IET Control Theory & Applications

WILEY

IET The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Nonlinear Controller Design and its Optimization for Real-Time Implementation for a Servo Motor Drive

Emre Çelik[1] | Davut İzci[2] | Serdar Ekinci[3] | Erdal Bekiroğlu[4] | Diego Oliva[5] | Mahmoud Abdel-Salam[6] | Ghanshyam G. Tejani[7,8] | Seyed Jalaleddin Mousavirad[9]

[1]Department of Electrical and Electronics Engineering, Engineering Faculty, Düzce University, Düzce, Türkiye | [2]Department of Electrical and Electronics Engineering, Bursa Uludağ University, Bursa, Türkiye | [3]Department of Computer Engineering, Bitlis Eren University, Bitlis, Türkiye | [4]Department of Software Engineering, Engineering Faculty, Gazi University, Ankara, Türkiye | [5]Departamento de Ingeniería Electro-Fotónica, Universidad de Guadalajara, CUCEI, Guadalajara, Jalisco, Mexico | [6]Faculty of Computer and Information Science, Mansoura University, Mansoura, Egypt | [7]Department of Research Analytics, Saveetha Dental College and Hospitals, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, India | [8]Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan, Taiwan | [9]Department of Computer and Electrical Engineering, Mid Sweden University, Sundsvall, Sweden

Correspondence: Emre Çelik (emrecelik@duzce.edu.tr) | Seyed Jalaleddin Mousavirad (Seyedjalaleddin.mousavirad@miun.se)

Received: 11 September 2025 | **Revised:** 21 April 2026 | **Accepted:** 27 May 2026

ABSTRACT

A proportional-integral (PI) controller is still the workhorse in the industry due to its ease of commissioning and reliability. Therefore, this paper introduces an exponential PI (EXP-PI) controller as a potential alternative. In the proposed scheme, two tuneable EXP functions acting nonlinearly on the error and the rate of change of the error are incorporated in cascade with the PI control architecture. This controller is implemented as a speed controller on a permanent magnet DC motor drive system. Five recent intelligent algorithms, namely stochastic fractal search (SFS), snake optimizer (SO), dragonfly search algorithm (DSA), symbiotic organisms search (SOS) and reptile search algorithm (RSA) are employed to identify the best performer for calibrating the controller gains. According to the statistical results, SFS is found to provide controller gains of higher quality, reducing the designed cost function value to 48.19. The superiority of SFS is verified by the nonparametric Wilcoxon rank-sum test. Several experimental results with SFS-calibrated EXP-PI controller and other existing control schemes are presented using the DSP of TMS320F28335. The results show that our proposal performs better than its competing opponents in terms of various performance metrics, including integral-based error criteria, stability margin, overshoot and settling time for plants with and without dead time.

1 | Introduction

1.1 | Problem Statement

Direct current (DC) machines have been at the heart of numerous electromechanical systems for over a century, owing to their simplicity, ease of control and well-understood dynamics. With the advancement of power electronics and digital control technologies, DC motors have continued to find applications in areas demanding precise and efficient speed regulation [1–3]. However, in recent years, the widespread adoption of three-phase alternatives such as brushless DC motors (BLDCs) [4] and permanent magnet synchronous motors (PMSMs) [5] (which offer greater power density, improved reliability and minimal maintenance) has somewhat diminished the dominance of traditional DC machines in industrial settings. Despite this trend, permanent-magnet DC (PMDC) motors remain highly relevant, particularly in low-power applications, where their compact design, high efficiency and linear torque-speed characteristics make them ideal for robotics, small-scale automation and embedded drive systems [6]. In these applications, an improved and adequate speed control scheme is an unavoidable necessity. Further, from

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
© 2026 The Author(s). *IET Control Theory & Applications* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

the engineering commissioning point of view, the feasibility and practical aspects of the speed control scheme are another consideration that needs to be addressed carefully.

1.2 | Literature Review

Achieving precise and stable speed control in PMDC motors is critical for ensuring high-performance operation, particularly in real-time applications that demand fast response and robustness under varying load conditions. Recent studies have explored various control strategies and optimization frameworks to enhance the performance of PMDC motor drive systems. For example, Çelik et al. [7] proposed a nonlinear PI control structure with an exponential (EXP) gain block and optimized its parameters using the salp swarm algorithm, showing improved speed tracking and disturbance rejection through real-time experiments. Similarly, Chalardsakul et al. [8] employed a fuzzy-tuned PI-D controller and validated its effectiveness on a DSP-based platform using the TMS320F28379D microcontroller, revealing its superiority over classical PID in handling the nonlinear dynamics of PMDC motors. Though the performance of the above works seems comparatively promising, there is some room to elevate it further.

Several studies have also focused on parameter estimation techniques as a prerequisite for robust control. Çorapsız [9] compared nonlinear least squares, pattern search and simplex search algorithms for identifying controller gains under varying operating conditions, demonstrating that optimized parameter tuning significantly improves dynamic performance. Siddiqi et al. [10] introduced a data-driven approach using artificial neural networks for estimating electromechanical and electrical parameters of PMDC motors, highlighting the method's accuracy. However, the computational efficiency of the study remains a concern. Furthermore, Shary and Nekad [11] conducted a simulation study where they compared multiple metaheuristic algorithms for the tuning of PID controllers installed in the speed control loop of a PMDC motor. While the best algorithm was chosen on the basis of the reduced overshoot and shortened settling time, the performance of the resulting controller in real-time is questionable. They may likely yield performance poorer than expected. In a PV-powered DC motor application, a fuzzy cognitive network and a fuzzy controller were both exploited in [12] to control the speed of a shunt DC motor while retrieving the maximum power of the PV array under different environmental circumstances. Dhinakaran and Manamalli [13] have emphasized the performance of simplified predictive control and shifted dynamic matrix control on a PMDC motor drive. Simulation results of this study demonstrated satisfactory performance of the proposed technique with regard to the considered performance measures such as integral-based error rates, rise time and settling time. However, the lack of experimental verification and formal stability analysis places the practical significance of the work in the question. In a hardware-oriented study, Mishra et al. [14] proposed a low-power digital hardware accelerator for auxiliary PMDC motors in electric three-wheelers, where a quadral-duty PWM technique was adopted, particularly for lowering the speed ripple. Since a step-down converter was employed in the work, the performance of the proposed approach was shown only under one-quadrant operation. Recently, a two-degree-of-freedom (2DOF) PID controller has been proposed by [15] for the purpose of accurate speed regulation of a DC motor. To reach the two key requirements of precise set point tracking and efficient rejection of load disturbances, the controller parameters were tuned using a new hybrid algorithm named chaotic state of matter search integrated with elite opposition-based learning. Results of this work affirmed the performance of the proposed approach.

The literature review can also be extended to include other control schemes developed for BLDCs and PMSMs, thereby providing further insight into the relevant domain. Öztürk and Çelik [16] implemented the speed control of PMSM using a fuzzy controller, where a genetic algorithm was employed to obtain the knowledge base of the controller without human experience or interpretation. Çelik and Karayel [17] introduced a cascade one proportional derivative incorporating filter (1PDf)-proportional integral (PI) controller (c-1PDf-PI) to effectively tackle the speed control issue of a BLDC motor. As per the experimental results obtained using the DSP of TI TMS320F28335, superior performance was attained against complicated control designs. Robust speed control of PMSM using the 2DOF controller was accomplished in [18], where a change of five times the nominal inertia was considered. According to the results under parameter variation and load disturbances, the proposed 2DOF controller was found to perform significantly more robustly than a classical PI controller. More recently, other advanced control schemes based on event-triggered generalized super-twisting sliding mode control [19], a data-driven RBF neural network approach [20] and model predictive control [21] for PMSMs have been recognized in the literature. Although the performance improvements reported in these studies are well acknowledged, they yield expensive control laws, often causing an impractical solution for engineering commissioning. Besides, since their computational budget is raised, they demand a high-speed microprocessor like FPGA. A comparison of the current and published works is presented in Table 1.

The reviewed literature collectively highlights a growing emphasis on intelligent control strategies and optimization techniques for PMDC motor systems. These studies demonstrate the potential of metaheuristic algorithms and nonlinear control approaches to improve system performance. However, a closer look also reveals several important limitations. A significant portion of the existing work remains confined to simulation-based evaluations without real-time hardware validation. Moreover, many of the proposed methods rely on conventional PI control structures, which, while widely used for their simplicity and ease of implementation, often fall short in handling nonlinearities, external disturbances and rapidly changing reference signals. Although intelligent tuning methods can enhance controller performance, their integration into practical, real-time systems is still relatively limited in the current literature.

1.3 | Motivation and Contributions

To address these challenges, the present study introduces an EXP-PI controller—a structure specifically designed to improve adaptability and dynamic responsiveness in PMDC motor drive systems. This controller is motivated by the observation that fixed-gain PI controllers often struggle to balance fast transient performance with steady state stability in the presence

TABLE 1 | Comparison of the current and published works.

References	Optimization algorithm	Controller structure/validation	Controller complexity	Signals used
[7]	Salp swarm algorithm	Nonlinear PI/Experimental	Low	Speed error
[8]	–	Fuzzy tuned PI-D/Simulation	Moderate	Speed error
[9]	Nonlinear least squares Simplex search Pattern search	PID/simulation	Low	Speed error
[11]	Ant colony optimization Grey wolf optimization Flower pollination algorithm	PID/simulation	Low	Speed error
[12]	–	Fuzzy cognitive network/experimental	Moderate	Speed error Change of speed error
[13]	–	Simplified predictive control and shifted dynamic matrix control/Simulation	High	Speed error
[14]	–	Quadral-duty digital pulse width modulation/experimental	Low	Speed error
[15]	Chaotic state of matter search integrated with elite opposition-based learning	2DOF PID/simulation	Low	Speed error
[16]	Genetic algorithm	Fuzzy PI/simulation	Moderate	Speed error Change of speed error
[17]	Snake optimizer	$1PD_f$-PI/experimental	Low	Speed error
[18]	–	2DOF/experimental	Moderate	Speed error
[19]	–	Event-triggered generalized super-twisting sliding mode control/experimental	High	Speed error
[20]	–	Data-driven RBF neural network approach/experimental	High	Speed error
[21]	–	Model predictive control/experimental	High	Speed error
[22]	Symbiotic organism search	PI/experimental	Low	Speed error
[23]	Improved variant of snake optimizer	PID/simulation	Low	Speed error Change of speed error
This work	Stochastic fractal search	PI/experimental	Low	Speed error Change of speed error

of load variations and dynamic set point changes. The EXP-PI structure enhances conventional PI control by responding more aggressively to large deviations while preserving smooth and stable behaviour as the system approaches the desired operating point. In doing so, the proposed method not only overcomes key limitations associated with conventional PI designs but also provides a practical and implementable solution for real-time motor control applications.

The proposed EXP-PI controller is governed by parameters associated with tuneable EXP functions and the PI controller itself, all of which must be carefully and jointly calibrated to ensure optimal performance. Instead of relying on time-consuming trial-and-error methods, the tuning task is framed as an optimization problem, with a composite cost function that combines the integral of squared error and the integral of absolute error [22]. This formulation is evaluated under a wide range of speed references and load torques, ensuring that the controller maintains robust performance across different operating scenarios. To solve the optimization problem, five recent nature-inspired metaheuristic algorithms are applied under identical conditions, which are stochastic fractal search (SFS) [24], snake optimizer (SO) [25], dragonfly search algorithm (DSA) [26], symbiotic organisms search (SOS) [27] and reptile search algorithm (RSA) [28]. Each of these algorithms is grounded in biological or physical metaphors and has demonstrated strong potential in tackling complex, nonlinear and high-dimensional optimization tasks. Their application in this study facilitates a comprehensive evaluation of their suitability for parameter tuning in motor control applications.

In this paper, the EXP-PI controller was implemented in real-time on a PMDC motor testbed comprising a DSP of TMS320F28335, a H-bridge power converter and a pair of PMDC motors and generators from Sanyo Denki. The permanent magnet DC generator (PMDG) mechanically coupled to the PMDC motor is used to exert load torque disturbances on the motor shaft. Experimental validation was carried out through a series of tests, including step speed reference tracking, trajectory tracking and torque disturbance rejection. The main contributions of this study can be outlined as follows:

- Two tuneable EXP functions acting nonlinearly on the signals of error and the rate of change of the error are integrated into the classical PI framework to develop the EXP-PI controller for performance improvement.
- The controller tuning problem is formulated as a multi-scenario optimization task, accounting for a variety of operating conditions to improve system robustness.
- A comparative analysis is conducted using five recent meta-heuristic algorithms, namely SFS, SO, DSA, SOS and RSA to assess their performance in calibrating the EXP-PI controller gains.
- The best-performing optimizer is employed for real-time implementation and the proposed technique is validated experimentally on a physical motor drive system using DSP-based hardware. Further, two second-order plants with delay are also considered to see how our proposal deals with dead time.
- The results are contrasted against those with the existing control schemes to demonstrate the true effectiveness of our proposal.

