algorithms

MDPI

Article

Metaheuristic and Heuristic Algorithms-Based Identification Parameters of a Direct Current Motor

David M. Munciño [1], Emily A. Damian-Ramírez [1], Mayra Cruz-Fernández [1,*], Luis A. Montoya-Santiyanes [1,*] and Juvenal Rodríguez-Reséndiz [2]

[1] Industrial Technologies Division, Universidad Politécnica de Querétaro, El Marqués, Querétaro 76240, Mexico; 120037491@upq.edu.mx (D.M.M.); 119034035@upq.edu.mx (E.A.D.-R.)
[2] Facultad de Ingeniería, Universidad Autónoma de Querétaro, Santiago de Querétaro 76010, Mexico; juvenal@uaq.edu.mx
* Correspondence: mayra.cruz@upq.edu.mx (M.C.-F.); luis.montoya@upq.edu.mx (L.A.M.-S.)

Abstract: Direct current motors are widely used in industry applications, and it has become necessary to carry out studies and experiments for their optimization. In this manuscript, a comparison between heuristic and metaheuristic algorithms is presented, specifically, the Steiglitz–McBride, Jaya, Genetic Algorithm (GA), and Grey Wolf Optimizer (GWO) algorithms. They were used to estimate the parameters of a dynamic model that approximates the actual responses of current and angular velocity of a DC motor. The inverse of the Euclidean distance between the current and velocity errors was defined as the fitness function for the metaheuristic algorithms. For a more comprehensive comparison between algorithms, other indicators such as mean squared error (MSE), standard deviation, computation time, and key points of the current and velocity responses were used. Simulations were performed with MATLAB/Simulink 2010 using the estimated parameters and compared to the experiments. The results showed that Steiglitz–McBride and GWO are better parametric estimators, performing better than Jaya and GA in real signals and nominal parameters. Indicators say that GWO is more accurate for parametric estimation, with an average MSE of 0.43%, but it requires a high computational cost. On the contrary, Steiglitz–McBride performed with an average MSE of 3.32% but required a much lower computational cost. The GWO presented an error of 1% in the dynamic response using the corresponding indicators. If a more accurate parametric estimation is required, it is recommended to use GWO; however, the heuristic algorithm performed better overall. The performance of the algorithms presented in this paper may change if different error functions are used.

Keywords: Steiglitz–McBride; Jaya; grey wolf optimizer; genetic algorithm; heuristic; metaheuristic; DC motor

Citation: Munciño, D.M.; Damian-Ramírez, E.A.; Cruz-Fernández, M.; Montoya-Santiyanes, L.A.; Rodríguez-Reséndiz, J. Metaheuristic and Heuristic Algorithms-Based Identification Parameters of a Direct Current Motor. *Algorithms* **2024**, *17*, 209. https://doi.org/10.3390/a17050209

Academic Editors: Łukasz Knypiński, Ramesh Devarapalli and Marcin Kaminski

Received: 10 March 2024
Revised: 6 May 2024
Accepted: 8 May 2024
Published: 11 May 2024

1. Introduction

A direct current (DC) motor has many applications in the industrial sector [1], such as aeronautics, robotics, automation, manufacturing, automotive, etc. Therefore, it is constantly subjected to multiple experiments or investigations in identification, estimation, and parametric control to create solutions in different areas of research that could have applications in real life. For this, the precise control of motor parameters is required to achieve efficient operation [2], and model optimization should help achieve this goal.

Sometimes, models are simulated mathematically, drifting in predictions, control monitoring, and diagnosis. Motor parameters vary with features such as misalignments that can lead to poor motor performance. The design of DC motor controllers could have drawbacks, as they are sensitive to high tolerances [3]. Usually, the parameters specified by manufacturers are somewhat robust, and it is necessary to generate more precise models that help us make a more efficient motor. With precise models, diagnostic systems can be developed, as shown in the research by ref. [4], where a diagnostic method based on a rotor

slip was designed to detect faults in the current and speed sensors during the induction motor operation. Another example is from ref. [5], where a technique based on current estimation for fault detection and speed sensor failure for an induction motor is developed. The precise knowledge of mathematical motor models allows for the bypassing of sensors and efficient speed control, as shown in the research by ref. [6], where high-performance field-oriented motor control requires the accurate knowledge of the flow of information and the motor speed. These investigations denote the importance of motor modeling and parameterization, since they allow for better fault-tolerant control, reduce the number of sensors, and even improve control precision.

For example, an improved loss minimization technique based on fuzzy logic for a brushless DC motor (BLDC) drive system is presented in [7]. In addition, the implementation of a platform for testing an algorithm for the closed-loop speed control of a DC motor was sought in [8]. These have not been the only works that have focused on controlling speed through algorithms. Speed control algorithms have also been designed based on the estimation of dynamic error parameters under uncertainty and load variation [9].

Heuristic algorithms solve optimization problems defined by intuitive approximations, and solutions to these problems are intelligently thought out even if the solution is not the best. In recent years, heuristic procedures have been developed to optimize software design problems and component reuse. Some research shows the usefulness of this type of algorithm and its application in parametric evaluation [10]. Some of the proposed limitations of heuristic algorithms have been overcome with more robust methods based on statistics, such as in ref. [11]. However, it is necessary to continue innovating [12]. Investigations such as [13] propose an algorithm that uses a stochastic method to reach optimal points within simple algorithms.

A comparison of heuristic algorithms for the identification of DC motor systems was carried out in [14] through discrete Proportional–Integral (PI) controllers to analyze the system response. Fast and accurate convergence rate results were obtained with the extended Kalman filter (EKF) algorithm. A heuristic method is a well-known set of steps that is used to quickly identify a high-quality solution to a given problem. It consists of intuitive mathematics with a design of static and manual rules that are supposed to give a good solution, although not necessarily the optimal solution, to the problem. For this reason, it is important to compare them with metaheuristic algorithms and observe which offer the highest optimization.

Metaheuristic algorithms have acquired an important role in recent years. The optimization field has been no exception, because metaheuristic algorithms are iterative and intelligently combine the principles of evolution, natural selection, and inheritance, inspired by physical phenomena and different behaviors of animals and even humans. This makes it possible to correctly explore the search space by offering a population of feasible solutions. Due to their simplicity, they are general purpose algorithms and similar phenomena inspire them. In recent years, these algorithms have successfully solved practical problems in different fields. They are easily implementable, as they do not require any particular changes to their structure when applying them across various themes unless optimization is sought. However, there may be limitations, since most of these optimization methods start from random solutions, resulting in approximate, not exact, solutions.

Recent contributions, such as [15], showed that the Artificial Fish Swarm Algorithm (AFSA) has advantages, including high convergence speed, flexibility, fault tolerance, and high accuracy. However, it has been shown that they contain high temporal complexity and a lack of balance in global and local searches. On the other hand, the work by ref. [16] shows that Dolphin Partner Optimization (DPO) offers a quick solution with optimal stability in different function targets. Another area of interest in algorithms is convergence. A study on the hybrid algorithm Particle Swarm–Ant Colony Optimization (PS–ACO) has been presented [17], where ACO was used as a parallel calculation mechanism and showed excellent robustness. Unfortunately, they faced the limitations of stagnation and premature convergence. The Particle Swarm Optimization (PSO) algorithm as an adaptation of reactive

power optimization [18] proved to be decisive. Refs. [19,20] use metaheuristic algorithms to optimize a fuzzy controller. Still, it needs predefined parameters for a user-determined issue. These problems can be solved through adaptive adjustments. In [21], the author offers solutions of low algorithmic complexity with optimal convergence results in Proportional–Integral–Derivative (PID) controllers. An example is the Cuckoo algorithm [22] which is a relatively simple method, having minor variation in parameters and containing fast convergence. However, it also has premature convergence defects and low calculation accuracy, which, in the worst case, results in inaccurate solutions.

Genetic algorithms (GA) are another method widely used to optimize a DC motor because of their low complexity of understanding and ease of adaptation. Such is the case in [23], where it is mentioned that GAs are appropriate for the estimation of platform parameters with nonlinear characteristics. Something similar occurs in [24], where it is described that GAs offer superior results in the estimation of parameters. Despite recent articles, GAs also have limitations, since they starts from random solutions. The algorithm requires improvements to optimize it, as shown in ref. [25]. Another example of this is shown in [26], where it is mentioned that although they are good at optimization, the traditional GA still has some shortcomings because most of the time, they require modifications. Therefore, a New Adaptive Genetic Algorithm (NAGA) is proposed to overcome the disadvantages of the traditional one.

Metaheuristic algorithms are sometimes used in combination with other methods to achieve better performance, such as in the case of ref. [27], where they are used for the speed control of a DC motor. Better dynamic and static performance is demonstrated thanks to the implementation of a PID controller with a Backtracking Search Algorithm (BSA) in comparison with a PSO. PID controllers are commonly implemented to drive DC motors and are tuned using different algorithms. This is the case in ref. [28], where a PID controller was tuned through the Jaya optimization algorithm to control the speed of the motor, obtaining better responses in the transitory stage. Speed control has been the subject of many experiments, as in [29], where the optimization of fuzzy rules using GA proved to be an effective method for rate accuracy. In other cases, the original metaheuristic algorithms are modified to optimize their performance. Modifications to the original cuckoo algorithm have been made [30], improving the parametric estimation.

