processing Applied for Electrical Drive with Elastic Shaft. *Energies* **2021**, *14*, 3389. [[CrossRef](#)]
43. Szczepanski, R.; Kaminski, M.; Tarczewski, T. Auto-Tuning Process of State Feedback Speed Controller Applied for Two-Mass System. *Energies* **2020**, *13*, 3067. [[CrossRef](#)]
44. Szabat, K.; Wróbel, K.; Drózd, K.; Janiszewski, D.; Pajchrowski, T.; Wójcik, A. A Fuzzy Unscented Kalman Filter in the Adaptive Control System of a Drive System with a Flexible Joint. *Energies* **2020**, *13*, 2056. [[CrossRef](#)]
45. Vašak, M.; Perić, N.; Szabat, K.; Cychowski, M. Patched LQR control for robust protection of multi-mass electrical drives with constraints. In Proceedings of the IEEE International Symposium on Industrial Electronics, Bari, Italy, 4–7 July 2010; pp. 3153–3158. [[CrossRef](#)]
46. Tang, Y.; Sun, W.; Wang, Y.; Zhai, X. Using Recurrent Fuzzy Wavelet Neural Network to Control AC Servo System. In Proceedings of the IEEE 5th International Power Electronics and Motion Control Conference, Shanghai, China, 14–16 August 2006; Volume 2, pp. 1–4. [[CrossRef](#)]
47. Song, J.; Shi, H. Dynamic system modeling based on wavelet recurrent fuzzy neural network. *Intern. Conf. Nat. Comput.* **2011**, *2*, 766–770. [[CrossRef](#)]
48. Liu, W.Y.; Gao, Q.W.; Zhang, Y. A novel wind turbine de-noising method based on the Genetic Algorithm optimal Mexican hat wavelet. In Proceedings of the 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence, Xi'an, China, 19–22 August 2016; pp. 1003–1006. [[CrossRef](#)]
49. Karnavas, Y.L.; Chasiotis, I.D. PMDC coreless micro-motor parameters estimation through Grey Wolf Optimizer. In Proceedings of the International Conference on Electrical Machines, Lausanne, Switzerland, 4–7 September 2016; pp. 865–870. [[CrossRef](#)]
50. Kaminski, M. Neural Network Training Using Particle Swarm Optimization—A Case Study. In Proceedings of the 24th International Conference on Methods and Models in Automation and Robotics, Międzyzdroje, Poland, 26–29 August 2019; pp. 115–120. [[CrossRef](#)]
51. Seizović, A.; Vojvodić, N.; Ristić, L.; Bebić, M. Energy efficient control of variable-speed induction motor drives based on Particle Swarm Optimization. In Proceedings of the International Symposium on Industrial Electronics and Applications, Kristiansand, Norwa, 9–13 November 2020; pp. 1–6. [[CrossRef](#)]