1.4 | Paper Organization

The PMDC motor model is presented in Section 2. After the EXP-PI controller's configuration is presented in Section 3, Section 4 reviews and compares the optimization algorithms and explains how the controller gains are tuned from the optimization point of view. A series of test results for the PMDC motor drive are shown in Section 5 and a frequency-domain performance analysis of different controllers is established in Section 6 for the stability analysis. Eventually, after revealing the performance of our proposal against the dead time in Section 7, the paper is concluded in Section 8.

2 | PMDC Motor Model

The dynamic model of a DC motor is created by an armature winding resistance R_a, a self-inductance L_a and an induced electromotive force (emf). To keep the parameters time-invariant and accordingly simplify the model, the effects of magnetic saturation and armature reaction are omitted in the paper. The armature voltage relationship is written as [7]

$$\upsilon = e + R_a i_a + L_a \frac{di_a}{dt} \tag{1}$$

where υ is the voltage applied to the armature circuit and i_a is the current flowing through this circuit. The induced emf, which is also referred to as the counter or back emf, is expressed as [7]

$$e = K\emptyset_f \omega_m \tag{2}$$

where $\emptyset_f$ is the field flux created by the field winding, ω_m is the angular speed and K is a positive scalar. Because the excitation poles in a PMDC motor are formed by permanent magnets, its field flux is constant, thus the induced emf is represented by [7]

$$e = K_b\, \omega_m \tag{3}$$

where K_b is the constant of induced emf represented by [7]

$$K_b = K\emptyset_f \text{ volt} \cdot \text{s/rad} \tag{4}$$

The electromagnetic torque is calculated using the armature power P_a divided by ω_m as [7]

$$T_e = \frac{P_a}{\omega_m} = \frac{ei_a}{\omega_m} = K_b\, i_a \tag{5}$$

Notice that the torque constant is the same as the emf constant for a constant-flux machine. The electromechanical modelling is presented next [7]

$$J\frac{d\omega_m}{dt} + B\omega_m + T_L = T_e \tag{6}$$

where T_e is the electromagnetic torque, T_L is the load torque, J and B are positive scalars denoting the moment of inertia and the viscous friction coefficient of the whole rotating system.

3 | Configuration of EXP-PI Controller

The EXP-PI controller block diagram is depicted in Figure 1. The innovative EXP block has appeared in, where it is connected in cascade with a classical PID controller to obtain an EXP-PID controller. The performance of this controller as a supplementary controller is tested over a variety of electric power systems. However, its performance has been demonstrated only through the simulation results, making the practical feasibility and effectiveness of the controller questionable. On the other hand, another variant for the nonlinear PI controller is presented in [7], where the EXP block acts only on the error signal by skipping its derivative. Within this scope, the EXP-PI controller we propose in this paper is novel and has not been the focus of any other previous research.

As seen in Figure 1, both the error and the rate of change of error are used in the controller. They are weighted by τ_1 and τ_2, then passed to the EXP functions and finally scaled by G_1 and G_2, respectively. The final signals $\delta_1(t)$ and $\delta_2(t)$ are then summed up to obtain $\delta\,(t) = \delta_1\,(t) + \delta_2(t)$. Here, $\delta(t)$ is a nonlinear reflection of the error signal $e(t)$. $\delta(t)$ is the input to the PI controller, which delivers the required control action to drive the system.

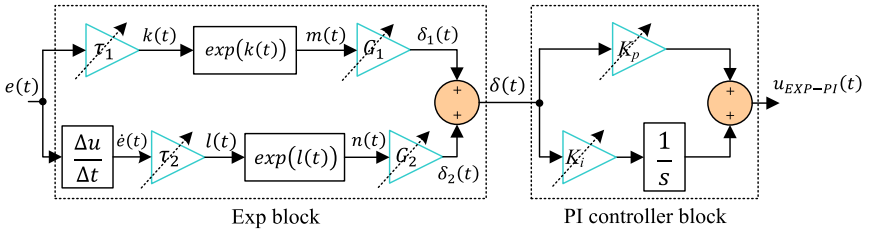


FIGURE 1 | The EXP-PI controller block diagram.

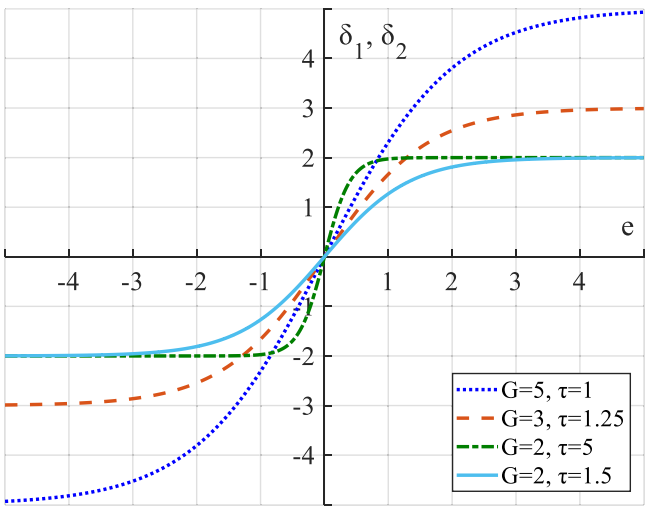


FIGURE 2 | Typical variations of δ_1 and δ_2 as a function of e for different values of G and τ.

The mathematical descriptions for $\delta_1(t)$ and $\delta_2(t)$ are given as [23].

$$\begin{aligned} \delta_1\ (t) &= G_1\ \left(\frac{2}{1+e^{-\tau_1 e(t)}}-1\right) \\ \delta_2\ (t) &= G_2\ \left(\frac{2}{1+e^{-\tau_2 \dot{e}(t)}}-1\right) \end{aligned} \tag{7}$$

where $e(t)$ is the error, $\dot{e}(t)$ is the rate of change of error with respect to time and G_1, G_2, τ_1, τ_2 are user-defined positive scalars. $\delta_1(t)$ and $\delta_2(t)$ are bounded by $|G_1|$ and $|G_2|$, respectively, when $|e| = \infty$. Furthermore, when $e = 0$, $\delta_1\ (t) = \delta_2\ (t) = 0$. τ_1 and τ_2 are responsible for altering the rate of variation of δ_1 and $\delta_2(t)$, respectively. Typical variations of δ_1 and δ_2 as a function of e for different values of G and τ are shown in Figure 2. It is noticeable from Figure 2 that G is used in the function to obtain the 'scaled' error. On the other hand, the sharpness of the function changes by τ. Bigger values of τ increase the sharpness, and make the function more similar to a step-wise function.

Adopting the proposed configuration in Figure 1 allows the controller parameters to adapt themselves to variable conditions in the system. In the EXP-PI controller, in addition to K_p and K_i, other parameters such as G_1, G_2, τ_1 and τ_2 are also important parameters affecting the control performance closely. All these parameters require joint calibration, using a suitable intelligent metaheuristic algorithm, in order to reach the most possible performance from the controller.

4 | Improved Speed Servo System

In this research, a classical PI controller is considered to be incorporated in cascade with the proposed method. In this controller, the PI controller law can be written as

$$u_{\mathrm{EXP-PI}}\ (t) = K_{\mathrm{p}}\ \delta\ (t) + K_{\mathrm{i}} \int_0^t \delta\ (t)\, dt \tag{8}$$

To protect against the iterated integral overflow, the time derivative of the two sides of Equation (8) is taken as

$$\frac{u_{\mathrm{EXP-PI}}\ (k) - u_{\mathrm{EXP-PI}}\ (k\ -\ 1)}{T} = K_{\mathrm{p}}\ \frac{\delta\ (k)\ -\ \delta\ (k\ -\ 1)}{T} + K_{\mathrm{i}} \delta\ (k) \tag{9}$$

where T is the sampling time with $T > 0$. Multiplying both sides of Equation (9) by T, we obtain

$$u_{\mathrm{EXP-PI}}\ (k) - u_{\mathrm{EXP-PI}}\ (k-1) = K_{\mathrm{p}}\ [\delta\ (k) - \delta\ (k-1)] + K_{\mathrm{i}} T \delta\ (k) \tag{10}$$

In Equation (10), the term $\delta(k) - \delta(k-1)$ shows the change in δ. Similarly, $u_{\mathrm{EXP-PI}}(k) - u_{\mathrm{EXP-PI}}(k-1)$ is the change obtained for the controller output $u_{\mathrm{EXP-PI}}$. Therefore, Equation (10) can be rewritten as

$$\Delta u_{\mathrm{EXP-PI}}\ (k) = K_{\mathrm{p}}\ \Delta\delta\ (k) + K_{\mathrm{i}} T \delta\ (k) \tag{11}$$

Note that in Equation (11), there is no accumulation term resulting from the integral operation. Therefore, using this form of the PI control rule, the integral overflow problem is eliminated. However, we find that $\Delta u_{\mathrm{EXP-PI}}$ is the change in the control signal. To obtain its true value, it is necessory to add $\Delta u_{\mathrm{EXP-PI}}$ to the previous control action as follows:

$$u_{\mathrm{EXP-PI}}\ (k) = \Delta u_{\mathrm{EXP-PI}}\ (k) + u_{\mathrm{EXP-PI}}\ (k-1) \tag{12}$$

In the next subsections, the calibration problem of the controller parameters is addressed through formulating it as an optimization problem.

4.1 | Cost Function

Algorithms search for the optimal solution by minimizing a problem-specific cost function. As previously discussed, the EXP-PI controller has a total of 6 parameters, namely $G_1, G_2, \tau_1, \tau_2, K_p$ and K_i, each affecting the overall performance to a varying extent. This problem is difficult to solve manually, because the problem has many possible solutions. Deciding the optimal or even the near-optimal solution in a reasonable time is undoubtedly impossible. As a result, we address the problem by considering

it from the optimization point of view. During optimization, as a figure of demerit (FOD), the following composite cost function comprising integral squared error (ISE) and integral absolute error (IAE) is employed [22].

$$\text{FOD} = \omega_1 \int_0^{t_s} e_s^2(t)\,dt + \omega_2 \int_0^{t_s} |e_s(t)|\,dt \tag{13}$$

where $e_s(t)$ represents the speed error between the reference and measured speed at time t, t_s is the simulation time in seconds, ω_1 and ω_2 are weight coefficients that decide the importance of each term in Equation (13). After many repetitions of optimization, promising results are achieved with $\omega_1 = 0.01$ and $\omega_2 = 1$.

Consequently, the problem of calibrating the controller parameters is formulated as follows:

Minimize FOD, subject to:

$$\begin{aligned} K_p^{\min} \le K_p \le K_p^{\max} \\ K_i^{\min} \le K_i \le K_i^{\max} \\ G_1^{\min} \le G_1 \le G_1^{\max} \\ G_2^{\min} \le G_2 \le G_2^{\max} \\ \tau_1^{\min} \le \tau_1 \le \tau_1^{\max} \\ \tau_2^{\min} \le \tau_2 \le \tau_2^{\max} \end{aligned} \tag{14}$$

Each controller parameter in Equation (14) is lower-bounded by the superscript min and upper-bounded by the superscript max. These limits are chosen as [0, 20] for all the parameters in order to search over a broader range. As the optimizer, several recent algorithms such as SFS, SO, DSA, SOS and RSA are employed to identify the best-performing algorithm.

4.2 | Review of SFS, SO, DSA, SOS and RSA

Today, numerous applications require optimization across many scientific areas such as engineering, economics and business. As technology advances and the world population grows, the importance of optimization has risen due to the growing complexity of problems. Particularly, optimization becomes increasingly important in the presence of limited resources and time. Optimization may be needed to minimize cost, risk or time and maximize profit, efficiency or quality. Computational intelligence has played a pivotal role as the backbone of optimization research for the last two decades. This way, various metaheuristic algorithms were developed by drawing inspiration from nature and deployed in different fields of optimization research. The popularity of these algorithms stem from their simplicity flexible nature and low computational complexity [29]. In this context, five recent metaheuristic optimizers such as SFS) [24], SO [25], DSA [26], SOS [27] and RSA [28]) have attracted attention for their effective performance across benchmark and engineering optimization tasks. Therefore, these algorithms have been adopted to perform performance analyses in this study. In this context, this section provides a brief review of these algorithms with respect to their underlying motivations, inspiration sources and mathematical formulations. For additional information, interested readers are recommended to kindly refer to the original works.

