APPLICATIONS OF
MODELLING AND SIMULATION
http://arqiipubl.com/ams eISSN 2600-8084 VOL 8, 2024, 213-224

System Identification and Control of Linear Electromechanical Actuator Using PI Controller Based Metaheuristic Approach

A. A. Abdullah Hashim[1], N. M. Abdul Ghani[1*], S. Ahmad[2,3] and A. N. K. Nasir[1]

[1]Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, Pahang, Malaysia
[2]College of Engineering and Technology, University of Doha for Science and Technology, Qatar
[3]Kulliyyah of Engineering, International Islamic University Malaysia, Malaysia

[*]Corresponding author: normaniha@umpsa.edu.my

Submitted 01 March 2024, Revised 03 July 2024, Accepted 12 July 2024, Available online 03 August 2024.

Abstract: This study focuses on the crucial need for effective control strategies design for linear electromechanical actuators (EMA), which are essential components in industrial automation. While the PI controller is commonly used due to its simplicity and versatility in various control scenarios, its effectiveness is limited by its complex tuning process, which requires significant time and effort to achieve the optimal performance. To address this issue, the paper focuses on employing metaheuristic approaches that are Spiral Dynamic Algorithm (SDA) and Artificial Bee Colony (ABC) to fine tune the PI parameters for controlling the position of EMA. The simulation results, implemented in MATLAB Simulink, show that PI-SDA and PI-ABC produce better performances with minimal steady-state error, reduced overshoot, faster settling time and rise time. Hardware-in-the-loop (HIL) testing proves the effectiveness of the controller's performance in real-world scenarios. PI-SDA achieved a steady-state error of 0.0623, zero overshoot, faster rise time of 1.6956 s, and faster settling time of 7.2166 s. As for validation, PI-ABC showed similar results in HIL environment with a steady-state error of -0.0314, an overshoot of 5.0007%, a rise time of 1.3805 s, and a settling time of 9.1002 s. These results highlight the effectiveness of metaheuristic methods in real-world implementation and outperform the traditional heuristic approaches.

Keywords: Artificial bee colony; Hardware-in-the-loop; Linear actuator; PI controller; Spiral dynamic algorithm.

1. INTRODUCTION

Linear electromechanical actuators (EMA) have become indispensable in various industries, particularly automotive and machinery [1], [2], due to their versatility, efficiency, and accuracy. They have revolutionized mechanical motion execution in industrial automation, offering precise and controlled movement across a wide range of processes [3]. In contrast, traditional actuators, such as pneumatic and hydraulic systems, face limitations. Pneumatic actuators (PAs) may produce large torques with lower inertia and enhanced safety due to air's natural compliance, yet they suffer from slow response times, compliance issues, and nonlinear dynamics, making precise control challenging [4]. Hydraulic actuators are susceptible to leakage, leading to pressure drops, reduced performance, significant power losses and increased operational cost [5], [6]. Furthermore, the extensive utilization of EMA in industries like automotive, manufacturing, and robotics is crucial for improving operational effectiveness and ensuring accurate control of critical processes. Their ability to transform electrical energy into linear motion not only provides a dependable method for regulating the location and motion of industrial machinery but also positions them as essential components in the development of electrified vehicles, a growing trend in today's automotive industry.

However, controlling EMA presents significant challenges. The nonlinear dynamics inherent in their operation, coupled with stringent precision requirements, make the design of effective control systems complex. Nonlinear dynamics can lead to unpredictable behavior under varying loads and operating conditions, which complicates the modeling and control of these actuators [7], [8]. Additionally, achieving and maintaining precise operation necessitates a robust feedback mechanism. This is because any deviation from the desired performance must be quickly detected and corrected to ensure stable and accurate functionality.

To overcome these challenges, a range of control strategies is utilized for linear electromechanical actuators, with feedback control playing a crucial role in ensuring precise and stable operation. Among these strategies, PID controllers stand out due to their simplicity, versatility, and proven effectiveness in controlling linear motion systems [9]. Unlike more complex methods like fuzzy logic or state-space controllers, PID controllers provide a straightforward yet powerful approach to motion control.

In [10], the authors introduce a PID controller-based control strategy for electromechanical actuators in aerospace applications, aiming to minimize power losses, heating, and current consumption while optimizing performance. Furthermore in [11], the research aimed to construct a linear-axis research system utilizing industrial servo and stepper motors, integrating Matlab/Simulink controllers into the Beckhoff TwinCAT 3 platform. The study confirmed the effectiveness of importing and utilizing Matlab/Simulink PID controllers on Beckhoff soft PLCs, enabling the deployment of advanced and faster control strategies, and facilitating the evaluation of various controller concepts such as non-linear and soft logic controllers. However, the research faced challenges in finding the suitable PID parameters due to their reliance on manual tuning methods, highlighting the need for more efficient tuning techniques in future studies.

In response to these challenges, researchers have begun exploring alternative tuning methods, with a particular focus on optimization techniques that automate the process of fine-tuning PID parameters, driven by the exploration of nature-inspired algorithms to further enhance PID controller performance in specific applications [12]. These nature-inspired algorithms include evolutionary algorithms like Genetic Algorithms (GA) [13], Particle Swarm Optimization (PSO) [14], Differential Evolution (DE) [15], Ant Colony Optimization (ACO) [16], Simulated Annealing (SA) [17], Artificial Bee Colony (ABC) [18], Firefly Algorithm (FA) [19], Cuckoo Search (CS) [20], Harmony Search (HS) [21], and Grey Wolf Optimization (GWO) [22]. Each algorithm offers unique strategies for optimizing PID parameters and enhancing control performance in other applications. Nature-inspired optimization algorithms provide diverse strategies for efficiently exploring and exploiting the solution space. These algorithms mimic natural phenomena such as genetic evolution, particle swarm behavior, ant foraging, and simulated annealing to search for optimal solutions. They iteratively adjust parameters, combining exploration and exploitation to fine-tune PID controller parameters effectively. By leveraging these algorithms, researchers can optimize control performance in many applications, thereby enhancing system efficiency and functionality.

Numerous efforts have been devoted to enhancing PID controller performance, particularly in the domain of EMA. Authors in [23] presented a nonlinear model of an EMA system for aerofin control, utilizing genetic algorithms for optimizing both PID controllers and their nonlinear modifications. They introduced a fuzzy gain scheduling approach combined with genetic optimization, demonstrating improved transient response and closed-loop frequency performance through simulations and hardware tests. The paper in [24] also utilizes a genetic algorithm to determine the optimal PID controller parameters for a non-linear EMA, focusing on improving transient response. The optimization targets rise time, peak time, settling time, and maximum overshoot. Simulation results demonstrate that the genetic algorithm is a fast and flexible tuning method, significantly reducing overshoot by about 80% compared to conventional methods while maintaining other performance metrics. Another study [25], introduced a novel optimal PID controller for the position and stiffness control of an Antagonistic Variable Stiffness Actuator (AVSA) using Hammerstein models. Genetic algorithms were employed to compute the optimal PID gains for various positions and stiffness values, demonstrating significant improvements in control performance. While genetic algorithm optimization has shown promising results in optimizing EMAs, one significant concern is the possibility of slow convergence speeds. This means that genetic algorithms may require a significant number of iterations to find an ideal solution. As a result, the optimization process can be time-consuming, especially for complex or large-scale optimization problems. This highlights the critical need to investigate various optimization algorithms, especially for electromechanical actuators, to further enhance PID controller performance in these applications.

