mathematics

MDPI

Article

Performance Evaluation of Metaheuristic Algorithms for Fractional-Order PID Controller Tuning in Non-Linear Systems

Basil Mohammed Al-Hadithi [1,2], Adrián Gómez García [1] and Manuel Barraza-Rodríguez [3,*]

[1] Department of Electrical, Electronics, Control Engineering and Applied Physics, Higher Technical School of Industrial Design and Engineering, Universidad Politécnica de Madrid, 28012 Madrid, Spain; basil.alhadithi@upm.es (B.M.A.-H.); adrian.gomezg@upm.es (A.G.G.)
[2] Centre for Automation and Robotics (UPM-CSIC), Universidad Politécnica de Madrid, 28006 Madrid, Spain
[3] Departamento de Ingeniería Eléctrica-Electrónica, Universidad de Tarapacá, Arica 1010069, Chile
* Correspondence: mabarrazar@academicos.uta.cl

Abstract

The use of metaheuristic algorithms has become very popular in recent years. One of their many applications is in tuning different controllers, especially those that would require extensive effort to tune manually. In some cases, due to the complexity of the controller, it is almost impossible to adjust the parameters manually. In this work, a fractional-order PID controller (FOPID) is used to control an inverted pendulum. This is a non-linear, multi-variable, and underactuated system that requires advanced techniques to achieve robust control. Three different algorithms have been implemented to compare their performance in solving this problem. The algorithms used are a Genetic Algorithm (GA), an Artificial Bee Colony algorithm (ABC), and Black Widow Optimization (BWO). The controllers have been evaluated through extensive tests on a physical model of the inverted pendulum, proving the validity of the designs. Furthermore, after running the algorithms several times, optimal configurations for the parameters of each algorithm have been obtained.

Keywords: metaheuristic algorithm; inverted pendulum; genetic algorithm; black widow optimization; fractional-order PID; intelligent control systems

MSC: 93C10; 93C40; 93B52; 26A33; 90C26; 90C59; 70Q05

1. Introduction

PID controllers are commonly used in industry because they are easy to implement and offer simple parameter-tuning methods, such as empirical approaches like Ziegler–Nichols tuning. However, for more complex systems, fractional-order $PI^{\lambda}D^{\mu}$ controllers (FOPIDs) can be used. These improve the performance of PID controllers due to the two additional degrees of freedom they offer. Nonetheless, tuning five parameters for each controller is required, which is not feasible through analytical methods (which only allow, not without added difficulties, tuning of FOPI controllers), unlike conventional PIDs, which can be adjusted analytically. Rule-based methods such as Ziegler–Nichols can be applied with limitations. Lately, the most popular tuning methods for these controllers are objective minimization methods, based on the reduction of different index values. In that regard, metaheuristic algorithms have become quite popular as they are efficient and easier to implement than other existing approaches [1].

Therefore, metaheuristic algorithms are employed to tune the parameters of FOPID controllers. The first metaheuristic algorithm was proposed in 1975 by J.H. Holland [2],

Check for updates

Academic Editor: Nataliya I. Kalashnykova

Received: 6 August 2026
Revised: 18 September 2026
Accepted: 22 September 2026
Published: 28 October 2026

Copyright: © 2026 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.

and since then numerous variants have been proposed. They offer low computational cost and are capable of finding optimal solutions even when the search space is very large.

Metaheuristic algorithms are nowadays a very popular computation technique for complex problem solving. According to the Web of Science database, since 2021, more than 2000 works about metaheuristic algorithms have been published each year, reaching more than 3800 in 2025. This allows choosing the most suitable algorithm for a given problem from a wide variety of algorithms, as each of them offers better performance in certain areas.

However, some authors have criticized the number of new publications. In [3], 111 publications were analyzed to find that 65% of them were improvements of existing algorithms. This is the case for the intelligent water drop algorithm, which is claimed to be a specific case of Ant-Colony Optimization (ACO) [4]. The same also occurs regarding the Harmony Search or Black Hole Search algorithms [4,5]. In any case, with this many new algorithms (either original or improved versions of existing ones), it is necessary to check their performance and utility in real scenarios rather than only on mathematical test-benches.

Many new algorithms appear each year—38 yearly in the last decade according to [6]—yet only a handful are responsible for the majority of citations in scientific publications. Using the number of citations in Google Scholar as a reference, the most popular algorithm is Particle Search Optimization (PSO), with 75,000 citations, followed by the GA. In terms of tendency over the last years, PSO remains the most popular one during the last years, with a slightly decreasing interest in the GA and an increasing interest in the Grey Wolf Optimizer (GWO) [6].

The case of PSO is very interesting. It has been one of the most popular algorithms for some time, and has led to a great number of engineering applications. PSO has been used in various applications from wireless communications to image processing and electrical power systems. There are not only multiple applications but also multiple variants of the algorithm. One of the most common ones is the use of inertia weights [7]. However, many other variations have been proposed [8–10] as well as hybridization with other existing algorithms, such as the GA [11], the ACO [12], or the differential evolution (DE) algorithm [13].

PSO is a good example of how there is a current tendency to modify existing algorithms with the aim of improving their performance. That also means that many metaheuristic algorithms can be classified into a limited number of groups that share similar underlying mechanisms, as analyzed later.

In this work, three metaheuristic algorithms (GA, ABC, and BWO) are used to design three FOPID controllers, one per algorithm.

A popular system used to test the effectiveness of different control techniques is the inverted pendulum. This is because it is a challenging system to control due to its inherent instability, as well as being a non-linear, underactuated, and multivariable system.

In the related scientific literature, there are several examples of the application of metaheuristic algorithms to FOPID tuning. These algorithms have been applied to many of the control techniques currently in use, with multiple case studies that demonstrate their effectiveness. Also, it is interesting to study the behavior of each algorithm for the solution of each problem, as, according to the No Free Lunch Theorem, there is no universally better metaheuristic algorithm [14].

The combination of using metaheuristic algorithms to tune a FOPID controller and applying it to an inverted pendulum is not commonly found in the literature. Eight publications up to mid-2026 have been identified in which a similar approach to the one mentioned previously has been used. The metaheuristic algorithms used in these studies are diverse. Apart from this, few experiments in the existing literature have tested the robustness of the

controllers on real systems. The following publications are closely related to certain aspects of the research carried out in this paper.

In [15], the GA and PSO were used to adjust the controller. Tests were conducted in both simulations and on the real system, although an exhaustive study of the system's behavior under disturbances was not performed. This publication also provides details about the parameters used in the execution of the algorithms.

In [16], the Sine–Cosine Algorithm (SCA) was used to tune the FOPID controller; however, few tests were carried out under disturbances on a real system. This research involved significant work to accurately identify the system, achieving a model that closely reflects reality. This is particularly important to ensure that the results obtained apply to real-world systems.

In [17], the GA with real encoding was used (solutions can be encoded in different ways; one common approach is binary encoding). Additionally, PSO and BFO were employed to tune the FOPID controller. A stability analysis of the system was conducted from a phase margin perspective. Various disturbances were also introduced during the tests. It is worth mentioning that, in addition to noise disturbances and reference changes, the length of the pendulum's vertical bar and the mass of the pendulum were altered as well.

In [18], dPSO (dynamic Particle Swarm Optimization) was used to tune the FOPID controller. An LQR controller and an integer-order PID controller were also implemented. This publication also presented results from a real system model. To test the system's robustness, the pendulum was manually perturbed.

In [19], a FOPID and a PID controller were used for the control of a two-wheeled inverted pendulum. Both controllers were tuned with metaheuristic algorithms in order to compare their performance. The metaheuristic algorithms employed were ABC, PSO, GWO, and Cuckoo Search (CS). It was determined that the best algorithm was ABC, although the configuration for each algorithm was not presented. It is also worth mentioning the research conducted in this publication regarding the objective function. The controllers were designed with different objective functions intended to achieve various system responses (for example, reducing settling time, overshoot, or angle and position deviations from the set point). This demonstrates that the definition of performance indices is key in achieving the desired results. Therefore, the objective function can and must be designed accordingly.

In [20], a similar approach to the one followed in [15] is used. The FOPID is designed with three different metaheuristic algorithms: the Starfish Optimization Algorithm (SFOA), Harris Hawks Optimization (HHO), and GWO. Afterwards, the transient behavior was studied. Disturbances were considered as deviations of the mass used in the model, up to 15%. However, disturbances were applied in the real system while it was in the inverted position to test the robustness of the design.

The authors of [21] also consider the applicability of metaheuristic algorithms to the case study of a FOPID for the control of a cart-inverted pendulum. In this case, the approach used was more focused on the combination of the Slime Mould Algorithm (SMA) and the Artificial Hummingbird Algorithm (AHA), and no real tests were conducted. However, in the simulations, the robustness of the system was tested under noise conditions.

Due to the high interest in this research field, several recent works have focused on the application of metaheuristic algorithms to the tuning of FOPID controllers. In [22], a cart-inverted pendulum system is controlled with a FOPID and a Fractional-Order PI (FOPI), both tuned with multiple algorithms such as the GA, PSO, and the Levenberg–Marquardt algorithm (LMA). Another cart-inverted pendulum system is controlled in [23] with a FOPID tuned using a different metaheuristic algorithm, the Teaching–Learning-Based Optimization (TLBO). This publication provides an exhaustive analysis of the system's robustness. In [24], PSO is applied to tune a fractional-order integral regulator and a

fractional-order PI regulator. This work, applied to an inverted pendulum on a cart system, includes stability analysis and several robustness tests. Although not explicitly an inverted pendulum, [25] addresses two-wheeled self-balancing robots, which work under the same principles as an inverted pendulum. This study uses several metaheuristic algorithms to tune the FOPID controller for system stabilization. Some popular algorithms employed are GWO and PSO, alongside lesser-known ones such as Beluga Whale Optimization, Dung Beetle Optimization (DBO), and Multi-strategy Improved Beluga Whale Optimization (MSBWO).

In another context, recent publications relate to the power sector, where control engineering is becoming increasingly important for managing power converters that connect renewable energy sources to the power grid. For example, in [26], BWO is used, demonstrating that this very recent algorithm has already been applied in various fields. This publication also presents another algorithm: Egret Swarm Optimization. Another example of metaheuristic algorithms applied to the electrical sector is [27], where BWO is used to control distributed power-flow controllers that improve power quality. In this work, the performance of BWO is compared with two other popular algorithms.

As can be seen, many metaheuristic algorithms have been proposed since the GA was first used. Even now, new metaheuristic algorithms are proposed. BWO, first introduced in 2020 by V. Hayyolalam and A. Kazem [28], has recently been applied to the solution of different problems, such as in the manufacturing industry for optimization [29], the configuration of neural networks [30], or for damage assessment [31].

It must be noted that improvements to existing metaheuristic algorithms are frequently proposed to improve performance. Such is the case in [31], where improvements to BWO are proposed, even though it is a very novel algorithm.

However, BWO has not been applied yet to the tuning of a FOPID for the control of an inverted pendulum, which is a challenging case for testing the capabilities of the algorithm.

This paper provides insight into the whole process for using metaheuristic algorithms for the tuning of a FOPID applied to an inverted pendulum. The configurations of the algorithms for solving this specific problem have been analyzed to provide recommendations on the more optimal adjustments of the algorithms. This paper also proves the applicability of BWO to the FOPID tuning problem. In Section 2, the fundamentals of the metaheuristic algorithms are described, and the GA, ABC algorithm, and BWO are described one by one in detail. Also, the kind of performance index suitable to this kind of problem is discussed. Afterwards, the inverted pendulum is described along with the equations for building a model of the system. In Section 3, the results are presented. First, the different configurations of the algorithms are tested. Then, the best configuration is used to obtain a controller that can be implemented for the control of the real system. Finally, three controllers (one designed with each algorithm) are tested in the real system. Their robustness is tested by applying multiple disturbances. The maximum levels of disturbance are obtained to check the robustness of the designed controllers. The typical ranges of values for each gain of the FOPID controller are also shown. The results and limitations of the study are discussed in Section 4. Finally, in Section 5, the conclusions are presented.

2. Methods

This section provides an introduction to the fundamentals of the FOPID controller, the three metaheuristic algorithms used, and the inverted pendulum.

2.1. Fundamentals of Metaheuristic Algorithms

Metaheuristic algorithms are used to find optimal solutions (or at least optimized solutions) for problems with multiple objectives, wide search spaces, and non-linearities,

or high-dimensional problems [32]. In many cases, the goal is not to find a perfect solution, but rather a sufficiently optimal one.

Since the genetic algorithm emerged in 1975 [33], researchers have developed numerous metaheuristic algorithms. Many are inspired by natural phenomena, such as the GA itself, while others are based on the behavior of specific animal species, including the ABC and BWO algorithms.

There are multiple classifications regarding the strategies, and, therefore, the inspiration, of metaheuristic algorithms. Generally, they are based on the behavior of certain animal species, the behavior of collectives, physics laws, or other naturally occurring phenomena. For example, in [34], metaheuristic algorithms are classified as nature-inspired, human-behavior-based, game-based, and population-based metaphor-free. The nature-inspired algorithms are the most common ones, including those based on biology and the laws of physics. The ones inspired by biology are based on evolution, populations, and swarms. The algorithms presented later on in this work are of biological inspiration. The GA and BWO, as will be latter analyzed, are based on evolution as a means to search and exploit. On the other hand, the ABC algorithm is based on the behavior of a population. Apart from nature-inspired algorithms, *human-inspired* algorithms are inspired by human social interactions, while *game-based* ones are inspired by the tactics and strategy of games to achieve solution by competition. Finally, population-based metaphor-free algorithms are based on mathematical operators to explore and exploit the search space [34].

As mentioned, no single taxonomy covers all metaheuristic algorithms. For example, [35] presents a very different classification. Metaheuristic algorithms are divided into trajectory-based, evolutionary, and population-based algorithms. The trajectory-based algorithms are based on exploiting a single solution. The evolutionary algorithms include those based on genetics and evolution, such as the GA and BWO. Finally, this classification includes population-based algorithms, such as swarm intelligence, bio-inspired algorithms, and physics- and chemistry-based algorithms. In this case, the ABC algorithm analyzed later in this work would be considered a population-based algorithm, with traits of both a bio-inspired algorithm and a swarm intelligence one.

However, metaheuristic algorithms fall into multiple taxonomies. Studying them allows us to identify the underlying mechanisms and behaviors of each algorithm, having a direct relation with the search and exploitation strategies used. In [36], a comprehensive study of the taxonomy of nature-inspired metaheuristic algorithms is presented. As the range of strategies is limited, it is found that there are certain algorithms upon which the vast majority of algorithms are based, specifically, the GA, ABC, PSO, DE, and ACO. This is the case for the GA and BWO, the latter being an improvement on the former. Studying this kind of classification provides a better understanding of the strategies underlying the algorithms. It is also the case that the black hole search is based on PSO [5], and the intelligent water drop algorithm [4] is based on ACO.

These algorithms work by progressively minimizing the value of a cost function. To achieve this, metaheuristic algorithms perform multiple tests to improve upon the best solution found so far. These trials can be conducted on either the real system or through model-based simulations. The former avoids modeling errors but can require significantly more time to perform all necessary trials. In contrast, simulations generally run much faster. For example, when simulating an inverted pendulum in Simulink, hundreds of trials can be conducted per minute. In this work, the simulations (and results presented) correspond to 10-second runs. If the simulations were performed on the real system, only two or three trials per minute could be carried out.

The GA is widely used and has proven its effectiveness in numerous applications, such as facility layout planning [37,38], operations planning and management [39], image

and video processing [40–42], and epidemiological predictions during the SARS-CoV-2 pandemic [43]. On the other hand, the ABC algorithm is worth studying, as it follows a structure different from the other two implemented algorithms. Lastly, BWO is a more recent development that, although it shares similarities with the GA, offers an interesting performance comparison. In a way, BWO could be considered an evolution of the GA.

Each of these algorithms is explained in detail in the following sections, with a flowchart provided for each.

An important aspect of every metaheuristic algorithm is the definition of termination criteria. Since no perfect solution may exist for the types of problems metaheuristic algorithms typically address, it is crucial to define when the algorithm should stop iterating. In this paper, three termination criteria are used for the implemented algorithms:

- Maximum index value. Reducing the index value beyond a certain point may be unnecessary. Therefore, when the algorithm reaches this "maximum index value", it stops execution.
- Maximum number of generations. Since convergence is not guaranteed, a generation limit is set to abort execution. This is equivalent to an execution time limit. Both termination criteria serve the same purpose.
- Stall condition. Usually, there comes a point when the algorithm continues running but achieves little improvement in the performance index. Continuing execution may be a waste of resources, so it is best to stop the execution. However, the algorithm might still slightly improve the solution in each generation, so a "no change" tolerance is introduced. This means that although the performance index may change, if these changes are smaller than the tolerance, the algorithm is considered to be stalled. This condition does not apply to consecutive generations, as the performance index often remains unchanged for several generations. Instead, there must be no significant changes for multiple generations.

