Journal of Intelligent & Robotic Systems (2025) 111:31
https://doi.org/10.1007/s10846-025-02228-1

REGULAR PAPER

Motor Speed Control of Four-wheel Differential Drive Robots Using a New Hybrid Moth-flame Particle Swarm Optimization (MFPSO) Algorithm

Mohamed Reda[1,2] · **Ahmed Onsy**[1] · **Amira Y. Haikal**[2] · **Ali Ghanbari**[1]

Received: 14 May 2024 / Accepted: 23 January 2025

Abstract
Speed control of DC motors is essential for automated vehicles and four-wheel differential drive (4WD) cars, which are distinct by their high level of maneuverability. The PID controller is one of the most popular techniques for controlling speed, but tuning its parameters is challenging. This paper presents a novel hybrid algorithm, the Moth-Flame Particle Swarm Optimization (MFPSO), which combines moth-flame optimization (MFO) and particle swarm optimization (PSO) to address the slow convergence of MFO and the premature convergence of PSO. The MFPSO is deployed for real-time interactive tuning of the PID controller to control the speed of DC motors in a 4WD car. Additionally, a novel practical procedure is proposed to build a robust four-wheel differential drive and maintain the synchronization of the four DC motors. Simulation results and statistical analysis demonstrate the superior performance of the MFPSO compared with the PSO, MFO, and other hybrid variants (HMFPSO and HyMFPSO), with MFPSO ranking first in the Friedman test on CEC2020/2021 and engineering optimization benchmark problems. Practical results and the transient response analysis of the speed control revealed that MFPSO significantly outperformed the traditional Ziegler-Nichols (ZN) method, MFO, PSO, HMFPSO, and HyMFPSO algorithms. Specifically, the MFPSO algorithm reduced settling time by 34.83%, 21.20%, 20.75%, 22.97%, and 31.59%, and overshoot by 86.11%, 64.99%, 71.02%, 74.37%, and 60.58% compared to the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms, respectively. The source code of the proposed algorithm is available at https://github.com/MohamedRedaMu/MFPSO-Algorithm.

Keywords Four-wheel differential drive (4WD) · DC motor Speed control · Moth-flame Optimization (MFO) · Particle Swarm Optimization (PSO) · CEC2020 benchmark; transient response

1 Introduction

1.1 Four-wheel Differential Drive and DC Motors

Unmanned vehicles require key control tasks for autonomous driving, including risk assessment [1], lane-keeping [2], steering control [3], trajectory control [4], and path planning [5]. Among these, motor speed control is critical, particularly in 4WD systems widely used in mobile robotics for maneuverability [6]. DC motors, central to 4WD systems, offer high torque at low speeds and precise control [7]. Each wheel's motor in a 4WD robot is independently controlled, enabling complex maneuvers [8]. Synchronizing these motors is vital, requiring consistent PWM frequencies via motor drivers and accurate alignment of encoders for speed measurement [9].

Accurate vehicle positioning and steering rely heavily on precise motor speed control, achieved by minimizing errors between desired and actual speeds using PID controllers [10]. PID controllers are widely applied in 95% of control systems due to their effectiveness, but they are challenging to tune [11]. The Ziegler-Nichols method is a traditional yet straightforward approach for PID tuning [12].

Meta-heuristic algorithms like Particle Swarm Optimization (PSO) and Moth-Flame Optimization (MFO) have shown better performance in tuning PID controllers for speed control [13]. However, PSO can suffer from limited diversity

✉ Mohamed Reda
mohamed.reda.mu@gmail.com; mramohamed@uclan.ac.uk; mohamed.reda@mans.edu.eg

[1] School of Engineering, University of Central Lancashire, Preston PR1 2HE, UK

[2] Computers and Control Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt

Published online: 19 February 2025

and premature convergence, while MFO excels in exploration but converges slower [14, 15]. A hybrid approach combining the strengths of PSO and MFO is proposed to enhance performance and overcome their limitations.

1.2 Contributions

The contributions of this research are as follows:

- Proposes a new hybrid algorithm, Moth-Flame Particle Swarm Optimization (MFPSO), addressing limitations of standard PSO and MFO algorithms.
- Introduces a practical procedure with four experiments to synchronize the motors of a four-wheel differential drive vehicle.
- Designs and implements a real-time PID-MFPSO controller for precise DC motor speed control, tuned by the MFPSO algorithm.
- Validates the MFPSO algorithm using CEC2020/2021 benchmarks and 13 engineering problems, outperforming state-of-the-art algorithms.
- Experimentally validates the PID-MFPSO controller's transient response on a DC motor, achieving significant improvements. Specifically, the MFPSO algorithm reduced overshoot by 86.11%, 64.99%, 71.02%, 74.37%, and 60.58%, and settling time by 34.83%, 21.20%, 20.75%, 22.97%, and 31.59% compared to the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms, respectively.

1.3 Paper Organization

The rest of the research is organized as follows: Section 2 reviews state-of-the-art methods and hardware, including the Elegoo smart robotic car, PID tuning methods, and the standard MFO and PSO algorithms. Section 3 details the proposed MFPSO algorithm. Section 4 analyzes the MFPSO parameters and provides recommendations. Section 5 discusses benchmark simulation results, while Section 6 validates the algorithm on engineering optimization problems. Section 7 describes the design and assembly of the 4WD prototype. Section 8 presents troubleshooting experiments for motor synchronization. Section 9 demonstrates MFPSO-based PID tuning for DC motor speed control. Section 10 compares the transient response of MFPSO with other algorithms. Finally, Section 11 summarizes the findings and outlines future work.