The aim of this discussion is to delve deeper into the advancements made in PID controller optimization, particularly focusing on EMA. This research employs metaheuristic method such as Spiral Dynamic Algorithm (SDA) and ABC, validated through extensive simulations and hardware-in-the-loop testing. One of the key advantages of using metaheuristic methods like SDA and ABC is their ability to efficiently optimize PID controller parameters for EMAs. These methods offer global optimization capabilities, allowing them to explore a wide range of solutions and find optimal or near-optimal control settings. Additionally, the adaptability of metaheuristic algorithms to nonlinear systems and their robustness in handling uncertainties such as friction, backlash, and dead zones make them well-suited for improving control performance in EMAs.

The contribution of this work lies in introducing alternative metaheuristic optimization approaches, particularly SDA and ABC, for EMAs. While previous studies often relied on GA for EMA optimization, this research explores the efficacy of SDA and ABC, showcasing their power and effectiveness in this context. By introducing novel optimization techniques that have not been extensively explored in the realm of EMAs, this study contributes to the advancement and diversification of optimization methodologies, paving the way for improved control systems in EMAs. Furthermore, the study contributes to the development of a model for linear EMAs using identification techniques. This approach addresses the limitations of traditional models that rely solely on mathematical equations and often neglect uncertainty parameters such as friction, backlash, and dead zones. By incorporating these real-world factors, the model provides a more accurate and reliable foundation for the design and optimization of control systems for EMAs.

2. METHODOLOGY

2.1 Data Acquisition

Figure 1 shows the Simulink system design for open loop, which begins with the introduction of an input voltage. This voltage goes through a series of Simulink blocks, first maintaining a specified range between 12 and -12, then shifting to a span of 0 to 24. The modified voltage signal is carefully converted to “uint8” format to ensure compatibility with an Arduino Uno. The data is recorded for further analysis or utilization within the System Identification Toolbox.

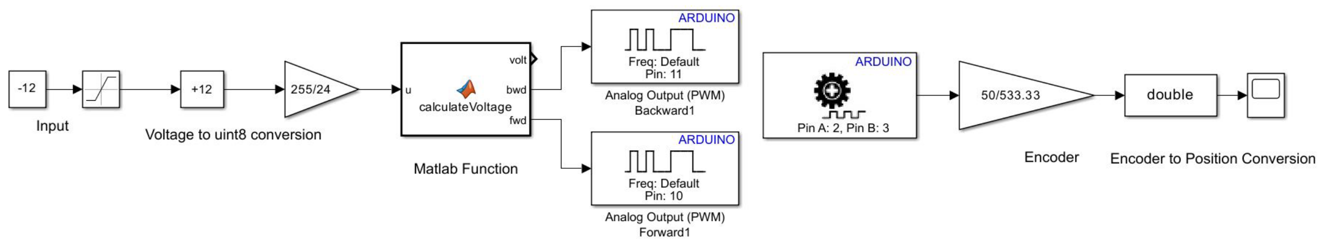

Figure 1. Open loop design of linear actuator system in Simulink.

2.2 System Identification for Optimization

The MATLAB System Identification Toolbox is a robust tool for analyzing and modeling dynamic systems using observed data. It includes features like model estimation, validation, and simulation to create accurate models that capture real-world complexities. This approach is particularly effective when physical laws are insufficient, or system dynamics are unclear. The toolbox helps engineers and researchers develop models that enhance understanding and performance in practical applications. Model estimation involves selecting a suitable model structure (e.g., ARX, ARMAX, NARX) based on system characteristics and data, followed by using optimization algorithms to adjust model parameters for better accuracy. Model validation, crucial for ensuring accuracy, includes methods like cross-validation and statistical metrics (RMSE, R2, AIC) to assess model fit and guide improvements. The goal is to achieve the best fit, accurately representing system dynamics and enabling accurate predictions and informed decision-making.

One of the well-known models in system identification is the Nonlinear Autoregressive with Exogenous Input (NARX) model. This model holds significance due to its capability to address inherent nonlinearities present in the behaviour of linear actuators. The regression model for the NARX architecture, the Equation (1). depicts the regressor model for the NARX architecture, which includes which includes delayed values for both input, *u* and output, *y*.

$$y(t) = f(u(t-1), u(t-2), \dots, y(t-1), y(t-2), \dots) + e(t) \tag{1}$$

where $y(t)$ signifies the current output, $u(t)$ represents the current input, and $e(t)$ denotes the model error. The function f encapsulates the system's underlying dynamics, encompassing the influence of both current and past input and output terms. This comprehensive regressor model proves essential in accurately capturing the nonlinear complexities inherent in the behaviour of a linear actuator, making the NARX model a valuable tool for system identification in such dynamic scenarios. Table 1 provides a summary of the parameters used for system identification in the different models.

Table 1. Summary of model parameters for system identification.

Model	Linear regressor	Input lag order	Output lag order	Sigmoid network	Number of poles	Number of zeros
NARX Model	Yes	[1, 7]	[1, 7]	Yes	-	-
Transfer Function Model 1	-	-	-	-	3	2
Transfer Function Model 2	-	-	-	-	8	7

2.3 Discrete PI Controller

PI controllers are conventional and widely used in industry due to their simplicity and effectiveness in handling noise issues and increasing system stability. Unlike PID controllers, PI controllers are preferred because they avoid the complexities introduced by the derivative (D) term. The derivative term in a PID controller amplifies high-frequency noise present in the system, leading to rapid fluctuations and oscillations in the control signal, which can destabilize the system [26]. While the derivative action can improve response times and reduce overshoot by damping the system, it also introduces phase lag, making the system more difficult to tune and potentially less stable. PI controllers, on the other hand, provide reliable performance by using only the proportional (P) and integral (I) gains, which simplifies the tuning process and avoids the noise amplification issues [27].

PI controllers exist in two-time domains: continuous time and discrete time. In continuous time, controllers are represented in the *s*-domain using Laplace transforms, while in discrete time, they are represented in the *z*-domain using *Z*-transforms. When testing a control system in a hardware-in-the-loop (HIL) setup, using a discrete PI controller in the *z*-domain is necessary because real-world hardware operates with digital signals and discrete-time sampling. In HIL testing, the controller is typically implemented on a digital platform such as a microcontroller, digital signal processor (DSP), or field-programmable gate array (FPGA), which process signals in discrete time intervals. The physical system (plant) and the controller exchange signals at discrete intervals, where continuous signals from sensors are sampled and quantized, and control signals are updated at each sampling period. To accurately test and validate the controller's performance as it would operate in the actual system, the controller in the HIL setup must operate in the same discrete-time manner. Thus, in the *z*-domain, the PI controller can be represented as Equation (2).

$$u(t) = Kp + Ki\,\frac{T}{Z-1} \tag{2}$$

where *Kp* is the proportional gain, *Ki* is the integral gain, *T* is the sampling period and *Z* is the complex frequency variable in the *Z*-transform domain.