This manuscript presents a parametric estimation of the dynamic model of a DC motor. The study was carried out using the following algorithms: Steiglitz–McBride, Jaya, Genetic Algorithm (GA), and Gray Wolf Optimizer (GWO). These algorithms were used to compare the optimization of the actual responses of current and angular velocity between heuristic and metaheuristic algorithms. Simulations were carried out in MATLAB/Simulink 2010. The evaluation criteria consisted of approximation error, computational cost, convergence time, and dynamic signal behavior. Stabilization times, overshoot values, and the average value of steady-state current were also discussed. This paper is organized as follows. The Section 2 presents the foundations of the DC motor dynamic model, the parameters that govern the development of the metaheuristic algorithms, and the conditions for simulations. The subsequent Section 3 recalls a brief background of the selected algorithms for optimization, diagrams, and conditions for parameter estimation. Sections 4 and 5 show the Results and Discussion, respectively. Here, the parameters estimated by the different algorithms are compared using different metrics. Finally, the Section 6 concludes the present work.

2. Mathematical Modelling

DC Motor Modelling

The dynamic system of the DC motor is composed of two differential equations (an electrical and a mechanical part). The electrical equation has variables such as voltage and current and parameters such as a resistor, which is the internal resistance of the motor, and an inductor, which represents the inductance generated by its windings. In the mechanical

equation, there are variables such as speed and torque and mechanical parameters such as the coefficient of friction and inertia, as shown in Figure 1.

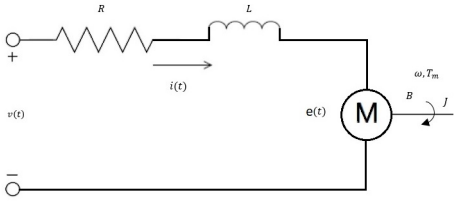

Figure 1. DC motor dynamic circuit diagram.

From the diagram in Figure 1, the following equations are obtained.

$$v(t) = Ri(t) + L\frac{di(t)}{dt} + e(t) \tag{1}$$

$$\tau(t) = J\frac{d\omega(t)}{dt} + B\omega(t) + T_L \tag{2}$$

$$e(t) = K_e\omega(t) \tag{3}$$

$$\tau(t) = K_m i(t) \tag{4}$$

$$K_e \approx K_m = K \tag{5}$$

where R is the resistance expressed in ohms Ω, L the inductance in H, J is the moment of inertia of the rotor, B the coefficient of friction between the rotor and the stator, and T_L the torque load; however, for parameter estimation, a motor free of load is used, which means that $T_L = 0$. $v(t)$ is the voltage induced to the armature, $i(t)$ is the current, $\omega(t)$ is the angular velocity of the rotor, $e(t)$ is the induced electric voltage, and $\tau(t)$ is the torque of the motor. K is the rate of change of the electromotive force with respect to the angular velocity, and the rate of change of the induced voltage is considered to be equivalent to both K_m and K_e. Equation (1) was derived using a simple Kirchhoff's voltage law analysis presented in Figure 1. Equation (2) was obtained by knowing that the sum of moments that make the rotor rotate to the symmetrical axis is the same as that of moments that oppose its movement. K is the rate of change of the electromotive force with respect to the angular velocity, and the rate of change of the induced voltage is considered equivalent to both K_m and K_e. This is because it is a DC motor and the excitation source is considered constant; hence, the previous constants are considered equal. This can be deduced by working with the instantaneous power equation, which relates to the instantaneous power supplied by the excitation source, the instantaneous mechanical power, and the instantaneous power dissipated by the motor windings [31].

The system of differential equations shown in Equations (6) and (7) is obtained from the combination and substitution of Equations (3) and (4) into (1) and (2), respectively.

$$\frac{di(t)}{dt} = \frac{v(t) - Ri(t) - K_e\omega(t)}{L} \tag{6}$$

$$\frac{d\omega(t)}{dt} = \frac{K_m i(t) - B\omega(t)}{J} \tag{7}$$

The equations represent the dynamic model of a DC motor, where Equation (6) corresponds to the mechanical part of the motor, and Equation (7) corresponds to the electrical part. The time response of the dynamic system was simulated using the nominal parameters, as shown in Table 1. Later, simulations were performed with the parameters selected

by the algorithms, as described in the following section. It should be noted that the model and nominal parameters may have differences from reality, although they are considered so small as to be negligible.

Table 1. Nominal values of M2 motor.

R	L	K	B	J
0.921042 Ω	0.0077590 H	0.073472	0.000678 $\frac{kg \cdot m^3}{s^2}$	0.000136 $kg \cdot m^2$

The simulation of differential equations was carried out through a block diagram in MATLAB/Simulink 2010. The arrangement consisted of six inputs, which are the five described in Table 1 and the input voltage. The current and angular velocity outputs were delivered as a result. The setup can be seen in Figure 2.

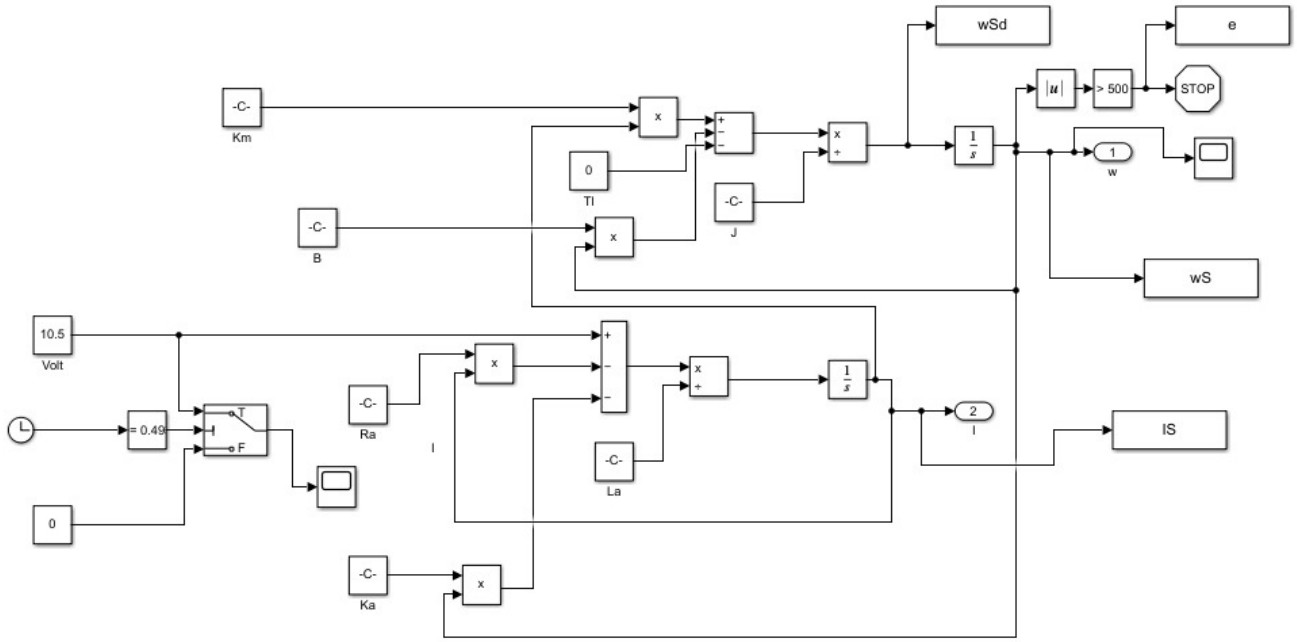

Figure 2. MATLAB/Simulink 2010 setup for the dynamic model.

3. Optimization Algorithms

3.1. Steiglitz–McBride Algorithm as a Parametric Estimator

The Steiglitz–McBride algorithm is a modification of the Minimum Square Recursive (MSR) algorithm. This algorithm was selected because it is a widely used heuristic algorithm equivalent to the GA in metaheuristics. Any heuristic problem can be solved by this algorithm. In the same way, this algorithm has already been subjected to comparisons with metaheuristic algorithms, obtaining favorable results [30]. The most attractive feature of this algorithm is its ability to find the Mean Square Error (MSE) for a linear system by sampling the inputs and outputs in an iterative process, calculating the parameters using a filter defined in the code equivalent to the system. The algorithm takes the results once the filter is applied to the input parameters. It uses them as a starting point in the next iteration, seeking to reduce the error on each iteration. In this way, the best solution is obtained. As mentioned above, the algorithm works in two phases: pre-filtering and the estimation of the parameters. The behavior of the algorithm is better explained in Figure 3, where the two phases of the algorithm, prefiltering and estimation, are shown. In the first phase, the image shows how it uses the input and output of the plant to adjust (in the second phase, it uses the least squares algorithm).

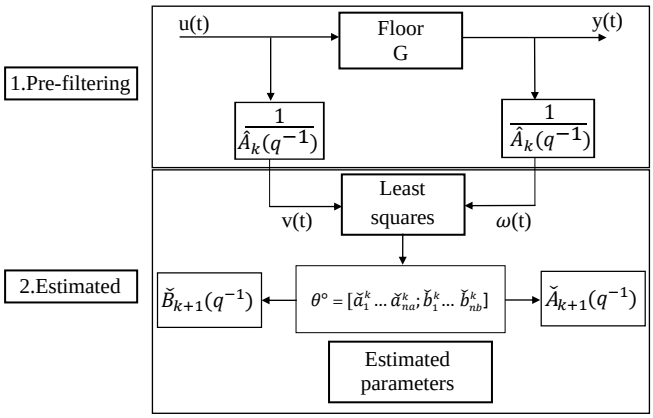

Figure 3. The flowchart of the Steiglitz–McBride algorithm.

To simulate the heuristic Steiglitz–McBride algorithm, the current and angular velocity transfer functions were used to convert the model from discrete to continuous time.

3.1.1. Pre-Filtering

Pre-filtering is implemented on the motor input, i.e., voltage, in addition to two measurable output parameters, where a transfer function defines current and angular velocity.

3.1.2. Estimation

The algorithm uses discrete models. For this reason, they are loaded by samples of signals involved in the estimation. The discrete model must go through the Steiglitz–McBride algorithm to become a continuous model and obtain the estimated motor parameters through calculations. The function changes from a discrete to a continuous model and is simulated in the continuous model, and the error is calculated with the resulting new functions generating a difference between the actual and estimated signals. Each time the coefficients are calculated, the process is repeated, seeking to minimize the error.