2 Review of Related Work and Algorithmic Foundations

The literature review is divided into two parts. The first explores related work, including PID tuning algorithms for motor speed control and the characteristics of the Elegoo V4 smart robotic car as a 4WD. The second discusses standard PSO and MFO algorithms, along with hybrid variants.

2.1 Literature Review

Various algorithms have been developed for PID tuning in motor speed control. Ziegler-Nichols (ZN) remains a widely used method due to its simplicity [12]. Particle swarm optimization (PSO) and moth-flame optimization (MFO) are frequently applied to improve performance. Qi et al. and Xie et al. used PSO to tune PID controllers for CAN-based and brushless DC motors, respectively [16, 17]. Garba et al. found PSO achieved a 5.81% better settling time than ABC in DC motor speed control [18]. Yazgan et al. showed PSO outperformed GA in motor speed control [19], while Ramya et al. demonstrated PSO's superiority over ZN and GA for brushless DC motor speed [20].

MFO-based PID controllers have also shown effectiveness. Acharyulu et al. applied MFO in an AGC system [21], and Bennaoui et al. reported MFO's superiority over PSO in DC-DC boost converter control [22]. Mustafa et al. and Sultan et al. demonstrated PSO's efficiency in improving settling and rise times for DC motors [23, 24]. Similarly, Valluru et al. showed that MFO outperformed GA and SA for steering autonomous underwater vehicles [25].

Recent work includes PSO-PID and MFO-PID applications in diverse systems. Safarzadeh et al. used PSO for reactor power regulation [26], while Vishnoi et al. and Yusubov et al. applied MFO-PID for temperature control and photovoltaic systems, respectively [27, 28]. Naik et al. used MFO-PID for electronic throttle control in hybrid electric vehicles, outperforming ZN [29]. Sharma et al. and Shary et al. applied PSO-PID to enhance DC motors' speed control [30, 31]. These studies highlight PSO and MFO's effectiveness in various tuning and control scenarios.

The Elegoo V4 car is an educational platform widely used in autonomous vehicle research. It features four geared DC motors with a TB6612 dual motor driver, but its design has limitations [32]. Febbo et al. validated self-driving systems using the Elegoo V2 car [33], while Latoui et al. utilized the V3 model for Q-learning in remote path planning [34]. Singh et al. developed a digital twin toolbox for the car's Li-ion batteries [35], and Farrugia used the V4 car for path planning and obstacle avoidance [36]. Other applications include active haptic guidance [37], object following [38], and SLAM algorithm validation [39].

Despite its popularity, the Elegoo V4 has significant drawbacks as a 4WD vehicle. It lacks motor encoders, hindering speed monitoring and localization. The car's four motors are controlled by two TB6612 drivers, reducing it to two-wheel drive and causing kinematic conflicts. Additionally, the Arduino UNO's limited IO pins restrict hardware

expansion. Therefore, multiple hardware modifications are required to convert the car into a standard four-wheel differential drive car.

2.2 Algorithmic Foundations of the Standard PSO and MFO Algorithms

PSO, introduced by Kennedy and Eberhart in 1995 [40], mimics the collective movement of birds searching for food. Candidate solutions are represented as particles, updating their velocities based on personal best $pbest$ and global best $gbest$ positions. The velocity v_i^t and position x_i^t of the i-th particle in the t-th iteration are updated by Eqs. 1 and 2, where w is inertia weight, (c_1, c_2) are acceleration coefficients, and $(rand_1, rand_2)$ are random numbers in [0,1].

$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot rand_1^t \cdot (pbest_i^t - x_i^t) + c_2 \cdot rand_2^t \cdot (gbest^t - x_i^t) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

MFO [41] is inspired by moths' navigation using moonlight, disrupted by artificial light sources. The population (M) consists of n moths (agents), while flames (F) are the sorted M, representing the best positions. The number of flames (F_{no}) decreases over iterations to balance exploration and exploitation using Eq. 3.

The position of the i-th moth in the t-th iteration M_i^t is updated using a logarithmic spiral equation Eq. 4, in which the i-th moth flies around the corresponding i-th flames F_i^t, where b is a constant for defining the shape of the logarithmic spiral and r is a random number in [-1,1]. This approach maintains diversity by ensuring flames are not fixed to specific moths across iterations.

$$F_{no} = round(n - t * \frac{n-1}{maxIter}) \quad (3)$$

$$M_i^{t+1} = \begin{cases} |F_i^t - M_i^t| * e^{b*r} * cos(2\pi r) + F_i^t & \text{if } t \le F_no \\ |F_i^t - M_i^t| * e^{b*r} * cos(2\pi r) + F_{F_{no}}^t & otherwise \end{cases} \quad (4)$$

Bingi et al. developed the hybrid HyMFPSO algorithm, combining PSO and MFO, which outperforms both on eight benchmark functions (2-10 dimensions) [42]. In HyMFPSO, PSO velocity (v_i^{t+1}) is added to the MFO position update, as in Eq. 5. Shaikh et al. proposed HMFPSO, an improved version of Hy-PSO-MFO by Yang et al. [43], validated on benchmark functions and power transmission applications [14]. HMFPSO introduces a PSO-based local attractor (Q_i^t) obtained using Eq. 6 for moth position updates. The $pBest_i^t$ and $gbest^t$ represent the local and global best flames, respectively, and ϕ is a random number in [0, 1] [43].

$$M_i^{t+1} = \begin{cases} |F_i^t - M_i^t| * e^{b*r} * cos(2\pi r) + F_i^t + v_i^{t+1} & \text{if } t \le F_{no} \\ |F_i^t - M_i^t| * e^{b*r} * cos(2\pi r) + F_{F_{no}}^t + v_i^{t+1} & otherwise \end{cases} \quad (5)$$

$$Q_i^t = \phi \times pBest_i^t + (1 - \phi) \times gbest^t \quad (6)$$

$$M_i^{t+1} = |F_i^t - M_i^t| * e^{b*t} * cos(2\pi t) + Q_i^t \quad (7)$$

3 The Proposed Moth Flame Particle Swarm Optimization (MFPSO) Algorithm

The standard PSO algorithm has a premature convergence problem, where they may get trapped in a local minimum [14]. The particle velocities are updated based on the local and global bests. The position of the particles is updated based only on the particle velocities, which increases the possibility of premature convergence.