The main importance of a PI controller lies in determining optimal values for *Kp* (proportional gain) and *Ki* (integral gain). which directly impact the controller's responsiveness, stability, and overall system performance. The process of fine-tuning these parameters can be challenging, as it requires a deep understanding of theoretical concepts such as control theory, system dynamics, and performance objectives. Therefore, this study includes an exploration of various methods and strategies for fine-tuning the PI parameters, aiming to optimize the controller's performance and achieve desired control objectives.

2.4 Metaheuristic Approach for Tuning PI controller

2.4.1 Spiral Dynamic Algorithm

The SDA is an innovative metaheuristic approach inspired by nature's spiraling patterns found in various biological and physical systems proposed by Tamura and Yashoda [30]. It integrates principles of diversification and intensification to efficiently explore and optimize parameter spaces, particularly in complex control systems such as PI or PID controllers. By leveraging the spiral trajectory, SDA facilitates an adaptive search process that balances exploration and exploitation, dynamically adjusting step sizes to navigate towards optimal solutions effectively.

The SDA optimization method begins with establishing the problem's objective function, which represents the metric to be improved. The parameter space also defines the value ranges for each parameter. The algorithm then employs a variety of potential solutions, commonly referred to as "agents". The movement of the agents begins at the outermost layer and progresses in a spiral trajectory towards the central optimal position. This can be achieved with a larger step size, fostering diversification as a primary objective. As the process advances, a shift towards intensification takes place. This involves dynamically decreasing the step size as the search points move through the innermost layer of the spiral. The combination of diversification and intensification methods aims to effectively guide the agents towards the optimal solution within the search area. Throughout this exploration, agents dynamically adjust their positions based on their fitness levels, providing insights into their relative performance with respect to the objective function. Figure 2 depicts a detailed flowchart explaining the step-by-step method by which the SDA efficiently adjusts the PI controller settings. The flowchart depicted deeply describes the sequential stages that involve initialization, iterative optimization, and comparative study of the result, revealing how SDA dynamically adapts agent positions based on the spiral pattern to gradually refine the PI parameters.

The spiral model is a vital component of the algorithm that specifies the spiral's characteristics and shape. The spiral model is primarily determined by two parameters: spiral radius r and spiral rotation angle θ. These parameters are crucial to ensure that the agents' movements align precisely with the intended spiral trajectory. Therefore, the equation mathematical modelling for SDA is represented by Equation (3).

$$x_i(k+1) = S_n(r,\theta)x_i(k) - (S_n(r,\theta) - I_n)x^* \tag{3}$$

where x^*represents the center of the spiral, $x_i(k+1)$ the updated position of searching agents at iteration of k and $x_i(k)$ denotes the iteration number, r signifies the rotation radius, and θ indicates the rotation angle. I_n *is* $n\times n$ identity matrix and $S_n(r,\theta)$ is a stable matrix given as. where $R^{(n)}(\theta)$ *)* is $n\times n$ rotational matrix.

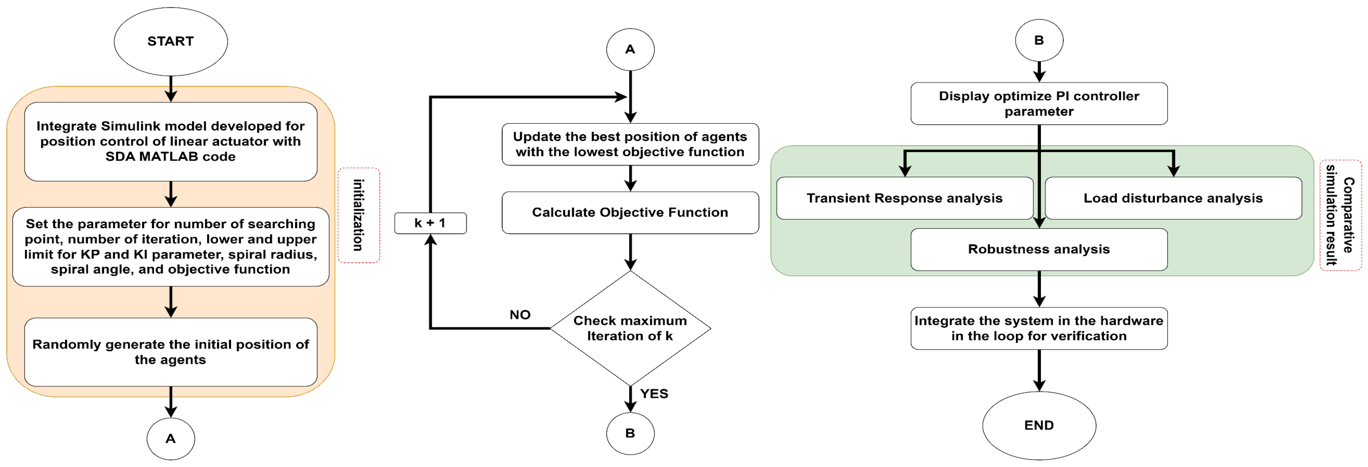
START
Integrate Simulink model developed for position control of linear actuator with SDA MATLAB code
Set the parameter for number of searching point, number of iteration, lower and upper limit for KP and KI parameter, spiral radius, spiral angle, and objective function
Randomly generate the initial position of the agents
initialization
A
A
Update the best position of agents with the lowest objective function
Calculate Objective Function
k + 1
NO
Check maximum Iteration of k
YES
B
B
Display optimize PI controller parameter
Transient Response analysis
Load disturbance analysis
Robustness analysis
Comparative simulation result
Integrate the system in the hardware in the loop for verification
END

Figure 2. Flowchart of the implementation of spiral dynamic algorithm.

2.4.2 Artificial Bee Colony

The ABC algorithm is a novel swarm intelligence method inspired by the foraging behavior of honeybees. The initial framework designed by Karaboga, emulates the collective foraging activities of a bee swarm to find optimal solutions for multi-variable and multi-modal continuous function [31]. In the natural world, bees foraging for food in a honeybee colony can be categorized into three groups: employed bees, onlooker bees, and scout bees. Each food source has a corresponding employed bee, meaning the number of employed bees is equal to the number of food sources. When a food source is abandoned, its employed bee transforms into a scout bee. Employed bees are linked to specific food sources based on factors such as proximity to the hive, richness or concentration of energy, and ease of energy extraction. For simplicity, the profitability of a food source can be represented as a single quantity. Employed bees search for available food sources and gather information, which they share with other bees in the hive through waggle dances that convey the distance, direction, and profitability of the food sources. Onlooker bees then choose the most promising food sources based on the information provided by the employed bees and further explore these sources. If the quality of a food source does not improve over a set number of cycles, known as the limit (an important control parameter in the ABC algorithm), the food source is abandoned, and the employed bee becomes a scout bee, searching for new food sources randomly near the hive. Figure 3, depicts the flowchart of the implementation of the ABC algorithm, illustrating the processes involved in each phase of the bees' activities.

