electronics

MDPI

Article

Metaheuristic Algorithm-Based Proportional–Integrative–Derivative Control of a Twin Rotor Multi Input Multi Output System

Ali Can Cabuker * and Mehmet Nuri Almalı

Department of Electrical-Electronics Engineering, Faculty of Engineering, Van Yuzuncu Yil University, Van 65000, Türkiye; mna1@yyu.edu.tr

* Correspondence: 19910001123@yyu.edu.tr; Tel.: +90-534-258-98-58

Abstract: Metaheuristic algorithms are computational techniques based on the collective behavior of swarms and the study of organisms acting in communities. These algorithms involve different types of organisms. Finding controller values for nonlinear systems is a challenging task using classical approaches. Hence, using metaheuristics to find the controller values of a twin rotor multi-input multi-output system (TRMS), one of the nonlinear systems studied in the literature, seems to be more appropriate than using classical methods. In this study, different types of metaheuristic algorithms were used to find the PID controller values for a TRMS, including a genetic algorithm (GA), a dragonfly algorithm, a cuckoo algorithm, a particle swarm optimization (PSO) algorithm, and a coronavirus optimization algorithm (COVIDOA). The obtained graphs were analyzed based on certain criteria for the main rotor and tail rotor angles to reach the reference value in the TRMS. The experimental results show that when the rise and settlement times of the TRMS are compared in terms of performance, the GA took 1.5040 s (seconds) and the COVIDOA took 9.59 s to increase the pitch angle to the reference value, with the GA taking 0.7845 s and the COVIDOA taking 2.4950 s to increase the yaw angle to the reference value. For the settling time, the GA took 11.67 s and the COVIDOA took 28.01 s for the pitch angle, while the GA took 14.97 s and the COVIDOA took 26.69 s for the yaw angle. With these values, the GA and COVIDOA emerge as the foremost algorithms in this context.

Keywords: twin rotor MIMO system (TRMS); metaheuristic algorithms; proportional–integral–derivative (PID) control; nonlinear system

check for updates

Citation: Cabuker, A.C.; Almalı, M.N. Metaheuristic Algorithm-Based Proportional–Integrative–Derivative Control of a Twin Rotor Multi Input Multi Output System. *Electronics* **2024**, *13*, 3291. https://doi.org/10.3390/electronics13163291

Academic Editors: François Auger and Jingyang Fang

Received: 18 July 2024
Revised: 1 August 2024
Accepted: 15 August 2024
Published: 19 August 2024

Copyright: © 2024 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

Aircraft play a major role in various fields today, and their safety and control are crucial. A twin rotor multi-input multi-output system is a type of aircraft that resembles a helicopter in structure but differs in terms of its aerodynamic force control, which is achieved through DC motors. The two DC motors in the system face control challenges due to coupling dynamics during their movements. The system's controller regulates the voltage supplied to the DC motors to achieve the desired horizontal and vertical positions [1,2]. Unlike a helicopter, a TRMS lacks flight capability and cyclic control [3]. A TRMS comprises two DC motors and a balance beam structure that creates a biaxial state in the plane of motion. One of the DC motors controls the main rotor, allowing the system to move up or down, while the other DC motor in the tail rotor enables the system to move right or left. The main purpose is to keep the pitch and yaw angles in the TRMS in the equilibrium position. To achieve this, four different variables are measured on the TRMS: the vertical angle, horizontal angle, and their individual angular velocities. The system is modeled using Newton's laws of motion and can be linearized under certain conditions in a helicopter, a type of TRMS used in real-life scenarios [4]. A TRMS can be operated as a single-axis

or two-axis system [5]. If the system is operated in two axes, two PID controllers and six values should be obtained [6].

These systems can be controlled by determining the parameters of the proportional–integral–derivative (PID) controller, a classical control method. The parameters of the PID controller can be determined using classical methods such as the Ziegler–Nichols and Astrom–Hagglund methods, as well as other methods [7,8]. PID controllers have recently been used for several different systems by converting these controllers to discrete-time controllers; at the same time, metaheuristic algorithms such as the particle swarm optimization algorithm have been used to determine the parameters of discrete-time PID controllers [9,10].

Solving complex optimization problems requires advanced techniques beyond traditional and classical methods. These techniques include metaheuristic algorithms, which are computational intelligence paradigms capable of handling high-dimensional, nonlinear, and hybrid problems. Metaheuristic algorithms are classified based on their types, such as swarm-based, evolutionary, physics-based, or human-based algorithms, as well as their nature-inspired or non-nature-inspired approach. Additionally, they can be categorized as population-based or single-solution-based algorithms [11,12].

Related studies of TRMSs have aimed to increase robustness by reducing the impact of external disturbances encountered in daily life, such as wind, and minimizing the uncertainties of the TRMS itself [13]. With controllers that increase the system's robustness, the TRMS's position can also be adjusted [3].

The different transfer functions TRMSs are applied to are the horizontal angle and the vertical angle and their effects on each other. These effects determine the coupling dynamics. Different discrete-time values can be determined for the sampling times of these transfer functions. With these discrete-time transfer functions, the efficiency of the PID controller can be increased, and more robust control can be provided. Therefore, the system should be controlled in discrete time to reduce the effect of coupling dynamics on TRMSs [14,15].

Metaheuristic algorithms can be used in nonlinear systems, such as an inverted-pendulum system. PID controller parameters are optimized using ant colony optimization to control the inverted pendulum. First, the system state is modeled in state space by considering the physical properties of the system. A PID controller controls the system variables specified in the state space in the best way possible and works to minimize the error in closed-loop control systems. The ant colony optimization algorithm determines the values that will achieve the best response in the system for the controllers. The performance of the inverted pendulum is evaluated based on the integral square error values. The ant colony optimization algorithm mainly ensures that each iteration achieves a cost function lower than the previous one. The integral squared error is then used as the fitness function to find the PID controller values [16].

Evolutionary computation methods and metaheuristic optimization algorithms have been used in many studies to find PID controller parameters for TRMSs. These studies have been carried out with classical swarm-based optimization algorithms, such as particle swarm optimization. In these studies, IAE, ITAE, ITSE, and ISE have been used as the performance measures for the fitness function value, and these performance values can be expressed as the sum of the main rotor and tail rotor errors [17–19].

2. A Mathematical Model of the Twin Rotor MIMO System

A TRMS uses two DC motors and a balance beam, and its mathematical model is based on its electrical and mechanical properties. Separate equations can be written for the main rotor and the tail rotor. Figure 1 shows the parameters that affect TRMSs on the horizontal and vertical axes [20–22]. To understand the flight dynamics and behavior of a TRMS, it is necessary to know the dynamic equations of the system.

A TRMS consists of a main rotor and a tail rotor, as shown in Figure 1. The twin DC motors that power these rotors are connected to a voltage input from an electrical unit

beneath the TRMS. This electrical unit is a communication switch between the TRMS and the computer and contains an emergency shutdown unit. The twin rotors are connected to DC inputs that produce torques τ1 and τ2, which turn the rotors.