As soon as one of the conditions above is reached, the execution terminates. Therefore, the three conditions are completely independent. Depending on the algorithm used and its configuration, one of the three conditions may be more likely to end the execution. For example, when the execution is successful and good results are achieved quickly, the "maximum value index" condition usually terminates the execution. By contrast, if the algorithm improves the results slowly, the maximum number of generations usually terminates the execution.

2.1.1. Genetic Algorithm

The basic working principle of the genetic algorithm is to generate an initial random population, evaluate each member, select the best individuals, and perform genetic operations on them, namely recombination and mutation. There are different ways to implement these algorithms, particularly regarding the selection process, as long as the principle of selecting the best—the "survival of the fittest"—is maintained [44].

One of the main advantages of these algorithms, besides not requiring extensive knowledge of the system to be controlled, is that they tend to converge relatively easily and often avoid local minima. This is because the search begins from multiple independent points and then evolves in parallel [45].

Figure 1 shows a flowchart that illustrates the structure of the genetic algorithm. However, there are multiple ways to implement this algorithm, as different techniques for encoding, selection, recombination, and mutation exist. Various possibilities are outlined in [46].

In this work, the algorithm was implemented with real encoding, where each individual is represented by a vector of 10 decimal values, corresponding to the 10 gains to be tuned for the two FOPIDs used. For the selection process, all individuals were ranked according

to their cost function values, and the n individuals with the lowest values were chosen (as defined here, the lower the cost function value, the better the system's response). For the crossover operation, pairs of individuals were selected and combined using the following process.

For each pair (x_1, x_2), one random value (α) between 0 and 1 is generated for each dimension of the problem; for example, 10 α values are generated when tuning the FOPIDs of the inverted pendulum. Once generated, the two individuals of the pair are combined (variable by variable) using the following equation to produce a new individual (y):

$$y = x_1\alpha + (1 - \alpha)x_2. \tag{1}$$

For mutation, each value was randomly altered within limits defined by the mutation rate, which restricts the maximum allowable variation. The mutation is represented by a random value, δ, and is applied as follows:

$$y' = y \cdot (1 + \delta). \tag{2}$$

If the mutation limit is β, then δ is generated as a random value between $-\beta$ and β. The mutation is applied individually to each dimension, representing a completely random phenomenon, similar to what occurs in nature.

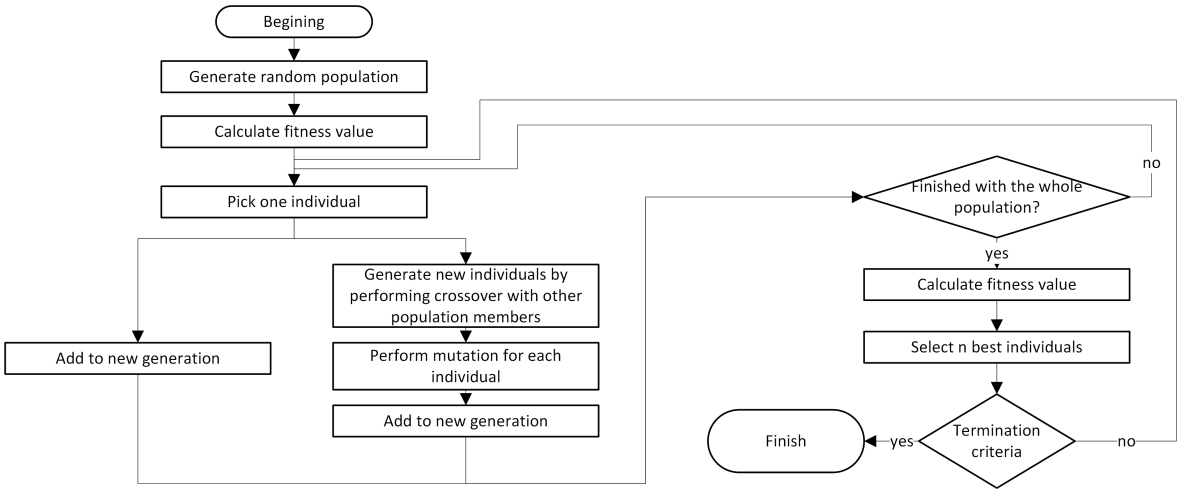


Figure 1. GA flowchart.

In this work, this algorithm has been chosen to provide a baseline for comparing with other ones, as it is one of the most popular algorithms since being proposed in the 1970s. Although the field has improved and researchers have proposed modifications to this algorithm, it remains a useful reference for comparisons.

2.1.2. Artificial Bee Colony Algorithm

The ABC algorithm was proposed in 2005 by Dervis Karaboga [47], and is inspired by the behavior of a bee colony. For this specific implementation, the ideas presented in [48] have been largely followed. The algorithm involves three entities: resources, employed bees, and unemployed bees.

The resources, which correspond to nectar sources for the bees, represent candidate solutions. In this case, as with the genetic algorithm, they are encoded as 10-element vectors, representing the configuration of the two FOPID controllers.

Employed bees are those that collect food from a resource. In the algorithm, "collecting food" refers to making small modifications to the solution, represented by the resource, to

find improvements. This is analogous to a bee searching for new nectar sources near the one it is currently exploiting.

Finally, unemployed bees can be divided into two types: scouts and onlookers. Scouts search for new nectar sources. In the algorithm's implementation, this means introducing randomly generated solutions (created within the specified initial search limits). This process replaces solutions that have been abandoned due to poor performance. On the other hand, onlooker bees wait for employed bees to communicate the locations of promising resources. Once informed, the onlookers become employed and join the workers in exploiting a certain resource, while also searching for new resources nearby.

To evaluate the resource's quality, a simulation is performed to calculate the cost function value. Based on this value, a resource will have a higher or lower probability of being exploited by a bee. When a resource is good, many bees work on it (employed bees communicate its location to the onlookers, who then join them), increasing the chances of finding further improvements. The unemployed bees are assigned to resources based on probabilities calculated from the fitness values of those resources.

In order to generate similar resources in the vicinity of an existing resource, a change rate (equivalent to the mutation rate in the GA) is applied as follows:

$$y' = y \cdot (1 + \delta). \tag{3}$$

Figure 2 presents a flowchart that describes the operation of the algorithm.

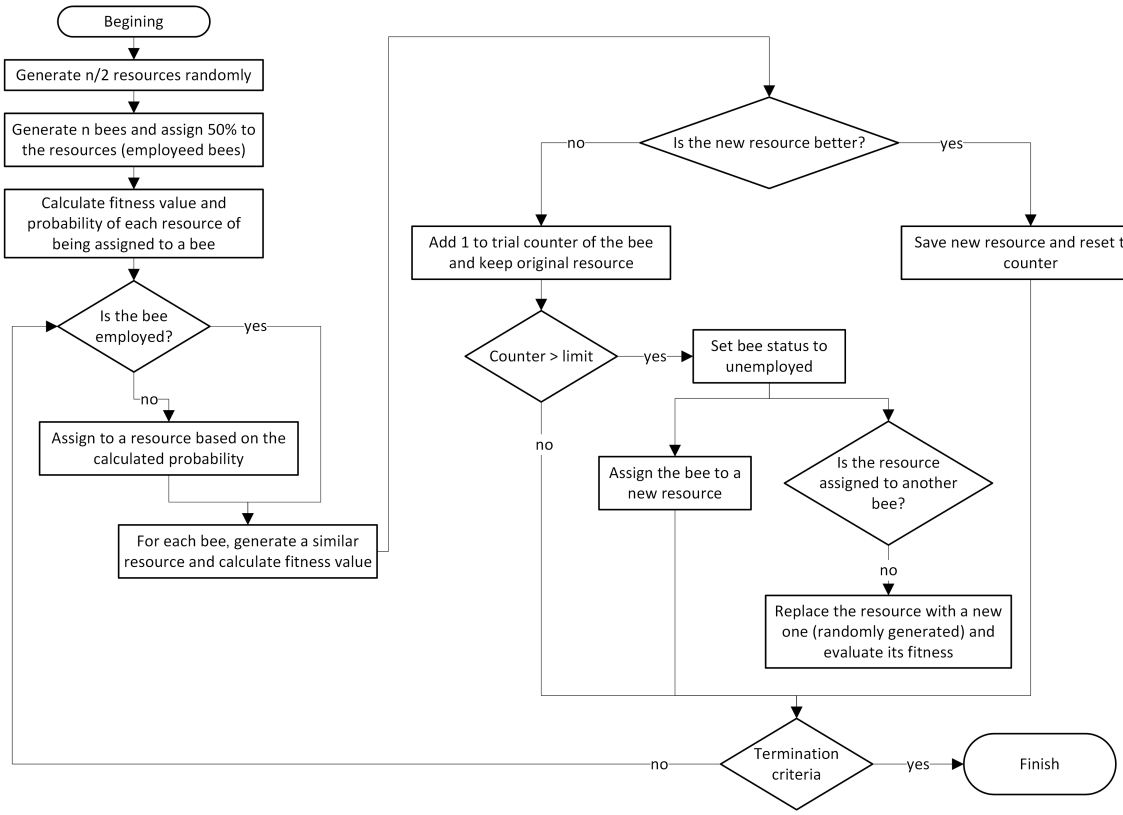


Figure 2. ABC algorithm flowchart.

This algorithm is quite different from the GA. The exploration part of the algorithm carries more weight than in the genetic algorithm compared to the exploitation, which also remains an important part for perfecting preliminary suitable solutions.

2.1.3. Black Widow Optimization

At the beginning of the execution, a population of black widow spiders is generated randomly. Each spider represents a solution to the problem. The next step is reproduction. For this purpose, couples are formed among the best individuals in the population, since these individuals have higher chances of reproducing. The number of reproducing couples (n_r) is determined by the reproduction rate parameter.

For each couple, four offspring are generated. These new spiders are created using the following equations, always incorporating a random component (α) in the combination of individuals:

$$y_1 = \alpha x_1 + (1 - \alpha)x_2; \quad (4)$$

$$y_2 = \alpha x_2 + (1 - \alpha)x_1. \quad (5)$$

The novelty of BWO, and its main difference from other algorithms, lies in what happens once the offspring are generated. The female black widow may eat the male after reproduction. Cannibalism also occurs among the offspring, ensuring the survival of the strongest individuals.

In the algorithm, the strongest individuals from both the reproducing couple and their offspring are selected together, based on the cannibalism rate and according to the fitness function values.

This process is repeated for each couple, with the surviving spiders forming the next generation.

The cycle continues through subsequent generations, as in other algorithms, until the termination criteria are met [28,49].

The flowchart of the algorithm is shown in Figure 3.

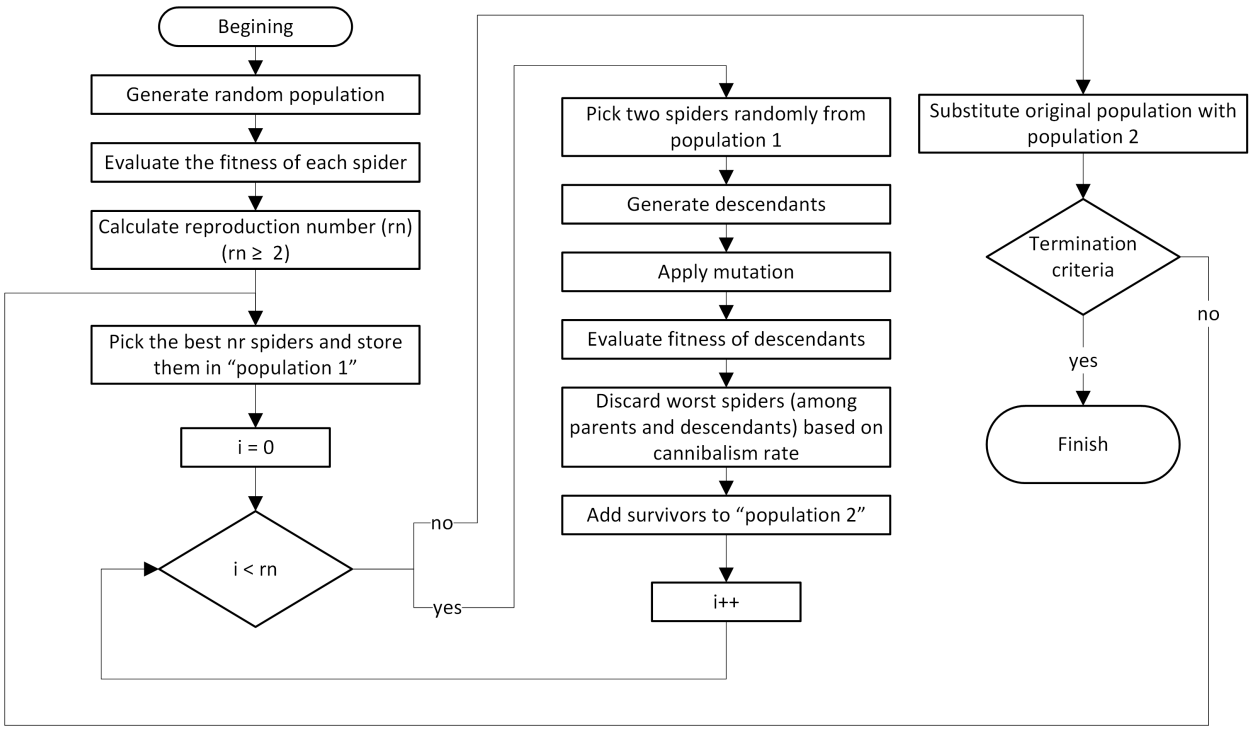


Figure 3. BWO flowchart.

BWO has been included in this work as it is one of the latest proposed algorithms and, to the best of the author's knowledge, it has not yet been used for the specific case of tuning a FOPID for controlling an inverter pendulum. Moreover, it is interesting to analyze the performance of an algorithm that shares structural similarities with the GA, as BWO can be considered, in some ways, to be an evolution of the GA.

2.2. Implementation of a FOPID Controller Tuned with Metaheuristic Algorithms

This section explains certain practical aspects of implementing a metaheuristic algorithm.

2.2.1. Performance Measurement Indices

Metaheuristic algorithms operate by minimizing a scalar performance index. The algorithm only knows how adequate the solution is based on this value, so it is of paramount importance to define it adequately, which means that it must be representative of the results obtained.

Standard evaluation metrics are integral performance indices, which compute the time integral of the error for each state or output variable independently. Although the literature offers various formulations, the most common indices are described in [33,50].

In Figure 4, the evolution of all the indices is shown, allowing one to distinguish which indices penalize each characteristic of the response. The system consists of a transfer function and a step input of 10 units. The system is underdamped and has a high overshoot.

In the graphic, the different indices have been multiplied by constants to match the order of magnitude achieved. In this way, the values for all the indices can be shown in a single graphic, allowing a simple comparison between them. The constants by which the indices have been multiplied are shown in the legend of Figure 4.

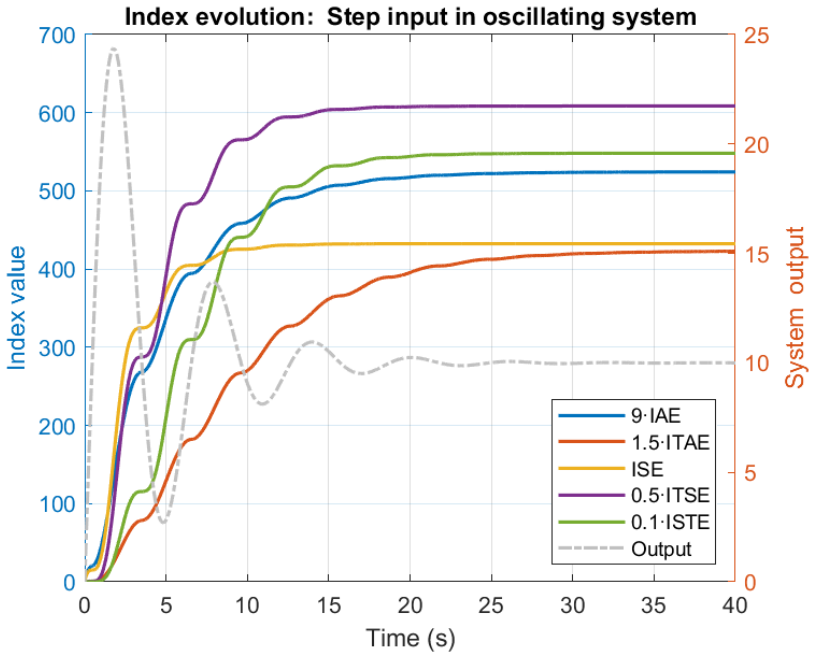


Figure 4. Index evolution comparison between different performance indices for an underdamped system with a steady-state output of 10 units.

It can be appreciated that the ITSE and the ISE do not strongly penalize the third oscillation nor the following ones. In certain systems, such as an inverted pendulum, an oscillation of the magnitude of the third one shown in the example of Figure 4 cannot be ignored, so these indices must be used consistently.

In the case of the ITAE especially, the index continues to grow, although the oscillations are very small. This is useful if a strong penalization of the oscillations (or steady-state error) is desired, especially after several seconds after the step in the input signal. Also, it partially disregards the strong oscillations at the beginning of the transient as it grows, but not at a high rate compared with its own growth with the subsequent oscillations or compared with the other indices. This is useful if high errors during the first instants of the transient are acceptable.