4.2.1 | Stochastic Fractal Search

The SFS algorithm is a nature-inspired metaheuristic that draws from the stochastic dynamics of fractal growth processes, particularly diffusion-limited aggregation and dielectric breakdown [24]. It follows a structured sequence of steps, emphasizing a balanced interplay between exploitation (local refinement) and exploration (diversification).

In the first step, a population of N particles (candidate solutions) is randomly generated within the predefined minimum and maximum limits using $P_j = LB + \varepsilon \times (UB - LB)$, where $\varepsilon \in [0, 1]$ is a random number and LB and UB denote the problem's minimum and maximum parameter limits, respectively. The second step is related to the fitness evaluation, where each particle's cost function value is computed to assess its quality. The best-performing solution is tracked as the current global best. In the following step, the exploitation (diffusion process) occurs. In this localized search phase, each particle diffuses around itself to refine its current position. A Gaussian walk is employed in this stage:

$$GW_1 = \text{Gaussian}(\mu, \sigma) + \varepsilon \times (BP - P_i) \tag{15}$$

Here, P_i is the current particle, BP is the best-known position and σ is a scale factor that decreases over time to intensify the search near good solutions. Only the best particle generated from the diffusion is retained. To avoid premature convergence, SFS incorporates a two-level random update mechanism in the exploration (updating process). In this mechanism, component-wise update and vector-wise update are performed. In the component-wise update, each variable within a particle may be adjusted using information from others (with a probability proportional to its rank) using $P_i'(j) = P_r(j) - \varepsilon \times (P_t(j) - P_i(j))$, where P_r and P_t are randomly chosen particles and $\varepsilon \in [0, 1]$. In the vector-wise update, entire particles are modified using either $P_i' = P_i - \varepsilon \cdot (P_r - BP)$ or $P_i' = P_i + \varepsilon \times (P_r - P_t)$ depending on whether if rand ≤ 0.5. Such an update mechanism encourages larger exploratory jumps based on interactions with other particles and the global best. Particles that improve upon their previous solutions are retained, while the best global solution is always preserved to guide the search. The above explained entire process repeats until a predefined number of generations is reached or a satisfactory solution is found. SFS has demonstrated competitive results on a broad suite of unconstrained and constrained benchmark functions, particularly excelling in avoiding local optima and promoting convergence accuracy [24]. Its ability to balance exploration and exploitation makes it suitable for various engineering design tasks [30–34].

4.2.2 | Snake Optimizer

The SO is a bio-inspired metaheuristic that simulates the reproductive and foraging behaviour of snakes [25]. Its search process is governed by environmental cues such as temperature and food quantity. These determine whether the algorithm should explore new regions or exploit known promising areas. In nature, snake mating and foraging behaviours are highly adaptive. In that sense, mating occurs only when food is abundant and the environment

is cool. Such behavioural rules form the basis of SO's algorithmic logic.

SO begins by generating a population of candidate solutions randomly distributed across the problem space. This ensures diversity in the initial generation. The initialization is given by $X_i = X_{min} + r \times (X_{max} - X_{min})$, where $r \in [0, 1]$ is a uniform random number and X_{min} and X_{max} denote the minimum and maximum limits of the solution space. The population is then evenly split into male and female subgroups, each evolving semi-independently based on group-specific behavioural models. At each iteration, two dynamic environmental parameters (temperature and food quality) are calculated. Temperature simulates seasonal shifts, decreasing over time, whereas food quantity reflects the availability of resources in the search space. These parameters determine which behavioural phase the algorithm should follow. Depending on temperature and food availability, the algorithm switches between exploration (when food is scarce) and exploitation (when food is abundant). In exploration, individuals move randomly in search of better regions. On the other hand, in exploitation, snakes either feed or engage in mating competitions based on the temperature. A key behavioural rule used during exploitation updates the position of a snake as follows:

$$X_i^{t+1} = X_{food} \pm c \times \text{Temp} \times \text{rand} \times \left(X_{food} - X_i^t\right) \tag{16}$$

Here, X_{food} represents the location of the best solution found so far, Temp is the current temperature and the sign ($\pm$) introduces diversity by allowing bidirectional updates. The last step of the SO mimics mating and offspring. When conditions are ideal (cool temperature and sufficient food), mating behaviour is triggered where males compete for the best females, females selectively mate to produce offspring. The worst-performing individuals may be replaced by these offspring to preserve population diversity and avoid stagnation. The explained structure of the SO has initially been benchmarked using CEC 2017 functions and several constrained engineering problems. It consistently outperformed or remained on par with state-of-the-art algorithms [25]. Besides, it has been demonstrated to perform well on different problems [17, 35, 36], as well.

4.2.3 | Dragonfly Search Algorithm

The DSA is a metaheuristic algorithm inspired by the swarming behaviours of dragonflies in nature [26]. Dragonflies exhibit two primary types of swarming namely static swarming and dynamic swarming. In static swarms, small and localized groups hover and move abruptly for hunting, which is analogous to exploration. In dynamic swarms, large and cohesive swarms migrate in a consistent direction so as to emphasize exploitation. DSA emulates these patterns to guide artificial dragonflies through a solution space, alternating between diverse search and focused refinement.

DSA simulates collective swarm behaviour by modelling five fundamental social interactions observed in dragonflies: separation, alignment, cohesion, attraction toward food and distraction from enemies. Separation helps individuals avoid overcrowding by maintaining distance from nearby neighbours, while alignment encourages them to synchronize their movement with surrounding individuals. Cohesion guides each dragonfly toward the centre of its local group, promoting swarm integrity. Meanwhile, attraction toward food sources directs the swarm toward promising areas in the search space and distraction from enemies ensures avoidance of non-optimal or deceptive regions. These behavioural components are adaptively weighted and combined to form the step vector, which determines both the direction and magnitude of movement (functioning analogously to the velocity vector in particle swarm optimization [37]). A key formula for the step update is:

$$\Delta X^{t+1} = s \times S_i + \alpha \times A_i + c \times C_i + f \times F_i + e \times E_i + w \times \Delta X^t \tag{17}$$

where s, a, c, f, e are the behavioural weights, S_i, A_i, C_i represent separation, alignment and cohesion vectors, F_i and E_i denote attraction to food and repulsion from the enemy, w is the inertia weight, X^t is the previous step vector. The position is then updated as $X^{t+1} = X^t + \Delta X^{t+1}$. DSA also adjusts swarm behaviour over iterations. For example, during early stages, high alignment and low cohesion promote wide-ranging exploration. Toward later stages, low alignment and high cohesion favour focused convergence. Additionally, neighbourhood size increases over time, allowing the swarm to behave more like a unified group as convergence is approached. When no neighbours are found, a Levy flight random walk is used to encourage broader exploration by following $X^{t+1} = X^t + \text{Levy}(d) \times X^t$. Through the described operating structure, DSA has demonstrated good performance on a variety of different optimization tasks [38–43].

4.2.4 | Symbiotic Organisms Search Algorithm

The SOS algorithm is inspired by the natural symbiotic relationships among organisms in ecosystems [27]. In nature, survival and adaptation often rely not on solitary behaviours, but on interactive biological relationships including mutualism, commensalism and parasitism. SOS converts these relationships into algorithmic strategies that collectively guide a population of candidate solutions toward an optimal solution in the search space.

In SOS, a population, named an ecosystem, is initially generated with randomly distributed organisms. Each organism represents a potential solution and is evaluated using the objective function to determine its fitness, analogous to biological adaptability. Then, the algorithm iteratively improves the ecosystem through three interaction phases. In the first phase (mutualism), two randomly selected organisms engage in a mutually beneficial interaction, updating their positions in a way that increases the fitness of both. The shared benefit is modelled via a mutual vector and the position of each organism is updated as follows:

$$X_i^{new} = X_i + \text{rand}\,(0,\ 1) \times (X_{best} - \text{Mutual}_{Vector} \times BF_1) \tag{18}$$

$$X_j^{new} = X_j + \text{rand}\,(0,\ 1) \times (X_{best} - \text{Mutual}_{Vector} \times BF_2) \tag{19}$$

where X_{best} is the current best organism, $\text{Mutual}_{Vector} = (X_i + X_j)/2$, BF_1 and BF_2 are random integers (either 1 or 2), representing benefits. Only fitter organisms are retained

after this interaction. In the second phase (commensalism phase), one organism (e.g., X_i) benefits from another (X_j), without affecting the latter. The update rule seeks to enhance the fitness of X_i using the difference between X_{best} and X_j using $X_i^{new} = X_i + \text{rand}(-1, 1) \times (X_{best} - X_j)$. If the modified X_i has a better fitness, it replaces the previous one. In the third phase (parasitism phase), an organism X_i generates a parasite vector by duplicating and randomly modifying itself. This parasite attempts to invade a randomly selected host X_j. If the parasite has a higher fitness, it replaces the host; otherwise, it is discarded. This mechanism introduces diversity and prevents stagnation. The algorithm initially demonstrate high performance on both benchmark and structural engineering optimization problems, particularly in terms of robustness, convergence speed and solution diversity [27]. Its efficacy has then further been shown on different problems [44–48].

4.2.5 | Reptile Search Algorithm

The RSA is a recently introduced metaheuristic inspired by the encircling and hunting behaviours of crocodiles [28]. Known for their intelligence, namely coordination and adaptable predatory strategies, crocodiles engage in both solitary and collaborative hunts depending on the context. RSA captures these behaviours in a mathematical framework that alternates between exploration and exploitation phases. Therefore, the algorithm is modelled after two key phases, namely the encircling phase and the hunting phase. The encircling phase involves high walking and belly walking strategies that help crocodiles scan a wide area for prey to promote exploration. On the other hand, the hunting phase represents focused exploitation, where crocodiles coordinate or cooperate to attack. This is achieved through hunting coordination (synchronized movement) and hunting cooperation (alternating attack roles). The search dynamics in RSA are structured to adapt over time, gradually shifting from exploration to exploitation as iterations progress.

RSA begins by initializing a population of candidate solutions X, randomly sampled within the defined search bounds using $x_{ij} = \text{rand} \times (UB - LB) + LB$. Here, x_{ij} is the jth dimension of the ith candidate, while LB and UB denote the minimum and maximum limits of the search space. In the encircling phase, the population explores the search space using two movement strategies:

- High walking is conducted in early iterations ($t \leq (T/4)$), simulating fast, high-stance movement.
- Belly walking is applied in intermediate iterations ($(T/4) < t \leq (T/2)$), representing stealthy, close-to-ground exploration.

The candidate solution's position is updated using a biologically inspired rule. One form of this update (during high walking) is:

$$x_{ij}^{t+1} = \text{Best}_j(t) - \eta_{ij}(t) \times \beta - R_{ij}(t) \times \text{rand} \tag{20}$$

where $\text{Best}_j(t)$ is the best solution found so far in the jth dimension, $\eta_{ij}(t)$ is a dynamic hunting operator, $R_{ij}(t)$ reduces the search area to enhance convergence and β is a fixed sensitivity parameter. These mechanisms generate diverse candidates and prepare the algorithm to transition into exploitation. When $t > (T/2)$, the algorithm switches to the hunting phase to satisfy exploitation:

- Hunting coordination is performed in $(T/2) < t \leq (3T/4)$, where individuals are aligned with the best-known solutions.
- Hunting cooperation is engaged in $(3T/4) < t \leq T$, involving a cooperative adjustment of positions.

One of the key equations for this phase is:

$$x_{ij}^{t+1} = \text{Best}_j(t) \times P_{ij}(t) \times \text{rand} \tag{21}$$

Here, $P_{ij}(t)$ is the relative difference between the current and best solutions, promoting intensified search near the optimal region. RSA has been initially benchmarked on over 70 test functions and multiple real-world engineering problems [28]. The obtained results confirmed its superiority over several recent algorithms, validating its robustness and global convergence capacity. Its performance was then further demonstrated on several different optimization problems [49–51].