3.2. Grey Wolf Optimizer (GWO)

GWO is a metaheuristic algorithm that takes only one inherent parameter to define its search in addition to the general search parameters. It should be noted that this algorithm has already been subjected to research where it has obtained better results than heuristic algorithms. This makes it a suitable algorithm for this research. In other investigations, the authors have managed to obtain good results with the GWO algorithm when adjusting the parameters of nonlinear models for vibration amplitude measurements [32]. Also, there are demonstrations where the estimation of the parameters of a control system can be optimized in real time to improve the performance of robotic manipulators, specifically in terms of tracking precision, robustness to uncertainties, and the smoothness of movement [33].

GWO is based on the social hierarchy that governs decision-making within a pack of gray wolves. The author of [34] mathematically shapes this social hierarchy to consider the best solution as "*alpha*", the second-best solution is called "*beta*", the third-best solution is represented as "*delta*", and the rest of the answers are assumed as "*omegas*". It should be noted that the hierarchy mentioned also influences the main phases of wolf hunting. This is of utmost importance, since the algorithm is responsible for molding or designing a mathematical order to describe each of these hunting phases based on their social hierarchy, since the optimization offered by this algorithm for the solution of different problems is guided by the "*alpha*", "*beta*", and "*delta*" solutions. The main hunting phases of gray wolves are as follows.

3.2.1. Encircling Prey

Mathematically speaking, the cornering of prey tells us that a gray wolf can randomly update its position within the hunting space according to the position of the prey of the best search agents. The author of [34] explains this in the mathematical order, adjusting the vectors of the search agents. This behavior is described mathematically in Equations (8a) and (8b).

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{8a}$$

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{8b}$$

where $\vec{X}$ is the position of the wolf, t denotes the iteration, $\vec{X}_p$ denotes the position of the prey, and $\vec{D}$ is the distance between wolf and prey. Coefficients $\vec{A}$ and $\vec{C}$ are calculated as

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{9a}$$

$$\vec{C} = 2\vec{r}_2 \tag{9b}$$

r_1 and r_2 are two random values of the vector within the range $[0, 1]$.

3.2.2. Hunting

This phase is mathematically simulated in the algorithm by saving the three best solutions obtained ($alpha, beta, delta$). These solutions will guide the other solutions (*omegas*) to update their position relative to them in the next generation, because the final solution is expected to be at a random place within the circle defined by the three best solutions. We can calculate their updated positions by using alpha, beta, and delta, which are $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$, as follows:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|\,,\ \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|\,,\ \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \tag{10a}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha)\,,\ \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta)\,,\ \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta) \tag{10b}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{10c}$$

3.2.3. Attacking Prey

The author of [34] proposes that this behavior can be simulated by decreasing the value $\vec{A}$, a random value in an interval of $[-2a, 2a]$, where a is reduced from 2 to 0 throughout iterations. When the arbitrary values of $\vec{A}$ are at $[1, 1]$, the next position of search agent "a" can be anywhere between its current situation and the position of the dam. With the hunting phases explained so far, the algorithm offers good search agents that converge to attack the dam (optimal solution). However, the algorithm is still prone to stagnation, which brings us to the exploration phase.

3.2.4. Search for Prey (Exploration)

In this stage, the grey wolves diverge from each other to search for prey and converge to attack prey. Some random values are selected to model the divergence to force the search agent to diverge from the prey. The $\vec{C}$ vector contains random values in $[0, 2]$. This is mainly performed to increase the range of exploration and avoid stagnation, perhaps finding a better target. Considering this scheme, the search hyperparameters are selected, and in this case, the prey acts as the vector of engine parameters. The algorithm evaluates each search agent and selects the one with the lowest fitness as the alpha wolf. The algorithm will repeat the process until the selected number of iterations is completed.

The flowchart in Figure 4 is explained as follows. First, the program initializes the population to search for the constants to optimize, within the upper and lower limits. The main program receives the parameters through Simulink in order to calculate the motor current and velocity responses and compare with the actual responses using the fitness

function. The algorithm chooses the three best solutions closest to the actual response which become the alpha, beta, and gamma agents, thus emulating the hunting behavior of wolves. The program is initialized again with a new iteration, now creating the agents within the range of solutions of alpha, beta, and gamma agents of the last iteration; in this way, the same cycle is repeated until reaching the maximum iteration, obtaining the best solution.

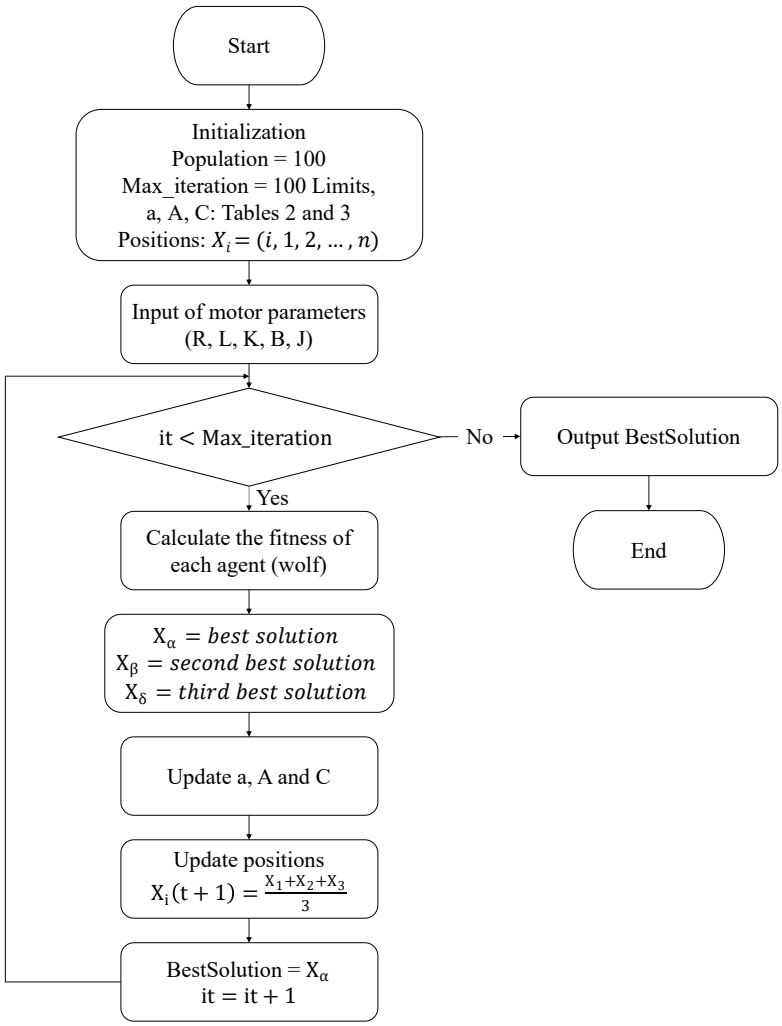

Figure 4. The flowchart of the GWO algorithm.

Table 2. General parameters applied in metaheuristic algorithms.

Parameter	Value	Description
Search agents	100	Number of random solutions proposed of each metaheuristic algorithm.
Upper limits	$R \leq 1$, $L \leq 0.1$, $K \leq 0.1$, $B \leq 0.001$, $J \leq 0.001$	Upper search limit for each DC motor parameter
Lower limits	$R \geq 0.1$, $L \geq 0.001$, $K \geq 0.005$, $B \geq 0.00001$, $J \geq 0.00001$	Lower search limit for each DC motor parameter
Iterations	100	Repetitions of each metaheuristic algorithm.
Fitness function	$Fitness = \frac{1}{\sqrt{\sum(I-I_s)^2+\sum(\omega-\omega_s)^2}}$	Function used to evaluate the performance of each random solution ($f : i$)

Table 3. Specific parameters applied in metaheuristic algorithms.

Algorithm	Specific Parameters	Description
Genetic Algorithm	Mutation: 20%	An operator that allows the random alteration of a gene to maintain search diversity.
	Selection: Roulette wheel ($s(i) = \frac{f_i}{\sum_{j=0}^{N} f_j}$)	Population with N individuals, for each chromosome i with corresponding fitness value f_i, probability $s(i)$ of selection
	Type: Crossover	A random point is selected to combine the chromosomes.
	Biological pressure: 30%	The percentage of individuals that reproduce
Grey Wolf Optimizer	a, A and C	Intrinsic parameters calculated by original method proposed (see original method in [34]).
Jaya	-	Jaya doesnt contain any specific parameter.

This algorithm has been used in multiple optimization works to estimate the parameters of DC motors. It offers good optimization in robust analyses, as in [35], where it is sought to optimize the fractional-order PID controller in motor speed control.