The other indices (IAE and ISTE) offer intermediate characteristics. The IE index is not shown, as it has a value close to zero. Moreover, it has a negative value, which complicates the comparison with the other indices.

2.2.2. Fractional-Order PID Implementation

FOPID controllers differ from classical PID controllers because they include terms with non-integer (fractional) exponents, as shown in the following equation:

$$R(s) = \frac{U(s)}{E(s)} = K_p + K_i s^{-\lambda} + K_d s^{\mu}. \tag{6}$$

To implement this in software such as MATLAB R2022b, an approximation of the fractional-order transfer function is required. For this purpose, the FOMCON library [51] is used, which performs the following operations.

First, the term raised to the fractional exponent is decomposed into the product of two terms. One term has an integer exponent, and the other has a fractional exponent between 0 and 1:

$$s^x = s^n s^{\alpha}, \tag{7}$$

where $0 < \alpha < 1$ and n is an integer.

The term s^{α} is approximated by the following fractional-order function:

$$s^{\alpha} \approx \left(\frac{d \cdot \omega_h}{b}\right)^{\alpha} \cdot \left(\frac{d \cdot s^2 + b \cdot \omega_h \cdot s}{d \cdot (1-\alpha) \cdot s^2 + b \cdot \omega_h \cdot s + d \cdot \alpha}\right) \cdot G_p, \tag{8}$$

where

$$G_p = \prod_{k=-N}^{N} \frac{s + \omega'_k}{s + \omega_k}, \tag{9}$$

$$\omega_k = \left(\frac{b \cdot \omega_h}{d}\right)^{\frac{\alpha+2k}{2N+1}}, \tag{10}$$

$$\omega'_k = \left(\frac{d \cdot \omega_b}{b}\right)^{\frac{\alpha-2k}{2N+1}}. \tag{11}$$

Here, ω_b is the lower frequency of the operating frequency range, ω_h is the upper frequency, and N is the approximation order. It is recommended to use $b = 10$ and $d = 9$.

The primary purpose of the FOMCON toolbox employed in this research is the design of FOPID controllers. The integration of its blocks into Simulink provides a more advanced and flexible framework for system control through this approach, as illustrated in Figure 5.

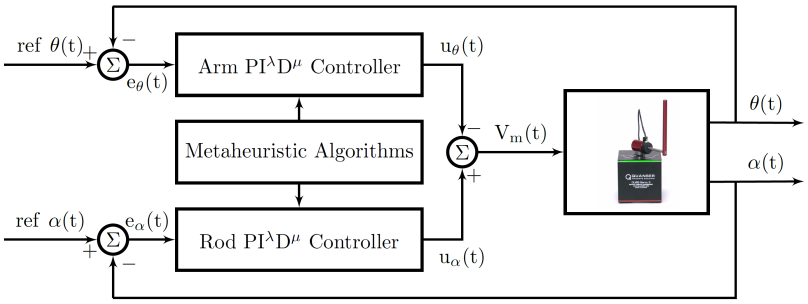


Figure 5. Rotatory inverted pendulum control architecture.

2.2.3. Fundamentals of the Rotatory Inverted Pendulum

In this paper, the controller designed with the algorithms is tested on a rotatory inverted pendulum, also known as *Furuta's pendulum*. In order to execute the algorithm, a model of the pendulum is required. The model is based on the differential equations of the pendulum, and is programmed in Simulink R2022b.

This kind of rotatory inverted pendulum was developed in the 90s, substituting the inverted pendulums built with wheeled carts, which were planar pendulums. In the rotatory inverted pendulum, an arm is driven by an electrical motor that rotates on the horizontal plane. Attached to the first arm, there is a second arm that rotates freely on the vertical plane, in a similar way to the rotation of the planar-inverted pendulum [52].

The system can be modeled by these two linear equations, in a similar way to what is achieved in [52,53], with the input being the voltage applied to the motor and the output the angles of the two rods. The model used is commercially available: the Qube-Servo 2, a rotary inverted pendulum by Quanser, as illustrated in Figure 6A. To facilitate the understanding of the dynamics associated with the rotary pendulum, the free-body diagram is presented in Figure 6B. The mathematical model of the system was derived through the Euler–Lagrange approach. Equations (12) and (13) describe the non-linear equations of motion of the system, while the electromechanical torque is given in Equation (14). All parameters, along with their nominal values, are detailed in Table 1.

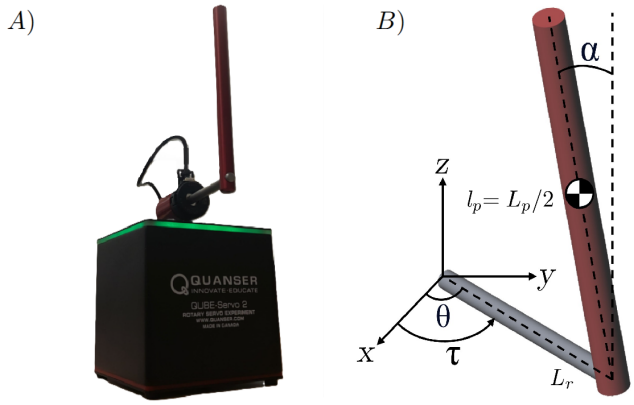


Figure 6. (**A**) QUBE-Servo 2 Quanser. (**B**) A free-body diagram for a rotary inverted pendulum.

$$\left(J_r + m_p L_r^2 + J_p \sin^2(\alpha)\right)\ddot{\theta} + 2J_p \sin(\alpha)\cos(\alpha)\dot{\alpha}\dot{\theta} - m_p L_r l_p \cos(\alpha)\ddot{\alpha} + m_p L_r l_p \sin(\alpha)\dot{\alpha}^2 = \tau - b_r\dot{\theta}; \tag{12}$$

$$J_p\ddot{\alpha} - m_p L_r l_p \cos(\alpha)\ddot{\theta} - J_p \sin(\alpha)\cos(\alpha)\dot{\theta}^2 - m_p g l_p \sin(\alpha) = -b_p\dot{\alpha}; \tag{13}$$

$$\tau = \frac{k_t}{R_m} \cdot (V_m - k_m \cdot \dot{\theta}). \tag{14}$$

Linearization around the upright position. Selecting the upper operating point ($\alpha = 0$), the following approximations are performed:

$$\cos(\alpha) \approx 1,$$
$$\sin(\alpha) \approx \alpha,$$
$$\dot{\theta}^2 = \dot{\alpha}^2 \approx 0.$$

Substituting into (12) and (13) gives the linearized model of the system:

$$(J_r + m_p L_r^2)\ddot{\theta} - m_p L_r l_p \ddot{\alpha} = \tau - b_r \dot{\theta}; \tag{15}$$

$$J_p \ddot{\alpha} - m_p L_r l_p \ddot{\theta} - m_p l_p g \alpha = -b_p \dot{\alpha}. \tag{16}$$

Solving for the angular acceleration variables $\ddot{\theta}$ and $\ddot{\alpha}$, the following results are obtained:

$$\ddot{\theta} = \frac{1}{J_T} \cdot \left(m_p^2 \cdot g \cdot L_r \cdot l_p^2 \cdot \alpha - J_p \cdot b_r \cdot \dot{\theta} - m_p \cdot L_r \cdot l_p \cdot b_p \cdot \dot{\alpha} + J_p \cdot \tau \right), \tag{17}$$

$$\ddot{\alpha} = \frac{1}{J_T} \left(\left(J_r + m_p L_r^2 \right) m_p g l_p \alpha - m_p L_r l_p b_r \dot{\theta} - \left(J_r + m_p L_r^2 \right) b_p \dot{\alpha} + m_p L_r l_p \tau \right), \tag{18}$$

$$J_T = J_p (J_r + m_p L_r^2) - m_p^2 L_r^2 l_p^2, \tag{19}$$

where

- V_m is the input voltage—for the studied case, it is limited to 12 V (either positive or negative);
- θ is the position of the horizontal rod—in this case, it is limited to a range of 180°;
- $\dot{\theta}$ is the speed of the horizontal rod (which is the same speed as the speed of the motor);
- α is the position of the vertical rod;
- $\dot{\alpha}$ is the speed of the vertical rod;
- $\ddot{\alpha}$ is the acceleration of the vertical rod;
- $\ddot{\theta}$ is the acceleration of the horizontal rod.

Table 1. Parameters of pendulum QUBE-Servo 2.

Symbol	Description	Value	Unit
J_r	Moment of inertia of the horizontal arm	$\frac{1}{3} m_r r^2$	kg m^2
J_p	Moment of inertia of the vertical rod	$\frac{1}{3} m_p L_p^2$	kg m^2
m_r	Mass of rotary arm	0.095	kg
L_r	Length of rotary arm	0.085	m
m_p	Mass of pendulum	0.024	kg
L_p	Total length of pendulum	0.129	m
l_p	Center of mass (COM) location of pendulum	$\frac{L_p}{2}$	m
R_m	Terminal resistance	8.4	Ω
k_m	Motor back-emf constant	0.042	V/(rad/s)
k_t	Torque constant	0.042	N m/A
b_r	Damping coefficient of rotary arm	10^{-3}	N m s/rad
b_p	Damping coefficient of pendulum	5×10^{-5}	N m s/rad
g	Gravitational acceleration	9.81	m s^{-2}

With these equations, a model can be programmed in order to execute the algorithms and test the controllers designed with them on simulations before testing them on the real system.

3. Results

After analyzing the fundamentals of the metaheuristic algorithms used, the FOPID controller, and the rotary inverted pendulum, this section presents the experimental tests. First, simulations are carried out to evaluate the performance of the algorithms. Then, the best results are implemented on a real system, where exhaustive tests are conducted to assess its robustness.

3.1. Configuration of the Metaheuristic Algorithms

In this section, various aspects related to the configuration of the parameters for each of the metaheuristic algorithms are discussed.

3.1.1. Selection of the Performance Index

After analyzing the general properties of different performance indices, it is necessary to evaluate their behavior to specifically assess the performance of the inverted pendulum system. The selected index must distinguish between stable and unstable system responses. Ideally, unstable systems should yield very high index values; oscillatory or slow responses should result in intermediate values; and responses that meet the desired criteria—fast settling, minimal overshoot, and low oscillations—should correspond to the lowest index values.

As seen in [19], the objective function can be defined with as many components as desired. In this work, it is considered that the main goal is to keep the system stable. Therefore, the chosen components are the performance index for the angle of the horizontal arm, the index for the angle of the vertical rod, and the energy cost (the voltage applied to the electric motor, or, in other words, the controller output).

To compare the different typical indices, the index values for the three systems shown in Figure 7 (unstable system), Figure 8 (oscillating system), and Figure 9 (ideal stable system) are presented in Table 2. For this work, an inverted pendulum on a cart is used as an example. It is important to note that there is no such thing as an absolutely ideal system. This term is used here to denote a very good response, which would be the desired target.

Table 2. Value of the indices for unstable, oscillating, and stable systems.

Type of Index	Component	Unstable	Oscillating	Stable (Ideal)
IE	Angle θ	40.93	0.01393	-1.208×10^{-3}
	Position x	−0.2741	0.04851	-5.280×10^{-2}
	Energy c	−2.111	0.9769	-8.888×10^{-2}
IAE	Angle θ	55.15	0.9419	2.372×10^{-2}
	Position x	0.8446	0.5886	6.076×10^{-2}
	Energy c	45.10	57.68	1.718
ITAE	Angle θ	244.3	4.740	1.361×10^{-2}
	Position x	2.797	2.909	5.998×10^{-2}
	Energy c	205.4	290.3	0.9523
ISE	Angle θ	631.1	0.1138	7.572×10^{-4}
	Position x	0.1802	0.04503	2.456×10^{-3}
	Energy c	251.3	390.0	4.373
ITSE	Angle θ	2404	0.5755	1.607×10^{-4}
	Position x	0.3463	0.2173	1.225×10^{-3}
	Energy c	1012	1971	0.6507

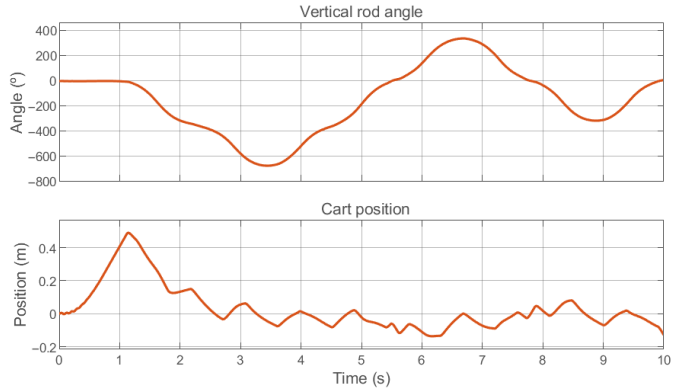


Figure 7. Unstable response of an inverted pendulum on a cart.

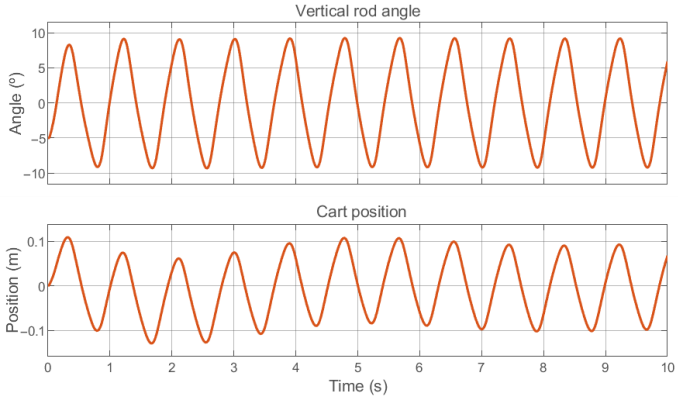


Figure 8. Oscillating response of an inverted pendulum on a cart.

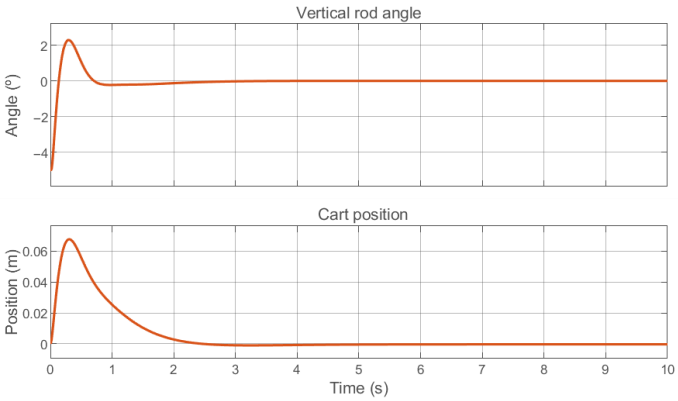


Figure 9. Stable response of an inverted pendulum on a cart.

According to the values in Table 2, the IE index is automatically ignored. The IAE exhibits similar behavior to the ITAE; however, with ITAE, the difference between the stable system and the other two is greater, making it preferable. The same occurs with the ISE and the ITSE, with the ITSE being the better option, although both show similar behavior. The indices regarding the angle of the vertical rod and horizontal rod, respectively, reach similar values. However, the term related to the energy ($ITAE_c$) has a different order of magnitude. As it is greater, it has to be multiplied to avoid canceling the effect of the other terms.

Also, it must be noted that at this stage, the objective function was intended for the stabilization of the system. Other aspects such as overshoot could be considered by adding additional terms with appropriate weights.

As a result, the global performance index is defined as follows:

$$J = ITAE_\theta + ITAE_x + 0.5 \cdot ITAE_c, \tag{20}$$

where $ITAE_\theta$ is the index related to the vertical rod angle, $ITAE_x$ is the index related to the horizontal arm angle, and $ITAE_c$ represents the energy cost.

3.1.2. Tuning of the GA

In order to study the best configuration in terms of balance between success rate and execution time, several tests were carried out.

For these tests, the same conditions were used to allow comparisons between different runs. The inverted pendulum, controlled by the FOPID, is the target system, and the performance index is calculated as shown in the previous section. The rest of the parameters are configured as shown in Table 3.

Figures 10 and 11 show the evolution of the performance index for the four tests conducted for each of the eight configurations tested. In each graphic, each line represents one execution of the algorithm. In this way, it can be compared to how each execution evolves, if it becomes stalled, or if it reaches a desirable solution.

These results indicate that a mutation rate of 0.25 is too low, and in several cases, the algorithm stalls before reaching the target index ($J = 1$). For a mutation rate of 0.75, the success rate depends on the population size. When using a 0.5 mutation rate, satisfactory results are obtained for both populations of 15 and 20 individuals. As the population size increases, so does the execution time, so it is desirable to keep the population as small as possible without compromising the success rate. It can be seen that a population of 15 individuals is sufficient to obtain good results. However, a larger population increases the probability of convergence.

Depending on the intended use of the algorithm, it may be preferable to keep the population size small (around 15 individuals) and re-run the algorithm if the first execution fails to converge. Alternatively, larger populations (20 individuals or more) can be used to increase the likelihood of convergence.