4.2.6 | Optimization of Controller Parameters

Based on the drive system model and the motor model derived, a series of simulations have been performed on a computer at several operating speeds and different loading scenarios of the motor. After the controller parameters are calibrated separately by utilizing different algorithms, they are then inserted in the DSP to assess their performance in practice.

As the EXP-PI controller has 6 parameters, each search agent in the selected algorithms consists of these parameters. Each algorithm begins its search by a number of search agents randomly initialized within the specified limits. In the simulation model, each candidate solution or the controller is utilized and it will provide a different speed response, with a unique value of cost function calculated by Equation (13). After setting a fitness value, the search agents are sent back to the optimization module to be modified and evolved for the next iterations. Here, for each fitness evaluation, the load torque and speed reference of the motor are changed from 0–1.2 Nm and from 0–2750 rpm, in steps of 0.2 Nm and 250 rpm, respectively, where the maximum limits depend on the motor nameplate. Thus, there exist 84 distinct working conditions and the final cost is computed by taking the average of these 84 costs, which is mathematically defined by

$$J = \left(\sum_{i=1}^{NC} \text{FOD}_i \right) \Big/ NC \tag{22}$$

where NC is the number of conditions that is equal to 84 and J is the average cost. Paying heed to the average cost (rather than a single cost calculated in one operating condition only) maintains the robustness and effectiveness of solutions across changeable operating conditions. The tuning procedure for the controller parameters described above is illustrated in Figure 3.

In line with the above-described procedure, SFS, SO, DSA, SOS and RSA are separately applied to the problem using a population size of 30 and maximum number of fitness function evaluations

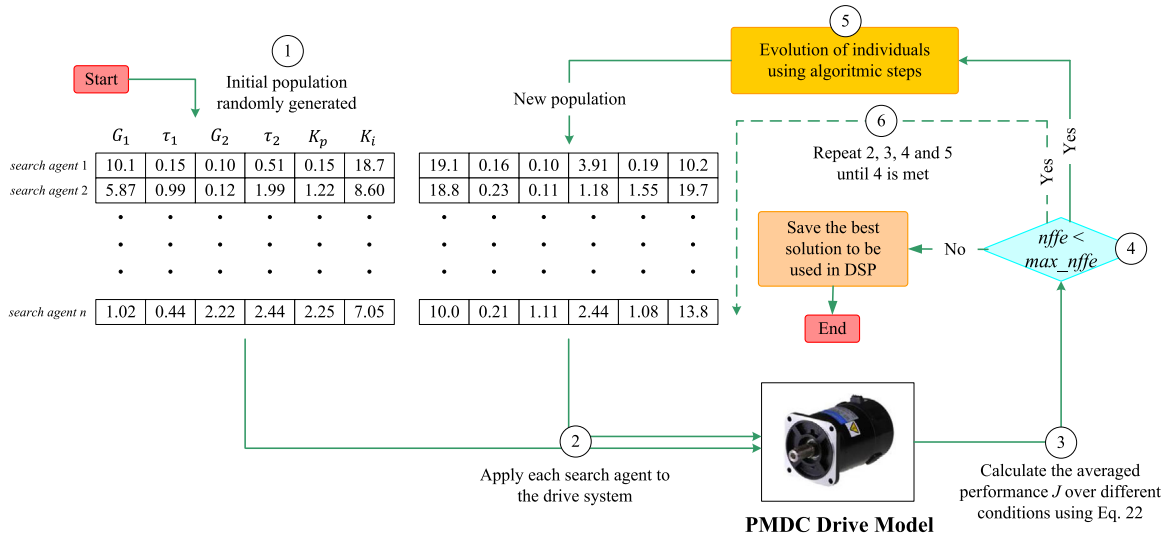


FIGURE 3 | Block diagram for the tuning procedure of controller parameters.

TABLE 2 | Controlling parameters of algorithms.

Algorithm	Parameter	Value
SFS	Maximum diffusion number (MDN)	1
SO	–	–
DSA	c	$0.1 - \text{iter} \times (0.1 - 0)/(\text{max_iter}/2)$
	Swarming factors:	
	s	$2 \times \text{rand} \times c$
	a	$2 \times \text{rand} \times c$
	c	$2 \times \text{rand} \times c$
	f	$2 \times \text{rand}$
	w	$0.9 - \text{iter} \times (0.9 - 0.4)/\text{max_iter}$
SOS	–	–
RSA	α	0.1
	β	0.005

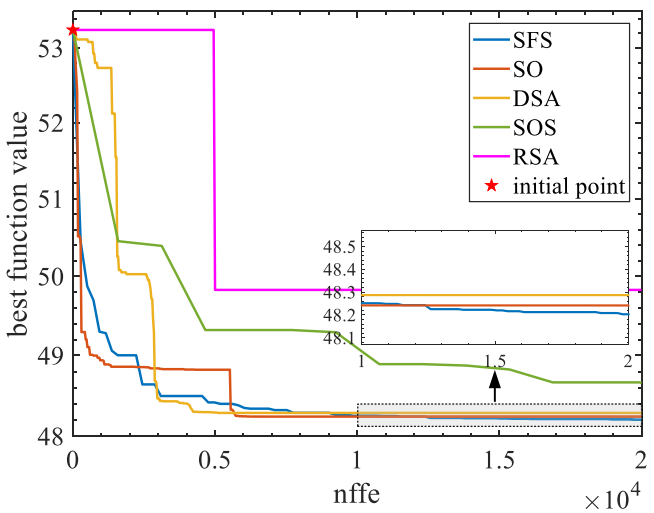


FIGURE 4 | Converging capacities of the algorithms during the minimization of J.

(max_nffe) of 20,000. Other controlling parameters unique to these algorithms are presented in Table 2. Note that they are the values recommended in the original works.

The sampling time set for the motor drive system is 0.01 ms and the system is simulated for 3 s to calculate the cost function. The best parameters of the EXP-PI controller along with the associated J value found using different algorithms are listed in Table 3. As expected, each algorithm has provided different controller parameters and thus the value of J differs. The minimum value of J belongs to SFS, therefore it ranks first, which is followed by SO, DSA, SOS and RSA, in order. On the other hand, RSA has the worst performance in this problem. It is also to be noted that J is the result of a total of 84 costs averaged. Therefore, although all the algorithms may seem to perform similarly, a small difference in J yields a significant difference in performance while the system is functioning around a certain point.

As the metaheuristic algorithms are stochastic in nature, each algorithm is independently run 51 times and the average performance of these runs is assessed in Table 4. Each row shows the best, worst, median and standard deviation of the best fitness values found at the last iteration. Although these performance measures offer us a confident comparison, a nonparametric statistical test named the Wilcoxon rank-sum test has been conducted at a 5% of significance level to see whether the obtained improvements are significant. The associated p-values are included in the last row of Table 4, where N/A is placed to refer to 'not applicable', because comparing the best algorithm with itself in the rank-sum test is not applicable. According to this test, p-values smaller than 0.05 signify a rejection of the null hypothesis, which is, in other words, strong evidence of remarkable performance. It is noticeable from Table 4 that the minimum values of best, worst, median, mean and std belong to SFS, so it has the highest performance of all. The respective p-values also affirm that these superiorities are statistically significant.

The converging capacities of the algorithms are also analysed in Figure 4. To have a fair comparison and avoid biased conclusions,

TABLE 3 | Best EXP-PI controller parameters found by different algorithms.

Algorithm	τ_1	G_1	τ_2	G_2	K_p	K_i	J (Equation 22)
SFS	0.1569	19.9952	0.1076	0.1000	0.1922	19.9759	**48.19**
SO	0.1460	19.9999	4.9899	0.1000	0.1922	19.9999	48.24
DSA	0.3696	20.0000	1.5211	0.1000	0.1922	20.0000	48.29
SOS	0.2029	19.9922	3.2861	0.1003	0.1895	19.9696	48.66
RSA	0.2136	18.1522	2.5233	0.1005	0.2214	17.6592	49.83

Note: *Better results are bolded.

TABLE 4 | Statistical results of the compared algorithms.

Metric	RSA	SOS	DSA	SO	SFS
Best	49.8300	48.6601	48.2990	48.2403	**48.1914**
Worst	51.7792	51.6274	50.8743	48.2993	**48.2171**
Median	49.8805	48.7401	48.5470	48.2481	**48.1956**
Mean	50.1810	48.9989	48.8281	48.2507	**48.1966**
Std.	0.56949	0.66933	0.68244	0.01399	**0.00511**
p-values	3.0199E-11	3.0199E-11	3.0199E-11	3.0199E-11	N/A

Note: *Better results are bolded.

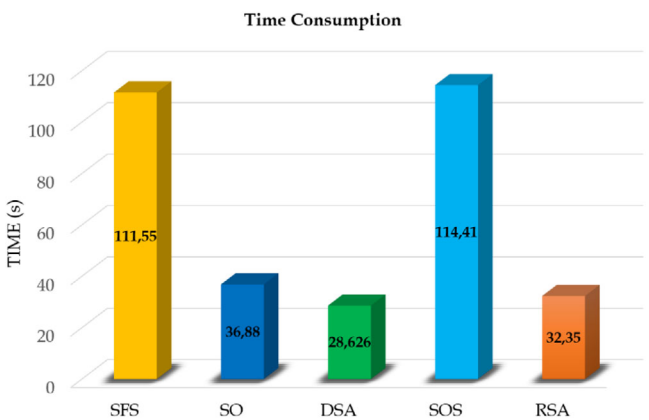


FIGURE 5 | Execution time in seconds per iteration.

all algorithms start from the same initial point (marked by the red pentagram). Critical analysis of Figure 4 reveals that SFS is capable of continuously improving the best solution gradually till the end of the optimization while the others could not. This is clear in the zoomed view. It is also apparent in Figure 4 that RSA is the algorithm that has the poorest convergence behaviour because the value of J does not change much, an indication of local optima stagnation or premature convergence. Consequently, from the results in Table 4 and Figure 4, we conclude that SFS is better than the other competing algorithms. Therefore, we will use the controller parameters offered by SFS in the remaining experiments to preserve integrity.

Finally, the averaged execution time of the compared algorithms is measured in seconds per iteration and the results are visualized in Figure 5. At first glance, time consumption may seem excessive for every algorithm, yet most of the time is spent by the cost function evaluation, whereby the system is consecutively simulated 84 times, subjected to variable speed references and different load conditions. It is seen from Figure 5 that both SFS and SOS require almost the same amount of time, which is greater than that consumed by the remainder of the algorithms. This is to be expected because SFS needs to rank the entire population two times in one iteration prior to the updating processes, which is quite a time-consuming task. On the other hand, organisms in SOS frequently exchange information among themselves and after any modification to the solution, the algorithm immediately evaluates its fitness whether to keep or delete it. This is the reason for the high time consumption of the SOS technique.

5 | Experimental Verification

The EXP-PI controller performance is verified experimentally on a PMDC motor drive system. Figure 6 illustrates the system block diagram. The adopted PMDC is mechanically coupled to a PMDG for the purpose of loading. The PMDC is driven through a H-bridge IGBT converter, which can effectively satisfy four-quadrant operation. The DSP of TMS320F28335 is used as the control unit, which reads the position feedback signal, measures the motor armature current and accordingly generates the required PWM signals to keep the operation at the desired point. The EXP-PI controller outputs the current reference i_{a_ref} by acting on the speed error and the rate of change of the error. Then, the hysteresis current controller readily generates switch signals by comparing i_{a_ref} with the measured current i_a.

The hardware platform is shown in Figure 7. Sanyo Denki R730T-042EL7 is the PMDC motor model to be controlled and the generator model is Sanyo Denki T730B-569EL8N. The H-bridge IGBT converter is supplied from the Chroma 62050P-100-100 DC power supply and IGBTs of IXGH48N60C3D1 are used in the converter. There is a current sensor, LA 55-P, located between the converter and the motor to sample the armature current. The adopted motor is equipped with a built-in tachogenerator with a voltage constant 7.4 V/krpm, through which the speed response is acquired. The motor armature current representing the output torque is also measured by a Fluke 80i-110s current probe. The waveforms pertaining to the motor speed and current are displayed and saved by the Tektronix DPO 2024B oscilloscope.

The PMDC motor nominal parameters are reported in Table 5.

Two cases of experiments, namely speed control and torque disturbance rejection are used to verify the proposed approach.