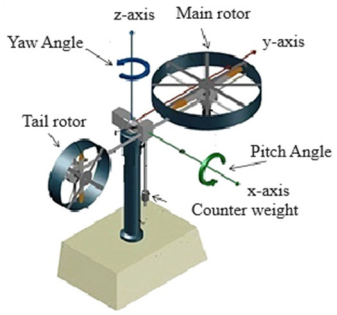


Figure 1. TRMS laboratory model.

The TRMS provides two position measurements—the pitch (ψ) angle and the yaw (ϕ) angle. The TRMS model can be divided into two parts: a DC motor model for the main and tail rotors and a mechanical model for the TRMS in the horizontal and vertical planes. Equations (1) and (2) refer to the torque of the pitch and yaw angles, respectively [16,20,21].

$$\frac{d\tau_1}{dt} = -\frac{T_{10}}{T_{11}}\tau_1 + \frac{k_1}{T_{11}}u_1 \quad (1)$$

$$\frac{d\tau_2}{dt} = -\frac{T_{20}}{T_{21}}\tau_2 + \frac{k_2}{T_{21}}u_2 \quad (2)$$

In Equations (1) and (2), T_{10}, T_{11}, T_{20}, and T_{21} are the denominator parameters, and k_1 and k_2 represent the motor gains. The torque equations give rise to mechanical equations that describe the TRMS's vertical and horizontal movements. These are expressed in Equations (3) and (4).

For vertical movement,

$$I_1\psi'' = M_1 - M_{FG} - M_{B\psi} - M_G \quad (3)$$

For horizontal movement,

$$I_2\phi'' = M_2 - M_{B\phi} - M_R \quad (4)$$

The horizontal and vertical expressions are provided in Equations (5)–(11).

$$M_1 = \alpha_1.\tau_1^2 + b_1.\tau_1 \quad (5)$$

$$M_2 = \alpha_2.\tau_2^2 + b_2.\tau_2 \quad (6)$$

$$M_{FG} = M_g.\sin\psi \quad (7)$$

$$M_{B\psi} = B_{1\psi}.\psi + \frac{0.0326}{2}.\sin(2.\psi).\phi^2 \quad (8)$$

$$M_{B\phi} = B_{1\phi}.\phi \quad (9)$$

$$M_G = k_{gy}.M_1.\phi.\cos\psi \quad (10)$$

$$M_R = \frac{kc.(T_0.s+1)}{(T_p.s+1)} \quad (11)$$

The TRMS model parameters are provided in Equations (1)–(11). I_1 and I_2 are moments of inertia for the vertical and horizontal movements of the rotors, respectively; M_1 and M_2 reflect the nonlinear static characteristics; αi and bi (1, 2) are the static characteristic

parameters; M_{FG} is momentum due to gravity; M_g is the moment of gravitational force; $M_{B\psi}$ and $M_{B\phi}$ are the momentum caused by friction forces; $B_{1\psi}$ and $B_{1\phi}$ are the friction momentum function parameters for the respective rotors; M_G is the momentum from the gyroscopic effects; and k_{gy} is the gyroscopic momentum parameter. The cross-reaction momentum between the two axes, M_R, is the output of a linear system [20].

Table 1 defines the expressions and numerical parameter values used in the TRMS equations.

Table 1. TRMS model parameters.

Parameter	Value
I_1—moment of inertia of vertical rotor	6.8×10^{-2} kg.m^2
I_2—moment of inertia of horizontal rotor	2×10^{-2} kg.m^2
α_1—static characteristic parameter	0.00135
b_1—static characteristic parameter	0.0924
α_2—static characteristic parameter	0.02
b_2—static characteristic parameter	0.09
Mg—gravity momentum	0.32 N.m
B_1—friction momentum function parameter	6×10^{-3} Nms/rad
B_2—friction momentum function parameter	1×10^{-3} Nms^2/rad
B_1—friction momentum function parameter	1×10^{-1} Nms/rad
B_2—friction momentum function parameter	1×10^{-2} Nms^2/rad
K_{gy}—gyroscopic momentum parameter	0.05 s/rad
k_1—motor 1 gain	1.1
k_2—motor 2 gain	0.8
T_{11}—motor 1 denominator parameter	1.1
T_{10}—motor 1 denominator parameter	1
T_{21}—motor 2 denominator parameter	1
T_{20}—motor 2 denominator parameter	1
T_p—cross-reaction momentum parameter	2
T_0—cross-reaction momentum parameter	3.5
kc—cross-reaction momentum gain	−0.2

Equations (12)–(15) describe the state-space model [5,23].

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{-M_G}{I_1} & \frac{-B_{1\varphi}}{I_1} & 0 & 0 & 0 & \frac{b_1}{I_1} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-B_{1\theta}}{I_2} & \frac{-1}{I_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{T_p} & \frac{-kc}{T_p}\left(\frac{-T_{10}}{T_{11}}T_0 + 1\right) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-T_{10}}{T_{11}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-T_{20}}{T_{11}} \end{bmatrix} \quad (12)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{k_1 k_c T_0}{T_{11} T_p} & 0 \\ \frac{k_1}{T_{11}} & 0 \\ 0 & \frac{k_2}{T_{21}} \end{bmatrix} \quad (13)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (14)$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (15)$$

Equations (12) and (13) correspond to the values for matrices A and B in Equations (16) and (17) [5,23]:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -4.7059 & -0.088 & 0 & 0 & 0 & 1.359 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & -50 & 0 & 4.5 \\ 0 & 0 & 0 & 0 & -0.5 & 0.22 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.909 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (16)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.35 & 0 \\ 1 & 0 \\ 0 & 0.8 \end{bmatrix} \quad (17)$$

The state space model of the TRMS includes the pitch, yaw, and the impact of these angles on each other. If desired, it can be expressed using transfer functions. Figure 2 illustrates this coupling effect with a block diagram [24,25].

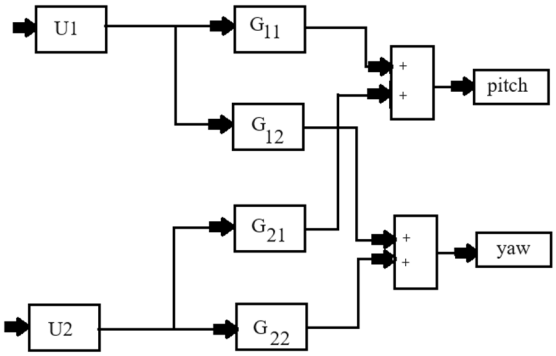


Figure 2. Block diagram of TRMS transfer function.