3.3. Jaya Algorithm

This algorithm is used in optimization and is one of the simplest algorithms to implement. It does not contain any intrinsic or extrinsic search parameters. It only uses general search parameters, which makes it very attractive compared to other metaheuristic algorithms that occupy specific search parameters. The Jaya algorithm is a metaheuristic algorithm proposed by Rao [36] and can be defined as a modification of PSO in which the attraction for the best local functions is removed. Jaya replaces this by moving the worst local solutions away from the next iteration, thus producing optimized solutions for the problem of interest. Compared to other algorithms, Jaya only occupies three input parameters: the size of the population, the number of generations (iterations), and the range of limits. The Jaya optimization process focuses on determining the most optimized solution, leaving aside the lower solutions by disregarding generations that do not offer a better solution than the current generation. When it comes to finding a better solution, the algorithm eliminates the current generation, as it does not keep track of the best solutions. The procedure of the Jaya algorithm is discussed in the following steps [37]. *Step 1*: Initialize population size N and iteration number T. The constrained problem is as follows:

$$\begin{gathered} \min f(\mathbf{x}) \\ S.t \\ g_j(\mathbf{x}) = c_j \quad \forall j = (1, 2, \ldots, n) \end{gathered}$$

where $f(\mathbf{x})$ is the objective function used to calculate fitness value of the solution $x = (x_1, x_2, \ldots, x_D)$, where x_i is a decision variable assigned by a value in the lower and upper limits such that $x_i \in [X_i^{min}, X_i^{max}]$. g_j is the j^{th} equality constraint, and h_k is the k inequality constraint. *Step 2*: Constructing the initial population for Jaya. Note that JM is an augmented matrix of size $N \times D$ corresponding to the Jaya Memory and shown in Equation (11). Conventionally, a solution is randomly constructed: $JM_{i,j} = X_j^{min} + (X_j^{min} - X_j^{max}) \times rnd, \forall i \in (1, 2, \cdots, N) \bigwedge \forall j \in (1, 2, \cdots, N)$. rnd is a uniform function that generates a random value between 0 and 1.

$$\mathbf{JM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_D^1 \\ x_1^2 & x_2^2 & \cdots & x_D^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^N & x_2^N & \cdots & x_D^N \end{bmatrix} \begin{bmatrix} f(\mathbf{x}^1) \\ f(\mathbf{x}^2) \\ \vdots \\ f(\mathbf{x}^N) \end{bmatrix} \tag{11}$$

The objective function $f(\mathbf{x}^i)$ for each solution is also calculated, and the JM solutions are sorted in ascending order. The best solution is $\mathbf{x}^1$ and the worst solution is $\mathbf{x}^N$. *Step 3*:

Jaya evolution process. The decision variables of all solutions in the JM undergo changes using the Jaya operator formulated in Equation (12). Note that $\mathbf{x}_j^{'i}$ is the new updated solution, x_j^i is the current solution, x_j^1 is the decision variable j in the best solution, and x_j^N is the decision variable j in the worst solution. r_1 and r_2 are two independent random numbers generated from a uniform distribution $U(0,1)$. *Step 4*: The JM solutions at every iteration will be updated. The objective function value of the new $f(\mathbf{x}^{'i})$ is calculated. The current solution $\mathbf{x}^i$ will be replaced by the new solution $\mathbf{x}^{'i}$ if $f(\mathbf{x}^{'i}) \leq f(\mathbf{x}^i)$. This process will be repeated as many as N times. *Step 5*: Stop rule. The algorithm repeats Step 3 and Step 4 until the maximum number of iterations T is reached.

$$x_j^{'i} = x_j^i + r_1 \times (x_j^1 - |x_j^i|) - r_2 \times (x_j^N - |x_j^i|) \tag{12}$$

Figure 5 shows the flowchart for Jaya algorithm.

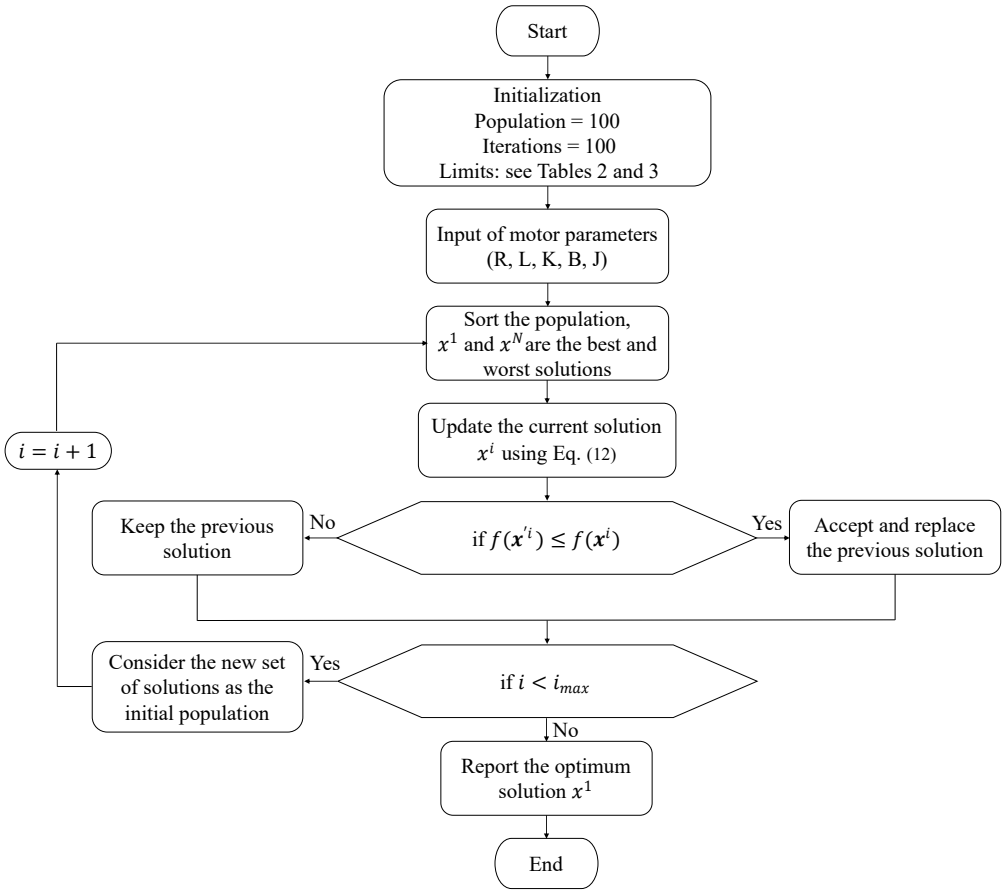

Figure 5. The flowchart of the Jaya algorithm.

3.4. Genetic Algorithm (GA)

GA is the oldest and most widely used metaheuristic algorithm, and it has many antecedents in the field of DC motors. This background extends to the following applications of the Genetic Algorithm: in PID controllers, to tune the parameters of DC motors when they have a time-dependent nature [38]; in the optimization of slotless BLDC motors and the effect on the distribution of magnetic flux and temperature [39,40]; to identify the dynamic state of a DC motor using the least squares error as a metric [41]; in a geared DC motor to improve the actual angular trajectory [23]; and to design a PID controller for a DC shunt motor considering a third-order model [42]. Also, the usefulness of GA for time scheduling and optimization in industrial robotized tasks has been demonstrated, improving efficiency and production time [43].

The main drawback of the algorithm is that it has several specific search parameters. The GA helps us to optimize the search for problem solutions based on natural selection mechanisms and biological and genetic evolution as part of evolutionary computation. The algorithm works through generations, where it first initializes them with random solutions. These are called populations, where each individual or chromosome will be evaluated by the selection function (fitness), which is mathematically molded depending on the problem in question. This process happens in each iteration (generation). Thanks to the selection mechanism, the next generation can be created from the selected chromosomes of the previous generation with the help of the algorithm function called crossing, which modifies the chromosomes through the mutation operator. Thereby, the algorithm finds an optimal solution to the problem in question. Figure 6 shows the operation of the algorithm.

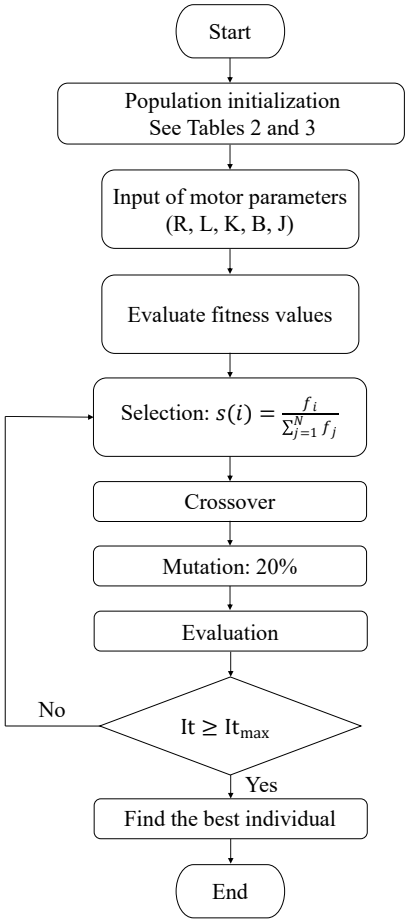

Figure 6. The flowchart of the Genetic Algorithm.

3.5. Implementation of Metaheuristic Algorithms

The parameters used for the development of the algorithms are described here. Much of the optimization is based on a good selection of search parameters that govern the search for acceptable solutions to the problem in question.

The same "fitness function" was applied to each metaheuristic algorithm for a fair comparison. This equation was based on the inverse of the Euclidean distance, as shown in Equation (13), where I_s is the estimated current, I is the actual current, ω_s is the estimated angular velocity, and ω is the actual angular velocity. It should be noted that this function is applied to each iteration of the metaheuristic algorithms.

$$Fitness = \frac{1}{\sqrt{\sum(I - I_s)^2 + \sum(\omega - \omega_s)^2}} \tag{13}$$

The MSE was used as an error evaluation criterion; see Equation (14). This function measures the average of the differences between the real and predicted values and allows us to penalize the largest differences to obtain a more robust model where, if there are large deviations, we can obtain a good approximation of the results. Since the DC motor is a system of differential equations with two variables, both must be included in the error estimate. That is why the fitness function must consider the MSE in current and speed. Since they are two different variables, the Euclidean distance is used to associate them. This statistical method will be used to analyze performance in the parametric estimations of the DC motor. The result will be a percentage of error in each vector of the five parameters [R, L, K, B, J]. It is important to highlight that the error function selected can affect the behavior of the metaheuristic algorithm. In addition to the MSE error, some other error functions have shown similar performance [44,45]; for example, the Integral Squared Error (ISE), the Integral Absolute Error (IAE), the Integral of Time multiplied by the Absolute Error (ITAE), and the Integral of Time multiplied by the Square Error (ITSE). Studying the effects of these error functions is beyond the scope of this manuscript.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{14}$$

This method results in an error calculated by adding the actual values minus the estimated values squared and divided by the number of parameters.