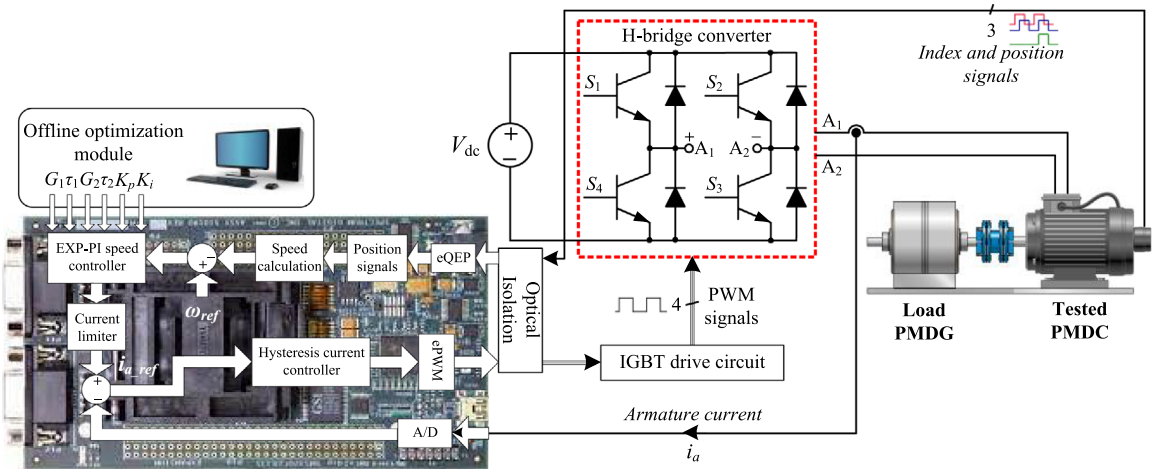


FIGURE 6 | PMDC motor drive system.

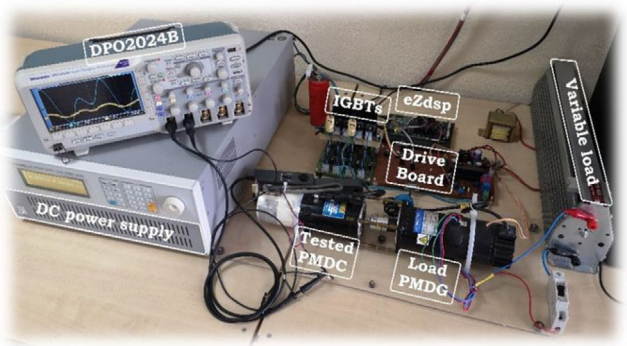


FIGURE 7 | The hardware platform.

TABLE 5 | Sanyo Denki R730T-042EL7 motor nominal parameters.

Variable	Value
Rated power	300 W
Rated armature current	4.8 A
Rated voltage	80 V
Rated speed	2500 rpm
Rated torque	1.15 Nm
Armature resistance (R_a)	1.75 Ω
Electrical time constant (τ_e)	1.5 ms
Voltage constant (k_e)	28.6 V/kmin^{-1}
Nominal inertia (J)	$5.8{\cdot}10^{-4}$ kg.m^2
Viscous-friction coefficient (B)	0.00055 Nm
Encoder resolution	4000 ppr

In the former, variable step speed references and a reference trajectory are used to test the motor in tracking them, while in the latter torque disturbances are applied at steady state to see whether the operating speed is preserved. To show the true contribution of our proposal (Method #1), it is compared with a classical PI (Method #2) and a nonlinear PI (Method #3) control schemes recently published in [7] and [22], respectively. The controller parameters and the associated average cost function value J are shown in Table 6, where N/R denotes 'not reported' in the respective work. According to the outcomes in Table 6, one can interpret that Method #1 outperforms Method #3 because it has a yielded less J value (= 48.19), which is 48.898 in the case of Method #3. Detailed quantitative comparisons will be carried out in the next subsections. Herein, it is to be noted that in all the reported experiments, those controller parameters are used and kept the same as in Table 6.

5.1 | Speed Control

Speed control is realized when the speed reference is initially a step function set to $n^* = 1500$ rpm at $t = 0$ s. In this test, we keep the PMDG terminals open so that it does not generate any opposing torque. Figure 8 shows the speed and current responses to this step reference, where the measured speed, n, is indicated by the turquoise curve with a speed scaling factor of 285.7 rpm/div. The blue curve indicates the measured i_a current with a scaling factor of 5 A/div. It is evident from Figures 8(a), 8(b) and 8(c) that the motor speed increases up to 1500 rpm and later it settles to its reference in all the methods. Although their steady state performances seem identical, Method #1 outpaces its rivals by achieving a more favourable transient response with less overshoot and smaller settling time. While approaching the speed reference, the current in the case of Method #1 drops from its maximum allowable value, i.e. 5 A to its new value much quicker than that of other methods.

Next, the controllers are tested in regulating the motor speed with variable step references. The speed reference signal is configured such that it undergoes step changes of 0, 1000, 2000, 0 and 500 rpm, respectively. The resulting responses are shown in Figure 9. Note that in this test, we turn the PMDG terminals on and thus the PMDC motor load changes as the speed changes. This can be easily viewed by observing the motor current at steady state. Comparing Figure 9(a) with Figures 9(b) and 9(c) reveals that Method #1 exhibits a transient performance superior to the other methods in terms of the speed overshoot and settling time.

TABLE 6 | Parameters of different techniques.

—	τ_1	G_1	τ_2	G_2	K_p	K_i	J (Equation 22)
Method #2	—	—	—	—	0.265	20.00	N/R
Method #3	0.01	0.425	—	—	0.192	19.970	48.898
Method #1	0.1569	19.9952	0.1076	0.1000	0.1922	19.9759	48.19

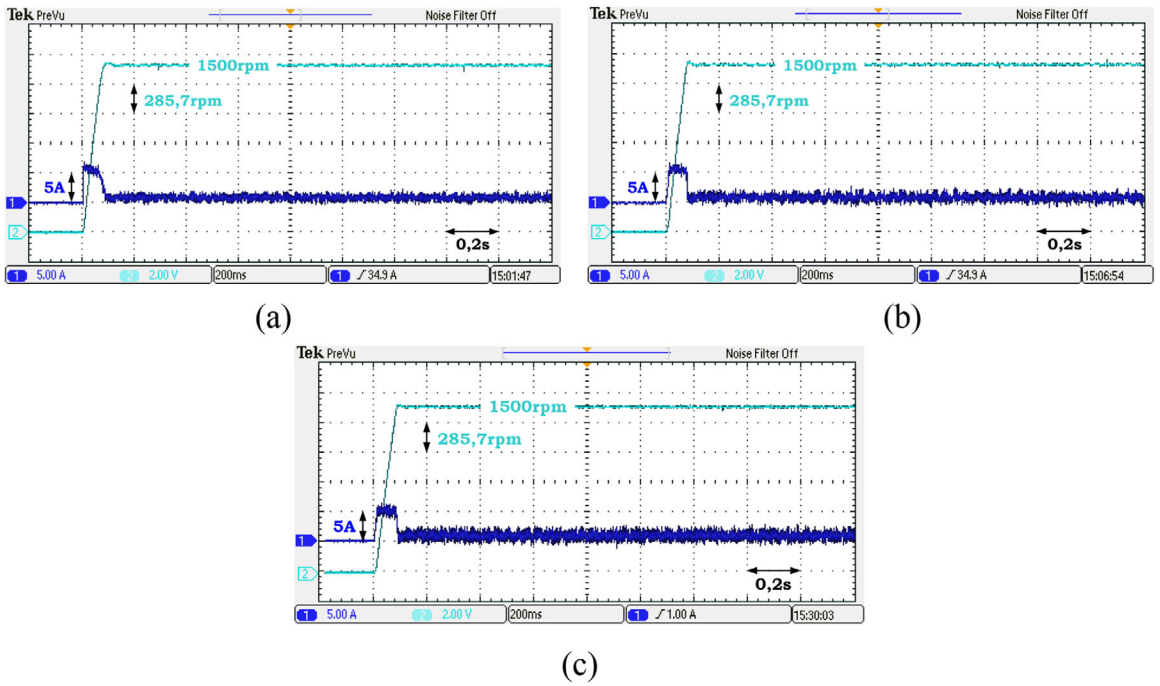


FIGURE 8 | Comparison of the speed and current responses to the step reference (a) Method #2; (b) Method #3; and (c) Method #1.

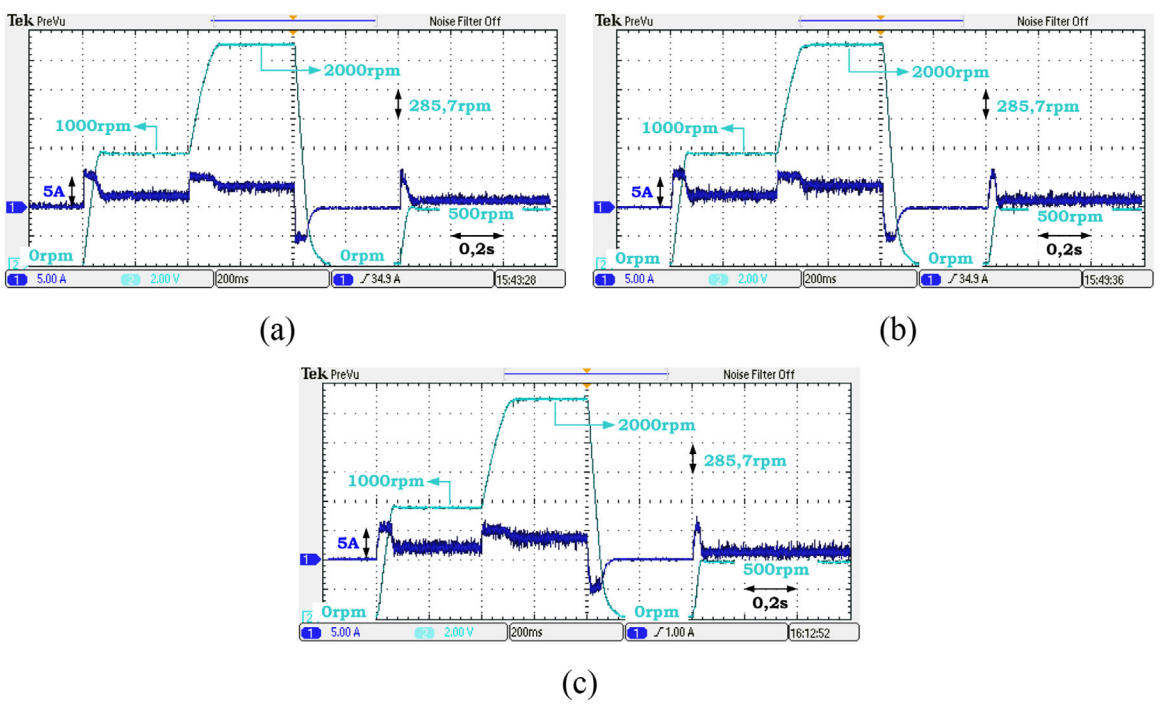


FIGURE 9 | Comparison of the speed and current responses to variable step references (a) Method #2; (b) Method #3; and (c) Method #1.

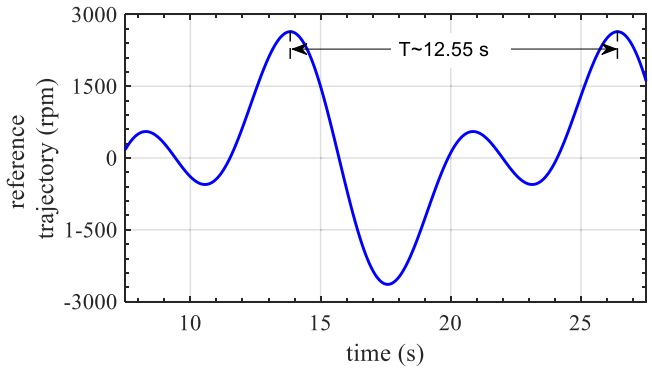


FIGURE 10 | Reference trajectory.

To evaluate and compare the performance against a continuously-changing speed reference, all the controllers are subjected to follow a trajectory. The reference trajectory chosen in this test is $n^* = 1500 \times (\sin(t) + \cos(0.5\,t))$, which is displayed in Figure 10. This trajectory has a period of $T \cong 12.55$ s and it ranges between −2640 and 2640 rpm.