3.1 The Concept of the Proposed MFPSO Algorithm and its Equations

The proposed Moth Flame Particle Swarm Optimization (MFPSO) algorithm aims to improve the performance of the original PSO by integrating elements from both the Particle Swarm Optimization (PSO) and Moth Flame Optimization (MFO) algorithms. The MFO algorithm, inspired by the natural navigation behavior of moths, excels in exploration due to its flame update mechanism [41]. By combining the strong exploration capabilities of MFO with the intense exploitation strengths of PSO, the MFPSO algorithm seeks to avoid premature convergence in the standard PSO and achieve a better balance between exploration and exploitation.

In the MFPSO algorithm, the concept of particle velocities from PSO is incorporated into the moths in the MFO algorithm. The population in MFPSO consists of a set of moths, each defined by a velocity $v_{i,d}^{(t)}$ and a position $x_{i,d}^{(t)}$, where i represents the i-th moth, d denotes the d-th dimension, and t is the iteration number. The position of each moth represents a candidate solution, corresponding to the PID parameters in the speed control application. The flames population F consists of the sorted moths based on their fitness values, with $f_{i,d}^{(t)}$ indicating the position of i-th flame in the d-th dimension at iteration t.

The moth's velocity $v_{i,d}^{(t)}$ is updated based on the differential step among the current moth's position $x_{i,d}^{(t)}$, the global best moth's position in the whole population $gbest_d$, and the personal best moth's position found by the i-th moth $pbest_{i,d}$. Equation 8 shows the moth's velocity update, where

w is the inertia weight, r_1 and r_2 are random numbers in the range [0,1], and c_1 and c_2 are the acceleration coefficients that influence the impact of personal and global best positions. This mechanism mirrors the velocity update in the PSO algorithm.

$$v_{i,d}^{(t+1)} = w \cdot v_{i,d}^{(t)} + c_1 \cdot r_1 \cdot (pbest_{i,d} - x_{i,d}^{(t)}) + c_2 \cdot r_2 \cdot (gbest_d - x_{i,d}^{(t)}) \quad (8)$$

The primary idea of the MFPSO algorithm is to combine the explorative strengths of MFO with the exploitative abilities of PSO during the moth's position update, thereby enhancing performance and preventing premature convergence. The moth population is divided into two groups. In the first group, the position $x_{i,d}^{(t+1)}$ is updated based on the newly calculated velocity $v_{i,d}^{(t+1)}$. In the second group, the position is updated based on a logarithmic spiral movement around the corresponding target flame position $P_{i,d}^{(t)}$. The hybrid position update equation is given by Eq. 9, where b is a constant defining the shape of the spiral (set to 1), and $l_{i,d}$ is a random number in the range [-1,1] for the d-th dimension of the i-th moth.

$$x_{i,d}^{(t+1)} = \begin{cases} x_{i,d}^{(t)} + v_{i,d}^{(t+1)}, & \text{if rand} \leq 0.5 \\ P_{i,d}^{(t)} + |x_{i,d}^{(t)} - f_{i,d}^{(t)}| \cdot e^{b \cdot l_{i,d}} \cdot \cos(2\pi \cdot l_{i,d}), & \text{otherwise} \end{cases} \quad (9)$$

The number of target flames, F_{no}, decreases with each iteration, starting from the population size, as shown in Eq. 3. The d-th dimension of the i-th target flame position $P_{i,d}^{(t)}$ is determined by Eq. 10, where t is the iteration number, $f_{i,d}^{(t)}$ represents the d-th dimension of the i-th flame's position, and $f_{F_{no},d}^{(t)}$ represents the d-th dimension of the moth located at the F_{no} index in the population.

$$P_{i,d}^{(t)} = \begin{cases} f_{i,d}^{(t)}, & \text{if } t \leq F_{no} \\ f_{F_{no},d}^{(t)}, & \text{otherwise} \end{cases} \quad (10)$$

3.2 Step-by-step Description of the MFPSO Algorithm

Algorithm 1 illustrates the steps of the MFPSO algorithm, and Fig. 1 shows the flowchart of the MFPSO sequence. First, the MFPSO parameters, including the inertial weight and the acceleration coefficients, are initialized. The second step of the MFPSO algorithm initializes a random population of moths M that contains all candidate solutions, where $nPop$ is the population size. Each moth has two attributes: the position x_i, which represents the solution itself (e.g., the PID parameters), and the velocity v_i, which is an inherited feature from the PSO algorithm used for the position.