The optimization offered by metaheuristic algorithms depends on the iterations, since the algorithm produces a more optimal solution in each iteration. However, if many iterations are used, the algorithm will require a higher computational cost. For this work, the number of iterations is a fixed parameter aiming for a good performance in both computational cost and parametric estimation, obtaining a good balance between effectiveness and efficiency. The iterations used were 100 for each program.

The range of limits (upper and lower) determines the search space for each motor parameter. The limits adjust each range proportionally to the magnitude of the parameter. Varying these limits increases or reduces the search space. Although an infinite number of values can be selected, it is not essential, as long as the nominal value is within the search range. Thanks to this, the algorithm has a defined search space, where it begins to evaluate random values in search of the most appropriate solution. Mutation and creating a new population are performed within the limits of the GA. In the case of GWO and Jaya, new search agents within the range are checked in each iteration, guaranteeing that the search for all algorithms is performed within the limits. The ranges selected for this work must be positive since, by definition, any parameter takes negative values. On the other hand, the final selection of values is based on the typical values expected for this type of motor. For example, investigations such as refs. [30,46] have successfully searched for values in this range in similar DC motors.

Considering the above, each algorithm uses common general parameters, such as limits and iterations; see Table 2. However, they also use parameters that are specific to each method. The summary with the values of all the parameters used in each metaheuristic algorithm is shown in Table 3.

Simulations were run for 3 s for a constant voltage of 10.5 V. Likewise, a numerical method with a fixed step of 0.001 s was used to adapt to the data acquisition system. In executing the three metaheuristic algorithms, MATLAB was used to model the dynamic system of the DC motor through Simulink, and to obtain the responses of current and angular velocity, Equations (6) and (7) were used, respectively. MATLAB is responsible for executing the heuristic and metaheuristic algorithms; however, to evaluate performance, the Simulink scheme shown in Figure 7 is used, which evaluates the proposed solutions and subsequently returns the current and voltage signals obtained to MATLAB.

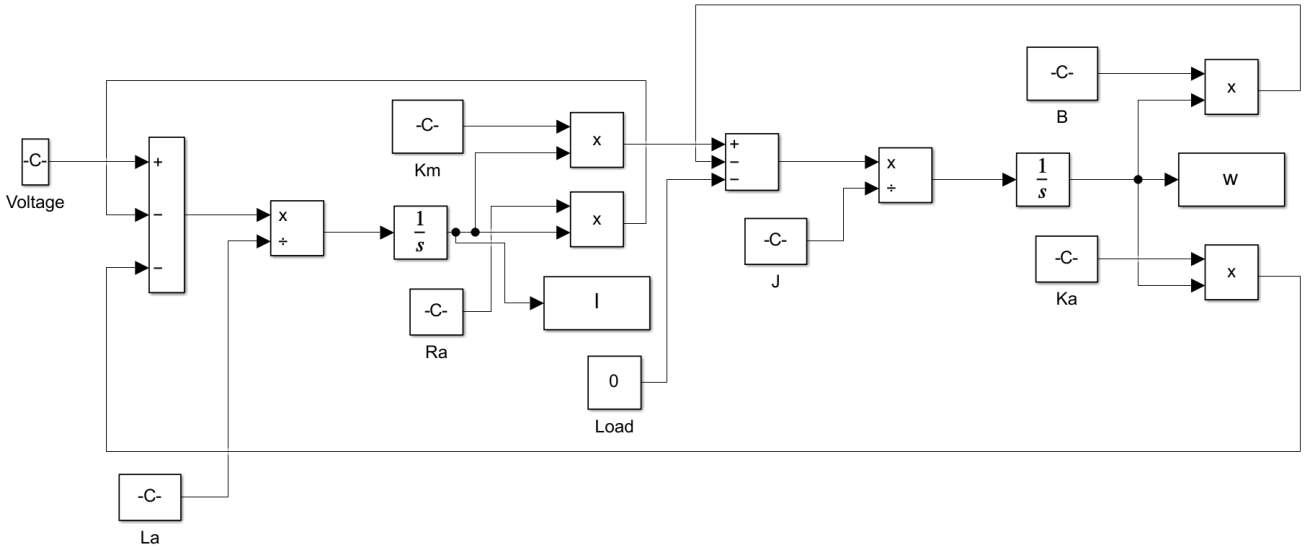

Figure 7. Fitness evolution for Genetic Algorithm.

In the same way, the input parameters of the M2-Robokits motor were the same as in the heuristic algorithm. The same period, voltage, and step mentioned above were used for each simulation.

4. Results

The minimum fitness calculated by each metaheuristic algorithm may vary because it is based on random values. Therefore, cross-validation is required. For this, each metaheuristic algorithm was executed ten times. The best, worst, and average are shown in Table 4. Furthermore, the performance graphs are shown for the GWO algorithm in Figure 8, for the Jaya algorithm in Figure 9, and for the GA in Figure 10.

Table 4. The cross validation for metaheuristic algorithms.

Algorithm	Fitness Value in Fifty Runs		
	Best	**Worse**	**Average**
GWO	0.029	0.062	0.042
Jaya	0.35	1.92	0.99
GA	0.044	1.06	0.058

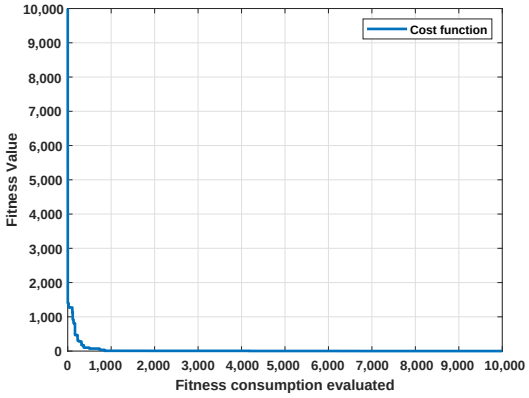

Figure 8. Fitness evolution for Gray Wolf Optimizer.

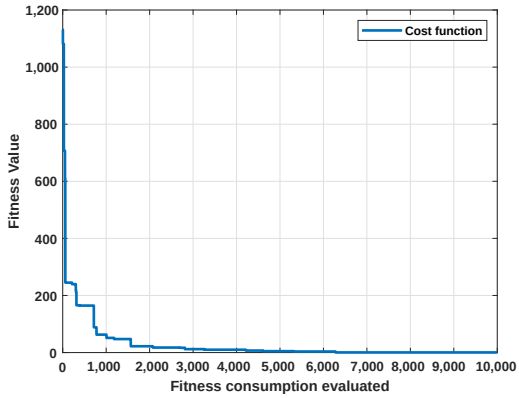

Figure 9. Fitness evolution for Jaya.

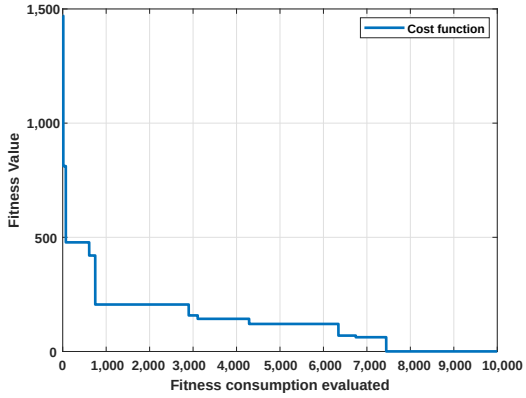

Figure 10. Fitness evolution for Genetic Algorithm.

The parameters calculated for all algorithms are shown in Tables 5 and 6. Two real signals from the M2 Motor were used for this. The current signal was hardware-filtered, and the velocity signal was filtered using a Chebyshev software filter. Both signals are considered real, although there may be variations due to acquisition and filtering.

- Figure 11 shows the comparison between the real current of the motor and the one estimated by the Steiglitz–McBride algorithm. In the same way, Figure 12 shows the comparison of the real velocity of the motor against that estimated by the Steiglitz–McBride algorithm.
- Figure 13 shows the comparison between the real current of the motor and the one estimated by the GWO algorithm. In the same way, Figure 14 shows the comparison of the real velocity of the motor against that estimated by the GWO algorithm.
- Figure 15 shows the comparison between the real current of the motor and the one estimated by the Jaya algorithm. In the same way, Figure 16 shows the comparison of the real velocity of the motor against that estimated by the Jaya algorithm.
- Figure 17 shows the comparison between the real current of the motor and the one estimated by the GA algorithm. In the same way, Figure 18 shows the comparison of the real velocity of the motor with that estimated by the GA algorithm.

Table 5. Comparison of Steiglitz–McBride and GWO algorithms against nominal values.

		Steiglitz–McBride		GWO	
Parameter	**Nominal Value**	**Value**	**MSE**	**Value**	**MSE**
$R\ (\Omega)$	0.921042	0.914735	00.68%	0.923696	0.42%
L (H)	0.007759	0.008893	14.49%	0.007752	0.96%
K	0.073472	0.073547	00.10%	0.073466	0.08%
$B\left(\frac{\text{kg}\cdot\text{m}^3}{\text{s}^2}\right)$	0.000678	0.000680	00.28%	0.000677	0.16%
$J\ (\text{kg}\cdot\text{m}^2)$	0.000136	0.000135	01.03%	0.000136	0.55%

Table 6. Comparison of Jaya and GA algorithms against nominal values.