In Figure 2, U1 and U2 are the controller outputs; the transfer function of the TRMS can be represented by a 2 × 2 matrix, comprising four transfer functions. Equation (18) expresses the transfer functions in the Laplace (s) domain, expressed as a 2 × 2 matrix denoted by G(s) [25–27].

$$G(s) = \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix}, \; G(s) = \begin{pmatrix} \frac{1.359}{s^3+0.997\times s^2+4.786\times s+4.278} & 0 \\ \frac{1.617}{s^3+0.997\times s^2+4.545\times s} & \frac{3.6}{s^3+6\times s^2+5} \end{pmatrix} \quad (18)$$

Four transfer functions were created in the literature to study the main rotor and tail rotor angles of TRMSs. Among these, G_{11} represents the transfer function of the pitch angle, G_{22} represents the transfer function of the yaw angle, G_{12} represents the transfer function of the yaw angle to the pitch angle, and G_{22} represents the transfer function of the pitch angle to the yaw angle. To improve system control and increase the PID controller's efficiency, we converted the transfer functions written in the Laplace domain to the z-domain, i.e., a discrete-time form based on the sampling time Ts = 0.001. Equation (19) provides the z-transfer function of the TRMS as a 2 × 2 matrix.

$$G(z) = \begin{pmatrix} G_{11(z)} & G_{12(z)} \\ G_{21(z)} & G_{22(z)} \end{pmatrix}, G(z) = \begin{pmatrix} \frac{(2.264e-10)z^2+(9.055e-10)z+(2.263e-10)}{z^3+2.999\times z^2+2.998\times z-0.999} & 0 \\ \frac{(8.069e-7)z+(8.053e-7)}{z^2-1.994\times z+0.9941} & \frac{(1.796e-6)z+(1.793e-6)}{z^2-1.994\times z+0.994} \end{pmatrix} \quad (19)$$

The reason for the z-domain conversion is that the TRMS is connected to a computer during real-time operation. This transformation ensures that the data flow correctly between the TRMS and the computer, the difficulties encountered in the s domain are overcome, and efficiency is increased. The selected sampling time was first simulated on the computer, and then the controller parameters were found and run in the real system. Since the Ts (sampling time) value was quite small, the parameters were obtained on the computer, and experimental studies were carried out. In this way, metaheuristic algorithms can cope with the sampling time against the real system.

3. Metaheuristic Algorithms

A TRMS is a complex system that takes in multiple inputs and produces multiple outputs. In all of the algorithms presented below, the fitness function is computed by summing up the integral time squared errors (ITSEs) resulting from differences between the real-time data values obtained from the pitch and yaw angles (the output values) and the reference value. This approach identifies the parameter values that result in the least error during system operation. The mathematical expression of the cost function is provided in Equation (20).

$$\text{cost} = (0.7 \times \int_0^\infty e_{\text{pitch}}^2(t)tdt + 0.3 \times \int_0^\infty e_{\text{yaw}}^2(t)tdt \Big) \quad (20)$$

In Equation (20), e_{pitch} indicates the amount of error between the reference value and the instantaneous pitch angle value, and e_{yaw} indicates the amount of error between the reference value and the instantaneous yaw angle value. Modeling the system in discrete time allows the nonlinear TRMS to be controlled using linear modeling. The sampling time in discrete time more clearly shows the influence of the main rotor and the tail rotor on each other. Accounting for this, since the main rotor has a greater impact on the system's operation and reaching the reference value, 70% of the error weight is assigned to the main rotor and 30% to the tail rotor. This decision is based on the coupling effect. In this study, we compared the performance of traditional and old metaheuristic algorithms—previously studied on another TRMS—and a new algorithm inspired by a viral infection that causes death by hijacking the new-generation immune system on our TRMS. This new algorithm responded with less overshoot at the first launch than the previous algorithms.

3.1. Particle Swarm Optimization

Nature has been an inspiration for scientific development throughout the years. For instance, the biological behavior of animal herds has led to innovations such as artificial intelligence. Optimization methods derived from the joint action of herds have provided substantial benefits in solving complex computational problems. One optimization method is the particle swarm optimization (PSO) algorithm, which is widely used in various studies by developing variants, mainly in engineering and computer sciences. In the PSO algorithm, initial parameters are defined. Then, a swarm is created, and each particle is compared with the others to determine the best position in the search space within the specified range. The movement of each particle varies according to its previous position and its determined velocity. Each particle placed in the search space updates its position in each iteration. As a result of these updates, the optimal fitness function can be approached [28,29]. The mathematical expressions of particle swarm optimization are presented in Equations (21) and (22).

$$v_{k+1} = w \times v_k + c_1 r_1 (pbest_k - x_k) + c_2 r_2 (gbest_k - x_k) \quad (21)$$

$$x_{(k+1)} = x_k + v_{k+1} \quad (22)$$

Equation (21) consists of three components: $w \times v_k$ represents the initial velocity, c_1r_1 is the cognitive component, and c_2r_2 is the social component. The current position of x_k is also a part of this equation. Equation (22), on the other hand, refers to the fast updating of the optimization's position.

3.2. Dragonfly Algorithm

The dragonfly algorithm is an optimization method that has inspired metaheuristic algorithms. It is based on the behavior of dragonflies in nature, specifically how they hunt and prey. Dragonflies have approximately 3000 species in nature, and they tend to live in flocks and become predators and hunters when they reach the adult stage [30].

The dragonfly algorithm involves various behaviors, such as alignment, separation, cohesion, attraction to food, and distraction from enemies. These behaviors determine the movements of the algorithm's particles and help achieve its objective. To create artificial dragonflies, these behaviors are integrated into a computer program, as described in Equation (23).

$$\Delta P_i^{t+1} = (s.S_i + \alpha.A_i + c.C_i + f.F_i + e.E_i) + w.\Delta P_i^t \quad (23)$$

In Equation (23), s is the separation weight, a is the alignment weight, c is the alignment weight, f is the food factor, and e is the enemy factor. The indented expressions S_i, A_i, C_i, F_i, and E_i denote the ith individual's separation, alignment, cohesion, food, and enemy factors at instantaneous iterations in computer simulations [31–33]. Every algorithm has a flow diagram that shows the sequence of operations performed by the computer. The dragonfly algorithm is widely used to solve optimization problems in various fields, such as power and energy systems, medical systems, and audio and image processing. Moreover, there are different versions of the dragonfly algorithm, including single-objective problems, the binary dragonfly algorithm, the multi-objective version, and hybrid versions [34].

3.3. Cuckoo Search Algorithm

The cuckoo algorithm is a swarm intelligence-based optimization algorithm developed in recent years. Cuckoos are one of many bird species found in nature, but what sets them apart is their parasitic lifestyle. They are also known for their beautiful sounds and aggressive breeding behavior. Cuckoos can lay their eggs in the nests of other host birds, and this algorithm is based on their parasitic, Levy flight behavior. The breeding and nesting behaviors of cuckoos optimize the breeding and reproduction conditions for the algorithm, solving difficult optimization problems [35,36].