Algorithm 1 The Proposed MFPSO Algorithm.

```
Initialize the MFPSO parameters.
Initialize a random population of moths M with positions and velocities (size nPop).
Evaluate the fitness of each moth using the fitness function
while Termination Condition do ▷ Main loop of the algorithm
    Sort the population M to obtain the flames F.
    Update velocities of all moths using equation Eq. 8.
    for each moth i = 1 to nPop do
        Get the next moth's position x_i from the population M.
        Generate random number r from 0 to 1.
        if r ≤ 0.5 then ▷ Group 1: PSO-based update (Exploitation)
            Get the moth's velocity v_i for the moth x_i.
            Update the moth's position x_i based on its velocity v_i (1st branch of Eq. 9).
        else ▷ Group 2: MFO-based update (Exploration)
            Update the flame numbers F_no using Eq. 3.
            Obtain the target flame P_i using Eq. 10.
            Update moth's position x_i based on target flame P_i (2nd branch of Eq. 9).
        end if
        Evaluate the fitness of the new moth's position.
    end for
    Update the global best solution gbest in the entire population based on their fitness.
end while
Return the global best gbest as the best-obtained solution.
```

In the third step of the initialization process, the fitness of each moth in the population is evaluated using the fitness function that indicates the quality of the moth. The fitness function depends on the problem and will be defined in the following sections for the PID tuning problem. Then, the loop of the MFPSO algorithm starts.

The first step inside the loop is inherited from the MFO algorithm, in which the moth population M is sorted based on the fitness values to generate a population of flames F. For each flame position x_i in the population M, there is a corresponding flame f_i from the population F.

The second step in the loop is inherited from the PSO algorithm, in which the velocity for each moth v_i is updated using Eq. 8. This update equation represents the exploitation power and the local search in the MFPSO algorithm, where the new moth's velocity v_i^{t+1} is generated by moving the current moth's position x_i^t towards the local best moth $pbest_i$ and the global best moth of the whole population $gbest$. The new moth's velocity v_i^{t+1} represents a differential step that makes the algorithm converge to the best solution and emphasizes exploitation.

The third step inside the loop divides the moths' and flames' populations into two equal groups. The division process is implemented by looping each moth and flame in the population and generating a random number r for each moth-flame pair. If r is less than or equal to 0.5, then the moth lies in group 1. Otherwise, it lies in group 2.

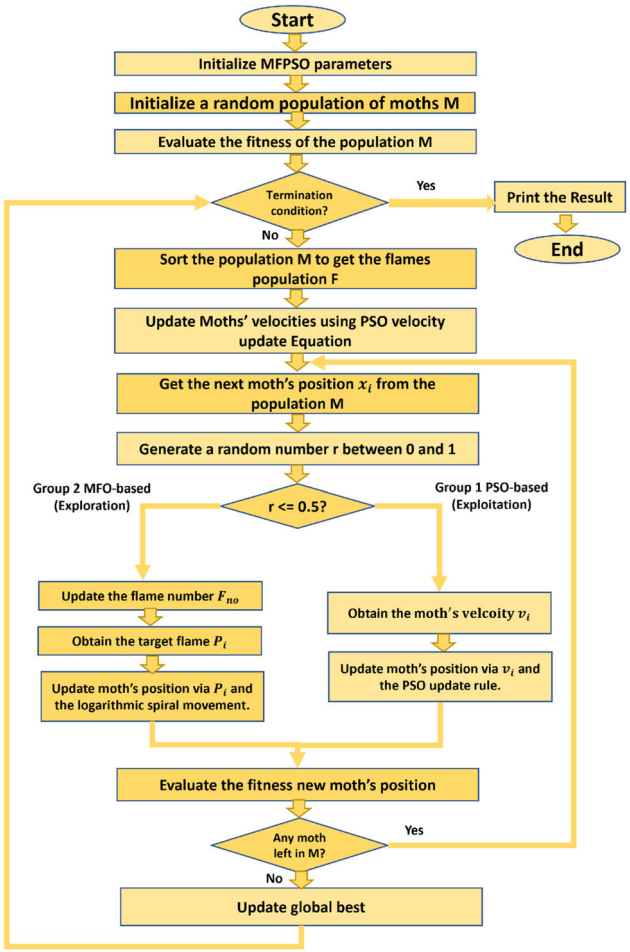


Fig. 1 Flowchart of the proposed MFPSO algorithm

Group 1 updates the moth's position based on the PSO methodology in which the new moth position moves towards the best solution, emphasizing the exploitation in the MFPSO algorithm. First, the moth's velocity v_i is obtained. Then, the new moth's position x_i^{t+1} is generated based on the moth's velocity v_i using the first branch in Eq. 9.

Group 2 updates the moth's position based on the logarithmic spiral movement of the MFO algorithm, in which the new moth moves away from its current position in a spiral movement, emphasizing the exploration in the MFPSO algorithm. First, the flame number F_{no} is updated using Eq. 3, and the target flame P_i is obtained from Eq. 10. Then, the new moth's position x_i^{t+1} is updated based on the target flame using the second branch in Eq. 9.

The new moth's position is evaluated using the fitness function for both groups. The final step in the main loop is to update the global best moth $gbest$, representing the moth with the best fitness value in the entire population. The termination condition is checked, and the algorithm terminates once met. Otherwise, the algorithm repeats the steps of the main loop. When the algorithm terminates, the algorithm returns the global best moth $gbest$ as the final best-obtained result of the optimization process.

The proposed grouping technique in the MFPSO algorithm ensures the diversity of the entire population, maintaining the balance between exploration and exploitation. By employing the original update equations from both PSO and MFO, the algorithm preserves these methods' strengths, enhancing the solutions' quality. In Group 1, the solutions are updated using the PSO-based approach, leveraging the exploitation strength of PSO to fine-tune solutions and improve convergence speed. Meanwhile, in Group 2, the positions are updated using the MFO-based approach, capitalizing on the exploration strength of MFO to prevent premature convergence and maintain diversity within the population. This balanced approach allows the MFPSO algorithm to combine the advantages of both methods, resulting in a more robust and effective optimization process that overcomes the limitations of traditional algorithms.