The actual trajectories and the measured motor currents are shown in Figure 11. The PMDG terminals are kept closed in this test, thus, the load applied to the motor shaft countiniously changes with the speed. During the experimentation, Method #2 cannot accomplish the desired response from time to time, as seen in Figure 11(a). The motor current is found to change with high frequency between its allowable edges, i.e., ±5 A, so the speed regulation is lost temporarily. On the other hand, the response of Method #1 is shown in Figure 11(c), where it obviously divulges the capacity of the system with the proposed controller to track the reference trajectory. Tracking of the trajectory may be considered tight by observing the period of the actual trajectory that is equal to that of the reference trajectory in Figure 10. Similar results can be observed when Method #3 replaces Method #1, as seen in Figure 11(b).

5.2 | Robustness to Load Torque Disturbances

The objective of this test is to explore the robustness of our proposal to load torque disturbances. Disturbance rejection is one of the control objectives to be satisfied. Given this, at steady state, the PMDC motor is loaded by its rated torque and then the load is removed instantly after nearly 1 s, while the operational speed is 2000 rpm. As we can see in Figures 12(a), 12(b) and 12(c), all the three methods respond satisfactorily to accommodate the new condition, where the speed deviates from the reference one first and then recovers following either the application or removal of the torque load. However, in the case of Method #1, the disturbance rejection performance is far better than that of the other methods because torque disturbances almost never affect the desired speed response. Notice that the motor current (or the output torque) is at its rated value after the load is applied.

The quantitative comparison between our proposal and the existing strategies presented in Table 7 is obtained from the above responses. The FOD value that is defined in Equation (13) and considers a combination of ISE and IAE is experimentally measured. According to the results in Table 7, Method #1 is the best-performing method in offering the minimum FOD value in every condition tested. Method #3 seems to be the second-best method, except for the variable step reference condition. Furthermore, compared with Method #2, the FOD value of Method #1 is maximally reduced by up to 70.1%, from 0.4384–0.1311 for the load disturbance rejection. The FOD value of Method #1 is maximally reduced by up to 11.79%, from 29.5155–26.0775 under variable step reference in comparison with Method #3.

6 | Stability analysis

Checking the stability of any theoretical control scheme prior to manufacturing and commissioning is an essential task. A stability analysis formally reveals whether or not the system under control remains bounded for any limited input. To address the stability analysis, at first step, we obtain the dynamic model of the entire system using the parameter values in Tables 2. Shown in Figure 13, the input and output of this model are the speed reference $r(s)$ and the actual speed $\omega(s)$, respectively. It is a negative feedback control system with an EXP block, PI controller and PMDC motor, which are connected in cascade. Note that the EXP terms given by Equation (7) are represented by the first order transfer functions within the EXP block.

Using the block diagram in Figure 13, the closed-loop transfer function of the system is obtained as follows:

$$\frac{\omega(s)}{r(s)} = \frac{0.0016345\,s^3 + 0.29235\,s^2 + 13.77124\,s + 108.2434}{1.6154 \times 10^{-8} s^5 + 1.7403 \times 10^{-5}\,s^4 + 0.0031595\,s^3 + 0.31304\,s^2 + 13.84564\,s + 108.2434} \tag{23}$$

It is clear from Equation (23) that the controlled system has an equivalent transfer function of fifth-order. The characteristic equation of the system can be identified from the denominator of Equation (23) as:

$$1.6154 \times 10^{-8}\,s^5 + 1.7403 \times 10^{-5}\,s^4 + 0.0031595\,s^3 + 0.31304\,s^2 + 13.84564\,s + 108.2434 = 0 \tag{24}$$

Calculating the roots of this polynomial yields the following poles:

$$S_1 = -878.55$$

$$S_{2,3} = -57.7806 \pm 85.3492\,i$$

$$S_4 = -73.4320$$

$$S_5 = -9.7773$$

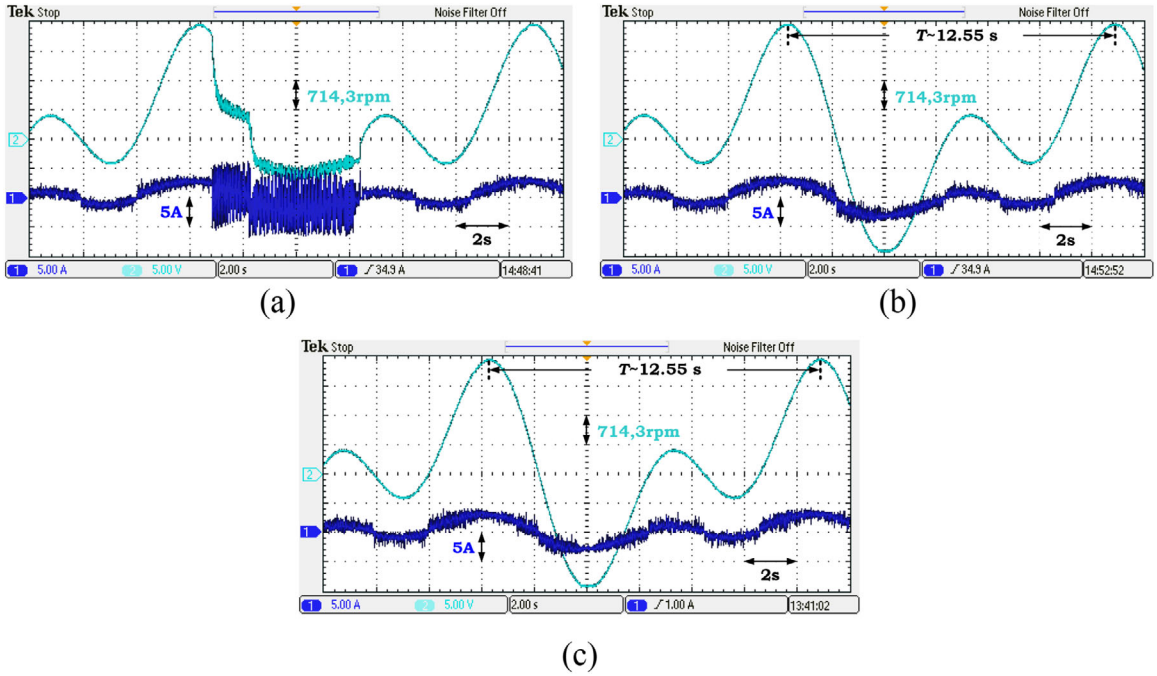


FIGURE 11 | Comparison of the speed and current responses to the reference trajectory (a) Method #2; (b) Method #3; and (c) Method #1.

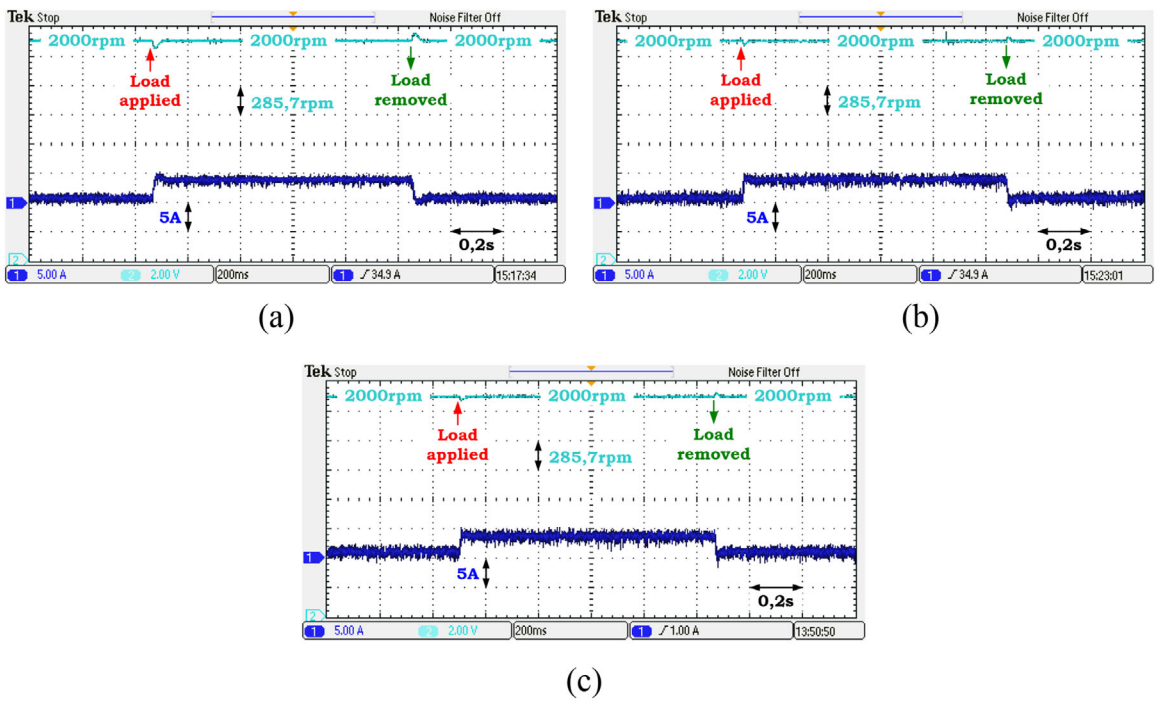


FIGURE 12 | Comparison of the speed and current responses under load disturbance (a) Method #2; (b) Method #3; and (c) Method #1.

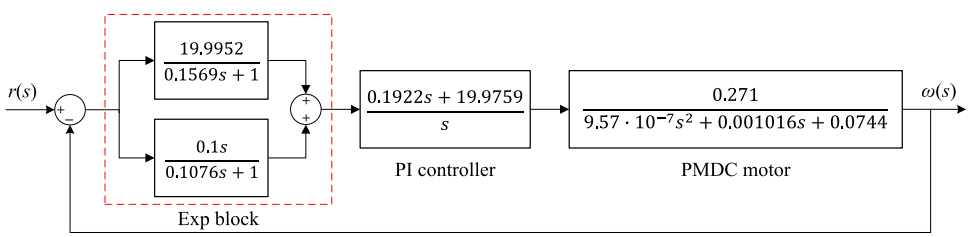


FIGURE 13 | Block diagram of the EXP-PI controller-based PMDC motor drive.

TABLE 7 | Quantitative comparison between our proposal and existing strategies under different conditions.

Condition	Strategy	FOD (Equation 13)
Step reference	Method #2	12.8474
	Method #3	12.8249
	Method #1	12.7959
Variable step reference	Method #2	28.1865
	Method #3	29.5155
	Method #1	26.0775
Reference trajectory	Method #2	5.2301
	Method #3	2.8140
	Method #1	2.7117
Load disturbance	Method #2	0.4384
	Method #3	0.1333
	Method #1	0.1311

Note: *Better results are bolded.

which discloses that three of the five poles are real and negative, while the others are complex conjugates with negative real parts. In other words, all of them lie in the left half of the complex plane, a necessary and sufficient condition to prove that the system of Equation (23) is stable.

We have also conducted a comparison study with Method #2 and Method #3 regarding frequency response analysis. The closed-loop transfer function of the system with Method #2 is derived as:

$$\frac{\omega(s)}{r(s)} = \frac{0.071815\, s + 5.42}{9.57 \times 10^{-7} s^3 + 0.001016\, s^2 + 0.146215\, s + 5.42} \tag{25}$$

Equation (25) reveals that the system based on Method #2 is of order 3 and it also performs stably because of the three poles that are all real and negative as:

$$S_1 = -898.6478$$
$$S_2 = -99.9469$$
$$S_3 = -63.0563$$

Eventually, the closed-loop transfer function of the system with Method #3 is given in Equation (26) [7].

$$\frac{\omega(s)}{r(s)} = \frac{2.2114s + 230.35}{9.57 \cdot 10^{-7} s^4 + 1.1117 \cdot 10^{-3} s^3 + 0.176 s^2 + 9.6514 s + 230.35} \tag{26}$$

According to Equation (26), the order of the system based on Method #3 is 4 and the system is stable with all the poles located in the left half of the complex plane, which are computed as:

$$S_1 = -985.5$$
$$S_2 = -102.12$$
$$S_{3,\ 4} = -37.18 \pm 31.76\, i$$

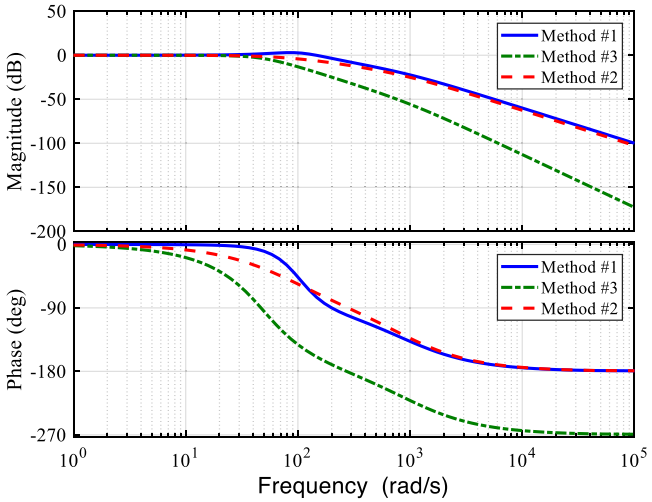


FIGURE 14 | Comparison of the frequency responses with different controllers.