The basic ABC algorithm divides the population into three types of bees: employed bees, following bees (also known as onlooker bees), and scout bees. Each type of bee operates in a distinct search phase: the employed bee phase, the following bee phase, and the scout bee phase. Initially, the algorithm generates the initial population through random initialization, as described by the following Equation (4).

$$x_{ij} = x_{ij}^{min} + \text{rand}\ (0, 1)\,.\left(x_j^{max} + x_j^{min}\right) \tag{4}$$

where $i = 1,\ldots,$SN and j=1, …, D. Here, SN denotes the population size, D represents the problem dimension, rand (0, 1) is a random number between 0 and 1, and x_j^{min} and x_j^{max} are the lower and upper bounds of the j-th dimension of an individual, respectively.

In the employed bee phase, each employed bee searches for new food sources by performing a random search within the feasible domain. This search is governed by the following Equation (5).

$$v_{ij} = x_{ij} + \Phi_{ij}\left(x_{ij} + x_{kj}\right) \tag{5}$$

where $k\in\{1, \ldots,\text{SN}\}$ is a randomly selected index different from i, ensuring that the new candidate solution is influenced by another solution in the population. $j\in\{1, \ldots, D\}$ is a randomly selected dimension, meaning that only one dimension is altered. Φ_{ij} is a random number uniformly distributed in the range $[-1,1]$.

In the following bee phase, the following bees select food sources based on the information passed back by the employed bees. The selection is done using a probabilistic approach, calculated by the following probability Equation (6).

$$P_i = \frac{fit_i}{\sum_{J=1}^{SN} fit_i} \tag{6}$$

Where fit_i denotes the fitness value of the i-th food source. The nectar collection process by the following bee also uses the Equation (5), to update the food source randomly.

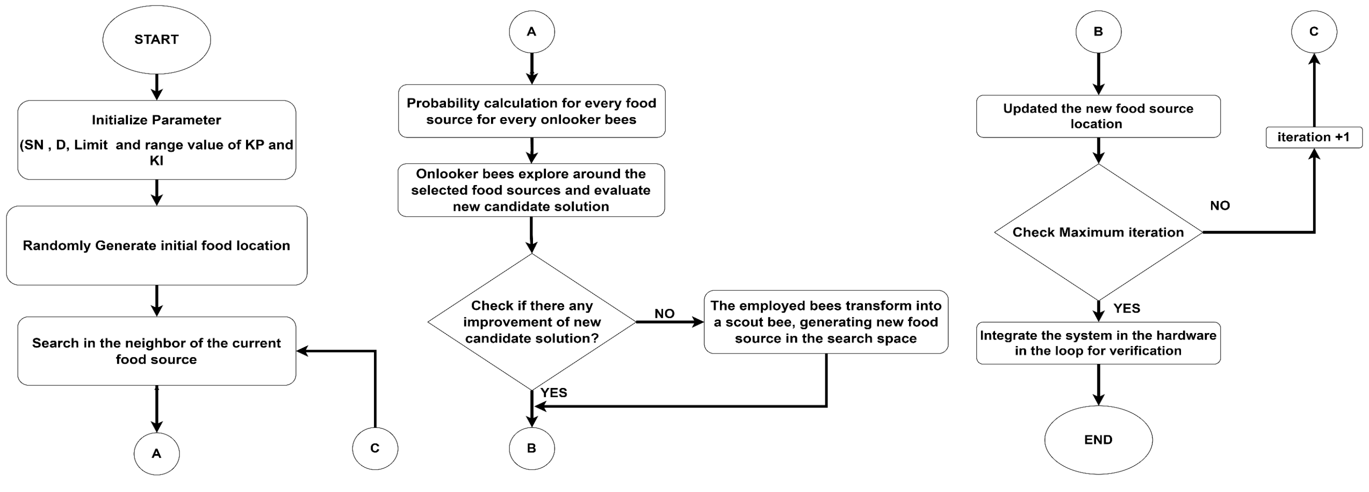
START
Initialize Parameter
(SN , D, Limit and range value of KP and KI
Randomly Generate initial food location
Search in the neighbor of the current food source
A
C
A
Probability calculation for every food source for every onlooker bees
Onlooker bees explore around the selected food sources and evaluate new candidate solution
Check if there any improvement of new candidate solution?
NO
The employed bees transform into a scout bee, generating new food source in the search space
YES
B
B
Updated the new food source location
Check Maximum iteration
NO
iteration +1
C
YES
Integrate the system in the hardware in the loop for verification
END

Figure 3. Flowchart of the implementation of artificial bee colony.

2.5 Simulation Model Design

Figure 4, shows the Simulink system block diagram to simulate control dynamics with a Discrete PI Controller and a linear actuator NARX model. This simulation environment is designed to replicate the behavior of the actual system, allowing for in-depth analysis. In this simulation, the desired position represents the target location for the linear actuator. This simulation employs the desired position as the linear actuator target. In discrete time, the Discrete PI Controller processes the error signal (the difference between desired and actual locations) to control the actuator. The NARX model better represents the actuator's nonlinear behavior than linear models. The simulation provides the actuator's actual position, which reflects its dynamic response to control signals. This complete simulation setup offers parameter adjustment and insight into the position control system's performance under various scenarios.

A crucial aspect of this simulation requires accurately choosing the numerical methods. In this study, a fixed-step solver employing the ode4 (fourth-order Runge-Kutta) method with a sample time of 0.2 seconds was selected for simulation purposes. The choice of a fixed-step solver was motivated by the need for predictability in simulation timing, deterministic behavior for repeatability, and suitability for real-time systems and hardware-in-the-loop (HIL) testing environments. The ode4 method was specifically chosen due to its balance between accuracy and computational cost, making it more accurate than lower-order methods while remaining computationally efficient compared to higher-order methods like ode45.

2.6 Hardware-in-the-Loop Techniques

2.6.1 MATLAB Simulink Setup for Hardware-in-the-Loop (HIL)

Figure 5 illustrates the closed-loop HIL setup process, where a discrete PI controller generates a control signal based on the discrepancy between the desired and actual positions of the linear actuator. As outlined in Figure 1, for the open-loop setup configuration, the primary components remain unchanged.

2.6.2 Comprehensive HIL Testing Setup and Parameter Analysis

Figure 6 depicts a comprehensive HIL testing setup. In this configuration, the closed-loop control system for the linear actuator's position is systematically executed using Simulink. The PI controller generates precise control signals, which are then processed by various Simulink blocks to manage voltage ranges and communicate with the Arduino Uno via the L298N Motor driver. The Arduino Uno is a critical component, converting these signals into practical commands that control the actuator's movement with extreme precision. Meanwhile, real-time feedback from the encoder, which captures the actuator's current position, is seamlessly integrated back into Simulink, ensuring continuous feedback loop completion. Table 2 provides details about the parameters of the linear actuator used in this study, offering insights into its specifications and functionality within the HIL testing setup. Additionally, in this HIL setup, as discussed in the simulation, the numerical configuration is crucial for running MATLAB Simulink smoothly. Therefore, the setup configuration during both simulation and hardware testing remains consistent to ensure consistency and accuracy in the system's performance evaluation.

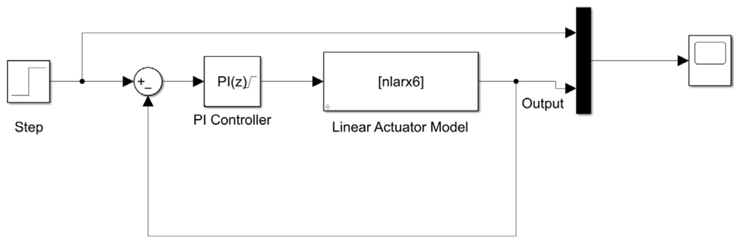

Figure 4. Closed-loop of the PI controller in MATLAB Simulink.