		Jaya		GA	
Parameter	**Nominal Value**	**Value**	**MSE**	**Value**	**MSE**
$R\ (\Omega)$	0.921042	0.996974	08.24%	0.817929	13.44%
L (H)	0.007759	0.008941	15.04%	0.011908	89.00%
K	0.073472	0.000676	01.28%	0.076071	05.75%
$B\left(\frac{\text{kg}\cdot\text{m}^3}{\text{s}^2}\right)$	0.000678	0.291812	01.61%	0.000661	17.07%
$J\ (\text{kg}\cdot\text{m}^2)$	0.000136	0.000128	08.75%	0.000135	17.67%

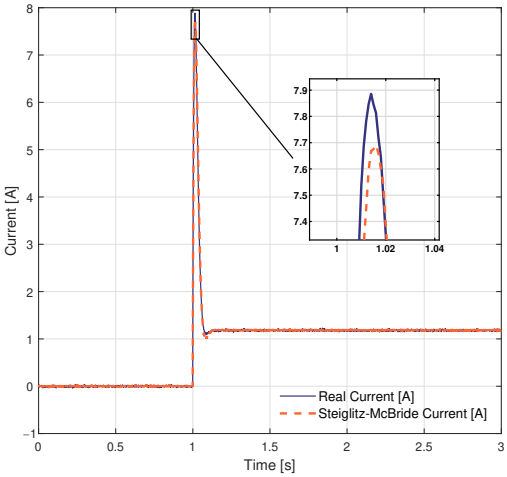

Figure 11. Real current signal vs. Steiglitz–McBride signal.

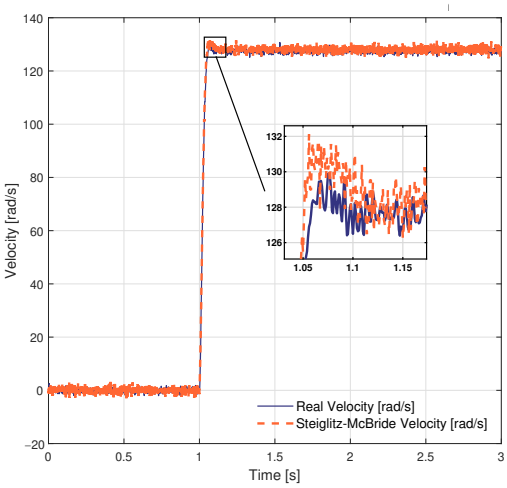

Figure 12. Real velocity signal vs. Steiglitz–McBride signal.

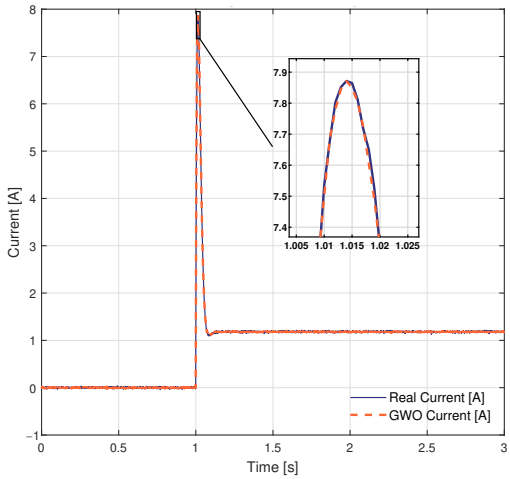

Figure 13. Real current signal vs. GWO signal.

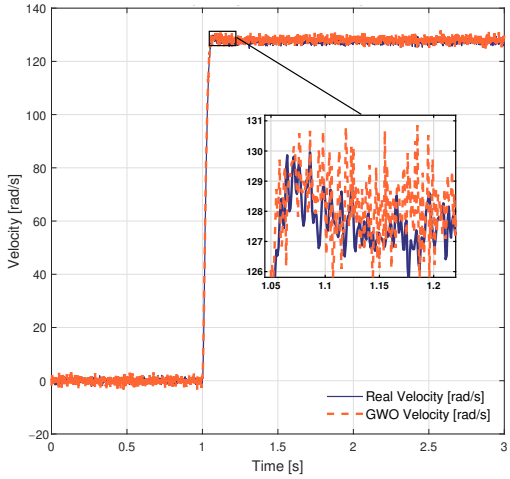

Figure 14. Real velocity signal vs. GWO signal.

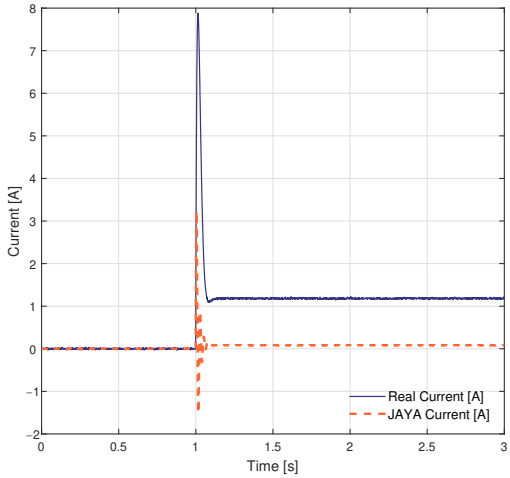

Figure 15. Real current signal vs. Jaya signal.

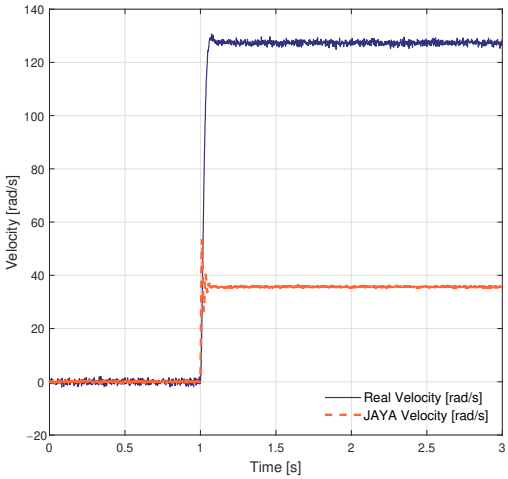

Figure 16. Real velocity signal vs. Jaya signal.

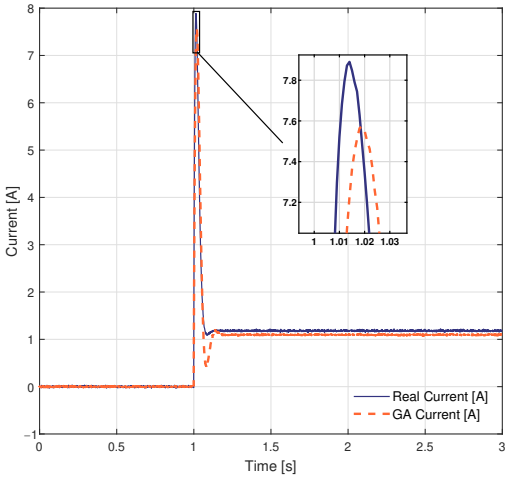

Figure 17. Real current signal vs. GA signal.

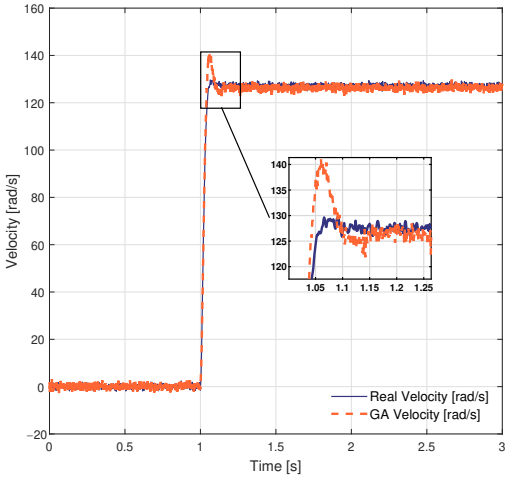

Figure 18. Real velocity signal vs. GA signal.

The computation times for each algorithm to run the 100 iterations are outlined in Table 7. The parameters and errors estimated by the Steiglitz–McBride and GWO algorithms are shown in Table 5 and compared against the nominal values. The parameters and errors estimated with the Jaya and GA algorithms are displayed in Table 6 and compared against the nominal values. As another metric to compare the algorithms, Table 8 includes the standard deviations that resulted from the estimates of each algorithm for each parameter analyzed.

Table 7. Computation time.

Algorithms	Average Time (min)
Steiglitz–McBride	0.102
GWO	49.53
Jaya	51.16
GA	60.23

Table 8. Standard deviation of estimates from each algorithm.

	Algorithms			
Parameters	**Steiglitz–McBride**	**GWO**	**Jaya**	**GA**
R	0.0023	0.0040	0.0086	0.1340
L	6.49×10^{-5}	9.22×10^{-5}	8.69×10^{-4}	0.0114
K	4.53×10^{-5}	7.24×10^{-5}	0.6940	0.0044
B	1.55×10^{-4}	1.15×10^{-6}	1.08×10^{-5}	9.94×10^{-7}
J	3.57×10^{-5}	6.99×10^{-7}	1.20×10^{-5}	5.27×10^{-7}

These results were obtained with real current and velocity signals. Better results are expected from all algorithms with computer-generated signals. However, it is necessary to address the noise inherent in real signals for practical application. The motor model parameterization process is usually performed offline, since the parameters remain constant. However, the parameterization of a motor can allow the use of control schemes that consider the dynamics of the system and not only the error, as in the case of the PID, which can allow for more precise control.

5. Discussion

Table 7 shows the computation performance of each algorithm. In this way, the computation times are compared among themselves to see whether they influence each algorithm's performance, since in works such as ref. [47], it has been reported that the increase in iterations negatively affects the optimization of the solutions. This is contrary to what occurs in our research, where the increased iterations and, consequently, high computational cost improved the parametric estimates. Even though parameterization is an offline process, the response times of the algorithms are too high for the response time expected in a control system. Thus, this type of algorithm is usually avoided in processes that require a high response speed.