Figure 14 shows the Bode plots of the systems given by Equations (23), (25) and (26). From Figure 14, it is easy to notice that the responses with Method #1 and Method #2 have nearly the same characteristics, where Method #1 has a mild amplification around 100 rad/s, a sign of faster responsiveness. On the other hand, they deliver almost the same bandwidth of 191.8 rad/s, whereas that of Method #3 has decreased to 43.2 rad/s. The gain and phase margins are measured as Inf dB and 105.6° for Method #1, 32.8 dB and 180° for Method #3 and Inf dB and 180° for Method #2. Here, it is to be stressed that a phase margin of 105.6° is more acceptable considering the trade-off between speed and stability because a high value of phase margin, i.e., 180° makes the response sluggish.

7 | Performance Evaluation Under Dead Time

In this section, we have tested the performance of EXP-PI controller under dead time. Within this context, the following second-order plant with delay, which was also used in [52] has been chosen as the case study.

$$P(s) = \frac{2(0.5\, s + 1)\, e^{-Ds}}{(s+1)(4\, s+1)},\ D = 0.5 \tag{27}$$

The plant has a dead time of 0.5 s and it is controlled by the EXP-PI controller. For the purpose of comparison, the results are also obtained when a conventional PID (C-PID) controller replaces the EXP-PI controller. SFS-based tuned gains for the EXP-PI controller and the reported gains for the C-PID controller are as follows:

- C-PID controller [52]: $K_p = 5.73$, $K_i = 5.00$, $K_d = 1.43$.
- EXP-PI controller: $\tau_1 = 0.99$, $G_1 = 9.98$, $\tau_2 = 0.0009$, $G_2 = 1.13$, $K_p = 9.98$, $K_i = 9.9020$.

The responses owing to the changes in reference and disturbance signals are displayed in Figure 15(a) with zoomed views in Figures 15(b) and 15(c). The performance metric, i.e., FOD is computed as 1.124 for the C-PID controller and 0.1395 for the EXP-PI controller. It can be seen from Figure 15 that the

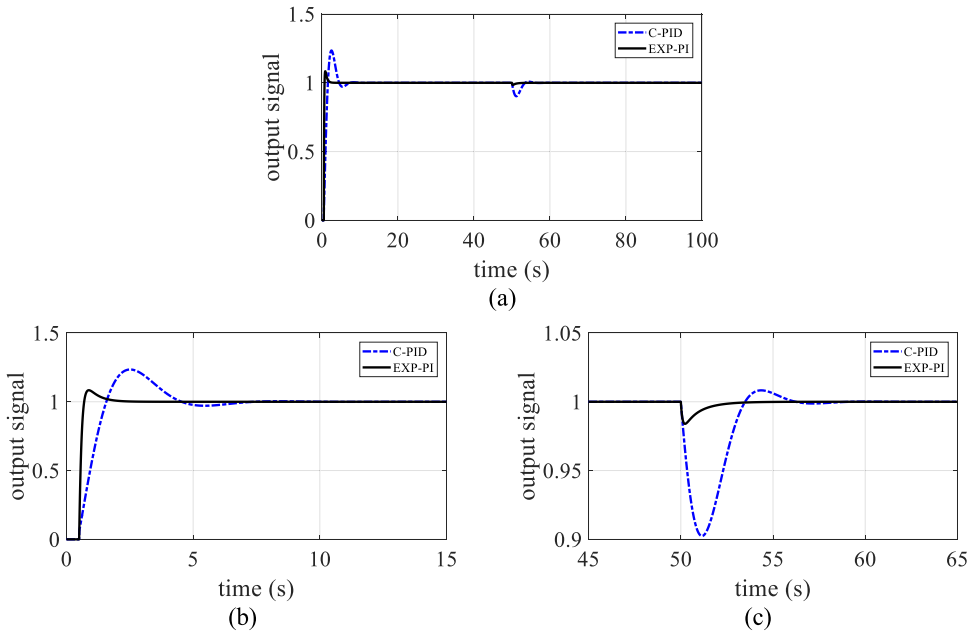


FIGURE 15 | Comparison of the responses for the system $P(s)$ (a) changes under reference and disturbance signals; (b) zoomed view from 0 to 15 s; and (c) zoomed view from 45–65 s.

EXP-PI controller enhances the performance of the C-PID controller significantly and provides not only a faster response but also excellent rejection of the disturbance applied.

Additionally, the control performance evaluation is made on the following test system $Q(s)$.

$$Q\ (s) = \frac{25}{s^2\ + 20\ s + 25} \tag{28}$$

To challenge the proposed controller, a comparison with a fixed-gain PI controller and a nonlinear gain PI controller published in [53] is also presented. In the nonlinear gain PI control scheme, a nonlinear gain k was incorporated in cascade with the classical PI controller, where the gain was declared as a function of the error e. Given this, for the fixed-gain PI controller, k can be considered constant and equal to 1. The gains of fixed-gain and nonlinear gain PI controllers along with those of the EXP-PI controller, procured through the employment of SFS, are presented as step-down:

- Fixed-gain PI controller [53]: $K_\mathrm{p} = 5.00$, $K_\mathrm{i} = 5.00$.
- Nonlinear gain PI controller [53]: $K_\mathrm{p} = 5.00$, $K_\mathrm{i} = 5.00$, nonlinear gain $k = 1 + 4[1 - \mathrm{sech}(e)]$.
- EXP-PI controller: $\tau_1 = 8.57$, $G_1 = 9.85$, $\tau_2 = 0.074$, $G_2 = 8.67$, $K_\mathrm{p} = 9.99$, $K_\mathrm{i} = 9.54$.

The unit step reference is delayed and applied at $t = 0.1$ s and the resulting system responses are comparatively given in Figure 16. The value of FOD is 0.1738 for the fixed-gain PI controller, 0.1047 for the nonlinear gain PI controller and 0.030 for the EXP-PI controller, respectively. Note that the smaller FOD value corresponds to the enhancement in the step response. Figure 16 indicates that the unit step response with the fixed-gain PI controller has a sluggish transient response and there is a small error persisting at steady state. Thanks to the inclusion of the nonlinear gain, the transient response of the closed-loop unit step response becomes faster with a very mild overshoot, but the steady state error still persists. Comparing the black solid trace with the other traces reveals that the EXP-PI controller has enhanced the system performance by significantly accelerating the transient response without causing sustained oscillations. While the response settles to the input reference hastily, the error at steady state is also eliminated.

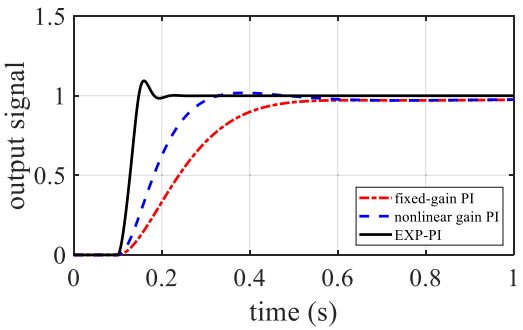


FIGURE 16 | Comparison of the responses for the system $Q(s)$.

As a consequence and an observation of the responses and numerical results in this section, it is affirmed that our proposal preserves its superior performance over the published approaches, even under different control systems.

8 | Conclusion

In this article, an enhanced version of a PI controller, namely the EXP-PI controller, was introduced for precise speed regulation in a PMDC motor. The control scheme that we present contains two EXP functions acting nonlinearly on the error and the rate of change of the error, respectively. Placed at the PI controller

front end to work with it in cascade, they are scalable and their rate of variation is changeable to have more degrees of freedom. Five recent algorithms including SFS, SO, DSA, SOS and RSA were applied to identify the best algorithm for the calibration of controller gains. The credibility and feasibility of the proposed method were evaluated on a PMDC motor drive system by conducting a series of practical tests. Besides, we also considered different plants to further challenge the proposed method. The outcomes of these tests were compared with the existing control schemes in the literature. Regarding the algorithmic comparisons, SFS was found to be the best performer, which could provide the minimum value of the designed cost function as 48.19. On the other hand, better results in terms of various performance metrics including integral-based error criteria, stability margin, overshoot and settling time were achieved using our proposal than using the available control techniques.

Ongoing research continues to explore how the EXP-PI controller reacts in the presence of permanent magnet synchronous motor and brushless DC motor drives. In lieu of the nonlinear functions used in the EXP block, other known or user-defined functions may be tried out for possible improved performance. Applying other optimization techniques (preferably more powerful than the SFS) to calibrate the controller parameters is also worth investigating to search for more accurate controller parameters.

Author Contributions

Emre Çelik: supervision, conceptualization, methodology, software, formal analysis, investigation, writing – original draft. **Davut İzci:** data curation, validation, visualization, writing – review and editing. **Serdar Ekinci:** methodology, software, formal analysis, writing – review and editing. **Erdal Bekiroğlu:** resources, data curation, validation. **Diego Oliva:** supervision, methodology, writing – review and editing. **Mahmoud Abdel-Salam:** formal analysis, visualization, writing – review and editing. **Ghanshyam G. Tejani:** investigation, validation, resources, writing – review and editing. **Seyed Jalaleddin Mousavirad:** writing – review and editing.

Funding

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the first author upon reasonable request.

References

1. N. L. Manuel, N. İnanç, and M. Lüy, "Control and Performance Analyses of a DC Motor Using Optimized PIDs and Fuzzy Logic Controller," *Results in Control and Optimization* 13 (2023): 100306, https://doi.org/10.1016/j.rico.2023.100306.

2. S. Ekinci, D. Izci, M. H. Almomani, et al., "Advanced Control Parameter Optimization in DC Motors and Liquid Level Systems," *Scientific Reports* 15 (2025): 1394, https://doi.org/10.1038/s41598-025-85273-y.

3. S. Ekinci, D. Izci, and M. Yilmaz, "Efficient Speed Control for DC Motors Using Novel Gazelle Simplex Optimizer," *IEEE Access* 11 (2023): 105830–105842, https://doi.org/10.1109/ACCESS.2023.3319596.

4. A. Sen, B. Singh, K. Mahtani, A. Moradzadeh, and S. M. Muyeen, "Optimized Design of a Permanent Magnet Brushless DC Motor for Solar Water-Pumping Applications," *Results in Engineering* 26 (2025): 104633, https://doi.org/10.1016/j.rineng.2025.104633.

5. N. El Bazi, N. Guennouni, M. Mekhfioui, A. Goudzi, A. Chebak, and M. Mabrouki, "Predicting the Temperature of a Permanent Magnet Synchronous Motor: A Comparative Study of Artificial Neural Network Algorithms," *Technologies* 13, no. 3 (2025): 120, https://doi.org/10.3390/technologies13030120.

6. G. F. Lukman and C. Lee, "Towards Digital Twin Modeling and Applications for Permanent Magnet Synchronous Motors," *Energies* 18, no. 4 (2025): 956, https://doi.org/10.3390/en18040956.

7. E. Çelik, G. Bal, N. Öztürk, et al., "Improving Speed Control Characteristics of PMDC Motor Drives Using Nonlinear PI Control," *Neural Computing and Applications* 36 (2024): 9113–9124, https://doi.org/10.1007/s00521-024-09568-3.

8. T. Chalardsakul, C. Piliyasilpa, and V. Sukontanakarn, "TMS320F28379D Microcontroller for Speed Control of Permanent Magnet Direct Current Motor," *IAES International Journal of Artificial Intelligence (IJ-AI)* 13, no. 3 (2024): 2816–2828, https://doi.org/10.11591/ijai.v13.i3.pp2816-2828.

9. M. R. Çorapsız, "Comparison of Parameter Estimation Techniques for Control Parameter Identification of Permanent Magnet Direct Current Motor Speed Control," *Advanced Control for Applications* 7, no. 1 (2025): 1–18, https://doi.org/10.1002/adc2.70005.