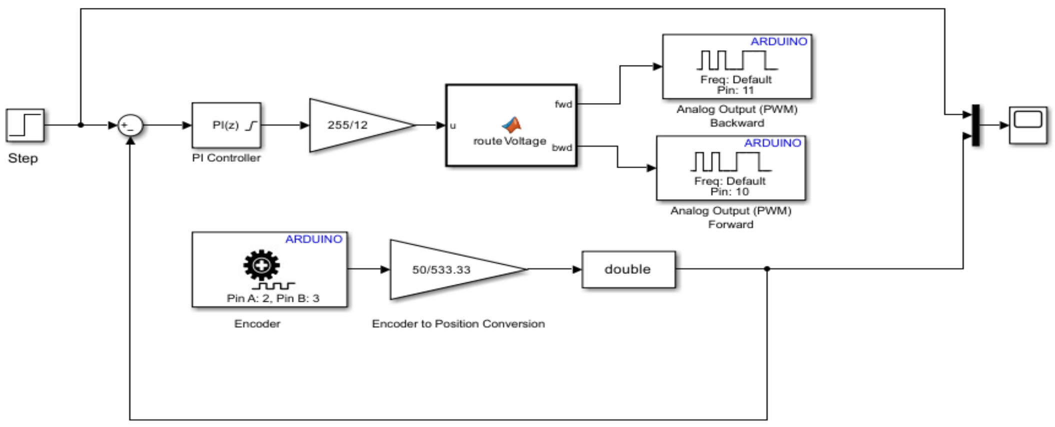

Figure 5. Closed-loop of the PI controller in MATLAB Simulink for hardware in the loop.

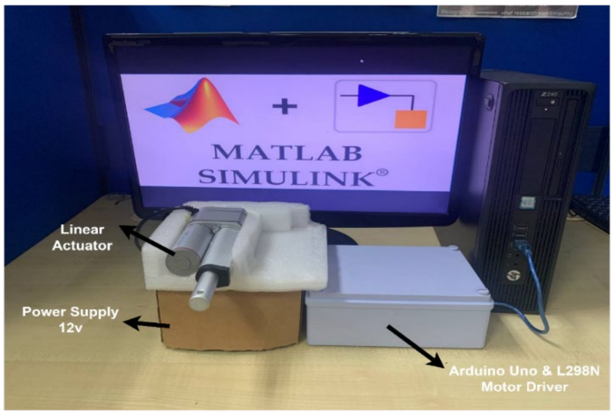

Figure 6. Hardware-in-the-loop (HIL) testing setup.

Table 2. Linear actuator parameter.

Parameter	Value
Voltage	12 v
Stroke Length	50 mm
Speed	24 mm/s
Gear Ratio	19: 11
Lead Screw diameter	6 mm
Push Load	500 N
Encoder Resolution	16 LPR

Table 3. Model validation comparison.

Type of model	Best fits (%)
Transfer Function model 1	10.98
Transfer Function model 2 (high zeros poles)	30.12
NARX model	90.18

3. RESULT AND DISCUSSION

3.1 System Identification and Model Selection

The performance of various models was compared, and the results are summarized in Table 3. The primary metric used for comparison is the percentage of best fits, which indicates how well each model matches the observed data. The term best fits percentage refers to the model's ability to closely match the observed data, with a higher percentage indicating a better fit and less error in capturing the system's behavior. In this context, the NARX Model's exceptional accuracy in achieving the validation signal and its highest best fits percentage highlight its strong capability in accurately capturing and predicting the system's behavior during the system identification process. The NARX Model stands out with a best fits percentage of 90.18%, demonstrating its superior ability to capture the underlying patterns and dynamics of the system being modeled. This high percentage indicates that the NARX model is highly accurate and reliable in predicting the output based on the given inputs.

In contrast, the Transfer Function Model 1 shows a significantly lower best fits percentage of 10.98%. This suggests that this model is less effective in fitting the observed data, potentially due to its simpler structure or assumptions that do not align well with the actual system behavior. The Transfer Function Model 2, which incorporates high zeros and poles, performs better than the first transfer function model, with a best fits percentage of 30.12%. While this is an improvement, it still falls short compared to the NARX model. The inclusion of more zeros and poles likely adds complexity and flexibility to the model, but not enough to surpass the NARX model's performance. In summary, the NARX model's highest best fits percentage underscores its robustness and suitability for the given application, making it the most effective choice among the models evaluated.

3.2 Comparative Analysis of Convergence Behavior

Figure 7 shows the convergence graph generated by the SDA over 20 iterations. This graph highlights the algorithm's rapid convergence and optimization efficiency. The SDA, designed to minimize errors using Root Mean Square Error (RMSE) as the cost function, typically converges within just 2 to 3 iterations. This quick convergence results from the careful configuration of the algorithm's parameters, particularly the PI parameters, which are restricted to ranges that consistently produce the lowest errors. By focusing exploration within these ranges, the algorithm speeds up convergence and avoids unnecessary exploration. The best cost function achieved during the 20 iterations is 2.9974708725032, demonstrating the effectiveness of the SDA-PI in optimizing the objective function.

On the other hand, the convergence graph in Figure 8 produced by the ABC algorithm over 20 iterations, presents a different convergence pattern. The ABC algorithm tends to converge approximately every 2 or 3 iterations, displaying a gradual convergence behavior with intermittent periods of convergence and maintenance. This approach allows the algorithm to steadily approach an optimal solution over multiple iterations, balancing exploration and exploitation in the search space. The key strength of the ABC algorithm lies in its ability to explore the search space comprehensively. It evaluates promising regions while avoiding premature convergence or stagnation in suboptimal solutions. Although the convergence speed may not match that of the SDA, the ABC algorithm's thorough exploration often leads to robust optimization and a deeper understanding of the solution space. The best cost function achieved by the ABC algorithm during these iterations is approximately 3.81, indicating successful error minimization and optimization of the specified objective function. This gradual convergence pattern, characterized by intermittent periods of convergence followed by maintenance, reflects the algorithm's balanced exploration strategy.

3.3 PI Controller Simulation System

The data presented in Table 4 showcases the distinct PI parameters utilized by various tuning methods, reflecting diverse control strategies. For instance, PI SDA and PI ABC exhibit specific parameter values that contribute to their respective efficiency in controlling the linear actuator. Meanwhile, Figure 9 visually illustrates the position of the linear actuator under different tuning methods, providing a clear comparison of their performance.

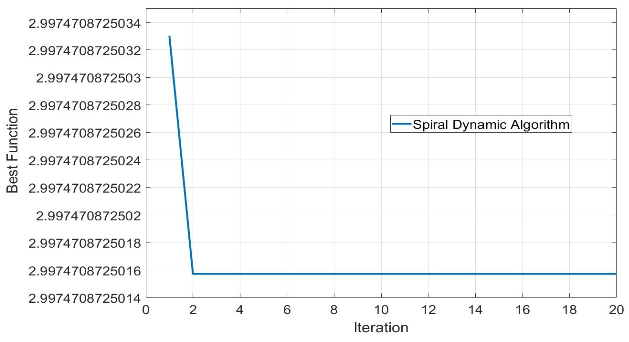

Figure 7. Convergence comparison of SDA and ABC algorithms.