The cuckoo algorithm is a simple algorithm that follows three basic rules. Each cuckoo lays one egg at a time and randomly places it in a nest. The nests with high-quality eggs are considered the best and are passed on to the next generation. The number of available host nests is fixed, and the hosts have a probability of discovering a foreign egg [0, 1] [37]. The cuckoo algorithm uses Levy flight to help birds find new solutions during optimization. This flight pattern directs the cuckoos to their next position. Equation (24) determines the movement of a cuckoo to the next position using Levy flight.

$$X_i^{(t+1)} = X_i^{(t)} + \alpha \oplus \text{Levy}(\lambda)(1) \quad (24)$$

Equation (24) defines α as the step size that can expand or contract the search space. Equation (25) provides the Levy distribution from which the Levy flight is obtained.

$$\text{Levy} \sim u = t - \lambda, (1 < \lambda \leq 3) \quad (25)$$

The cuckoo algorithm belongs to a class of population-based swarm intelligence algorithms. It utilizes a step length that randomizes the process and, therefore, enhances its efficiency. The algorithm requires fewer parameter inputs than particle swarm op-

timization, providing an edge. Before running the cuckoo optimization algorithm, the optimization habitat must first be created. The habitat is obtained by defining an Nvar dimensional optimization model based on the size of the problem. The habitat is created as a 1 × Nvar array that represents the location of the cuckoos. The array is represented by Equation (26) [35].

$$\text{habitat} = [x_1, x_2, \ldots\ldots \text{xNvar}] \tag{26}$$

The cuckoo search algorithm maximizes a profit function (Equation (27)) that depends on habitat and cost.

$$\text{profit} = \text{cost}(\text{habitat}) = f_c(x_1, x_2, \ldots\ldots \text{xNvar}) \tag{27}$$

In an optimization problem, var_{hi} and var_{low} are set as the lower and upper bounds. These bounds define the "laying radius" of the cuckoo flocks in the search space. Each cuckoo has its own laying radius, calculated using Equation (28).

$$\text{Eggradius} = \alpha \times \frac{\text{Number of current cuckoos eggs}}{\text{total number of eggs}} \times (\text{var}_{\text{hi}} - \text{var}_{\text{low}}) \tag{28}$$

3.4. Genetic Algorithm

Animal herds can adapt their bodies to the environments they live in. This process is known as adaptation. The genetic algorithm involved in this process comprises stages such as natural selection, crossover, and mutation. These stages bring about changes in the genotypes and phenotypes of animals. The ability of living organisms to adapt to changing environmental and climatic factors is crucial. Individuals who fail to adapt to their surroundings cannot pass on their genes to the next generation through natural selection. Therefore, the fittest individuals survive and thrive in animal herds [38–40].

The genetic algorithm involves a population of individuals living in a habitat. Each individual has chromosomes in its cellular structure, which contain genes. The gene sequences within chromosomes determine an organism's genotype and phenotype. The genetic algorithm follows a flow diagram similar to that of other metaheuristic algorithms. In the population creation stage, genes are created using binary values of 0 and 1. These genes are then placed in chromosomes to form a population of individuals. The chromosomes are evaluated to ensure they meet predetermined conditions. Next, the crossover process begins, where gene sequences from different chromosomes are combined to create new chromosomes without affecting their biological properties. The crossover process can be single-point or multi-point depending on the specific requirements [41]. During the mutation stage of genetic algorithms, gene sequences in chromosomes undergo alteration. The mutation process takes place at a particular rate, usually defined within a range of 0–1. In the final stage, before reaching the maximum number of iterations, selection is performed. This selection process involves choosing individuals to be matched using various methods, such as the roulette wheel and tournament method [42].

3.5. Coronavirus Disease Optimization Algorithm (COVIDOA)

The COVIDOA is an optimization algorithm inspired by the coronavirus disease that emerged in late 2019 and affected the whole world. As a structure, the coronavirus microbe contains RNA and the N protein glycoprotein. It also harbors its genetic structure in an oily membrane as a cellular structure. This genetic structure is encoded in RNA and is read by the infected cell in the body [43]. The coronavirus consists of four structural proteins. The coronavirus can be transmitted from person to person through coughs and sneezes and causes death with conditions such as septic shock and multiple organ failure.

The biggest danger this virus poses to human health is its rapid replication. Here, the replication mechanism of the coronavirus was used for the proposed algorithm to achieve a specific goal. Figure 3 shows the flow diagram of the COVIDOA used to find the PID parameter for TRMS [44].

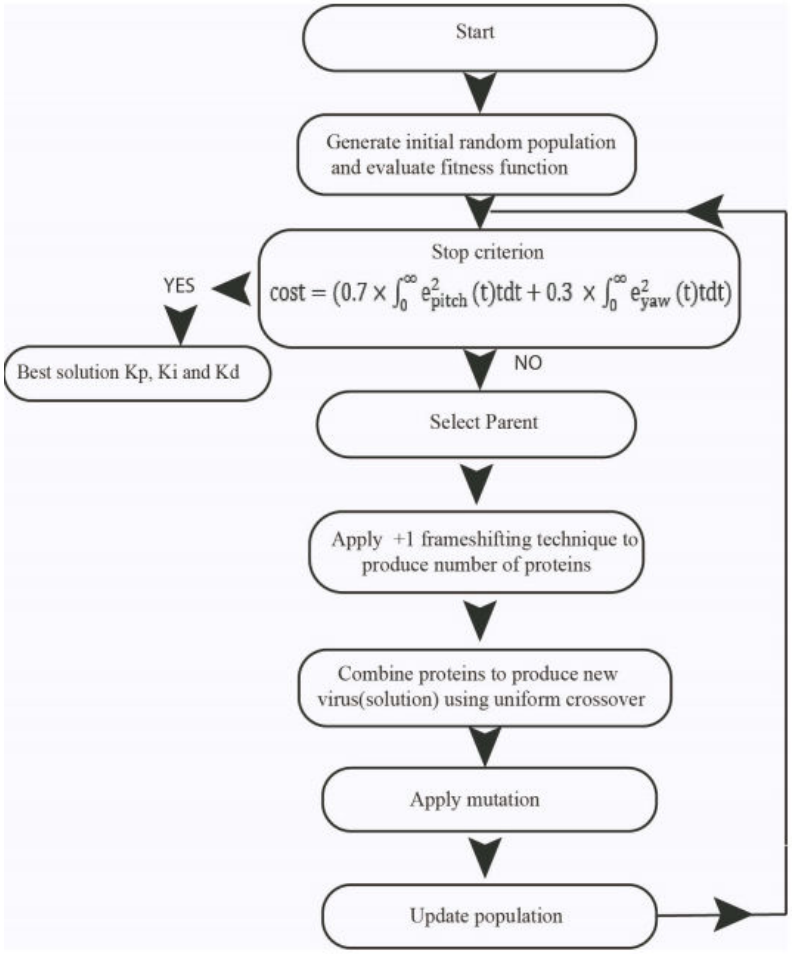


Figure 3. Flowchart of COVIDOA with TRMS.

The PID controller values were determined using metaheuristic algorithms, and their codes were generated. These values were then used to conduct physical experiments on the TRMS. The results section of this paper includes graphs and their respective ISE, IAE, ITSE, and ITAE values for the 2 DOF system. Figure 3 shows an example diagram of the COVIDOA for the TRMS. The flow diagram in Figure 3 shows that the cost function for all algorithms is placed in them and weighted for the TRMS.