As can be seen in Table 7, the Steiglitz–McBride heuristic algorithm has a lower computational cost compared to the metaheuristic algorithms. This is because the algorithm itself works through pure mathematics and is in charge of finding a unique solution. However, among the metaheuristic algorithms, a similar computational cost was observed, with a time variation of 10.7 min.

GWO has been used to estimate the parameters of a DC motor by applying an Integral Squared Error (ISE) object function [3]. It is reported that a smaller number of iterations could be detrimental, as it causes stagnation and local optima problems. Because of this, the authors decided to implement 500 iterations per test and few search agents compared to iterations. The use of 500 iterations in our research significantly increased the computation

time. GWO has also been used as a parameter tuner for a PID controller [48], where 50 iterations and 30 search agents were applied. Taking into account the above, the configuration we decided to use in our research consisted of a smaller number of iterations and a larger number of search agents to reduce the computational cost and achieve a low error compared to the literature. This represented an estimation with an average error rate of 1.73%, which was 55% lower than that shown in [3].

The algorithm with the highest average MSE percentage was the GA algorithm, which was 28.59%. Among all the algorithms, the algorithm that showed the lowest average percentage error was the GWO, with 0.43%. However, the average MSE percentage for Steiglitz–McBride, which is not a metaheuristic algorithm, was lower than Jaya and GA; see Tables 5 and 6. Although GWO generated the best parametric estimation in this research, the Steiglitz–McBride algorithm could work as another alternative when the application requires low computational cost and a low standard deviation in the estimation of each parameter; see Table 8. For example, in [49], it has been shown that the Steiglitz–McBride algorithm is useful in the parametric estimation of electrical machines such as DC motors, brush DC, and brushless AC and gear machines. Also, it is frequently used as an optimizer.

The current signals estimated by GWO and Steiglitz–McBride were very similar to the nominal ones. For this reason, a comparison of the real and estimated signals obtained by the algorithms was performed with the following indicators: settling time, overshoot, and steady-state error.

The stabilization times of the current signals were as follows: 1.107 s in the real signal, 1.118 s in Steiglitz–McBride, 1.103 s in GWO, and 1.167 s in GA. However, the current signal estimated by the Jaya algorithm failed to reach the stabilization value; see Figure 15. The current signal obtained by GWO is the closest estimate to the real motor signal; see Figure 13.

The average current values in the steady-state zone of each algorithm were as follows: 1.1821 A for the real signal, 1.1840 A for Steiglitz–McBride, 0.0827 A for Jaya, 1.1805 A for GWO, and 1.0968 A for GA. This means that GWO and Steiglitz–McBride were the algorithms that estimated the closest current response to the real one.

The comparison of the overshoot values in the current signals were as follows: 7.8717 A for the real signal, occurring in 1.014 s; 7.6930 A for the Steiglitz–McBride algorithm, occurring in 1.016 s; 3.2511 A for Jaya, occurring in 1.004 s; 7.8738 A for GWO, occurring in 1.014 s; and, finally, 7.5736 A for GA, occurring in 1.018 s. The GWO algorithm proved to be the most accurate when discussing the maximum peaks, obtaining the same result as the real signal for time. For current, a 0.22% difference was displayed.

In [3], the authors reported that a percentage error comparison cannot be reliable as a measure of the effectiveness of these algorithms. Since the algorithms do not show consistent solutions because they start from random solutions, performing the comparison by means of the standard deviation would be more suitable, which is a more stable metric for random data. Table 8 shows the difference in the standard deviation between the estimated parameters of each algorithm and the nominal values. The algorithms with the most significant deviation were Jaya and GA, with Jaya being the algorithm with the greatest variation in the estimates. On the other hand, Steiglitz–McBride and GWO had less variation in the estimates, where Steiglitz–McBride was the algorithm with the lowest deviation of 37.34% with respect to GWO. This may be due to Steiglitz–McBride having consistent estimated values, because it starts from a mathematical model. In contrast, metaheuristic algorithms start from random solutions which are adjusted to find the most suitable solution during iterations.

6. Conclusions

This work performed a parametric estimation of a DC motor using a heuristic algorithm (Steiglitz–McBride) and three metaheuristic algorithms (Jaya, GWO, and GA). The MSE and standard deviation were used as statistical indicators to evaluate the performance of each algorithm as well as other values of the dynamic response, such as settling time,

overshoot, and steady-state error. In this way, this work determined which algorithm provides a parametric estimation closer to the nominal parameters and real DC motor signals. The comparison between metaheuristic and heuristic algorithms is limited to parametric estimation in direct current motors. As the No Free Lunch theorem states, no algorithm is best for all applications. Another limitation is the selection of hyperparameters, because there is no way to guarantee the optimal selection of these parameters in metaheuristic algorithms. There are algorithms for selecting hyperparameters, but they do not guarantee optimal selection. Therefore, the maximum performance of a metaheuristic algorithm may not be found.

According to the aforementioned aspects, the two best-performing algorithms were Steiglitz–McBride and GWO. GWO was the best parametric estimator for this application, having the lowest MSE of all algorithms for the nominal parameters. Likewise, this algorithm had the closest responses to the real ones, with the best overshoot and settling time approximation. However, Steiglitz–McBride also obtained good results as a parametric estimator, even better than Jaya and GA, but with the lowest computational cost of all and a less variable parametric estimation.

Although GWO is a better parametric estimator, it does not mean that it is better than the Steiglitz–McBride algorithm, since implementing the mathematical model in a metaheuristic algorithm can be simpler. However, the search parameters required by metaheuristic algorithms are obtained after many tests, which generate a high computational cost. Although the Steiglitz–McBride is mathematically more complex, if appropriately implemented, it may require fewer tests and fewer corrections within the algorithm, resulting in a lower computational cost.

In this work, the heuristic algorithm produced fewer problems during tests, with lower computational cost and low variation in estimates. However, if a more precise parametric estimation is required, it is recommended to use GWO. Considering these aspects, the heuristic algorithm has more advantages in the parametric estimation of DC motors. The performance of the algorithms presented in this paper may change if different error functions are used, as well as other search parameters. In addition, performance depends on the final application of the algorithm.

Author Contributions: Conceptualization, L.A.M.-S., E.A.D.-R., M.C.-F. and J.R.-R.; methodology, M.C.-F., D.M.M. and E.A.D.-R.; writing—original draft preparation, L.A.M.-S., M.C.-F. and J.R.-R.; writing—review and editing, D.M.M., E.A.D.-R. and L.A.M.-S.; supervision, L.A.M.-S. and M.C.-F. All authors have read and agreed to the published version of the manuscript.

Funding: No funding was received to conduct this study.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: The authors thank the Polytechnic University of Queretaro (UPQ) for the support.

Conflicts of Interest: The authors declare that they have no conflicts of interest. The datasets generated during and/or analyzed during the current study are available from the corresponding authors on reasonable request.

References

1. Fazdi, M.F.; Hsueh, P.-W. Parameters Identification of a Permanent Magnet DC Motor: A Review. *Electronics* **2023**, *12*, 2559. [CrossRef]
2. Beltran-Carbajal, F.; Tapia-Olvera, R.; Aguilar-Mejia, O.; Favela-Contreras, A.; Lopez-Garcia, I. An online algebraic estimation approach of parameters and variable mechanical torque in shunt DC motors. *Int. Trans. Electr. Energy Syst.* **2018**, *28*, e2474. [CrossRef]
3. Karnavas, Y.L. Application of recent nature-inspired meta-heuristic optimization techniques to small permanent magnet DC motor parameters identification problem. *J. Eng.* **2020**, *2020*, 877–888.