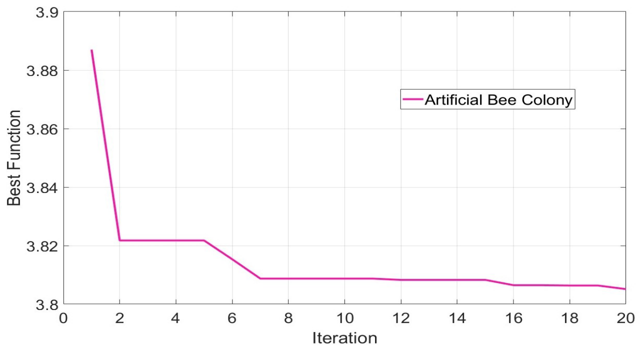

Figure 8. Convergence plot by using artificial bee colony.

Table 4. PI parameters for different tuning method at 25 mm step input.

Method	PI parameters	
	P	I
PI SDA	3.6428	2.0910
PI ABC	1.261047	0.617906
PI Ziegler Nichols	4.95	14.88714345
PI Trial and Error	2	3
PI Auto-Tune	6.8523	13.0526

Table 5. System performance analysis for different tuning methods at 25 mm step input.

Method	SSE	OS (%)	RT (s)	ST (s)
PI SDA	9.8547e-04	0	1.8537	6.3901
PI ABC	5.1777 e-04	5.6094	2.0707	6.4153
PI Ziegler Nichols	0.0029	29.3504	1.6999	6.6278
PI Trial and Error	-0.4907	23.0827	1.7152	10.8058
PI Auto-Tune	-0.0066	23.4308	1.6979	7.7179

SSE – Steady state error; OS – Overshoot; RT – Rise time; ST – Settling time.

Table 5 provides a comprehensive analysis of system performance under different tuning methods in response to a 25 mm step input. Starting with PI SDA, it exhibits a remarkably low steady state error of 9.8547e-04, indicating high precision in maintaining the desired position once the system stabilizes. Furthermore, it shows no overshoot, signifying a smooth and precise response without oscillations. The rise time of 1.8537 s indicates how quickly the system reaches its final position after the step input. Moving to PI ABC, while it maintains a low steady state error of 5.1777e-04, it experiences a slight overshoot of 5.6094%, suggesting minor transient oscillations before stabilizing. The rise time of 2.0707 seconds and settling time of 6.4153 s further indicate its ability to achieve stable control within a reasonable time frame. On the other hand, PI Ziegler Nichols shows a higher steady state error of 0.0029, along with a significant overshoot of 29.3504%, which indicates more pronounced oscillations before settling. However, it achieves a relatively low-rise time of 1.6999 s, showcasing a swift response to the step input. In contrast, PI Auto-Tune exhibits a negative steady state error (-0.4907), suggesting overcompensation leading to instability in maintaining the desired position. This is accompanied by an overshoot of 23.0827% and a relatively shorter rise time of 1.7152 s but a longer settling time of 10.8058 s, indicating delayed stabilization. Lastly, PI Trial and Error also shows a negative steady state error (-0.0066) but with an overshoot of 23.4308% and similar rise time and settling time values as PI Ziegler Nichols, indicating comparable performance in terms of response time and stability. Overall, this detailed analysis highlights how each tuning method affects the system's accuracy, stability, and responsiveness in achieving precise control of the linear actuator in response to a step input.

Figure 10 depicts the robustness testing process, which included a signal builder to expose the system to a variety of inputs and evaluate its adaptability under different conditions. They demonstrate that PI SDA and PI-ABC exhibit exceptional performance, showcasing consistently low overshoot and a stable response devoid of oscillations across a wide spectrum of input signals. In contrast, both PI Ziegler-Nichols and PI Auto-Tune show oscillations in their response to each input, indicating less stable and adaptive behavior. PI Trial and Error also displays similar oscillatory behavior, further highlighting its limitations compared to the metaheuristic approaches of PI-SDA and PI-ABC. This graphical representation underscores the robustness of the metaheuristic approach, particularly PI-SDA and PI-ABC, in maintaining precision and stability within the simulation system. This superiority is especially evident in applications that necessitate consistent and reliable control of the linear actuator's position under dynamic and changing conditions.

3.4 PI Controller in Hardware-in-the-Loop (HIL)

For further validation of the designed controller, Hardware-in-the-Loop (HIL) techniques were employed. Table 6 provides an overview of the PI controller parameters used during HIL testing, showcasing the parameters across various tuning methods. The parameters during the simulation phase, as shown in Table 4, remain consistent with those utilized in the HIL phase. This consistency is crucial as it ensures the robustness of the controller's performance in real-world experimentation scenarios, providing insights into how effectively the controller operates in a simulated environment compared to its behavior in an actual hardware setup.

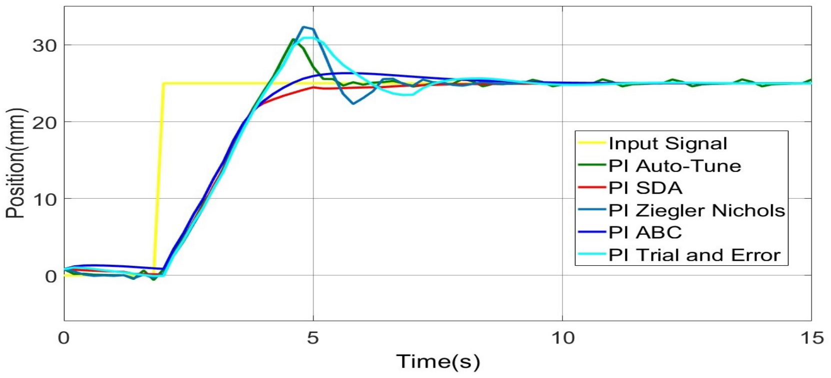

Figure 9. Position of linear actuator at step input 25 mm with various tuning methods.

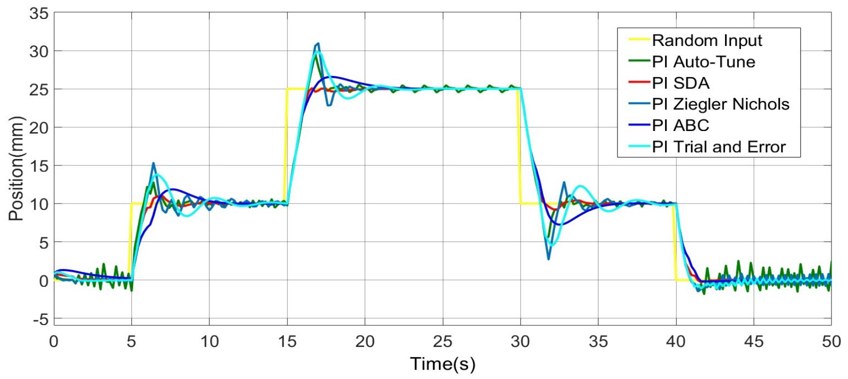

Figure 10. Position of linear actuator at random position with different tuning methods.