In the COVIDOA, first of all, a population is created randomly, and the cost is calculated for each solution. When we look at the mathematical equations for each structural protein mentioned, if there is a +1 frameshift,

$$S_k(1) = \text{rand}(\text{minVal}, \text{maxVal}) \tag{29}$$

$$S_k(2:D) = P(1:D-1) \tag{30}$$

If there is a -1 frameshift,

$$S_k(D) = \text{rand}(\text{minVal}, \text{maxVal}) \tag{31}$$

$$S_k(2:D) = P(1:D-1) \tag{32}$$

S_k refers to the kth generated protein, P is the parent solution, and D is the problem dimension (the number of variables in each solution). The frameshifting results represent a new protein sequence. The COVIDOA's cost is noted in Figure 4 as an example of minimizing cost (fitness) values for the above metaheuristic algorithms over iterations.

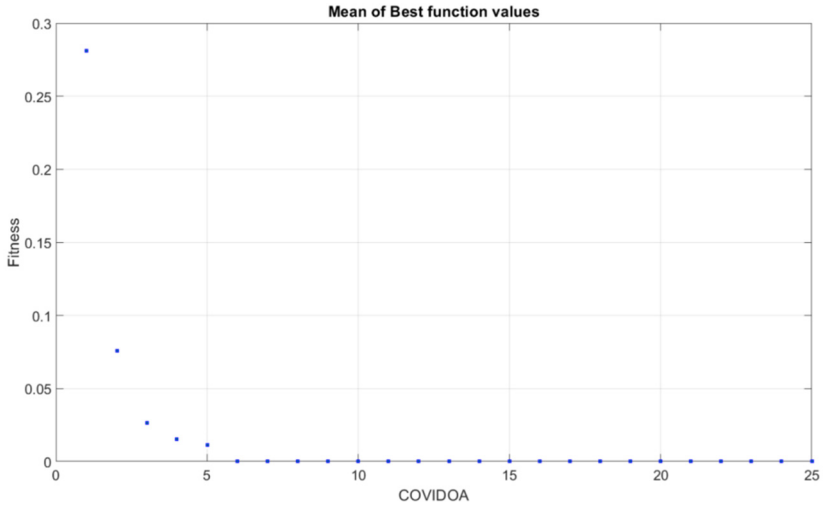


Figure 4. Cost (fitness) of COVIDOA.

4. Tuning PID with Metaheuristic Algorithms and Discrete PID Controller

To obtain PID values, the system should load values obtained from metaheuristic algorithms. Then, the iterations should be improved using the fitness function provided in Equation (20). A PID-tuning block diagram is provided in Figure 5.

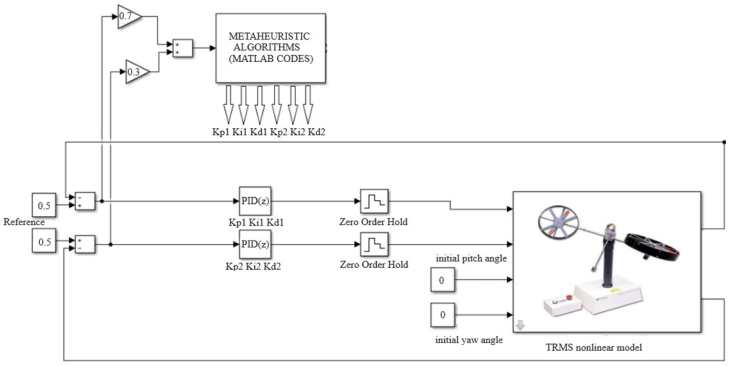


Figure 5. TRMS; discrete-time control.

The discrete-time PID controller is used to control digital systems. To use transfer functions and PID controllers continuously, they must be transformed from the s-domain into the z-domain. The z-transfer function changes with the sampling time, obtaining different transfer functions. The metaheuristic algorithm block in Figure 5 indicates the codes written for the algorithms. With these codes, six discrete-time PID parameters are obtained in each iteration (simulation) according to the amount of error in the system. Three parameters are obtained as the pitch angle and three as the yaw angle. The PID controller used in continuous time consists of K_p (proportional), K_i (integral), and K_d (derivative). It is converted into discrete time through the backward difference method. Using this method, the discrete transfer functions of a numerical integrator and numerical differentiator are determined.

Equation (33) is a mathematical expression for the discrete-time PID controller [9].

$$\mathrm{PID}(z) = [K_p + \frac{K_i T_s . z}{z-1} + \frac{K_d}{T_s}\frac{z-1}{z}].e[z] \quad (33)$$

Equation (33) is a discrete-time PID formula based on the difference between times z and $z-1$ within the sampling time. Ts is the sampling time, and e[z] is the error in the system. e[z] is also the input of the discrete-time PID controller. A block diagram of the discrete-time PID controller in the Matlab (2019) program was used. Unlike a traditional

PID controller formula, there is a difference concerning the sampling time. z is any moment in the sampling time. The graphs from the simulation model were accessed through the block diagram of the model parameters of the TRMS in the Matlab program. The K_{p1}, K_{i1}, and K_{d1} and K_{p2}, K_{i2}, and K_{d2} parameters obtained for pitch and yaw were generated in each iteration by searching within a solution space (lower and upper bounds) determined for the algorithms. During the iterations, the interaction between the parameters, i.e., the estimated mean value range (0–10), emerges by minimizing the Fitness function value. For simulation calculations, the m-file was used for coding in connection with Matlab/Simulink (2019).

5. ISE, IAE, ITSE, and ITAE Error Performance

An error performance analysis determined how much error the TRMS gains in reaching the targeted reference value. To analyze the performance of the value responsible for controlling the TRMS when reaching the reference value, at least one of the integral squared error (ISE), integral absolute error (IAE), integral time squared error (ITSE), and integral time absolute error (ITAE) values must be considered. The formulas for these errors are provided in Equations (34)–(37) [42].

$$\mathrm{ISE} = \int_0^\infty e^2(t)dt \tag{34}$$

$$\mathrm{ITSE} = \int_0^\infty e^2(t)tdt \tag{35}$$

$$\mathrm{IAE} = \int_0^\infty |e(t)|dt \tag{36}$$

$$\mathrm{ITAE} = \int_0^\infty t|e(t)|dt \tag{37}$$

The ITSE was used as the fitness function. The ITSE performance value is provided in Equation (20) as 70% for the main rotor and 30% for the tail rotor, including the total error of the system.

6. Results

Experimental studies were conducted on a physical TRMS, and the values obtained from each algorithm were compared for the pitch and yaw angles. The results are analyzed and compared in Figure 6 (pitch angle) and Figure 7 (yaw angle). The algorithm values were obtained by setting the maximum iteration number average to 25 iterations and the value search range to 0–10 for the PID parameters. We selected PID parameters in this range to prevent excessive disturbances in the system's dynamic behavior. In other words, we aimed to reduce the percentage of exceedance amounts. For TRMS, the actuator condition (ITSE) was caused by the main rotor, and its ratio in the fitness function is taken more than in the tail rotor (Equation (20)).