10. F. U. R. Siddiqi, S. Ahmad, T. Akram, M. U. Ali, A. Zafar, and S. W. Lee, "Artificial Neural Network-Based Data-Driven Parameter Estimation Approach: Applications in PMDC Motors," *Mathematics* 12, no. 21 (2024): 3407, https://doi.org/10.3390/math12213407.

11. D. K. Shary and H. J. Nekad, "Position and Speed Control for Permanent Magnet DC Motor Based on Different Optimization Algorithms," *Journal Européen des Systèmes Automatisés* 57 (2024): 1705–1711, https://doi.org/10.18280/jesa.570618.

12. T. L. Kottas, A. D. Karlis, and Y. S. Boutalis, "A Novel Control Algorithm for DC Motors Supplied by PVS Using Fuzzy Cognitive Networks," *IEEE Access* 6 (2018): 24866–24876, https://doi.or/10.1109/ACCESS.2018.2822051.

13. P. Dhinakaran and D. Manamalli, "Novel Strategies in the Model-Based Optimization and Control of Permanent Magnet DC Motors," *Computers and Electrical Engineering* 44 (2015): 34–41, https://doi.org/10.1016/j.compeleceng.2015.04.002.

14. P. Mishra, A. Banerjee, M. Ghosh, N. K. Vemula, P. K. Meher, and B. Chitti Babu, "A Cost-Effective Hardware Accelerator for PMDC Motor-Based Auxiliary Component Automation of Electric Three-Wheelers," *AEU-International Journal of Electronics and Communications* 195 (2025): 155766, https://doi.org/10.1016/j.aeue.2025.155766.

15. N. Khanduja, "Speed Control of a DC Motor Using Hybrid Metaheuristic-Tuned 2-DOF PID Controller," *SN Computer Science* 6 (2025): 981.

16. N. Öztürk and E. Çelik, "Speed Control of Permanent Magnet Synchronous Motors Using Fuzzy Controller Based on Genetic Algorithms," *International Journal of Electrical Power and Energy Systems* 43, no. 1 (2012): 889–898, https://doi.org/10.1016/j.ijepes.2012.06.013.

17. E. Çelik and M. Karayel, "Effective Speed Control of Brushless DC Motor Using Cascade 1PDf-PI Controller Tuned by Snake Optimizer," *Neural Computing and Applications* 36 (2024): 7439–7454, https://doi.org/10.1007/s00521-024-09470-y.

18. F. Mendoza-Mondragón, V. M. Hernández-Guzmán, and J. Rodríguez-Reséndiz, "Robust Speed Control of Permanent Magnet Synchronous Motors Using Two-Degrees-of-Freedom Control," *IEEE Transactions on*

Industrial Electronics 65, no. 8 (2018): 6099–6108, https://doi.org/10.1109/TIE.2017.2786203.

19. Z. Kang, X. Lin, X. Shen, Z. Liu, Y. Gao, and J. Liu, "Event-Triggered Generalized Super-Twisting Sliding Mode Control for Position Tracking of PMSMs," *IEEE Transactions on Industrial Informatics* 21, no. 7 (2025): 5701–5711, https://doi.org/10.1109/TII.2025.3556086.

20. X. Lin, R. Xu, W. Yao, et al., "Observer-Based Prescribed Performance Speed Control for PMSMs: A Data-Driven RBF Neural Network Approach," *IEEE Transactions on Industrial Informatics* 20, no. 5 (2024): 7502–7512, https://doi.org/10.1109/TII.2024.3357194.

21. R. Xu, X. Shen, X. Lin, Z. Liu, D. Xu, and J. Liu, "Robust Model Predictive Control of Position Sensorless-Driven IPMSM Based on Cascaded EKF-LESO," *IEEE Transactions on Transportation Electrification* 11, no. 4 (2025): 8824–8832, https://doi.org/10.1109/TTE.2025.3547259.

22. E. Çelik and N. Öztürk, "First Application of Symbiotic Organisms Search Algorithm to Off-Line Optimization of PI Parameters for DSP-Based DC Motor Drives," *Neural Computing and Applications* 30 (2018): 1689–1699, https://doi.org/10.1007/s00521-017-3256-5.

23. E. Çelik, "Exponential PID Controller for Effective Load Frequency Regulation of Electric Power Systems," *ISA Transactions* 153 (2024): 364–383, https://doi.org/10.1016/j.isatra.2024.07.038.

24. H. Salimi, "Stochastic Fractal Search: A Powerful Metaheuristic Algorithm," *Knowledge-Based Systems* 75 (2015): 1–18, https://doi.org/10.1016/j.knosys.2014.07.025.

25. F. A. Hashim and A. G. Hussien, "Snake Optimizer: A Novel Meta-Heuristic Optimization Algorithm," *Knowledge-Based Systems* 242 (2022): 108320, https://doi.org/10.1016/j.knosys.2022.108320.

26. S. Mirjalili, "Dragonfly Algorithm: A New Meta-Heuristic Optimization Technique for Solving Single-Objective, Discrete, and Multi-Objective Problems," *Neural Computing and Applications* 27 (2016): 1053–1073, https://doi.org/10.1007/s00521-015-1920-1.

27. M. Y. Cheng and D. Prayogo, "Symbiotic Organisms Search: A New Metaheuristic Optimization Algorithm," *Computers and Structures* 139 (2014): 98–112, https://doi.org/10.1016/j.compstruc.2014.03.007.

28. L. Abualigah, M. A. Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile Search Algorithm (RSA): A Nature-Inspired Meta-Heuristic Optimizer," *Expert Systems with Applications* 191 (2021): 116158, https://doi.org/10.1016/j.eswa.2021.116158.

29. R. Salgotra and U. Singh, "Application of Mutation Operators to Flower Pollination Algorithm," *Expert Systems With Applications* 79 (2017): 112–129, https://doi.org/10.1016/j.eswa.2017.02.035.

30. X. Chen, H. Yue, and K. Yu, "Perturbed Stochastic Fractal Search for Solar PV Parameter Estimation," *Energy* 189 (2019): 116247, https://doi.org/10.1016/j.energy.2019.116247.

31. R. Bhatt, G. Parmar, R. Gupta, and A. Sikander, "Application of Stochastic Fractal Search in Approximation and Control of LTI Systems," *Microsystem Technologies* 25 (2019): 105–114, https://doi.org/10.1007/s00542-018-3939-6.

32. R. Saini, G. Parmar, and R. Gupta, "An Enhanced Hybrid Stochastic Fractal Search FOPID for Speed Control of DC motor," in *Fractional Order Systems and Applications in Engineering* (Elsevier, 2023): 51–67.

33. S. Xu and H. Qiu, "A Modified Stochastic Fractal Search Algorithm for Parameter Estimation of Solar Cells and PV Modules," *Energy Reports* 8 (2022): 1853–1866, https://doi.org/10.1016/j.egyr.2022.01.008.

34. H. Rezk, T. S. Babu, M. Al-Dhaifallah, and H. A. Ziedan, "A Robust Parameter Estimation Approach Based on Stochastic Fractal Search Optimization Algorithm Applied to Solar PV Parameters," *Energy Reports* 7 (2021): 620–640, https://doi.org/10.1016/j.egyr.2021.01.024.

35. C. Mai, L. Zhang, and X. Hu, "An Adaptive Snake Optimization Algorithm Incorporating Subtraction-Average-Based Optimizer for Photovoltaic Cell Parameter Identification," *Heliyon* 10, no. 15 (2024): e35382, https://doi.org/10.1016/j.heliyon.2024.e35382.

36. Y. Wang, B. Xin, Z. Wang, and J. Sun, "Improved Snake Optimizer Based on Forced Switching Mechanism and Variable Spiral Search for Practical Applications Problems," *Soft Computing* 29 (2025): 803–838, https://doi.org/10.1007/s00500-025-10404-6.

37. T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access* 10 (2022): 10031–10061, https://doi.org/10.1109/ACCESS.2022.3142859.

38. S. Singh, A. Ashok, M. Kumar, Garima, and T. K. Rawat, "Optimal Design of IIR Filter Using Dragonfly Algorithm," In: H. Malik, S. Srivastava, Y. Sood, and A. Ahmad (eds) *Applications of Artificial Intelligence Techniques in Engineering. Advances in Intelligent Systems and Computing*, vol 698 (Springer, Singapore, 2019), https://doi.org/10.1007/978-981-13-1819-1_21.

39. C. M. Rahman and T. A. Rashid, "Dragonfly Algorithm and its Applications in Applied Science Survey," *Computational Intelligence and Neuroscience* 2019 (2019): 1–21, https://doi.org/10.1155/2019/9293617.

40. K. S. Sree Ranjini and S. Murugan, "Memory Based Hybrid Dragonfly Algorithm for Numerical Optimization Problems," *Expert Systems With Applications* 83 (2017): 63–78, https://doi.org/10.1016/j.eswa.2017.04.033.

41. Ş. Gülcü, "Training of the Feed Forward Artificial Neural Networks Using Dragonfly Algorithm," *Applied Soft Computing* 124 (2022): 109023, https://doi.org/10.1016/j.asoc.2022.109023.

42. X. Bao, H. Jia, and C. Lang, "Dragonfly Algorithm With Opposition-Based Learning for Multilevel Thresholding Color Image Segmentation," *Symmetry* 11, no. 5 (2019): 716, https://doi.org/10.3390/sym11050716.

43. J. Xu and F. Yan, "Hybrid Nelder–Mead Algorithm and Dragonfly Algorithm for Function Optimization and the Training of a Multilayer Perceptron," *Arabian Journal for Science and Engineering* 44 (2019): 3473–3487, https://doi.org/10.1007/s13369-018-3536-0.

44. C. Han and Y. Zhou, "Optimization IIR Filter Design Using Symbiotic Organism Search algorithm," in *Proceedings of the 2019 International Conference on Modeling, Simulation and Big Data Analysis (MSBDA 2019)* (Atlantis Press, 2019).

45. H. Mohammadzadeh and F. S. Gharehchopogh, "Feature Selection With Binary Symbiotic Organisms Search Algorithm for Email Spam Detection," *International Journal of Information Technology and Decision Making* 20, no. 01 (2021): 469–515, https://doi.org/10.1142/S0219622020500546.

46. A. E. S. Ezugwu, A. O. Adewumi, and M. E. Frîncu, "Simulated Annealing Based Symbiotic Organisms Search Optimization Algorithm for Traveling Salesman Problem," *Expert Systems With Applications* 77 (2017): 189–210, https://doi.org/10.1016/j.eswa.2017.01.053.

47. Y. Zhou, F. Miao, and Q. Luo, "Symbiotic Organisms Search Algorithm for Optimal Evolutionary Controller Tuning of Fractional Fuzzy Controllers," *Applied Soft Computing* 77 (2019): 497–508, https://doi.org/10.1016/j.asoc.2019.02.002.

48. D. Guha, P. K. Roy, and S. Banerjee, "Symbiotic Organism Search Algorithm Applied to Load Frequency Control of Multi-Area Power System," *Energy Systems* 9 (2018): 439–468, https://doi.org/10.1007/s12667-017-0232-1.

49. R. Saravanan, O. Sobhana, M. Lakshmanan, and P. Arulkumar, "Fuel Cell Electric Vehicles Equipped With Energy Storage System for Energy Management: A Hybrid JS-RSA Approach," *Journal of Energy Storage* 72 (2023): 108646, https://doi.org/10.1016/j.est.2023.108646.

50. S. Al-Dahidi, H. Alahmer, B. Rinchi, et al., "Multistep PV Power Forecasting Using Deep Learning Models and the Reptile Search Algorithm," *Results in Engineering* 27 (2025): 106265, https://doi.org/10.1016/j.rineng.2025.106265.

51. Q. Sun and J. Su, "Optimization of University Management Based on Reptile Search Algorithm Combined With Short-Duration Memory Neural Network," *Systems and Soft Computing* 6 (2024): 200101, https://doi.org/10.1016/j.sasc.2024.200101.

52. A. Fereidouni, M. A. S. Masoum, and M. Moghbel, "A New Adaptive Configuration of PID Type Fuzzy Logic Controller," *ISA Transactions* 56 (2015): 222–240, https://doi.org/10.1016/j.isatra.2014.11.010.

53. H. Seraji, "A New Class of Nonlinear PID Controllers With Robotic Applications," *Journal of Robotic Systems* 15, no. 3 (1998): 161–181, https://doi.org/10.1002/(SICI)1097-4563(199803)15:3⟨161::AID-ROB4⟩3.0.CO;2-O.