Table 6. PI parameter for different tuning method at 25 mm step input for HIL testing.

Method	PI parameters	
	P	I
PI SDA	3.6428	2.0910
PI ABC	1.261047	0.617906
PI Ziegler Nichols	4.95	14.88714345
PI Trial and Error	2	3
PI Auto-Tune	6.8523	13.0526

Table 7. System performance analysis for different tuning methods at 25 mm step input for HIL testing.

Methods	SSE	OS (%)	RT (s)	ST (s)
PI SDA	0.0623	0	1.6956	7.2166
PI ABC	-0.0314	5.0007	1.3805	9.1002
PI Ziegler Nichols	1.0936	44.3759	1.6747	36.0940
PI Trial and Error	-0.4064	27.8758	1.3083	6.8583
PI Auto-Tune	0.0623	24.8758	1.7361	9.8074

Figure 11 visually represents the position of the linear actuator in response to a 25 mm step input across various tuning methods during hardware testing. This graphical depiction offers a direct comparison of how each tuning method influences the actuator's position control in the physical hardware setup. Table 7 provides a detailed analysis of the system performance under different tuning methods during HIL testing. Specifically focusing on PI-SDA, it exhibits a steady-state error of 0.0623, indicating a relatively small deviation from the desired position once the system stabilizes. Moreover, it shows no overshoot, signifying a smooth and controlled response without oscillations. The rise time of 1.6956 s suggests how quickly the system achieves its final position after the step input, while the settling time of 7.2166 s reflects the time taken for the system to stabilize within a specified tolerance band around the desired position.

Comparatively, PI-ABC shows a negative steady-state error of -0.0314, indicating a slight overcompensation leading to a minor deviation from the desired position once stabilized. However, it does exhibit a small overshoot of 5.0007%, suggesting some transient oscillations before stabilization. The rise time of 1.3805 s and settling time of 9.1002 s further characterize its response time and stability. On the other hand, PI Ziegler-Nichols, PI Trial and Error, and PI Auto-Tune show higher steady-state errors (1.0936, -0.4064, and 0.0623, respectively), significant overshoots (44.3759%, 27.8758%, and 24.8758%, respectively). indicating less precise and stable control compared to PI-SDA and PI-ABC.

Robustness testing is essential for evaluating the controller's performance in both simulation and hardware testing environments. This process is critical in assessing the controller's responsiveness during both the extension and retraction phases of the actuator. Such transitional phases can potentially introduce instability into the system, underscoring the importance of verifying the controller's ability to effectively manage these changes and maintain stability throughout the operation. Thus, Figure 12 illustrates the testing of various input signals across different tuning methods. As expected, both PI-SDA and PI-ABC consistently produce excellent results free of oscillations, demonstrating their stability and effectiveness. This finding confirms that the implementation of the metaheuristic approach with SDA and ABC can indeed achieve robust and stable performance, establishing them as dependable options for maintaining precise control over the linear actuator's position, particularly in dynamic and unpredictable condition.

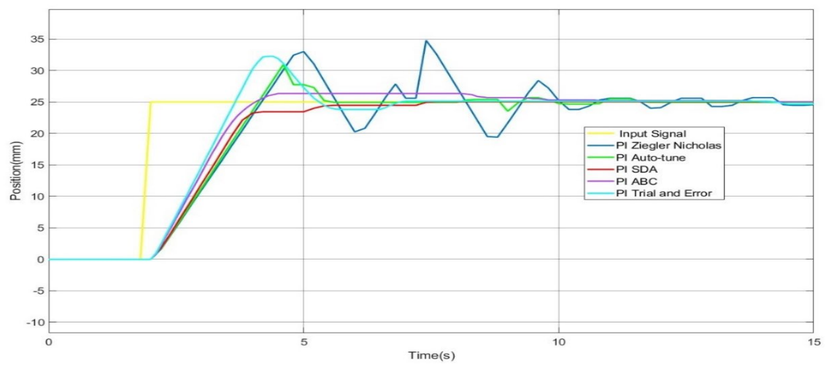

Figure 11. Position of linear actuator at step input 25 mm with various tuning methods for HIL testing.

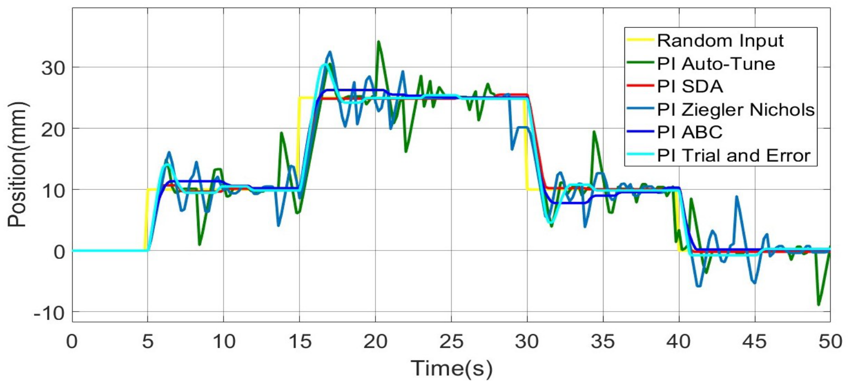

Figure 12. Position of linear actuator at random position with different tuning methods for HIL testing.

4. CONCLUSION

This paper introduced a PI controller designed to effectively control the position of a linear EMA, addressing a significant research gap in the field of linear EMAs. This study incorporates system identification using the NARX model to enhance the understanding of the actuator's dynamics and various tuning methods such as Ziegler-Nichols, Trial and Error, Auto-Tune, ABC algorithm, and the SDA for controller optimization. The NARX model was found to be superior in accurately capturing and predicting system behavior, achieving a best fits percentage of 90.18%, significantly higher than other models. This superior performance of the NARX model underscores its suitability for ensuring optimal control and reliable operation in real-world applications. In the comparative assessment of tuning methods, the SDA and ABC algorithm were identified as highly effective metaheuristic approaches. Simulation results indicated that these metaheuristic methods provided low overshoot and steady-state error. The effectiveness of the PI-SDA controller was particularly evident in HIL testing, where it achieved a steady-state error of 0.0623, zero overshoot, a rise time of 1.6956 s, and a settling time of 7.2166 s. Similarly, the PI-ABC controller also showed impressive results in HIL testing, achieving a steady-state error of -0.0314, an overshoot of 5.0007%, a rise time of 1.3805 s, and a settling time of 9.1002 s. Robustness testing revealed that both PI-SDA and PI-ABC maintained stability and effectiveness across a wide range of input signals, showcasing their suitability for applications requiring precise and stable control under dynamic and unpredictable conditions. These findings highlight the advantages of metaheuristic methods over traditional heuristic approaches like Ziegler-Nichols, Trial and Error, and Auto-Tune, which exhibited more significant oscillations and higher steady-state errors. Future work will employ various metaheuristic approaches, including advanced strategies to address issues like premature convergence, and will implement these in applications requiring precise positioning, such as power windows and auto brake systems.