4 Sensitivity Analysis of the MFPSO Algorithm

This section discusses an in-depth sensitivity analysis of the parameters of the MFPSO algorithm. This analysis aims to understand how variations in the MFPSO parameters affect the algorithm's performance.

4.1 The MFPSO Parameters and Experiment Setup

The MFPSO algorithm incorporates both MFO-based and PSO-based parameters. The spiral constant (b) is an MFO parameter that defines the spiral shape and is fixed at 1. The convergence constant (a) is an adaptive MFO parameter that linearly increases from -2 to -1 over iterations [41]. The PSO parameters-inertia weight (w) and acceleration coefficients (c_1, c_2)-critically influence MFPSO performance. The cognitive component (c_1) reflects personal experience, driving moths toward their personal best, while the social component (c_2) encourages movement toward the global best [44].

The inertia weight (w) balances exploration and exploitation, making it the most crucial parameter for tuning, while c_1 and c_2 have lesser significance [45]. Harrison et al. recommended ranges for PSO parameters: $w \in [0.4, 0.8]$, $c_1 \in [0.5, 1.0]$, and $c_2 \in [2.5, 3.0]$ [46]. Isiet and Gadala further suggested that $c_1 = 2.5$, $c_2 = 1.0$, and the sum of c_1 and c_2 should not exceed 4, with w typically ranging from 0.1 to 1 [47].

This study explores the sensitivity of w, c_1, and c_2 on MFPSO performance. The parameter values tested include $w = 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$, $c_1 = 0.5, 1, 1.5, 2, 2.5, 3$, and $c_2 = 0.5, 1, 1.5, 2, 2.5, 6$, resulting in 216 configurations,

which are evaluated on the ten benchmark functions from CEC2020/2021 [48]. Each configuration runs five independent times per function, with termination conditions of 50000 function evaluations for 10D and 100000 for 20D cases. The Friedman test analyzes mean error results, and mean ranks are used for ranking. Results are visualized using heatmaps and bar charts, where lighter heatmap colors and shorter bars indicate better performance.

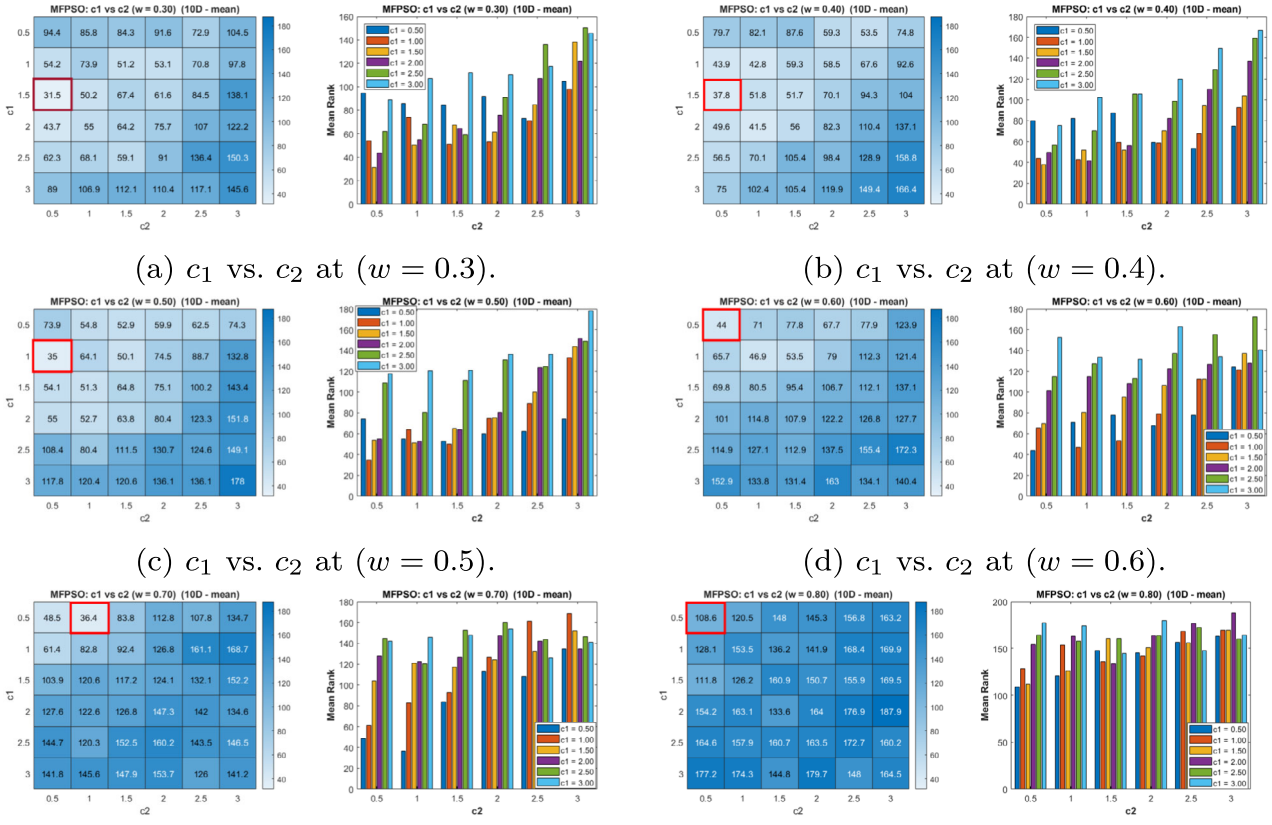
4.2 Recommendations and Results Discussion

In the 10D case, Fig. 2 visualizes the mean ranks for c_1 and c_2 across fixed w values. Additional projections and visualizations are provided in the Supplementary Materials (Section S1) for a more comprehensive analysis. At lower w (0.3-0.4), optimal performance is observed for $c_1 \approx 1.5$ and $c_2 \in [0.5, 1.0]$, achieving mean ranks between 31.5 and 51.8. For moderate w (0.5-0.6), the optimal c_1 decreases to [0.5, 1.0], while c_2 remains effective around 0.5. Higher w (0.7-0.8) results in degraded performance, though $c_1 = 0.5$ and $c_2 = 0.5$ still perform relatively better.