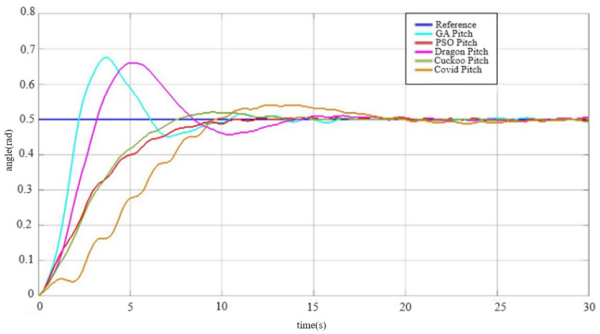


Figure 6. Pitch angle response with particle swarm, genetic, dragonfly, cuckoo search, and coronavirus optimization algorithms.

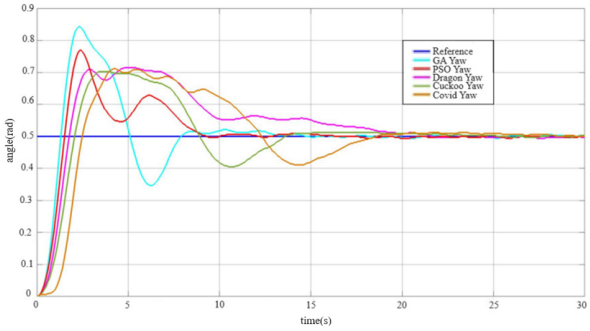


Figure 7. Yaw angle response with particle swarm, genetic, dragonfly, cuckoo search, and coronavirus optimization algorithms.

The ISE, IAE, ITAE, and error performance values were obtained for the pitch and yaw angles with the different algorithms. Table 2 shows the error performance values for the ISE, IAE, ITAE, and ITSE. Table 3 shows the K_p, K_i, and K_d values obtained from the PSO, dragonfly, cuckoo, genetic, and coronavirus optimization algorithms. Table 4 shows the rise time, settling time, peak value, and percentage overshoot amounts.

Table 2. ISE, IAE, ITAE, and ITSE performance values.

Algorithm	Angle	ISE	IAE	ITAE	ITSE
GA	Pitch	0.3219	1.425	15.05	0.4602
GA	Yaw	0.4205	1.772	15.92	1.86
PSO	Pitch	0.4596	1.738	13.26	3.59
PSO	Yaw	0.3105	1.479	15.12	2.225
Dragon	Pitch	0.4148	1.727	13.44	0.5172
Dragon	Yaw	0.4964	2.601	29.17	15.55
Cuckoo	Pitch	0.4838	1.733	14.11	1.523
Cuckoo	Yaw	0.4613	2.179	18.53	5.244
Coronavirus	Pitch	0.8739	2.734	12.16	5.308
Coronavirus	Yaw	0.6368	2.682	16.78	5.937

Table 3. The controller values of the algorithms.

Algorithm	Angle	Controller	Kp	Ki	Kd
GA	Pitch	PID	2.1732	4.5021	4.6705
GA	Yaw	PID	4.0039	3.3815	3.1414
PSO	Pitch	PID	3.0366	1.0891	6.6809
PSO	Yaw	PID	3.1704	2.6631	2.0078
Dragon	Pitch	PID	1.7811	3.0868	6.6290
Dragon	Yaw	PID	2.7536	0.6523	1.1953
Cuckoo	Pitch	PID	1.9912	1.1608	5.4874
Cuckoo	Yaw	PID	1.6197	3.3800	4.7060
Coronavirus	Pitch	PID	2.0374	0.5431	4.3727
Coronavirus	Yaw	PID	3.4720	4.5185	3.0745

Table 4. Rise, settling times, peak value, and percentage overshoot amounts.

Algorithm	Angle	Peak Value	Rise Time	Settling Time	% Overshoot
GA	Pitch	0.6750	1.5040	11.67	34.96
GA	Yaw	0.8437	0.7845	14.97	70.80
PSO	Pitch	0.5062	6.0875	10.27	1.227
PSO	Yaw	0.7701	0.9544	8.974	53.98
Dragon	Pitch	0.6596	2.2350	16.74	31.90

Table 4. *Cont.*

Algorithm	Angle	Peak Value	Rise Time	Settling Time	% Overshoot
Dragon	Yaw	0.7178	1.0747	21.79	42.94
Cuckoo	Pitch	0.5216	4.9833	12.97	4.938
Cuckoo	Yaw	0.7026	1.2772	22.17	41.35
Coronavirus	Pitch	0.5400	9.5930	28.01	7.3171
Coronavirus	Yaw	0.7118	2.4950	26.69	42.33

Figures 8–10 present bar graphs for the rise time, settling time, and overshoot amounts given in Table 4.

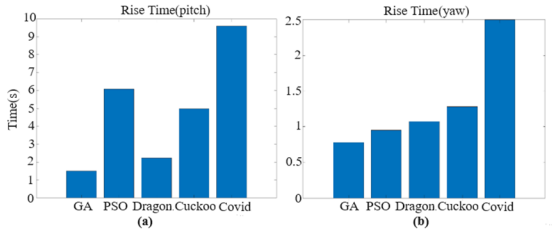


Figure 8. Bar graph of rise time: (**a**) pitch angle; (**b**) yaw angle.

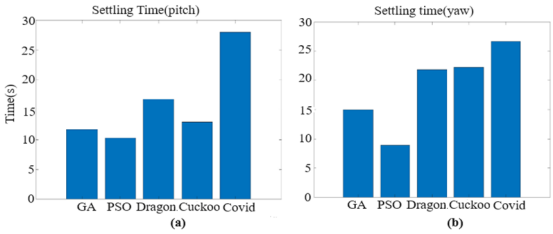


Figure 9. Bar graph of settling time: (**a**) pitch angle; (**b**) yaw angle.

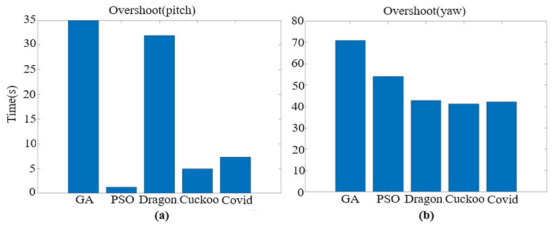


Figure 10. Bar graph of % overshoot: (**a**) pitch angle; (**b**) yaw angle.

7. Discussion

After an analysis of Table 2, the following results were obtained for the different optimization algorithms: For the main rotor angle, the genetic algorithm provided the lowest ISE value of 0.3219, the lowest IAE value of 1.425, and the lowest ITSE value of 0.4602. The coronavirus optimization algorithm provided the lowest ITAE value of 12.16. For the tail rotor angle, the particle swarm optimization algorithm provided the lowest ISE value of 0.3105, the lowest IAE value of 1.479, and the lowest ITSE value of 1.86. The lowest ITAE value was obtained from the particle swarm optimization algorithm. After analyzing the ITSE value used in the cost function and other error performance analyses, we determined that the genetic algorithm outperformed the other algorithms.