ACKNOWLEDGMENT AND FUNDING

This work has been financially supported by the Fundamental Research Grant Scheme (FRGS/1/2021/ICT02/UMP/02/2) with a university reference number (RDU210110) and FRGS/1/2021/TK0/UIAM/02/27 with university reference number (FRGS21-245-0854). Both have been awarded by the Ministry of Higher Education Malaysia via Research and Innovation Department, Universiti Malaysia Pahang Al-Sultan Abdullah and Research Management Centre, International Islamic University Malaysia respectively.

DECLARATION OF CONFLICTING INTERESTS

The authors declare no potential conflicts of interest with respect to the research and publication of this article.

REFERENCES

[1] Q. Sanders and D. J. Reinkensmeyer, Design and preliminary evaluation of a soft finger exoskeleton controlled by isometric grip force, *Machines*, 12(4), 2024, 230.

[2] K. A. Daniel, P. Kowol and G. Lo Sciuto, Linear actuators in a haptic feedback joystick system for electric vehicles, *Computers*, 13(2), 2024, 48.

[3] I. Boldea, Linear electromagnetic actuators and their control: A review, *EPE Journal*, 14(1), 2004, 43-50.

[4] B. Yang *et al.*, Quantitative comparative study on the performance of a valve-controlled actuator and electro-hydrostatic actuator, *Actuators*, 13(4), 2024, 118.

[5] J. Xu and G. M. Bone, Actuators for improving robotic arm safety while maintaining performance: A comparison study, *Actuators*, 13(2), 2024, 69.

[6] K. Sotoodeh, Actuator selection and sizing for valves, *SN Applied Sciences*, 1(10), 2019, 1207.

[7] G. Qiao, G. Liu, Z. Shi, Y. Wang, S. Ma and T. C. Lim, A review of electromechanical actuators for more/all electric aircraft systems, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 232(22), 2018, 4128-4151.

[8] M. Heydari Shahna, M. Bahari and J. Mattila, Robust decomposed system control for an electro-mechanical linear actuator mechanism under input constraints, *International Journal of Robust and Nonlinear Control*, 34(7), 2024, 4440-4470.

[9] A. Turan, PID controller design with a new method based on proportional gain for cruise control system, *Journal of Radiation Research Applied Sciences*, 17(1), 2024, 100810.

[10] I. Todić, M. Miloš and M. Pavišić, Position and speed control of electromechanical actuator for aerospace applications, *Technical Gazette*, 20(5), 2013, 853-860.

[11] V. Tič and D. Lovrec, Development and implementation of closed loop position control concepts on electromechanical linear system using Beckhoff controller, *Machines. Technologies. Materials.*, 16(4), 2022, 116-119.

[12] R. S. Patil, S. P. Jadhav and M. D. Patil, Review of intelligent and nature-inspired algorithms-based methods for tuning PID controllers in industrial applications, *Journal of Robotics and Control*, 5(2), 2024, 336-358.

[13] Y. B. Cui, J. Zheng, Y. T. Ju and J. Xu, Precise angle control of electromechanical actuator with fuzzy PID and genetic algorithm, *Applied Mechanics and Materials*, 300-301, 2013, 1479-1485.

[14] Y. Parvez, N. R. Chauhan and M. Srivastava, Vibration control and comparative analysis of passive and active suspension systems using PID controller with particle swarm optimization, *Journal of The Institution of Engineers (India): Series C*, 2024.

[15] E. Lybrech Talakua and E. Dhaniswara, Performance comparison between differential evolution and bat algorithm in PID tuning for optimization of speed control on parallel hybrid electric vehicle, *Jurnal Teknik Elektro*, 6(1), 2024, 64-74.

[16] Y. K. Poudel and P. Bhandari, Control of the BLDC motor using ant colony optimization algorithm for tuning PID parameters, *Archives of Advanced Engineering Science,* 2(2), 2023, 108-113.

[17] Z. Shi, L. Zhao and Y. Shi, Starting control of free piston engine linear generator based on simulated annealing algorithm to optimize fractional PID, *Proceedings of the 8th International Conference on Control Engineering and Artificial Intelligence*, New York, USA, 2024, 114-119.

[18] N. J. Singh, V. Chopra and S. Pandey, Artificial bee colony with predator effect algorithm for proportional integral derivative controller tuning, *Journal of Vibration and Control*, 0(0), 2024.

[19] P. Khare, M. Raju, S. C. Gupta and H. Shukla, Liberalized automatic generation control of interconnected thermal-hydro-gas system using firefly algorithm optimized PID controller, *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 37(1), 2024, e3136.

[20] N. Genc and Z. S. Kalimbetova, Cuckoo optimization algorithm based fuzzy logic speed controller for BLDC motor, *Electric Power Components and Systems*, 52(11), 2024, 2065-2077.

[21] A. Rajendran, M. Karthikeyan and G. Saravanakumar, Implementation of FOPID controller with modified harmony search optimization for precise modelling and auto-tuning of nonlinear systems, *Automatika*, 65(3), 2024, 881-893.

[22] K. Jagatheesan, D. Boopathi, S. Samanta, B. Anand and N. Dey, Grey wolf optimization algorithm-based PID controller for frequency stabilization of interconnected power generating system, *Soft Computing*, 28(6), 2024, 5057-5070.

[23] M. Ristanović, Ž. Ćojbašić and D. Lazić, Intelligent control of DC motor driven electromechanical fin actuator, *Control Engineering Practice*, 20(6), 2012, 610-617.

[24] A. Mahdi, Optimization of PID controller parameters based on genetic algorithm for non-linear electromechanical actuator, *International Journal of Computer Applications*, 94(3), 2014, 11-20.

[25] A. Javadi, H. Haghighi, K. Pornpipatsakul and R. Chaichaowarat, Data-driven position and stiffness control of antagonistic variable stiffness actuator using nonlinear Hammerstein models, *Journal of Sensor and Actuator Networks*, 13(2), 2024, 29.

[26] A. M. Ahmed, A. Ali-Eldin, M. S. Elksasy and F. F. Areed, Brushless DC motor speed control using both PI controller and fuzzy PI controller, *International Journal of Computer Applications*, 109(10), 2015, 29-35.

[27] S. Tufenkci, B. Baykant Alagoz, G. Kavuran, C. Yeroglu, N. Herencsar and S. Mahata, A theoretical demonstration for reinforcement learning of PI control dynamics for optimal speed control of DC motors by using twin delay deep deterministic policy gradient algorithm, *Expert System with Applications*, 213, 2023, 119192.

[28] H. A. Mintsa, G. E. Eny, N. Senouveau and R. M. A. Nzué, Optimal tuning PID controller gains from Ziegler-Nichols approach for an electrohydraulic servo system, *Journal of Engineering Research and Reports*, 25(11), 2023, 158-166.

[29] K. Devendranath *et al.*, Position tracking performance with fine tune Ziegler-Nichols PID controller for electro-hydraulic actuator in aerospace vehicle model, *Journal of Physics: Conference Series (International Conference on Man Machine System)*, 2107(1), 2021, 12064.
[30] M. B. Omar, K. Bingi, B. Rajanarayan Prusty and R. Ibrahim, Recent advances and applications of spiral dynamics optimization algorithm: A review, *Fractal and Fractional*, 6(1), 2022, 27.
[31] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, 39(3), 2007, 459-471.