It is important to note that these results apply to the specific case of an inverted pendulum. Depending on the number of variables to be tuned, a different configuration might be more suitable.

Table 3. Execution conditions for the performance tests of the GA.

Parameter	**Value**
Termination criteria (generation limit)	50
Termination criteria (”no change” tolerance)	5 %
Termination crit. (generations without significant changes)	10
Termination criteria (maximum index value)	1.0
Index type	ITAE
Controller	FOPID (position and angle)
System for which the controller is designed	Rotatory inverted pendulum
Angle controller upper search limit $[k_p, k_i, k_d, \lambda, \mu]$	[70, 10, 10, 2, 2]
Position controller upper search limit $[k_p, k_i, k_d, \lambda, \mu]$	[50, 8, 8, 2, 2]
Angle controller lower search limit $[k_p, k_i, k_d, \lambda, \mu]$	[0, 0, 0, 0.5, 0.5]
Position controller lower search limit $[k_p, k_i, k_d, \lambda, \mu]$	[0, 0, 0, 0.5, 0.5]

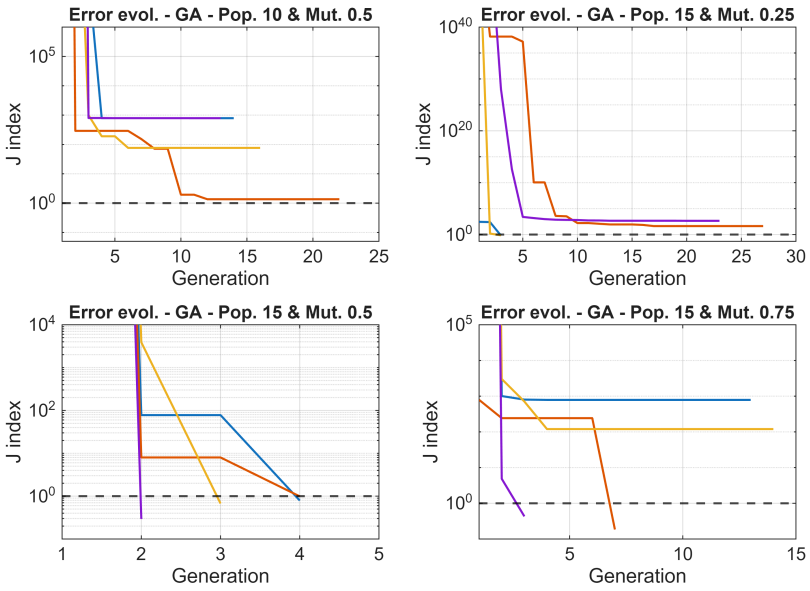


Figure 10. Index evolution for four different configurations of the GA, with populations (Pop) of 10 and 15 individuals and different mutation (Mut) rates.

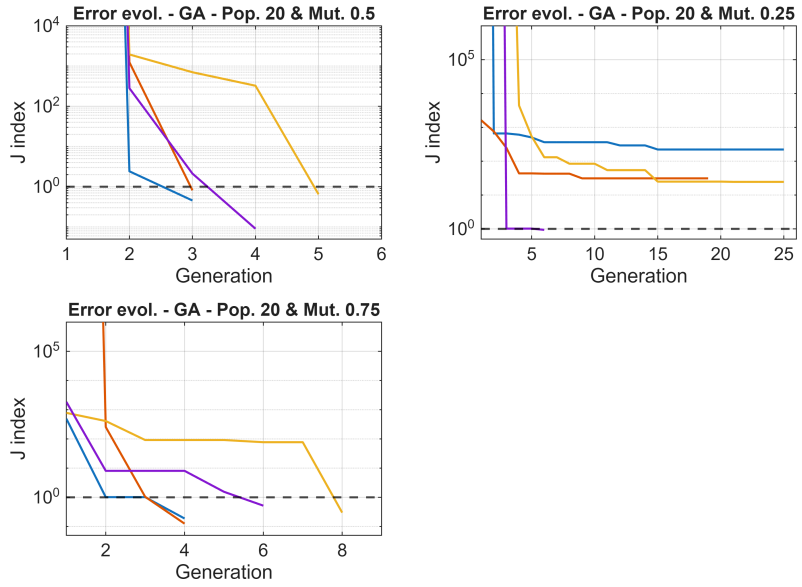


Figure 11. Index evolution for three different configurations of the GA, with a population (Pop) of 20 individuals and different mutation (Mut) rates.

3.1.3. Tuning of the ABC

Using the same configuration (for the general aspects such as termination criteria, number of CPUs, kind of index used, etc.) as for the GA, as shown in Table 3, several tests were conducted to assess the performance of the algorithm. Figures 12 and 13 show the evolution of the performance index over generations.

According to the results of Figure 12, high values for the mutation rate seem to prevent the algorithm from converging. Therefore, a change rate (mutation) of 0.5 is chosen. In Figure 13, small populations, such as those with 50 individuals, produce poor results. However, increasing the population to 150 individuals does not appear to significantly improve the convergence rate or the success rate. Thus, it can be concluded that a population of 100 individuals yields acceptable performance.

The trial limit has a greater impact. A low trial limit of three trials leads to a faster improvement in the index. In contrast, a limit of 15 trials can lead to stagnation. It should be noted that, for this specific configuration (population 100, mutation 0.5, and trial limit 15), the termination criterion was modified to avoid interrupting execution due to the stall condition. This is because the index is expected to remain unchanged for up to 15 generations in some cases.

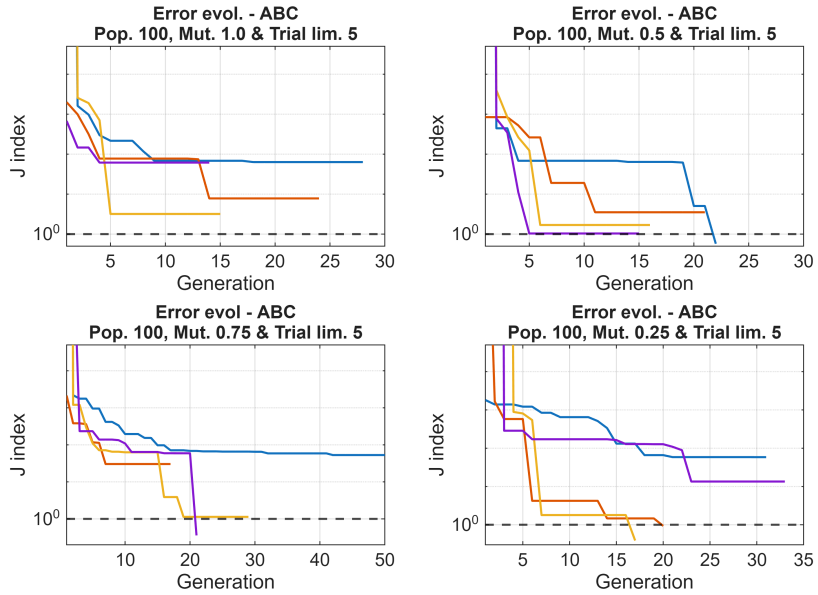


Figure 12. Index evolution for four different configurations of the ABC algorithm. Population (Pop) of 100 individuals with different mutation (Mut) rates and a trial limit (trial lim.) of five.

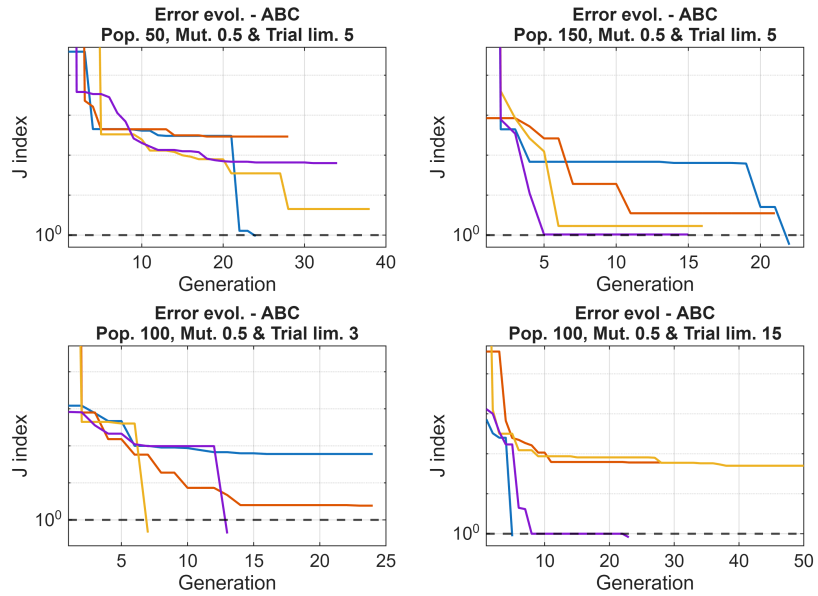


Figure 13. Index evolution for three different configurations of the ABC algorithm. Mutation (Mut) rate of 0.5 and different population (Pop) sizes and trial limits (Trial lim.).

3.1.4. Tuning of BWO

For BWO, the same configuration as in the case of the GA is used (see Table 3). The results obtained are shown in Figures 14 and 15.

In light of the results shown in Figures 14 and 15, with a population of 100 individuals, the success rate achieved was 100% for the four executions carried out with a 0.75 mutation rate, 0.4 reproduction rate, and 0.6 cannibalism rate.

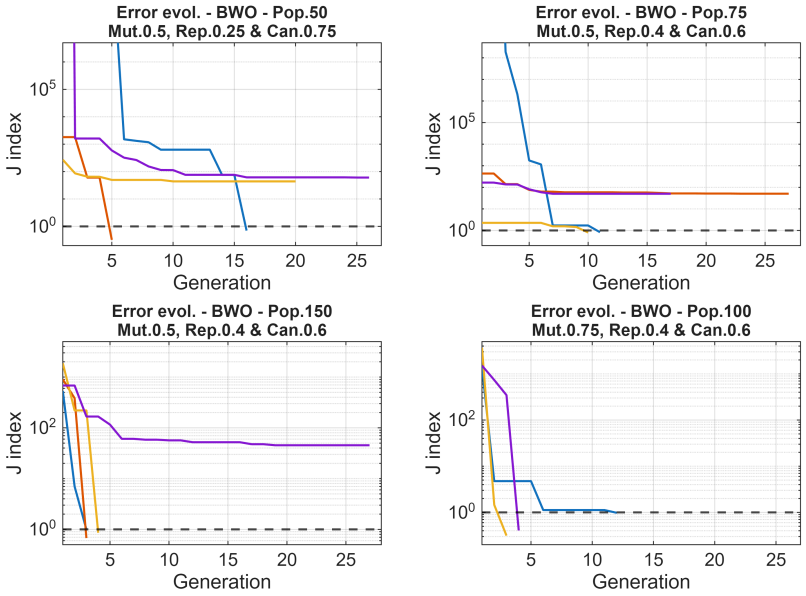


Figure 14. Index evolution for four configurations of the BWO algorithm, with different sizes of populations (Pop), and different values for the mutation (Mut), reproduction (Rep.) and cannibalism rates (Can.).

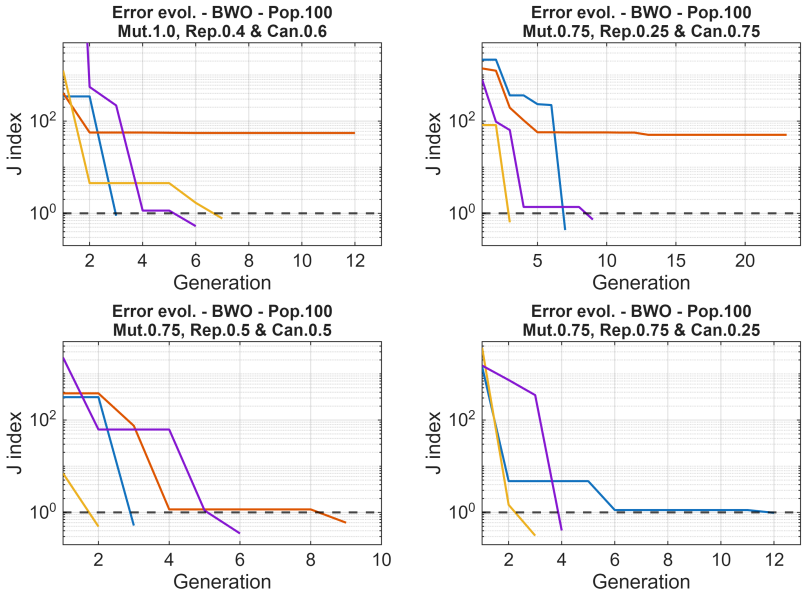


Figure 15. Index evolution for four configurations of the BWO algorithm, with populations of 100 individuals and different values for the mutation (Mut), reproduction (Rep.), and cannibalism rates (Can.).

Out of the 20 different executions with five different configurations, the success rate was 90%, which is considered acceptable. Only one configuration with a larger population was tested, and the success rate in this case was 75% (three out of four executions were successful). This does not imply that a larger population leads to worse results; with more executions, the success rate presumably would increase. However, it indicates that even with a larger population, convergence is not guaranteed. Additionally, the changes in other

configuration parameters may have influenced the results. Therefore, the trade-off between a slight improvement in the success rate and an increase in execution time must be evaluated for each specific case.

As for the mutation rate, based on the results of Figure 14, a mutation rate of 0.75 offers promising results. Thus, further tests with high mutation rates (0.75 and 1) were carried out, as shown in Figure 15. Three different combinations of reproduction and cannibalism rates were tested. Only the configuration with a reproduction rate of 0.25 and a cannibalism rate of 0.75 showed significantly worse results (one of the executions did not converge).

Therefore, the most suitable configuration is deemed to be a population of 100 individuals with a 0.75 mutation rate. As for the cannibalism and reproduction rates, further research is needed, as there are many possible parameter combinations (including mutation and population size). Nevertheless, a reproduction rate of 0.5 and a cannibalism rate of 0.5 have shown promising results.

3.1.5. Comparison Between the Three Algorithms

Having configured the parameters of all the algorithms, a comparison between them is established. It should be taken into account that, as previously stated, the configuration of the algorithms can be modified to achieve better success rates or faster execution times. Table 4 shows the configurations used for this comparison.

Table 4. Best configuration for each algorithm according to the run executions.

Algorithm	Population	Mutation Rate	Trial Limit	Reproduction Rate	Cannibalism Rate
GA	20	0.75	-	-	-
ABC	100	0.75	5	-	-
BWO	100	0.75	-	0.5	0.5

These configurations have been chosen because, as determined in the previous section, they represent an optimal trade-off between execution time and success rate.

The general configuration of the number of CPUs and the termination criteria, which are common settings for the three algorithms, is the same as that used for testing each algorithm, as shown in Table 3.

With these "optimal" configurations, 32 executions of each algorithm were performed. The results for the performance index and execution time are shown in Figure 16. The evolution of the index during 10 of the 32 executions of each algorithm is presented in Figure 17. In this figure, each line represents one execution, providing a general picture of each algorithm's behavior. Several executions of each algorithm are shown, as a single execution is not representative due to the randomness of metaheuristic algorithms.

With these results, several conclusions can be drawn. The fastest algorithm is the ABC (with an average execution time of approximately 3 min), although it has the lowest success rate. It is possible that increasing the population size could improve the success rate, but this would also increase the execution time; therefore, under the same conditions, it may not be much faster than the other two algorithms.

The BWO algorithm has an average execution time of around 8 min. However, its success rate is much higher (80%), and even when it did not reach the target index, the index value was close to the target, which was one. On the other hand, the GA had the worst execution times (around 10 min on average), although it achieved acceptable cost function values in the majority of executions.

As seen in Figure 17, several stalls can be observed in the ABC executions. Stalls are also present in the GA executions but with lower values of the cost function. In the case of BWO,

there is only one stall, and the value of the cost function had already been significantly reduced. Therefore, BWO is considered the best of the three algorithms.

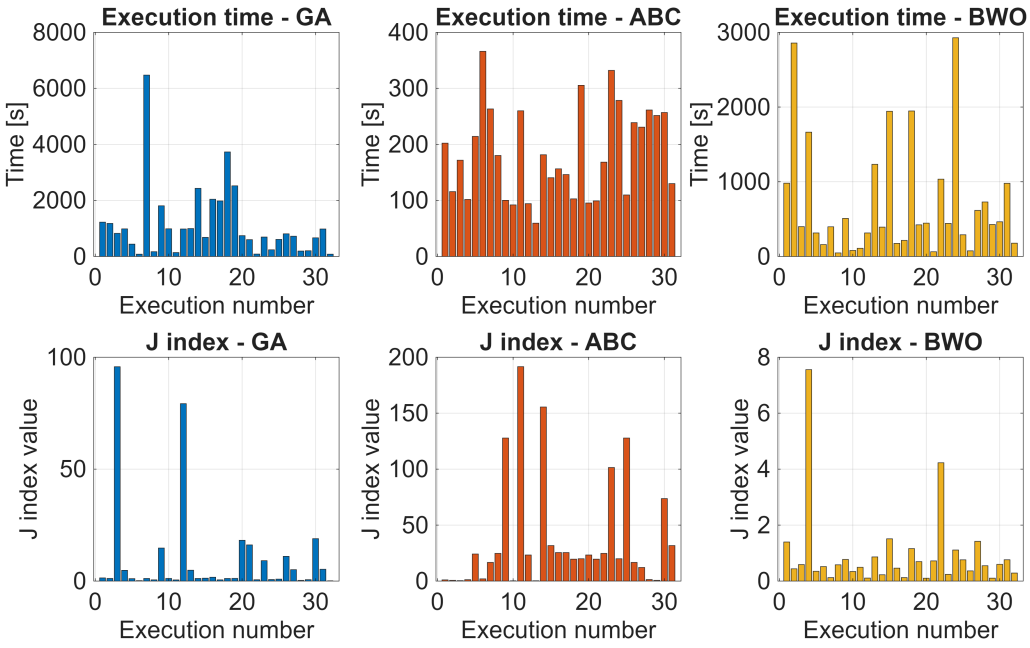


Figure 16. Execution time and final index value for ten executions of each algorithm using the "optimal" configurations from Table 4.