The best configuration for 10D is $c_1 = 1.5$, $c_2 = 0.5$, and $w = 0.3$, achieving a minimum mean rank of 31.5. Across all values of w in the 10D case, increasing c_1 initially improves MFPSO performance, peaking around $c_1 = 1.5$, before declining. Thus, a moderate c_1 prevents over-reliance on personal best and premature convergence, while $c_2 \in [0.5, 1.0]$ ensures sufficient attraction to the global best solution.

Figure 3 shows similar trends in the 20D case. Low w (0.3-0.4) yields better performance, with $c_1 = 1.5$ and $c_2 = 0.5$ achieving the best mean rank of 23.9. As c_2 increases, performance generally degrades, especially at medium w (0.5-0.6), where lower c_1 and c_2 values ($\approx$ 0.5-1.0) are more effective. High w (> 0.7) significantly reduces performance regardless of c_1 or c_2, emphasizing the negative impact of excessive inertia.

Keeping $w \in [0.3, 0.4]$ is critical for optimal performance for both dimensionalities. Within this range, c_1 should be [1.0, 1.5], avoiding over-reliance on individual experiences to prevent premature convergence. Similarly, $c_2 \in [0.5, 1.0]$ ensures sufficient attraction to the global best without excessive rigidity. High w values (> 0.6) should be avoided due to consistent performance degradation. The configuration

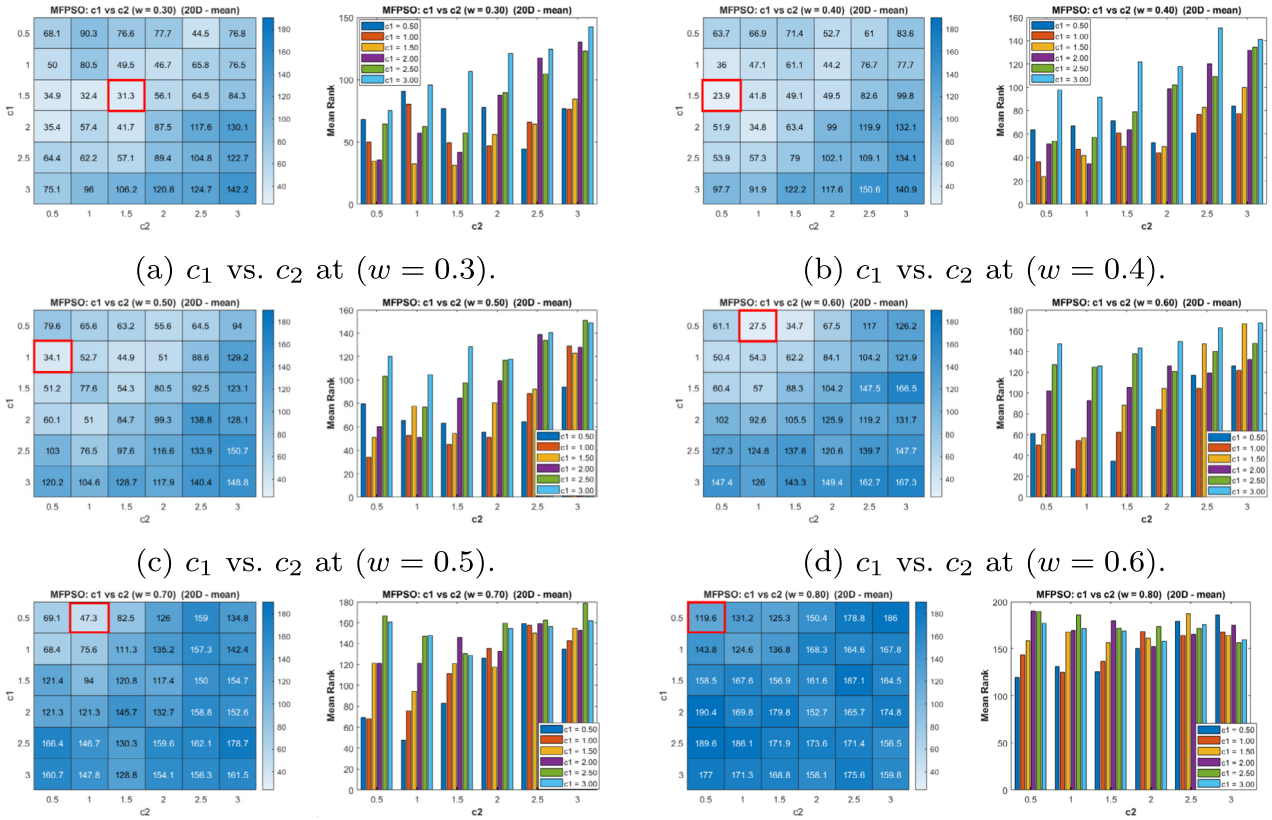


(a) c_1 vs. c_2 at $(w = 0.3)$. (b) c_1 vs. c_2 at $(w = 0.4)$.

(c) c_1 vs. c_2 at $(w = 0.5)$. (d) c_1 vs. c_2 at $(w = 0.6)$.

(e) c_1 vs. c_2 at $(w = 0.7)$. (f) c_1 vs. c_2 at $(w = 0.8)$.