4. Tran, C.D.; Kuchar, M.; Sobek, M.; Sotola, V.; Dinh, B.H. Sensor Fault Diagnosis Method Based on Rotor Slip Applied to Induction Motor Drive. *Sensors* **2022**, *22*, 8636. [CrossRef] [PubMed]
5. Chakraborty, C.; Verma, V. Speed and Current Sensor Fault Detection and Isolation Technique for Induction Motor Drive Using Axes Transformation. *IEEE Trans. Ind. Electron.* **2015**, *62*, 1943–1954. [CrossRef]
6. Merrassi, W.E.; Abounada, A.; Ramzi, M. Advanced speed sensorless control strategy for induction machine based on neuro-MRAS observer. *Mater. Today Proc.* **2021**, *45*, 7615–7621. ISSN 2214-7853. [CrossRef]
7. Nag, T.; Santra, S.B.; Chatterjee, A.; Chatterjee, D.; Ganguli, A.K. Fuzzy logic-based loss minimisation scheme for brushless DC motor drive system. *IET Power Electron.* **2016**, *9*, 1581–1589. [CrossRef]
8. Vikhe, P.S.; Punjabi, N.; Kadu, C.B. DC motor speed control using PID controller in lab view. *Int. J. Innov. Sci. Mod. Eng. (IJISME)* **2015**, 38–41.
9. Song, T.J.; Oh, K.S. Adaptive Velocity Control Algorithm for DC Motors based on Parameter Estimation of Error Dynamics Under Uncertainty and Load Variation. *Trans. Korean Soc. Mech. Eng. A* **2020**, *44*, 83–91. [CrossRef]
10. Majdoubi, R.; Masmoudi, L.; Bakhti, M.; Elharif, A.; Jabri, B. Parameters estimation of BLDC motor based on physical approach and weighted recursive least square algorithm. *Int. J. Electr. Comput. Eng. (IJECE)* **2021**, *11*, 133–145. ISSN 2088-8708. [CrossRef]
11. Knypiński, L.; Reddy, A.V.; Venkateswararao, B.; Devarapalli, R. Optimal design of brushless DC motor for electromobility propulsion applications using Taguchi method. *J. Electr. Eng.* **2023**, *74*, 116–121. [CrossRef]
12. Urra, E.; Cubillos, C.; Cabrera-Paniagua, D.; Mellado, R. hMod: A software framework for assembling highly detailed heuristics algorithms. *Softw.-Pract. Exp.* **2019**, *49*, 971–994. [CrossRef]
13. Zandavi, S.M.; Chung, V.Y.Y.; Anaissi, A. Stochastic dual simplex algorithm: A novel heuristic optimization algorithm. *IEEE Trans. Cybern.* **2021**, *51*, 2725–2734. [CrossRef]
14. Li, Y.; Cherednichenko, A.; Jiang, Z.; Shi, W.; Wu, J. A Novel Generalized Group-Sparse Mixture Adaptive Filtering Algorithm. *Symmetry* **2019**, *11*, 697. [CrossRef]
15. Neshat, M.; Sepidnam, G.; Sargolzaei, M.; Toosi, A.N. Artificial fish swarm algorithm: A survey 590 of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artif. Intell. Rev.* **2014**, *42*, 965–997. [CrossRef]
16. Yang, S.; Jiang, J.; Yan, G. A Dolphin Partner Optimization. In Proceedings of the 2009 WRI Global Congress on Intelligent Systems, Xiamen, China, 19–21 May 2009; Zhou, S.M., Wang, W., Eds.; IEEE Comp Soc; World Res Inst: Xiamen, China, 2009; Volume I, pp. 124–128.
17. Shuang, B.; Chen, J.; Li, Z. Study Hybrid PS-ACO Algorithm. *Appl. Intell.* **2011**, *34*, 64–73. [CrossRef]
18. Wu, E.; Huang, Y.; Li, D. An Adaptive Particle Swarm Optimization Algorithm for Reactive Power Optimization in Power System. In Proceedings of the 2010 8th World Congress on Intelligent Control and Automation (WCICA), Jinan, China, 7–9 July 2010; IEEE: Jinan, China, 2010; pp. 3132–3137.
19. Shi, J.; Mi, Q.; Cao, W.; Zhou, L. Optimizing BLDC motor drive performance using particle swarm algorithm-tuned fuzzy logic controller. *SN Appl. Sci.* **2022**, *4*, 293. [CrossRef]
20. Rodríguez-Abreo, O.; Rodríguez-Reséndiz, J.; García-Cerezo, A.; García-Martínez, J.R. Fuzzy logic controller for UAV with gains optimized via genetic algorithm. *Heliyon* **2014**, *10*, e26363. ISSN 2405-8440. [CrossRef]
21. Serradilla, F.; Canas, N.; Naranjo, J.E. Optimization of the energy consumption of electric motors through metaheuristics and PID controllers. *Electronics* **2020**, *9*, 1842. [CrossRef]
22. Wu, Z.; Du, C. Parameter Identification PMSM Based Improved Cuckoo Algorithm. *Neural Process. Lett.* **2019**, *50*, 2701–2715. [CrossRef]
23. Amiri, M.S.; Ibrahim, M.F.; Ramli, R. Optimal parameter estimation for a DC motor using genetic algorithm. *Int. J. Power Electron. Drive Syst. (IJPEDS)* **2020**, *11*, 1047–1054. [CrossRef]
24. Al-Azzawi, D.S. Evaluation of Genetic Algorithm Optimization in Machine Learning. *J. Inf. Sci. Eng.* **2020**, *36*, 231–241.
25. Cheng, Y.; Lyu, X.; Mao, S. Optimization design of brushless DC motor based on improved Jaya algorithm. *Sci. Rep.* **2024**, *14*, 5427. [CrossRef] [PubMed]
26. Yan, C.; Li, M.-X.; Liu, W. Application of Improved Genetic Algorithm in Function Optimization. *J. Inf. Sci. Eng.* **2019**, *35*, 1299–1309.
27. Ahamed, S.R.; Parumasivam, P.; Lipu, M.S.H.; Hannan, M.A.; Ker, P.J. A comparative evaluation of PID-based optimisation controller algorithms for DC motor. *Int. J. Autom. Control* **2020**, *14*, 634–655. [CrossRef]
28. Achanta, R.K.; Pamula, V.K. DC Motor Speed Control using PID Controller Tuned by Jaya Optimization Algorithm. In Proceedings of the 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, 21–22 September 2017; IEEE: Thandalam, India, 2017; pp. 983–987.
29. Niu, X.; Feng, G.; Jia, S.; Zhang, Y. Control of brushless DC motor based on fuzzy rules optimized by genetic algorithm used in hybrid vehicle. *J. Comput. Methods Sci. Eng.* **2021**, *21*, 951–968. [CrossRef]
30. Rodriguez-Abreo, O.; Hernandez-Paredes, J.M.; Rangel, A.F.; Fuentes-Silva, C.; Velasquez, F.A.C. Parameter Identification of Motors by Cuckoo Search Using Steady-State Relations. *IEEE Access* **2021**, *9*, 72017–72024. [CrossRef]
31. Pillay, P.; Krishnan, R. Modeling, simulation, and analysis of permanent-magnet motor drives. II. The brushless DC motor drive. *IEEE Trans. Ind. Appl.* **1989**, *25*, 274–279. [CrossRef]
32. Rodríguez-Abreo, O.; Rodríguez-Reséndiz, J.; Montoya-Santiyanes, L.A.; Álvarez-Alvarado, J.M. Non-Linear Regression Models with Vibration Amplitude Optimization Algorithms in a Microturbine. *Sensors* **2022**, *22*, 130. [CrossRef]

33. Rezoug, A.; Iqbal, J.; Tadjine, M. Extended grey wolf optimization–based adaptive fast nonsingular terminal sliding mode control of a robotic manipulator. *Proc. Inst. Mech. Eng. Part J. Syst. Control Eng.* **2022**, *236*, 1738–1754. [CrossRef]
34. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
35. Agarwal, J.; Parmar, G.; Gupta, R.; Sikander, A. Analysis of grey wolf optimizer based fractional order PID controller in speed control of DC motor. *Microsyst.-Technol.-Micro-Nanosyst.-Inf. Storage Process. Syst.* **2018**, *24*, 4997–5006. [CrossRef]
36. Rao, R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34. [CrossRef]
37. Zitar, R.A.; Al-Betar, M.A.; Awadallah, M.A.; Doush, I.A.; Assaleh, K. An Intensive and Comprehensive Overview of JAYA Algorithm, its Versions and Applications. *Arch. Comput. Methods Eng.* **2022**, *29*, 763–792. [CrossRef] [PubMed]
38. Godem Ali, M.I.; Karol, K.; Viliam, F. CAD of Cascade Controllers for DC Drives Using Genetic Algorithm Methods. *Procedia Eng.* **2014**, *96*, 182–189. [CrossRef]
39. Kamal, C.; Thyagarajan, T.; Kalpana, D.; Pragadheeshwaran, T. Multiobjective design optimization and analysis of magnetic flux distribution for slotless permanent magnet brushless DC motor using evolutionary algorithms. *J. Magn. Magn. Mater.* **2019**, *476*, 524–537. [CrossRef]
40. Rahideh, A.; Korakianitis, T.; Ruiz, P.; Keeble, T.; Rothman, M.T. Optimal brushless DC motor design using genetic algorithms. *J. Magn. Magn. Mater.* **2010**, *322*, 3680–3687. [CrossRef]
41. Lankarany, M.; Rezazade, A. Parameter Estimation Optimization Based on Genetic Algorithm Applied to DC Motor. In Proceedings of the 2007 International Conference on Electrical Engineering, Lahore, Pakistan, 11–12 April 2007; pp. 1–6. [CrossRef]
42. Thomas, N.; Poongodi, D.P. Position Control of DC Motor Using Genetic Algorithm Based PID Controller. In Proceedings of the World Congress on Engineering, London, UK, 1–3 July 2009; Volume 2.
43. Baizid, K.; Yousnadj, A.; Meddahi, A.; Chellali, R.; Iqbal, J. Time scheduling and optimization of industrial robotized tasks based on genetic algorithms. *Robot.-Comput.-Integr. Manuf.* **2015**, *34*, 0736–5845. [CrossRef]
44. Soni, Y.K.; Bhatt, R. BF-PSO optimized PID controller design using ISE, IAE, IATE and MSE error criteria. *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)* **2013**, *2*, 2333–2336.
45. Mousakazemi, H.; Mohammad, S. Comparison of the error-integral performance indexes in a GA-tuned PID controlling system of a PWR-type nuclear reactor point-kinetics model. *Prog. Nucl. Energy* **2021**, *132*, 103604. [CrossRef]
46. Rodríguez-Abreo, O.; Rodríguez-Reséndiz, J.; Fuentes-Silva, C.; Hernández-Alvarado, R.; Falcón, M.D.C.P.T. Self-Tuning Neural Network PID with Dynamic Response Control. *IEEE Access* **2021**, *9*, 65206–65215. [CrossRef]
47. Afifi, M.; Rezk, H.; Ibrahim, M.; El-Nemr, M. Multi-Objective Optimization of Switched Reluctance Machine Design Using Jaya Algorithm (MO-Jaya). *Mathematics* **2021**, *9*, 1107. [CrossRef]
48. Dutta, P.; Nayak, S.K. Grey Wolf Optimizer Based PID Controller for Speed Control of BLDC Motor. *J. Electr. Eng. Technol.* **2021**, *16*, 955–961. [CrossRef]
49. Jimenez-Gonzalez, J.; Gonzalez-Montanez, F.; Jimenez-Mondragon, V.M.; Liceaga-Castro, U.; Escarela-Perez, R.; Olivares-Galvan, J.C. Parameter Identification of BLDC Motor Using Electromechanical Tests and Recursive Least-Squares Algorithm: Experimental Validation. *Actuators* **2021**, *10*, 143. [CrossRef]