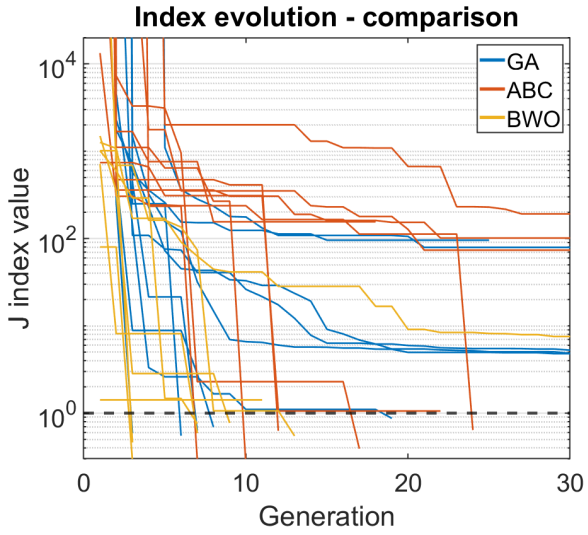


Figure 17. Index evolution for the executions shown in Figure 16.

3.2. Performance of the Designed Controllers on a Real System

After the algorithms had been run several times, the best controllers were chosen for further testing. To test their stability, different disturbances are applied in this section to check whether the controller is capable of keeping the inverted pendulum in the inverted position or not.

3.2.1. GA Performance

With the GA, the controllers shown in Table 5 were obtained.

To test the performance of the controller designed with the GA, several tests were carried out on a real system. Three kinds of disturbances were applied: noise, load, and set-point changes.

Table 5. FOPID parameters for the controllers designed with the GA.

Controller	k_p	k_i	k_d	λ	μ
Pendulum angle	40.5701	0.4158	2.7800	0.6624	1.0328
Horizontal angle	10.0549	2.8109	2.3428	0.1131	1.0388

First, noise disturbances were introduced by software. The noise disturbance consists of white noise generated with a Gaussian distribution. This noise is added to the signals of both encoders (the one that reads the horizontal angle and the one that reads the pendulum angle). The level of noise introduced is characterized by the standard deviation of the Gaussian distribution used to generate the noise, expressed in degrees. The results obtained are shown in Figure 18, where it can be seen that the regulator keeps the system stable under a noise level of $\sigma = 1.25°$.

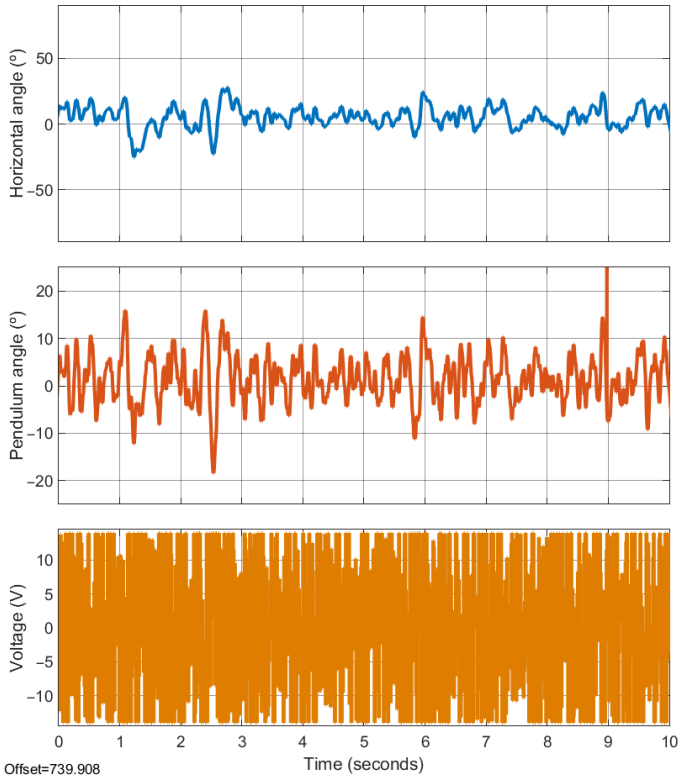


Figure 18. System response under $\sigma = 1.25°$ white noise. (Controller designed with the GA.)

The second disturbance applied is a change in the set point. This change is made abruptly, with a step input. This means the system must follow a rapid change in the set point while maintaining stability. If the set-point change was made gradually, the set point could be changed to any position. However, applying an abrupt change allows one to compare the performance of different regulators by assessing the maximum set-point change that does not destabilize the system. The largest change successfully handled by this regulator was 75°, as shown in Figure 19.

The last disturbance applied individually was a load change. This consists of adding weight to the pendulum. For this test, a 3D-printed support is attached to the vertical rod. On this support, weights are placed to carry out the tests, as illustrated in Figure 20. The results of placing 22 g (which includes the weight of both the support and the additional weights) are shown in Figure 21.

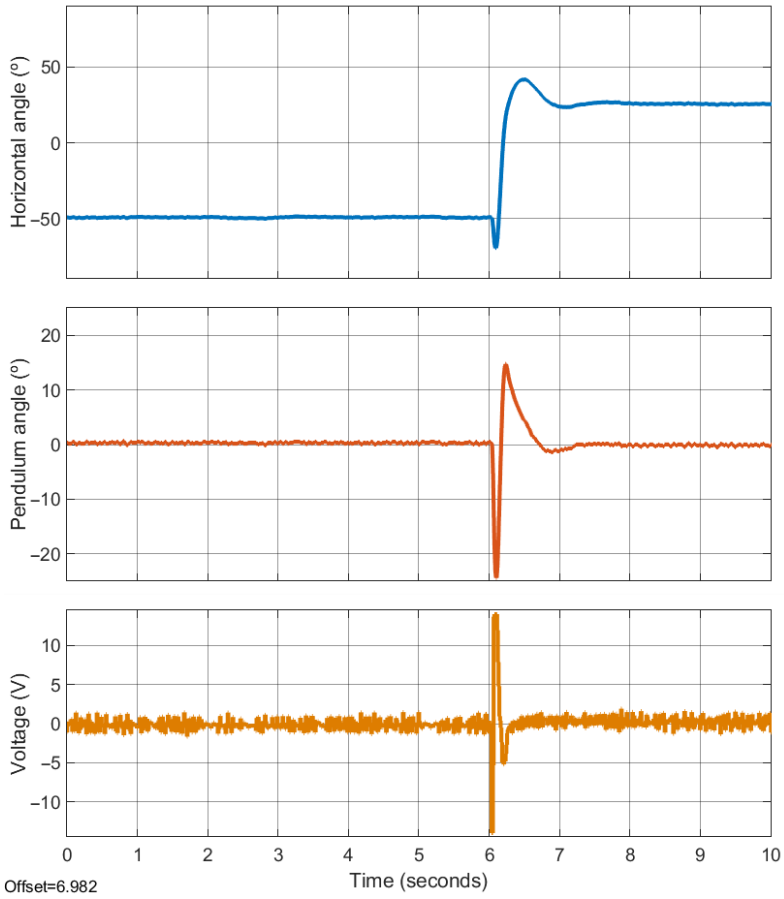


Figure 19. System response to a set-point change of 75°. (Controller designed with the GA.)



Figure 20. Photograph of the QUBE-Servo 2 Quanser in operation with the support base and weights.

Once performance with the three different disturbances was tested, combinations of multiple disturbances were evaluated. When multiple disturbances occurred simultaneously, the maximum level that the system could handle for each individual disturbance was generally lower than when applied separately. Given the infinite number of possible combinations, some significant cases are presented in Figures 22–25.

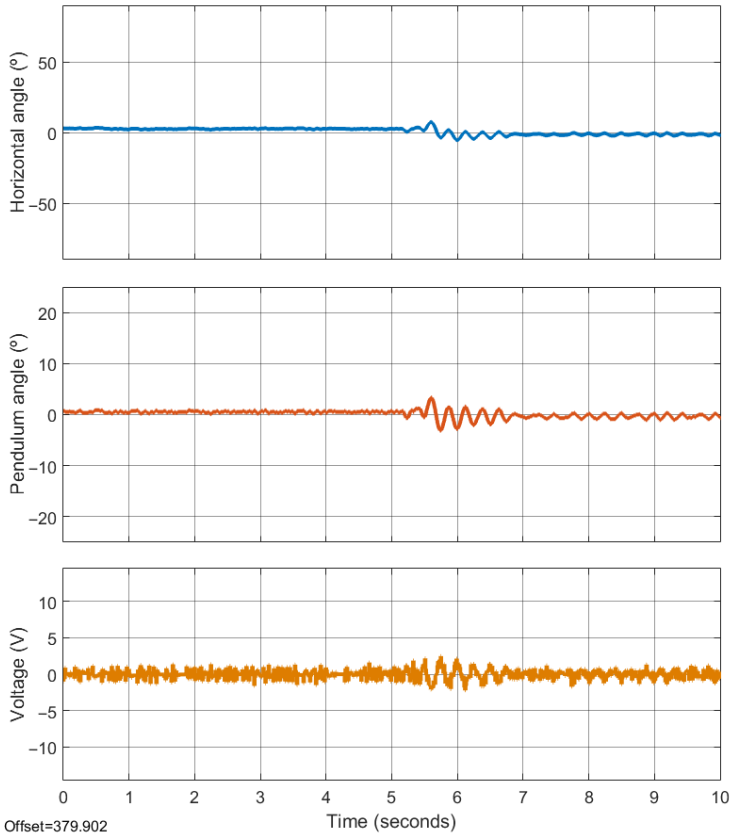


Figure 21. System response with 22 g of extra load. (Controller designed with the GA.)

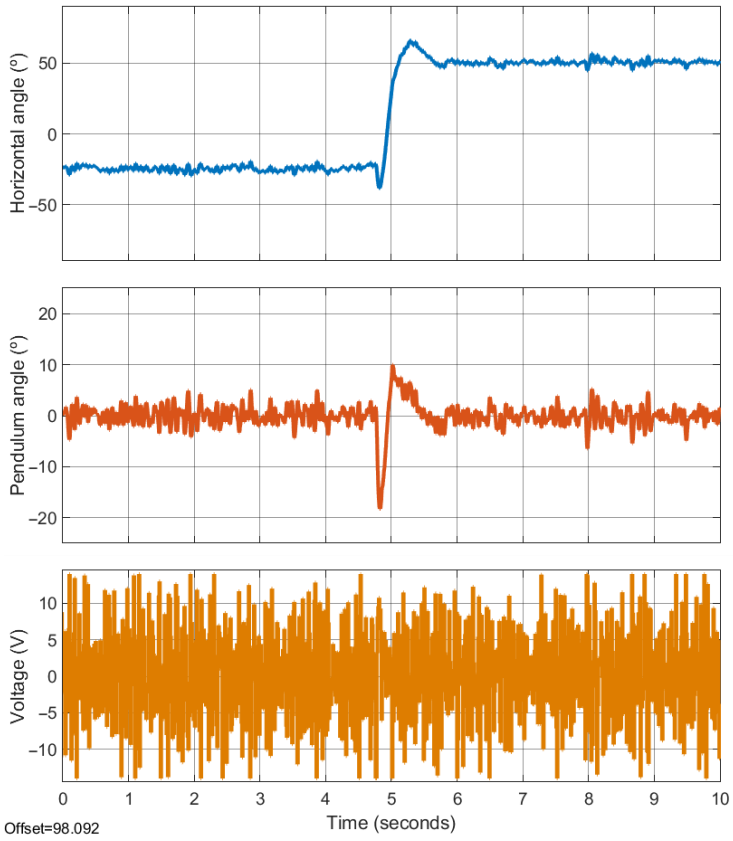


Figure 22. System response to a set-point change of 75° under $\sigma = 0.5°$ white noise. (Controller designed with the GA.)

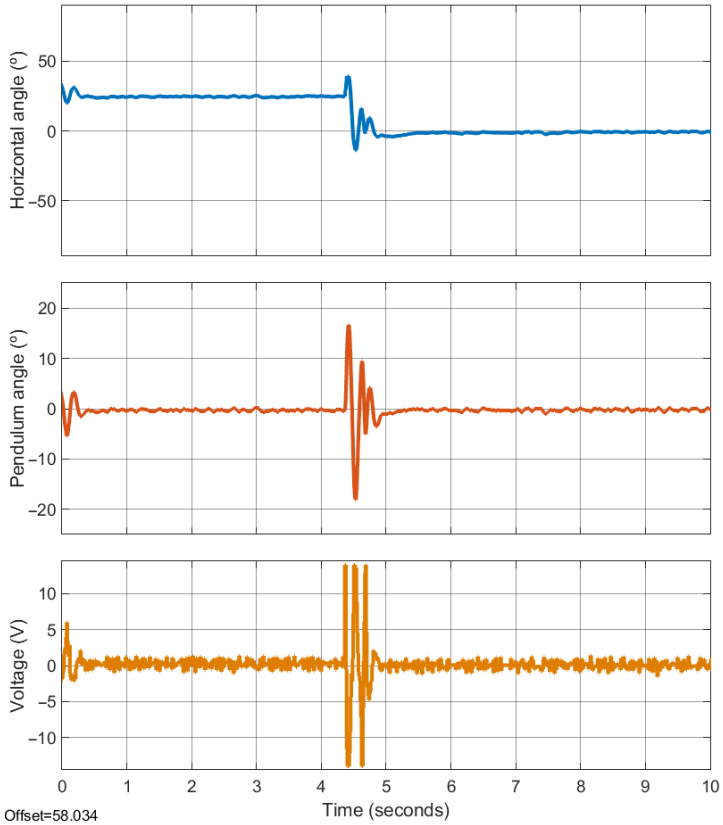


Figure 23. System response to a set-point change of 25° with 12 g of extra load. (Controller designed with the GA.)

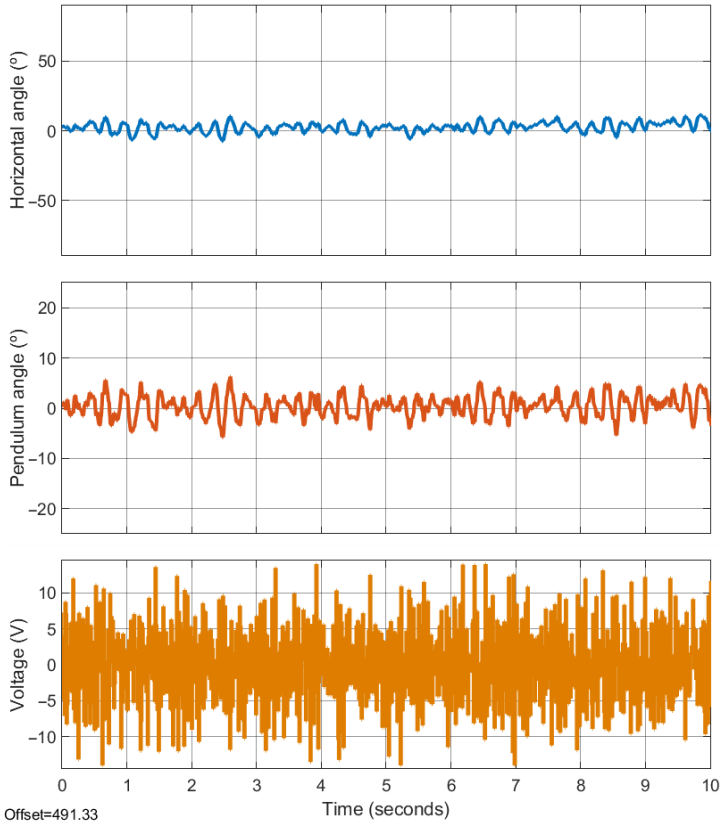


Figure 24. System response with 22 g of extra load under $\sigma = 0.5°$ of white noise. (Controller designed with the GA.)