Fig. 2 Heatmaps and bar charts for 10D functions for c_1 vs c_2 at fixed values of w



(a) c_1 vs. c_2 at $(w = 0.3)$. (b) c_1 vs. c_2 at $(w = 0.4)$.

(c) c_1 vs. c_2 at $(w = 0.5)$. (d) c_1 vs. c_2 at $(w = 0.6)$.

(e) c_1 vs. c_2 at $(w = 0.7)$. (f) c_1 vs. c_2 at $(w = 0.8)$.

Fig. 3 Heatmaps and bar charts for 20D functions for c_1 vs c_2 at fixed values of w

$c_1 = 1.5$, $c_2 = 0.5$, and $w \in [0.3, 0.4]$ emerges as the most robust and generalizable setup for the MFPSO algorithm.

5 Benchmark Testing on CEC2020/2021

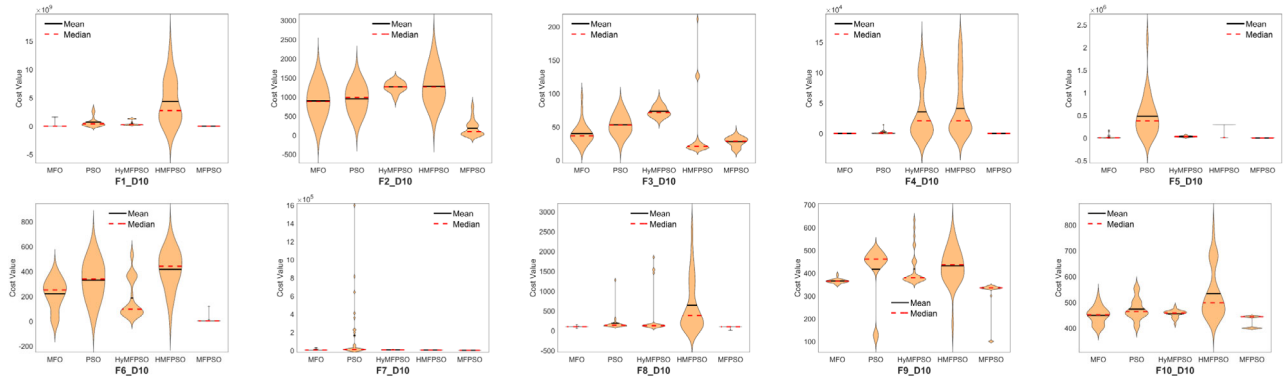
5.1 Parameter Settings and Results Collection

The proposed MFPSO algorithm is compared with PSO, MFO, and two recent hybrid algorithms, HyMFPSO [42] and HMFPSO [14, 43], using 10D and 20D CEC2020/2021 benchmark functions [48, 49]. Parameter settings for all five algorithms are summarized in Table 1. The termination condition is set to a maximum of 200000 function evaluations (MaxFES) for 10D and 500000 for 20D. The fitness function for each benchmark is the error, $E_i(x)$, calculated as the difference between the global best solution fitness $F_i(X)$ and the known optimal fitness $F_i(X^*)$. Algorithms terminate if the error is less than or equal to 1.0×10^{-8}. All algorithms are run 30 times per function, with dimensions ranging from $[-100, 100]$ [48].

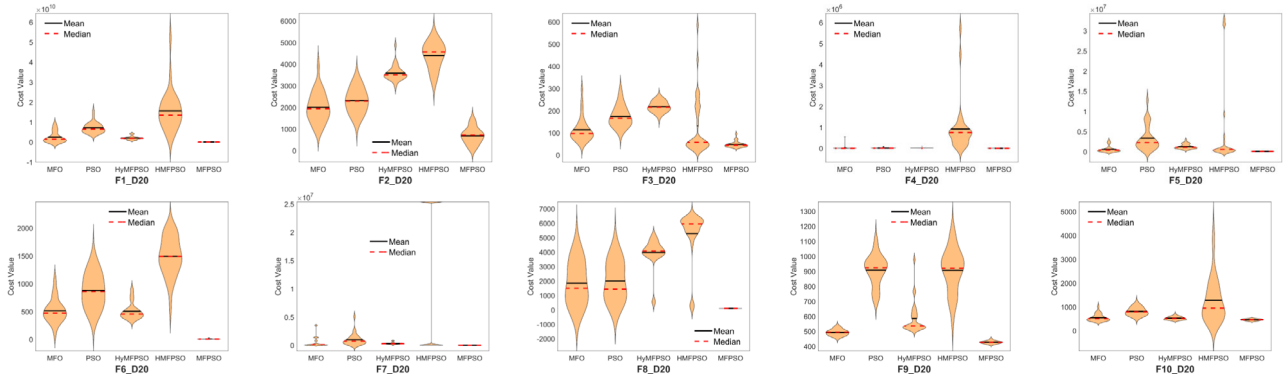
Error results across all 10D and 20D functions are visualized using violin plots, with Fig. 4a showing 10D results and Fig. 4b showing 20D results. MFPSO demonstrates the most reliable and repeatable distributions, with compact plots indicating high consistency. It achieves the lowest errors and the

Table 1 Parameter Settings for all the Algorithms

Algorithm	Parameter Values
All	population size = 30
MFO [41]	$b = 1, a_{max} = -1, a_{min} = -2$
PSO [50]	$c_1 = 1, c_2 = 1, w = 0.3$
HyMFPSO [42]	$c_1 = 2, c_2 = 2, w_{max} = 0.9, w_{min} = 0.2, b = 1, a_{max} = 1, a_{min} = -1$
HMFPSO [14, 43]	$b = 1, a_{max} = 1, a_{min} = -1$
MFPSO	$c_1 = 1.5, c_2 = 0.5, w = 0.4, b = 1, a_{max} = -1, a_{min} = -2$



(a) 10 Dim.