After analyzing the data in Table 4, which include the rise, settlement, and percentage overshoot amounts, we found that the particle swarm optimization algorithm had the lowest overshoot amount of 0.5062 for the main rotor angle, while the genetic algorithm had the highest overshoot amount of 0.6750. The fastest rise time occurred in the genetic

algorithm with 1.5040, while the particle swarm optimization algorithm had the fastest settling time of 10.27. Additionally, the lowest percentage overshoot occurred in the particle swarm optimization algorithm at 1.227%. Comparing these values, we concluded that the PID parameters obtained from the genetic algorithm provided the fastest reaction, but it was behind the particle swarm optimization algorithm in terms of settlement and percentage overshoot. For the tail rotor angle, the cuckoo algorithm had the lowest overshoot of 0.7026, while the genetic algorithm had the highest overshoot of 0.8437. The fastest rise time was observed in the genetic algorithm at 0.7845, while the particle swarm optimization algorithm had the fastest settling time of 8.974. Moreover, the cuckoo algorithm had the lowest percentage overshoot at 41.35%. Comparing these values, we concluded that the PID parameters obtained from the genetic algorithm provided the fastest reaction, but it was behind the particle swarm optimization and cuckoo algorithms regarding settlement and percentage overshoot. Determining which values obtained from the algorithms have the best or worst response will provide an advantage in knowing what kind of control response TRMS users want. At this point, the user should choose the algorithm for the TRMS. Whether the system reaches the reference value safely or the reference value of fast and sudden take-off should be determined by the user. This will naturally directly affect the comfort of the system. If the TRMS user does not care about high overshoots at the first take-off moment of the system, they should use the coefficients obtained from the genetic algorithm; if the user does not want high overshoots, they should use the coronavirus optimization algorithm.

8. Conclusions

Our study shows that the TRMS moves suddenly to reach the reference value determined at the first take-off moment. With the PID controller parameters used, although there was a problem at the input moment, the reference value was reached shortly. In the last case, the reference value was reached, and robust control was provided with small overshoots. In a real-time experimental study, the coronavirus optimization algorithm showed that the TRMS slowly and safely reached the reference value. The main rotor made fewer overshoots than the genetic and dragonfly algorithms. For the tail rotor, the genetic algorithm made fewer overshoots than the cuckoo algorithm. Since the coronavirus optimization algorithm is a new algorithm, its performance was close to that of the particle and cuckoo algorithms in terms of the main rotor and close to that of the dragonfly and cuckoo algorithms in terms of the tail rotor.

From this point of view, the coronavirus algorithm provided successful results for the TRMS. However, it needs to be improved. The discrete-time PID controller parameters were determined by reducing the cost function value below 0.05 for each algorithm for the TRMS to fit the specified reference value. As a result, if the system needs to settle to the reference value safely and slowly, the parameters obtained from the coronavirus optimization algorithm should be used. If it is thought that the system moving quickly at the first start will not cause a problem, the parameters obtained from the genetic algorithm should be used. From these perspectives, the genetic algorithm and the coronavirus optimization algorithm come to the fore. The graphs obtained from the PSO, cuckoo, and dragon algorithms provided intermediate values between the genetic algorithm and the coronavirus optimization algorithm. It cannot be said that these algorithms are very good or very bad because they are between the comfort values provided by the coronavirus algorithm or the fast movement values of the genetic algorithm. In terms of the performance measures, since the system is nonlinear, it is not possible to reach the optimal value in all aspects. On the other hand, a gain in the rise or settlement time causes losses in terms of overshoot. In addition, a high gain in one angle can cause losses in the other.

When considering aircraft with two degrees of freedom, such as TRMSs, the coronavirus optimization algorithm can be used to reach reference values for a safe journey. In practice, the discrete-time PID controller provided better results in controlling nonlinear

systems with a suitable sampling time than the classical PID controller. It is advantageous to digitize a PID controller for a specific sampling time.

Author Contributions: All of the authors made significant contributions to this work. A.C.C. was involved in writing, reviewing, and editing in the original draft preparation and validation. M.N.A. helped with supervision, conceptualization, and methodology and contributed to visualization and investigation. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting the findings of this study are available upon request from the corresponding author.

Acknowledgments: The authors thank the referee.

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

References

1. Chalupa, P.; Přikryl, J.; Novák, J. Modelling of twin rotor MIMO system. *Procedia Eng.* **2015**, *100*, 249–258. [CrossRef]
2. Huu, T.D.; Ismail, I.B. Modelling of twin rotor MIMO system. In Proceedings of the 2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA), Ipoh, Malaysia, 25–27 September 2016; pp. 1–6.
3. Tiwalkar, R.G.; Vanamane, S.S.; Karvekar, S.S.; Velhal, S.B. Model predictive controller for position control of twin rotor MIMO system. In Proceedings of the 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, 21–22 September 2017; pp. 952–957.
4. Harshath, K.; Manoharan, P.S.; Varatharajan, M. Model predictive control of TRMS. In Proceedings of the 2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE), Bengaluru, India, 21–23 January 2016; pp. 1–5.
5. Chaudhary, S.; Kumar, A. Control of twin rotor mimo system using 1-degree-of-freedom PID, 2-degree-of-freedom PID and fractional order PID controller. In Proceedings of the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 12–14 June 2019; pp. 746–751.
6. Ghazali, M.R.; Ahmad, M.A.; Ismail, R.R. Data-driven neuroendocrine-PID controller design for twin rotor MIMO system. *Int. J. Phys. Conf. Ser. IOP Publ.* **2020**, *1529*, 042080. [CrossRef]
7. Antony, A.P.; Varghese, E. Comparison of performance indices of PID controller with different tuning methods. In Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 18–19 March 2016; pp. 1–6.
8. Housny, H.; El Fadil, H. Fuzzy PID control tuning design using particle swarm optimization algorithm for a quadrotor. In Proceedings of the 2019 5th International Conference on Optimization and Applications (ICOA), Kenitra, Morocco, 25–26 April 2019; pp. 1–6.
9. Chander, S.; Agarwal, P.; Gupta, I. Auto-tuned, discrete PID controller for DC-DC converter for fast transient response. In Proceedings of the India International Conference on Power Electronics 2010 (IICPE2010), New Delhi, India, 28–30 January 2011; pp. 1–7.
10. Zeghlache, S.; Benyettou, L.; Djerioui, A.; Ghellab, M.Z. Twin rotor MIMO system experimental validation of robust adaptive fuzzy control against wind effects. *IEEE Syst. J.* **2020**, *16*, 409–419. [CrossRef]
11. Souza, D.A.; Batista, J.G.; dos Reis, L.L.; Júnior, A.B. PID controller with novel PSO applied to a joint of a robotic manipulator. *J. Braz. Soc. Mech. Sci. Eng.* **2021**, *43*, 377. [CrossRef]
12. Dutta, L.; Das, D.K. A New Adaptive Explicit Nonlinear Model Predictive Control Design for a Nonlinear MIMO System: An Application to Twin Rotor MIMO System. *Int. J. Control Autom. Syst.* **2021**, *19*, 2406–2419. [CrossRef]
13. Rajabi Moshtaghi, H.; Toloie Eshlaghy, A.; Motadel, M.R. A comprehensive review on meta-heuristic algorithms and their classification with novel approach. *J. Appl. Res. Ind. Eng.* **2021**, *8*, 63–89.
14. Ozdemir, A.; Mucuk, I.; Ilhan, S. Çift Pervaneli Çok Girişli Çok Çıkışlı (TRMS) Sistemin Kontrolü için Ayrık Zaman PID Kontrolör Tasarımı ve Gerçek Zaman Uygulaması. *Acad. Perspect. Procedia* **2020**, *3*, 597–608. [CrossRef]
15. Ozdemir, A.; Erdem, Z. A new approach for calculation of PID parameters with model based compact form formulations. *Elektron. Ir Elektrotechnika* **2014**, *20*, 3–10. [CrossRef]
16. Jacknoon, A.; Abido, M.A. Ant Colony based LQR and PID tuned parameters for controlling Inverted Pendulum. In Proceedings of the 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), Khartoum, Sudan, 16–18 January 2017; pp. 1–8.
17. Choudhary, S.K. Optimal feedback control of twin rotor MIMO system with a prescribed degree of stability. *Int. J. Intell. Unmanned Syst.* **2016**, *4*, 226–238. [CrossRef]
18. Feedback Instument. *Twin Rotor MIMO System Control Experiments Manual 33–949S*; Feedback Instruments Ltd.: Crowborough, UK, 2010.