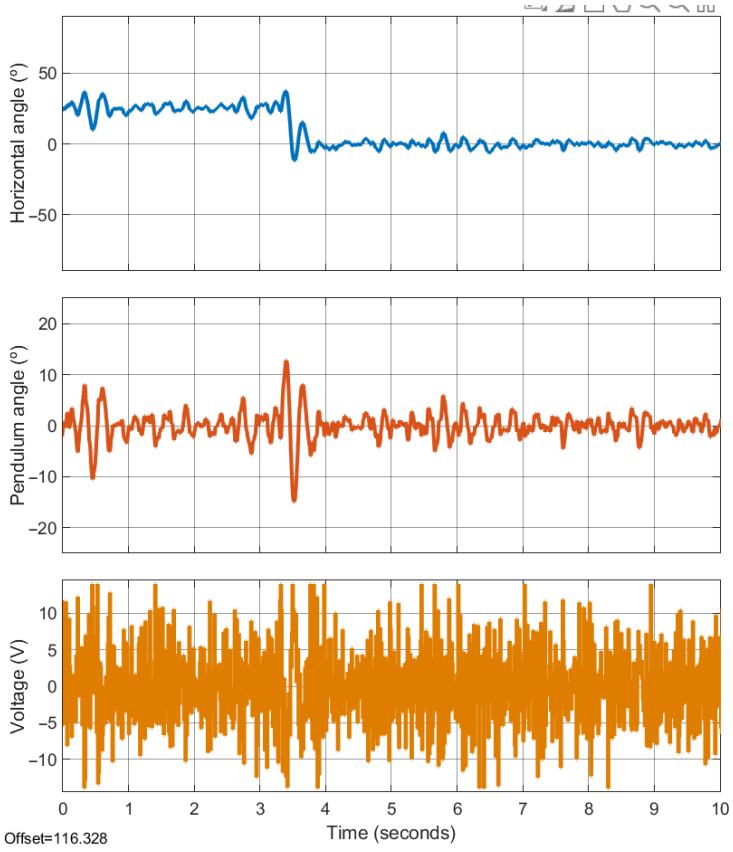


Figure 25. System response with all disturbances combined: 12 g load, $\sigma = 0.5°$, and 25° set-point change. (Controller designed with the GA.)

The results of the previous seven tests are summarized in Table 6.

Table 6. Summary of the results of the disturbance tests with the GA.

Test	Noise (°)	Set-Point Change (°)	Additional Mass (g)
#1	1.25	–	–
#2	–	75	–
#3	–	–	22
#4	0.5	75	–
#5	–	25	12
#6	0.5	–	22
#7	0.5	25	12

3.2.2. ABC Performance

With the ABC algorithm, the controllers shown in Table 7 were obtained.

Table 7. FOPID parameters for the controllers designed with the ABC algorithm.

Controller	k_p	k_i	k_d	λ	μ
Pendulum angle	63.2082	9.0499	1.3976	1.0916	1.1968
Horizontal angle	4.6069	0.8126	1.8680	0.1692	1.0070

The same experiments conducted for the GA were conducted to test the robustness of the controller designed with the ABC algorithm. In Figure 26, it is shown that the system can withstand noise up to $\sigma = 1.3°$. Figure 27 demonstrates that the system can handle a set-point change of 125°. Regarding the load change, Figure 28 presents the results of adding 22 g of extra load.

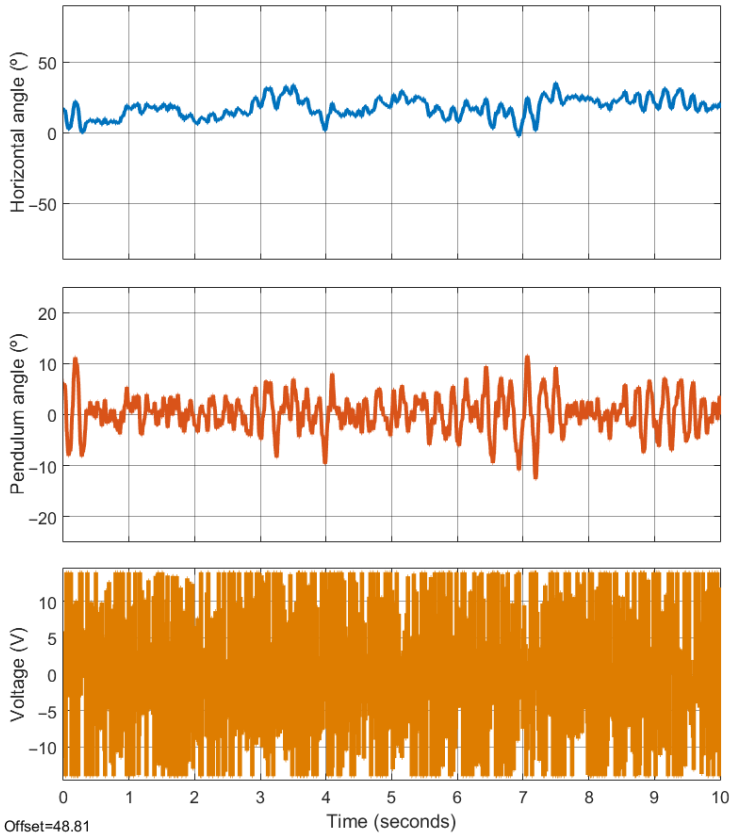


Figure 26. System response under $\sigma = 1.3^{\circ}$ white noise. (Controller designed with the ABC algorithm.)

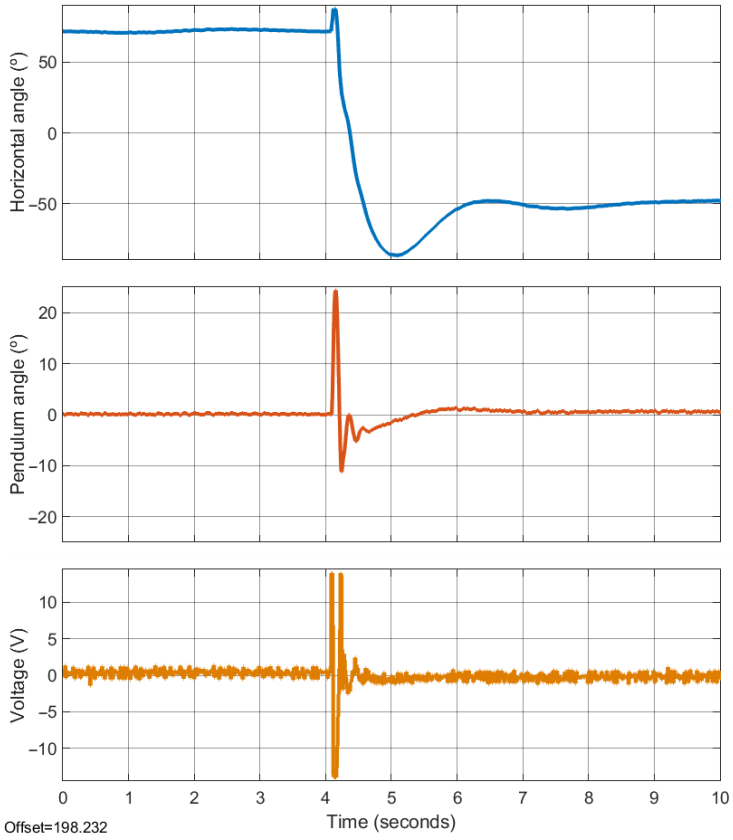


Figure 27. System response to a set-point change of 125°. (Controller designed with the ABC algorithm.)

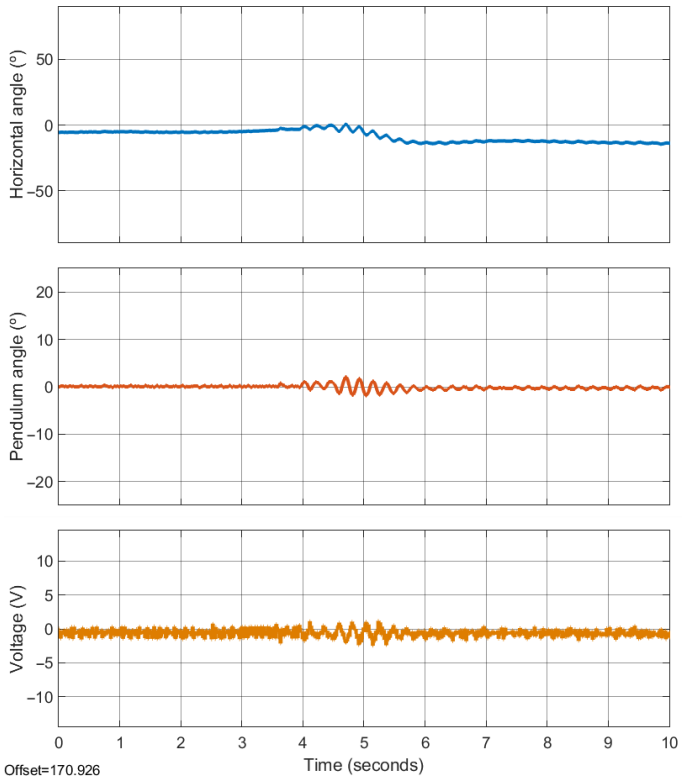


Figure 28. System response with 22 g of extra load. (Controller designed with the ABC algorithm.)

Having verified the limits that the system can withstand for each individual disturbance, combinations of disturbances were tested. For a combination of noise and a set-point change, the system could withstand $\sigma = 0.7°$ while undergoing a set-point change of 125°, as seen in Figure 29.

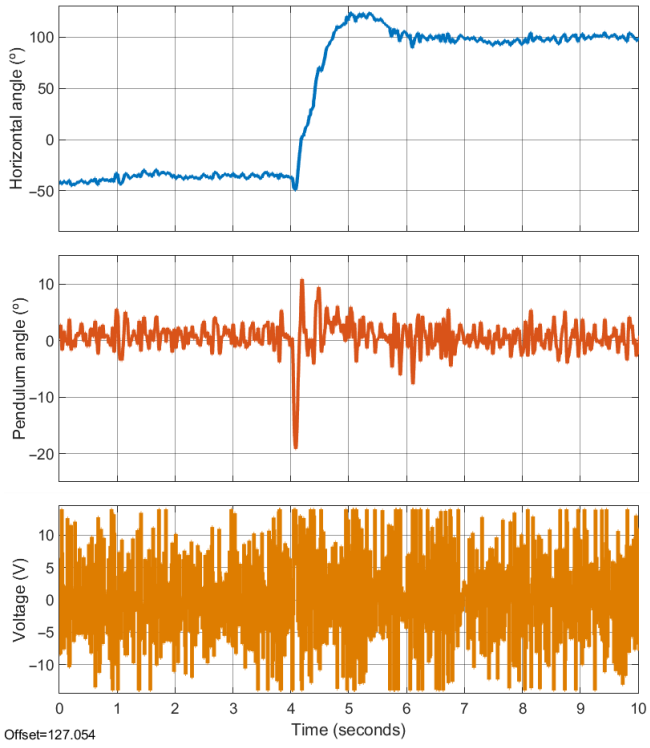


Figure 29. System response to a set-point change of 125° under $\sigma = 0.7°$ white noise. (Controller designed with the ABC algorithm.)

For a combination of load and set-point change, the results for a set-point change of 125° with an extra load of 12 g are shown in Figure 30. For the combination of load and noise, with a noise of $\sigma = 0.5°$, 22 g of extra load could be added, as seen in Figure 31.

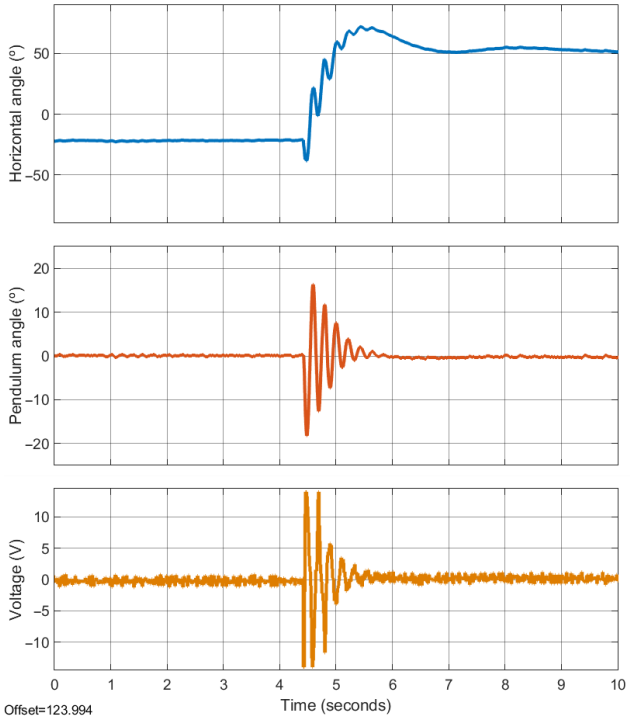


Figure 30. System response to a set-point change of 75° with 12 g of extra load. (Controller designed with the ABC algorithm.)

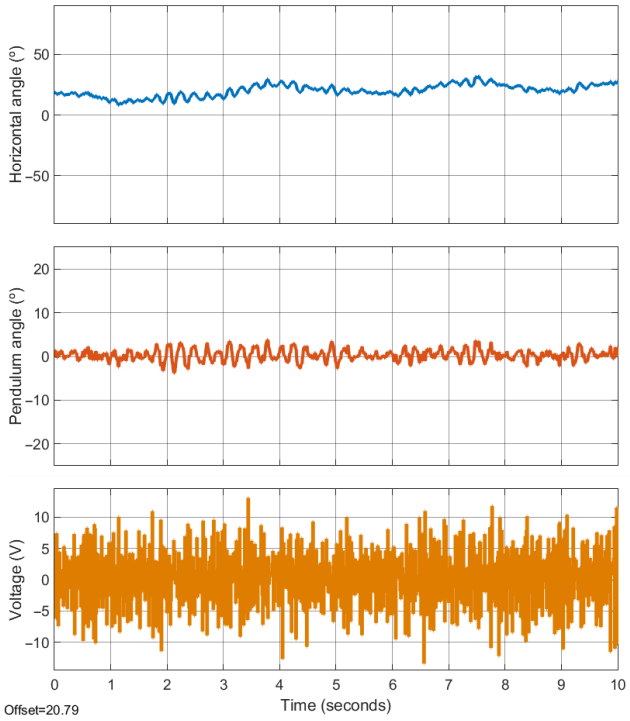


Figure 31. System response with 22 g of extra load under $\sigma = 0.5°$ of white noise. (Controller designed with the ABC algorithm.)

Finally, all the disturbances were combined, achieving 12 g of extra load, with $\sigma = 0.75°$ and a set-point change of 50° (see Figure 32).

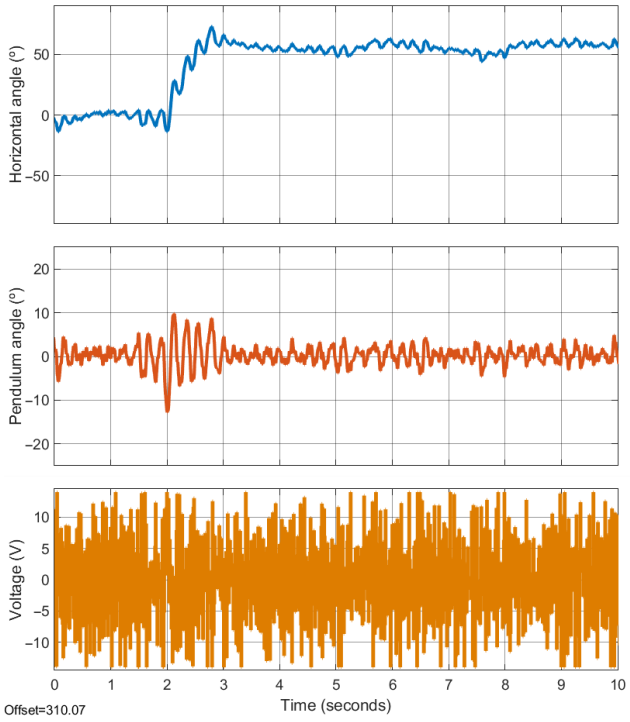


Figure 32. System response with all disturbances combined: 12 g load, $\sigma = 0.75°$, and 50° set-point change. (Controller designed with the ABC algorithm.)

The results of the previous seven tests are summarized in Table 8.

Table 8. Summary of the results of the disturbance tests with the ABC algorithm.

Test	Noise (°)	Set-point Change (°)	Additional Mass (g)
#1	1.3	–	–
#2	–	125	–
#3	–	–	22
#4	0.7	125	–
#5	–	75	12
#6	0.5	–	22
#7	0.5	50	12

3.2.3. BWO Performance

As was shown previously with the controllers designed using the GA and ABC algorithm, the controller designed with the BWO algorithm was tested on a real inverted pendulum. The values of the FOPID gains are shown in Table 9.

Table 9. FOPID parameters for the controllers designed with the BWO algorithm.

Controller	k_p	k_i	k_d	λ	μ
Pendulum angle	47.5270	1.9886	2.0927	0.7109	1.1632
Horizontal angle	16.4606	1.0742	1.3525	0.7849	1.1895

First, tests were conducted with individual disturbances. The system could withstand a noise level of $\sigma = 1.0°$, as seen in Figure 33. In the case of a set-point change, the maximum

change achieved was 100°, as shown in Figure 34. Regarding extra load, 40 g was placed on the vertical rod. The results are shown in Figure 35.

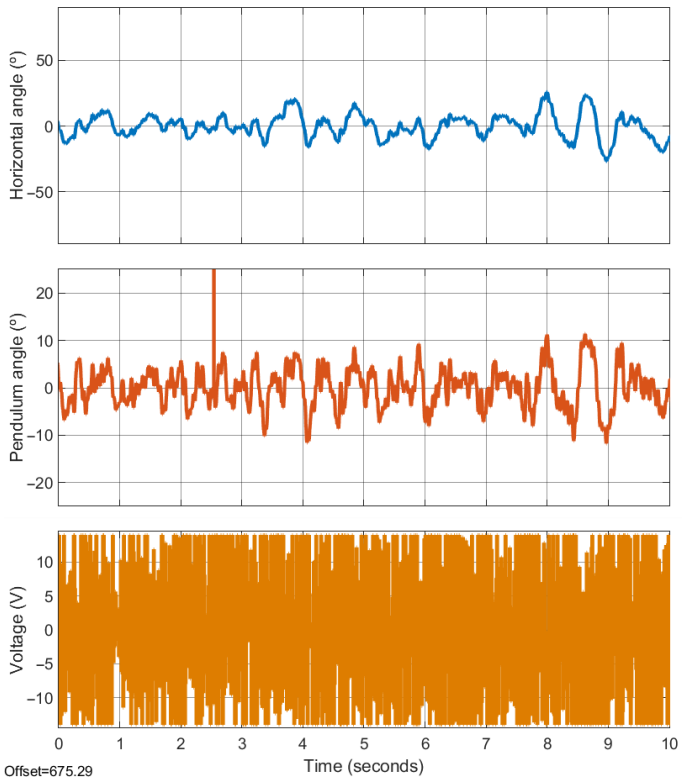


Figure 33. System response under $\sigma = 1.0°$ white noise. (Controller designed with the BWO algorithm.)

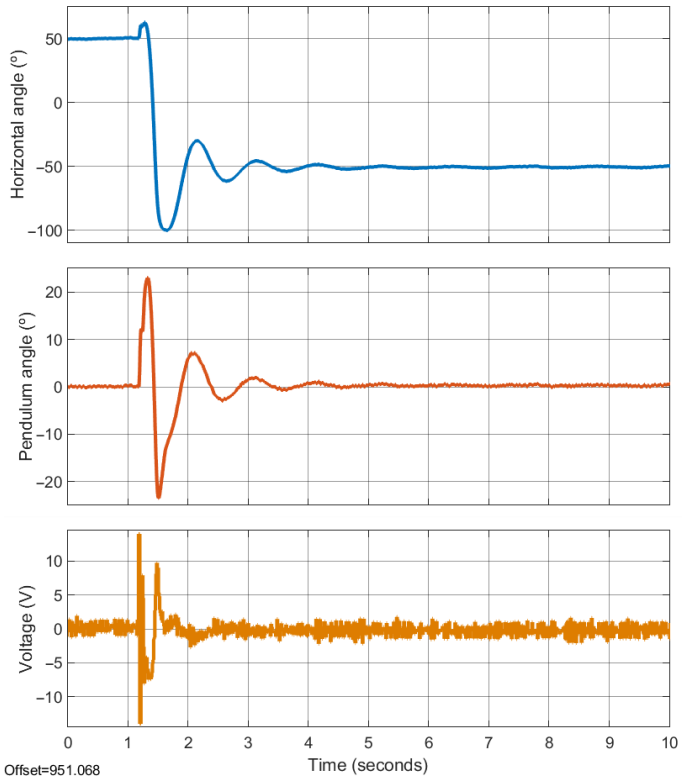


Figure 34. System response to a set-point change of 100°. (Controller designed with the BWO algorithm.)

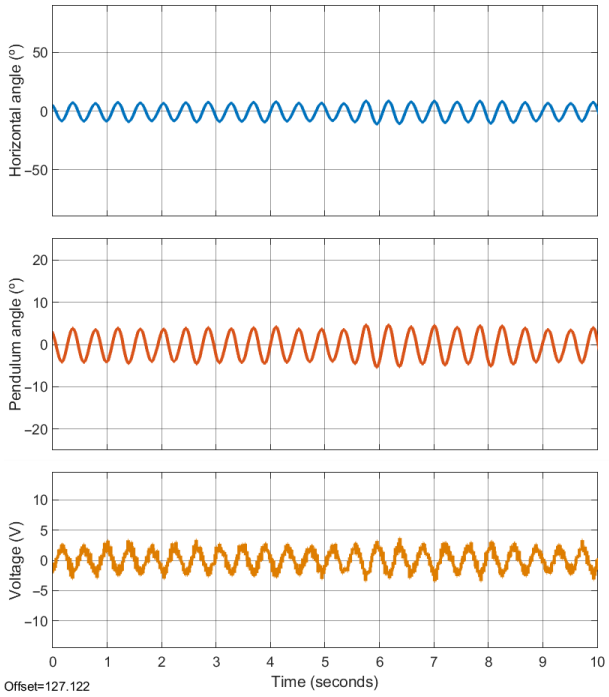


Figure 35. System response with 40 g of extra load. (Controller designed with the BWO algorithm.)

Having tested the robustness of the system with individual disturbances, combinations of those disturbances were applied. First, the system could withstand a combination of a 50° set-point change while white noise characterized by $\sigma = 1°$ was applied to the sensor signals, as shown in Figure 36.

Next, a combination of 32 g of extra load and a set-point change of 50° was tested, as can be seen in Figure 37. Finally, a combination of extra load (30 g) and noise ($\sigma = 0.5°$) was also tested. The results are shown in Figure 38.

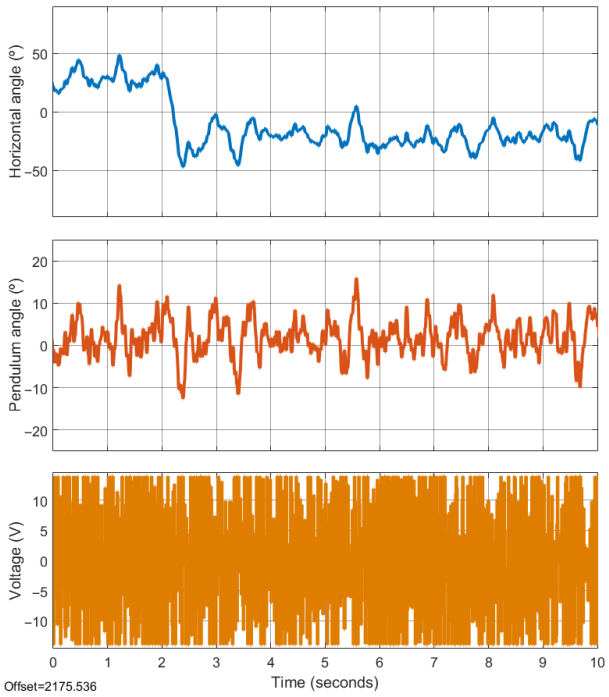


Figure 36. System response to a set-point change of 50° under $\sigma = 1°$ white noise. (Controller designed with the BWO algorithm.)

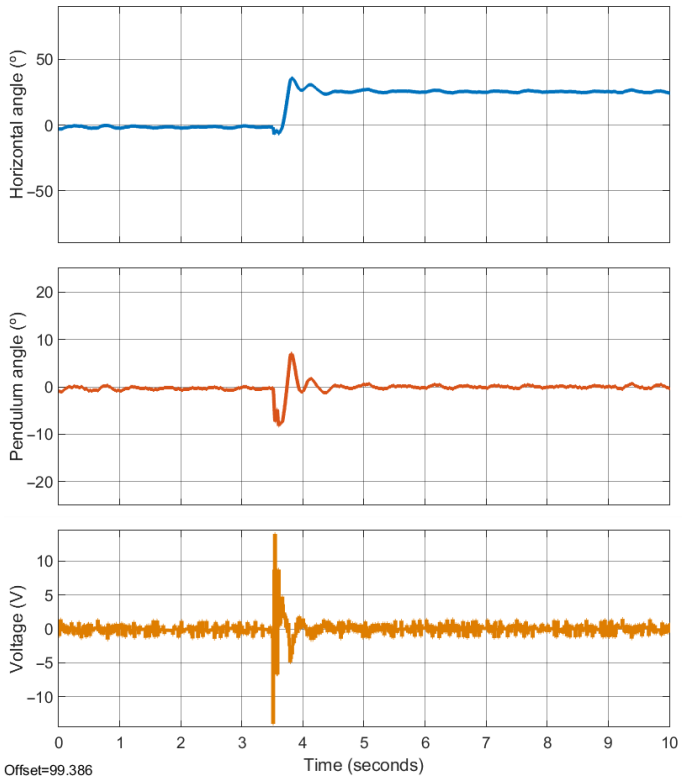


Figure 37. System response to a set-point change of 25° with 32 g of extra load. (Controller designed with the BWO algorithm.)

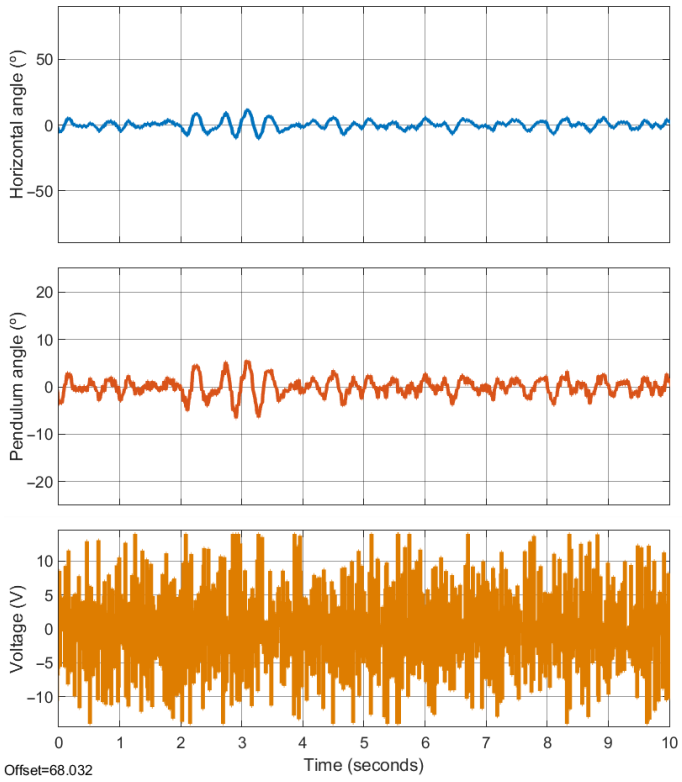


Figure 38. System response with 30 g of extra load under $\sigma = 0.5°$ of white noise. (Controller designed with the BWO algorithm.)

Last, the three disturbances were applied simultaneously. The result of applying 20 g of extra load, white noise characterized by $\sigma = 0.5°$, and a set-point change of 40° is shown in Figure 39.

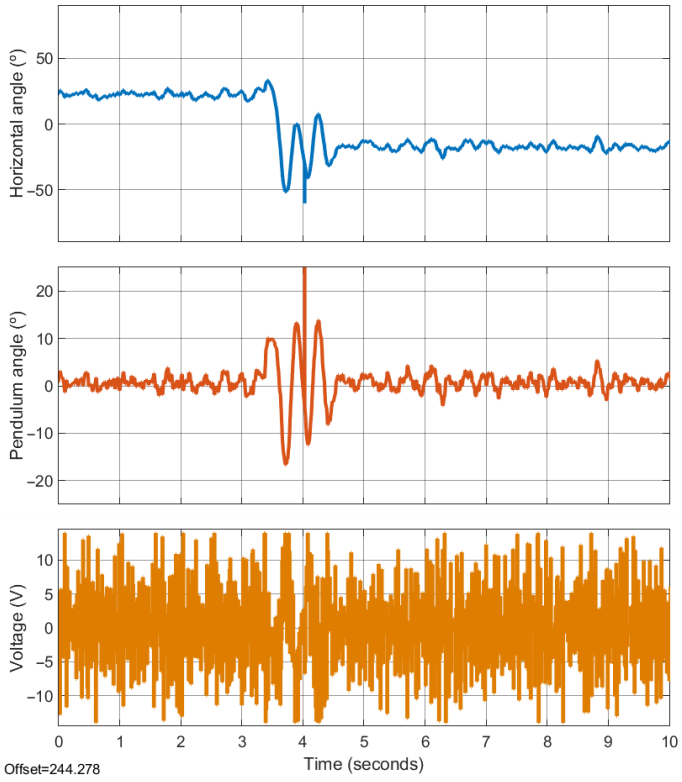


Figure 39. System response with all disturbances combined with 20 g load, $\sigma = 0.5°$, and a set-point change of 40°. (Controller designed with the BWO algorithm.)

The results of the previous seven tests are summarized in Table 10.

Table 10. Summary of the results of the disturbance tests with BWO.

Test	Noise (°)	Set-Point Change (°)	Additional Mass (g)
#1	1.0	–	–
#2	–	100	–
#3	–	–	40
#4	1.0	50	–
#5	–	25	32
#6	0.5	–	30
#7	0.5	40	20

To summarize the tests carried out for the three controllers, Table 11 is provided. However, it must be taken into account that due to the randomness of the algorithms, the performance of the algorithm and the controller are independent. That is, a controller designed with a theoretically worse algorithm might still have better performance due to said randomness of the results. Moreover, as in all cases where the execution terminated when the objective function reached the established threshold, the controllers have a similar overall performance, being each of them better at dealing with certain disturbances than others.

Table 11. Summary of the results of the disturbance tests with the three controllers.

Disturbance	GA	ABC	BWO
Noise (°)	1.25	1.3	1.0
Set-point change (°)	75	125	100
Additional mass (g)	22	22	40

3.3. Typical Values of the FOPID Controller Parameters

In order to provide further evidence about the performance of the algorithms, Figures 40 and 41 show the values obtained for 13 different controllers. These controllers were tested in the laboratory to ensure that they were stable not only in simulations but also in reality.

In Figures 40 and 41, it can be seen that the five gains of each controller have roughly similar values. It is worth recalling that metaheuristic algorithms do not provide a perfect solution to the problem; they provide an optimized solution. Consequently, all the values are within a certain range that has been proven to be adequate.

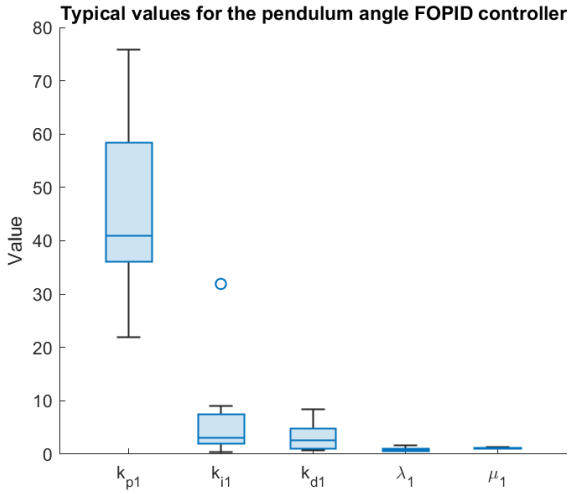


Figure 40. Typical values for the pendulum angle FOPID controller.

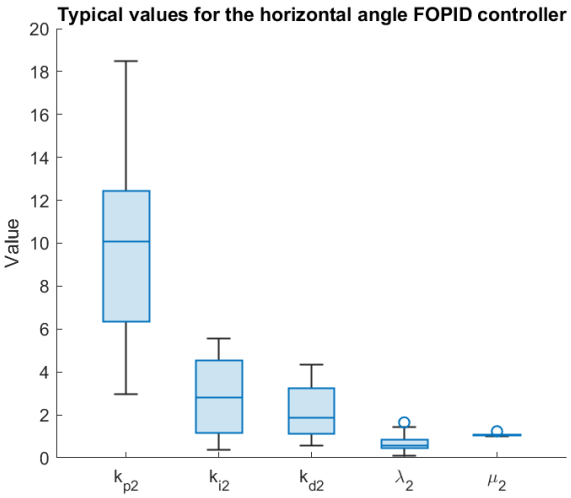


Figure 41. Typical values for the horizontal angle FOPID controller.

To obtain these 13 controllers, the same conditions as those in Table 3 were used, including the search limits. These limits were applied to establish the initial search boundaries;

however, the algorithm was allowed to explore beyond these limits, as can be observed in Figure 41, where the lowest parameter values fall well below the initial inner search limit.

4. Discussion

Previous works have addressed the implementation of metaheuristic algorithms. However, it is hard to find information on how to configure the algorithms themselves in a comprehensive way.

However, metaheuristic algorithms involve many parameters, and their behavior can vary significantly depending on the specific application. For instance, the number of parameters to tune greatly affects the performance and fitness of the chosen algorithm configuration.