19. Castillo, O.; Kutlu, F.; Atan, O. Intuitionistic fuzzy control of twin rotor multiple input multiple output systems. *J. Intell. Fuzzy Syst.* **2020**, *38*, 821–833. [CrossRef]
20. Abukan, Y.; Almali, M.N. İki serbestlik dereceli TRMS MIMO sistemin FOPID ve FOSTSMC yöntemi kullanılarak kontrolü. *Gazi Üniversitesi Mühendislik Mimar. Fakültesi Derg.* **2022**, *38*, 605–616. [CrossRef]
21. Paul, P.K.; Jacob, J. On the modeling of twin rotor MIMO system using chirp inputs as test signals. *Asian J. Control* **2017**, *19*, 1731–1740. [CrossRef]
22. İzcier, E.; Almalı, M.N.; Parlar, I.; Çabuker, A.C. Determining the Parameters of the Controllers of the TRMS System Using the Cuckoo Algorithm. *Int. Conf. Sci. Acad. Res.* **2023**, *1*, 51–57.
23. Pandey, S.K.; Dey, J.; Banerjee, S. Design and real-time implementation of robust PID controller for Twin Rotor MIMO System (TRMS) based on Kharitonov's theorem. In Proceedings of the 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), New Delhi, India, 4–6 July 2016; pp. 1–6.
24. Cleghorn, C.W.; Engelbrecht, A.P. Particle swarm convergence: An empirical investigation. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 2524–2530.
25. Gad, A.G. Particle swarm optimization algorithm and its applications: A systematic review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]
26. Meraihi, Y.; Ramdane-Cherif, A.; Acheli, D.; Mahseur, M. Dragonfly algorithm: A comprehensive review and applications. *Neural Comput. Appl.* **2020**, *32*, 16625–16646. [CrossRef]
27. Eröz, E.; Tanyildizi, E. Çok amaçlı metasezgisel optimizasyon algoritmalarının performans karşılaştırması. In Proceedings of the 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 21–22 September 2019; pp. 1–11.
28. Hammouri, A.I.; Mafarja, M.; Al-Betar, M.A.; Awadallah, M.A.; Abu-Doush, I. An improved dragonfly algorithm for feature selection. *Knowl. -Based Syst.* **2020**, *203*, 106131. [CrossRef]
29. Rahman, C.M.; Rashid, T.A.; Alsadoon, A.; Bacanin, N.; Fattah, P.; Mirjalili, S. A survey on dragonfly algorithm and its applications in engineering. *Evol. Intell.* **2021**, *16*, 1–21. [CrossRef]
30. Alshinwan, M.; Abualigah, L.; Shehab, M.; Elaziz, M.A.; Khasawneh, A.M.; Alabool, H.; Hamad, H.A. Dragonfly algorithm: A comprehensive survey of its results, variants, and applications. *Multimed. Tools Appl.* **2021**, *80*, 14979–15016. [CrossRef]
31. Joshi, A.S.; Kulkarni, O.; Kakandikar, G.M.; Nandedkar, V.M. Cuckoo search optimization—A review. *Mater. Today Proc.* **2017**, *4*, 7262–7269. [CrossRef]
32. Rajabioun, R. Cuckoo optimization algorithm. *Appl. Soft Comput.* **2011**, *11*, 5508–5518. [CrossRef]
33. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
34. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
35. Lambora, A.; Gupta, K.; Chopra, K. Genetic algorithm-A literature review. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 380–384.
36. Sohail, A. Genetic algorithms in the fields of artificial intelligence and data sciences. *Ann. Data Sci.* **2021**, *10*, 1–12. [CrossRef]
37. Karaköse, E. Sürü İnsansız Hava Araçlarının Görev Paylaşımı için Genetik Algoritma Tabanlı Bir Yaklaşım. *Fırat Üniversitesi Mühendislik Bilim. Derg.* **2022**, *34*, 351–360. [CrossRef]
38. Stankiewicz, W.; Roszak, R.; Morzyński, M. Genetic algorithm-based calibration of reduced order Galerkin models. *Math. Model. Anal.* **2011**, *16*, 233–247. [CrossRef]
39. Bar-On, Y.M.; Flamholz, A.; Phillips, R.; Milo, R. SARS-CoV-2 (COVID-19) by the numbers. *Elife* **2020**, *9*, e57309. [CrossRef]
40. Khalid, A.M.; Hosny, K.M.; Mirjalili, S. COVIDOA: A novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle. *Neural Comput. Appl.* **2022**, *34*, 22465–22492. [CrossRef] [PubMed]
41. Ezekiel, D.M.; Samikannu, R.; Oduetse, M. Modelling of the twin rotor mimo system (trms) using the first principles approach. In Proceedings of the 2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 22–24 January 2020; pp. 1–7.
42. Maiti, R.; Sharma, K.D.; Sarkar, G. PSO based parameter estimation and PID controller tuning for 2-DOF nonlinear twin rotor MIMO system. *Int. J. Autom. Control* **2018**, *12*, 582–609. [CrossRef]
43. Azar, A.T.; Sayed, A.S.; Shahin, A.S.; Elkholy, H.A.; Ammar, H.H. PID controller for 2-DOFs twin rotor MIMO system tuned with particle swarm optimization. In Proceedings of the International Conference on Advanced Intelligent Systems and Informatics, Cairo, Egypt, 26–28 October 2019; pp. 229–242.
44. Sivadasan, J.; Willjuice Iruthayarajan, M. Tuning of nonlinear PID controller for TRMS using evolutionary computation methods. *Tehnički Vjesn.* **2018**, *25*, 105–111.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.