In this study, an effort was made to demonstrate that a complex system such as a FOPID controller can be effectively tuned using different metaheuristic algorithms. Moreover, not just one controller was designed: three controllers were developed and exhaustively tested to validate both the metaheuristic algorithms and the proposed controller for a complex control problem. Additionally, thirteen different configurations were successfully tested on a real system, confirming their ability to maintain the inverted pendulum in a stable upright position, although the maximum disturbances tolerated were not quantitatively measured.

As there are multiple ways of configuring the algorithms, different configurations were tested on a limited number of executions. In that way, the randomness of the metaheuristic algorithms was reduced, allowing one to study the effect of each configured parameter on the performance of each algorithm. These results can be applied only to the scope of this work, as for other problems, other configurations might improve performance.

However, it is important to avoid certain comparisons between algorithms. The results obtained do not allow one to establish universal comparisons between the algorithms. For the case study presented in this work, BWO showed better performance in terms of convergence rate than the ABC algorithm and the GA. However, this might not be the case for other kinds of problems. It is also important to avoid comparing the algorithms based on the performance of the controllers when disturbances are applied. In this study, the controllers were designed without applying disturbances to the model; therefore, the algorithms did not search for specific solutions for those kinds of problems. The three controllers tested are similar, as their performance index (J) was similar. According to the results obtained, each controller better withstands certain disturbances; however, this cannot be attributed to the algorithms themselves, but to the inherent randomness of metaheuristic algorithms. To perform such a comparison, the algorithms should execute a simulation model during their search in which the disturbances are applied.

Furthermore, the algorithms only consider the performance index, which does not include information on how the system reacts to disturbances, since the simulations conducted during the algorithm execution do not include any form of disturbance (aside from the initial angle of the vertical rod).

Therefore, the results obtained from the three different algorithms serve to validate the applicability of all the algorithms rather than to establish direct comparisons between them. The main differences that can indeed be established from these results concern the success rate and the execution time, as shown in Figure 16.

5. Conclusions

First, an introduction explaining the need for the use of metaheuristic algorithms in control engineering was presented. After this, the fundamentals of metaheuristic algorithms were explained, and the specific fundamentals of each of the metaheuristic algo-

rithms used in this work were presented. In addition, detailed flowcharts of the algorithms were provided.

The performance index is a key component for the proper execution of the algorithm, and has a significant impact on the controller's performance. Therefore, the usual performance indices were analyzed to adequately choose the most suitable index for the specific case adopted in this work.

Then, the equations of the inverted pendulum were shown, as they are necessary to build a simulation model of the system, which is essential to execute the metaheuristic algorithms.

Subsequently, several executions of the algorithms were run to obtain the best configuration for each algorithm, considering that many parameters affect both the likelihood of convergence and the execution time.

In summary, the best configuration that achieves a balance between speed and success for the algorithms studied in this work is shown in Table 12.

Table 12. Recommended configurations for each algorithm for tuning a FOPID controller for an inverted pendulum.

Algorithm	Population	Mutation	Trial Limit	Reproduction	Cannibalism
GA	15–20	0.5–0.75	–	–	–
ABC	100	0.5–0.75	5	–	–
BWO	100	0.75	–	>0.4	0.25–0.75

It must be taken into account, as previously mentioned, that these results are based on a limited number of executions and on a specific example, so they may vary under different circumstances. They can also be adapted to the requirements of the users; for instance, prioritizing higher convergence rates despite longer execution times.

Having obtained the most suitable configuration for each algorithm, 32 executions were performed to compare their performance in this case study. Based on the results, it can be concluded that, in the tests performed, the BWO algorithm demonstrated better overall performance than the GA and the ABC algorithm, exhibiting the highest success rate while maintaining reasonable execution times. Taking into account only execution time, the ABC algorithm was faster than the others; however, it showed lower convergence reliability, as seen in Figure 16. For other kinds of problems, other algorithms might provide better performance, as, for example, it might be more desirable to explore more solutions rather than exploit them.

Finally, using the optimal configurations, three controllers were designed—one per algorithm—and tested to determine the maximum disturbances each could withstand.

Author Contributions: Conceptualization, B.M.A.-H. and A.G.G.; methodology, B.M.A.-H. and M.B.-R.; software, A.G.G. and M.B.-R.; validation, A.G.G. and B.M.A.-H.; formal analysis, A.G.G.; investigation, A.G.G. and M.B.-R.; writing—original draft preparation, A.G.G. and B.M.A.-H.; writing—review and editing, A.G.G. and M.B.-R.; supervision, A.G.G. and B.M.A.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: Manuel Barraza-Rodríguez thanks the financial support from Fortalecimiento Grupos de Investigación UTA N° 8764-25. This work is a result of grant PID2024-155948OB-C54

(project BRAINS: Building Real-world Applications with Intelligent, Neural, and Sustainable approaches) funded by MICIU/AEI/10.13039/501100011033 and ERDF/EU.

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

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Artificial Bee Colony Algorithm
ACO	Ant-Colony Optimization
AHA	Artificial Hummingbird Algorithm
BWO	Black Widow Optimization
CS	Cuckoo Search
DE	Differential Evolution
dPSO	Dynamic Particle Swarm Optimization
FOPI	Fractional-Order PI
FOPID	Fractional-Order PID Controller
GA	Genetic Algorithm
GWO	Grey Wolf Optimizer
HHO	Harris Hawks Optimization
PSO	Particle Swarm Optimization
LMA	Levenberg–Marquardt Algorithm
MSBWO	Multi-strategy Improved Beluga Whale Optimization
SCA	Sine–Cosine Algorithm
SFOA	Starfish Optimization Algorithm
SMA	Slime Mould Algorithm
TLBO	Teaching–Learning-Based Optimization

References

1. Bashishtha, T.K.; Singh, V.P.; Yadav, U.K.; Sahu, U.K. Fractional-order PID controllers and applications: A comprehensive survey. *Annu. Rev. Control* **2025**, *60*, 101013. [CrossRef]
2. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; University of Michigan Press: Cambridge, MA, USA, 1975.
3. Velasco, L.; Guerrero, H.; Hospitaler, A. A Literature Review and Critical Analysis of Metaheuristics Recently Developed. *Arch. Comput. Methods Eng.* **2024**, *31*, 125–146. [CrossRef]
4. Camacho-Villalon, C.L.; Dorigo, M.; Stutzle, T. Why the Intelligent Water Drops Cannot Be Considered as a Novel Algorithm. In *Proceedings of the Swarm Intelligence (ANTS 2018)*; Lecture Notes in Computer Science; Dorigo, M., Birattari, M., Blum, C., Christensen, A., Reina, A., Trianni, V., Eds.; Springer: Cham, Switzerland, 2018; Volume 11172, pp. 302–314. [CrossRef]
5. Piotrowski, A.P.; Napiorkowski, J.J.; Rowinski, P.M. How novel is the "novel" black hole optimization approach? *Inf. Sci.* **2014**, *267*, 191–200. [CrossRef]
6. Rajwar, K.; Deep, K.; Das, S. An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. *Artif. Intell. Rev.* **2023**, *56*, 13187–13257. [CrossRef] [PubMed]
7. Shami, T.M.; El-Saleh, A.A.; Alswaitti, M.; Al-Tashi, Q.; Summakieh, M.A.; Mirjalili, S. Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access* **2022**, *10*, 10031–10061. [CrossRef]
8. Zhang, H.; Yuan, M.; Liang, Y.; Liao, Q. A novel particle swarm optimization based on prey–predator relationship. *Appl. Soft Comput.* **2018**, *68*, 202–218. [CrossRef]
9. Gou, J.; Lei, Y.X.; Guo, W.P.; Wang, C.; Cai, Y.Q.; Luo, W. A novel improved particle swarm optimization algorithm based on individual difference evolution. *Appl. Soft Comput.* **2017**, *57*, 468–481. [CrossRef]
10. Zhang, X.; Wang, X.; Kang, Q.; Cheng, J. Differential mutation and novel social learning particle swarm optimization algorithm. *Inf. Sci.* **2019**, *480*, 109–129. [CrossRef]
11. Kao, Y.T.; Zahara, E. A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Appl. Soft Comput.* **2008**, *8*, 849–857. [CrossRef]
12. Kıran, M.S.; Gündüz, M.; Baykan, Ö.K. A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum. *Appl. Math. Comput.* **2012**, *219*, 1515–1521. [CrossRef]
13. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
14. Das, A.; Rai, R.; Sasmal, B.; Dhal, K.G.; Khurma, R.A.; Saha, R. Metaheuristic Algorithms Since 2020: Development, Taxonomy, Analysis, and Applications. *Arch. Comput. Methods Eng.* **2025**. [CrossRef]

15. Basant, T.; Narendra, K.; Mini, S. Robust Control of Rotary Inverted Pendulum Using Metaheuristic Optimization Techniques Based PID and Fractional Order $PI^{\lambda}D^{\mu}$ Controller. *J. Vib. Eng. Technol.* **2024**, *12*, 1–20. [CrossRef]
16. Shalaby, R.; El-Hossainy, M.; Abo-Zalam, B. Fractional order modeling and control for under-actuated inverted pendulum. *Commun. Nonlinear Sci. Numer. Simul.* **2019**, *74*, 97–121. [CrossRef]
17. Mondal, R.; Chakraborty, A.; Dey, J.; Halder, S. Optimal fractional order PIλDμ controller for stabilization of cart-inverted pendulum system: Experimental results. *Asian J. Control* **2020**, *22*, 1345–1359. [CrossRef]
18. Dwivedi, P.; Pandey, S.; Junghare, A. Stabilization of unstable equilibrium point of rotary inverted pendulum using fractional controller. *J. Frankl. Inst.* **2017**, *354*, 7732–7766. [CrossRef]
19. Erkol, H. Optimal $PI^{\lambda}D^{\mu}$ controller design for two-wheeled inverted pendulum. *IEEE Access* **2018**, *6*, 75709–75717.
20. Abut, T. Experimental Validation of Metaheuristic Optimized Fractional-Order PID Control for a Rotary Inverted Pendulum. *IEEE Access* **2026**, *14*, 85321–85331. [CrossRef]
21. Turktam, M.; Zorlu, H.; Soylu, S. Adaptive Hybrid Metaheuristic Optimization for FOPID-Based Controller-Energy CODESIGN in Nonlinear Underactuated Systems. *IEEE Access* **2026**, *14*, 113039–113059. [CrossRef]
22. Mondal, A.; Chakraborty, S. Design of two-loop FOPID-FOPI controller for inverted cart-pendulum system. *Eng. Res. Express* **2024**, *6*, 035354. [CrossRef]
23. Dey, B.; Pandey, S.K.; Sengupta, A. Design of modified fractional order PID controller for cart inverted pendulum system. *Int. J. Dyn. Control* **2025**, *13*, 1–17. [CrossRef]
24. Cheballah, F.; Mellah, R.; Saim, A. Design of fractional MOIF and MOPIF controller using PSO algorithm for the stabilization of an inverted pendulum-cart system. *IET Control Theory Appl.* **2025**, *19*, e12648. [CrossRef]
25. ZHANG, H.; Nor, N.M. Optimized fractional-order PID (FOPID) controller for two-wheeled self-balancing robots (TWSBR) using Multi-strategy improved Beluga Whale optimization algorithm (MSBWO). *Tech. Rep. Oct.* **2024**. [CrossRef] [PubMed]
26. PM, A. Egret swarm optimized distributed power flow controller for power quality enhancement in grid connected hybrid system. *J. Electr. Eng. Technol.* **2024**, *19*, 2047–2057. [CrossRef]
27. Goud, B.S.; Reddy, C.R.; Bajaj, M.; Elattar, E.E.; Kamel, S. Power Quality Improvement Using Distributed Power Flow Controller with BWO-Based FOPID Controller. *Sustainability* **2021**, *13*, 11194. [CrossRef]
28. Hayyolalam, V.; Kazem, A. Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103249. [CrossRef]
29. Figueroa-Torrez, P.; Duran, O.; Crawford, B.; Cisternas-Caneo, F. Enhancing Manufacturing Cell Formation Through Availability-Based Optimization Using the Black Widow Optimizer Metaheuristic. *Biomimetics* **2026**, *11*, 294. [CrossRef] [PubMed]
30. Rathod, V.V.; Rana, D.P.; Mehta, R.G. Black widow optimisation with deep learning-based feature fusion model for remote sensing image analysis. *Int. J. Comput. Sci. Eng.* **2025**, *28*, 56–70. [CrossRef]
31. Chen, Z.; Li, W.; Liu, X.; Wang, Y.; Chan, T.H.T. A Multistrategy Fusion-Improved Black Widow Optimization Algorithm for Structural Damage Identification. *Struct. Control Health Monit.* **2025**, *2025*, 2939779. [CrossRef]
32. Li, G.; Zhang, T.; Tsai, C.Y.; Yao, L.; Lu, Y.; Tang, J. Review of the metaheuristic algorithms in applications: Visual analysis based on bibliometrics. *Expert Syst. Appl.* **2024**, *255*. [CrossRef]
33. Bassi Joshep, S.; Gbenga Dada, E.; Abidemi, A.; Oyewola, D.; Khammas, B. Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. *Heliyon* **2022**, *8*, e09399. [CrossRef] [PubMed]
34. Ivanovski, T.; Brkic Bakaric, M.; Matetic, M. Recent Advances in Metaheuristic Algorithms. *Algorithms* **2025**, *19*, 19. [CrossRef]
35. Suman, A.; Gunjan; Udmale, S.S. Metaheuristic algorithms: A benchmark-driven functional taxonomy and performance analysis. *Comput. Sci. Rev.* **2026**, *60*, 100884. [CrossRef]
36. Molina, D.; Poyatos, J.; Del Ser, J.; Garcia, S.; Hussain, A.; Herrera, F. Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations. *Cogn. Comput.* **2020**, *12*, 897–939. [CrossRef]
37. Datta, D.; Amaral, A.; Figueira, J. Single row facility layout problem using a permutation-based genetic algorithm. *Eur. J. Oper. Res.* **2011**, *213*, 388–394. [CrossRef]
38. Sadrzadeh, A. A genetic algorithm with the heuristic procedure to solve the multi-line layout problem. *Comput. Ind. Eng.* **2012**, *62*, 1055–1064. [CrossRef]
39. Lee, C. A review of applications of genetic algorithms in operations management. *Eng. Appl. Artif. Intell.* **2018**, *76*, 1–12. [CrossRef]
40. Chouhan, S.; Kaul, A.; Singh, U. Soft computing approaches for image segmentation: A survey. *Multimed. Tools Appl.* **2018**, *77*, 28483–28537. [CrossRef]
41. Dhal, K.; Ray, S.; Das, A.; Das, S. A survey on nature-inspired optimization algorithms and their application in image enhancement domain. *Arch. Comput. Methods Eng.* **2019**, *26*, 1607–1638. [CrossRef]
42. Shyu, M.; Leou, J. A genetic algorithm approach to color image enhancement. *Pattern Recognit.* **1998**, *31*, 871–880. [CrossRef]

43. Ghosh, S.; Bhattacharya, S. A data-driven understanding of COVID-19 dynamics using sequential genetic algorithm based probabilistic cellular automata. *Appl. Soft Comput.* **2020**, *96*, 106692. [CrossRef] [PubMed]
44. Tang, K.; Man, K.; Kwong, S.; He, Q. Genetic algorithms and their applications. *IEEE Signal Process. Mag.* **1996**, *13*, 22–37. [CrossRef]
45. Bassi, J.; Dada, E. Automatic Tuning of Proportional-Integral-Derivative Controller using Genetic Algorithm. *Pac. J. Sci. Technol.* **2018**, *19*, 51–57.
46. Katoch, S.; Chauhan, S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
47. Karaboga, D. An Idea Based on Honey Bee Swarm for Numerical Optimization. Ph.D. Thesis, Erciyes University, Engineering Faculty, Kayseri, Türkiye, 2005.
48. El-Telbany, M. Tuning PID controller for DC motor: An artificial bees optimization approach. *Int. J. Comput. Appl.* **2013**, *77*, 18–21. [CrossRef]
49. Premkumar, K.; Vishnupriya, M.; Sudhakar Babu, T.; Manikandan, B.; Thamizhselvan, T.; Nazar Ali, A.; Rabiul Islam, M.; Kouzani, A.; Parvez Mahmud, M. Black widow optimization-based optimal PI-controlled wind turbine emulator. *Sustainability* **2020**, *12*, 10357. [CrossRef]
50. Tavazoei, M. Notes on integral performance indices in fractional-order control systems. *J. Process Control* **2010**, *20*, 285–291. [CrossRef]
51. Tepljakov, A. FOMCON: Fractional-order modeling and control toolbox. In *Fractional-Order Modeling and Control of Dynamic Systems*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 107–129.
52. Cazzolato, B.; Prime, Z. On the dynamics of the furuta pendulum. *J. Control Sci. Eng.* **2011**, *2011*, 528341. [CrossRef]
53. Castillo Gonzalez, V.D.; Molina Santiago, J.F.; Lopez Estrada, F.R.; Gomez-Penate, S. Modeling, parameterization, and state feedback control of a Furuta Pendulum. In Proceedings of the 2023 XXV Robotics Mexican Congress (COMRob), Xalapa/Veracruz, Mexico, 15–17 November 2023; pp. 87–92. [CrossRef]

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.