(b) 20 Dim.

Fig. 4 Violin plots for all the algorithms CEC2020/2021 functions

smallest spread compared to other algorithms. Comprehensive error metrics, including best, worst, mean, median, and standard deviation, as well as detailed box plots, are provided in the Supplementary Materials (Section S3.1).

5.2 Statistical Analysis of the Results

The statistical analysis, summarized in Table 2, evaluates the significance of the MFPSO algorithm's performance compared to other algorithms. The Friedman test was conducted to determine overall rankings across all metrics and functions for 10D and 20D cases. The MFPSO algorithm consistently ranked first across all metrics and dimensions with the lowest mean rank and a p-value < 0.05, confirming its superior performance. The MFO algorithm ranked second overall but fell to third or fourth in some cases, such as the 20D worst (third) and 20D SD metrics (fourth). HyMFPSO ranked third in the mean and median metrics for both dimensions.

Paired comparisons using the sign test and Wilcoxon test confirmed the dominance of the MFPSO algorithm. It outperformed all other algorithms in all metrics and dimensions, with p-values < 0.05 in 49 out of 50 comparisons. The only exception was the best metric for the 20D case against MFO ($p = 0.084$). For mean and median metrics, MFPSO achieved better results in at least 9 or 10 functions compared to MFO, PSO, HyMFPSO, and HMFPSO. These results demonstrate the MFPSO algorithm's significant and consistent superiority across all metrics and functions.

5.3 Confidence Interval (CI) Analysis and Confidence Curves

The confidence interval (CI) analysis evaluates the performance of the MFPSO algorithm against MFO, PSO, HyMFPSO, and HMFPSO. Using MFPSO as the reference, unpaired comparisons were conducted across 30 independent runs for each CEC2020 benchmark function, following a non-parametric ranking approach. All mathematical details and equations are provided in the Supplementary Materials (Section S2.1) and in [51]. Table 3 presents the CI results for 10D and 20D functions, where UB and LB represent the upper and lower bounds. If the CI includes zero, the difference is not statistically significant (pCI = FALSE); otherwise, it is significant (pCI = TRUE). A positive CI indicates MFPSO performs better, while a negative CI indicates the other algorithm is superior.

Table 2 Statistical analysis for 10D and 20D functions across all algorithms

Dim	Metric	Alg.	Friedman Test				Sign Test	Wilcoxon Test			
			SumRank	MeanRank	Rank	p-value	+/=/-	R+	R-	p-value	H
10D	Best	MFO	22	2.2	2	2.03E-05	+9/=0/-1	54	1	0.003906250	TRUE
		PSO	35	3.5	3		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	45	4.5	5		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	36	3.6	4		+9/=0/-1	54	1	0.00390625	TRUE
		MFPSO	12	1.2	1		NA	NA	NA	NA	NA
	Worst	MFO	29	2.9	2	2.53E-05	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	36	3.6	4		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	30	3	3		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	45	4.5	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	10	1	1		NA	NA	NA	NA	NA
	Median	MFO	23	2.3	2	3.54E-05	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	41	4.1	5		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	37	3.7	3		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	38	3.8	4		+9/=0/-1	54	1	0.00390625	TRUE
		MFPSO	11	1.1	1		NA	NA	NA	NA	NA
	Mean	MFO	23	2.3	2	6.61E-06	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	38	3.8	4		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	35	3.5	3		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	44	4.4	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	10	1	1		NA	NA	NA	NA	NA
	SD	MFO	26	2.6	2.5	1.07E-04	+8/=0/-2	46	9	0.064453125	TRUE
		PSO	40	4	4		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	26	2.6	2.5		+8/=0/-2	48	7	0.037109375	TRUE
		HMFPSO	44	4.4	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	14	1.4	1		NA	NA	NA	NA	NA
20D	Best	MFO	20	2	2	1.40E-05	+9/=0/-1	45	10	0.083984375	TRUE
		PSO	41	4.1	5		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	40	4	4		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	37	3.7	3		+9/=0/-1	53	2	0.005859375	TRUE
		MFPSO	12	1.2	1		NA	NA	NA	NA	NA
	Worst	MFO	29	2.9	3	1.17E-06	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	34	3.4	4		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	27	2.7	2		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	50	5	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	10	1	1		NA	NA	NA	NA	NA
	Median	MFO	24	2.4	2	8.27E-06	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	39	3.9	4		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	33	3.3	3		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	44	4.4	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	10	1	1		NA	NA	NA	NA	NA

Table 2 continued

Dim	Metric	Alg.	Friedman Test				Sign Test	Wilcoxon Test			
			SumRank	MeanRank	Rank	p-value	+/=/-	R+	R-	p-value	H
	Mean	MFO	26	2.6	2	3.36E-06	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	38	3.8	4		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	29	2.9	3		+10/=0/-0	55	0	0.001953125	TRUE
		HMFPSO	47	4.7	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	10	1	1		NA	NA	NA	NA	NA
	SD	MFO	35	3.5	4	8.35E-07	+10/=0/-0	55	0	0.001953125	TRUE
		PSO	34	3.4	3		+10/=0/-0	55	0	0.001953125	TRUE
		HyMFPSO	21	2.1	2		+9/=0/-1	53	2	0.005859375	TRUE
		HMFPSO	49	4.9	5		+10/=0/-0	55	0	0.001953125	TRUE
		MFPSO	11	1.1	1		NA	NA	NA	NA	NA

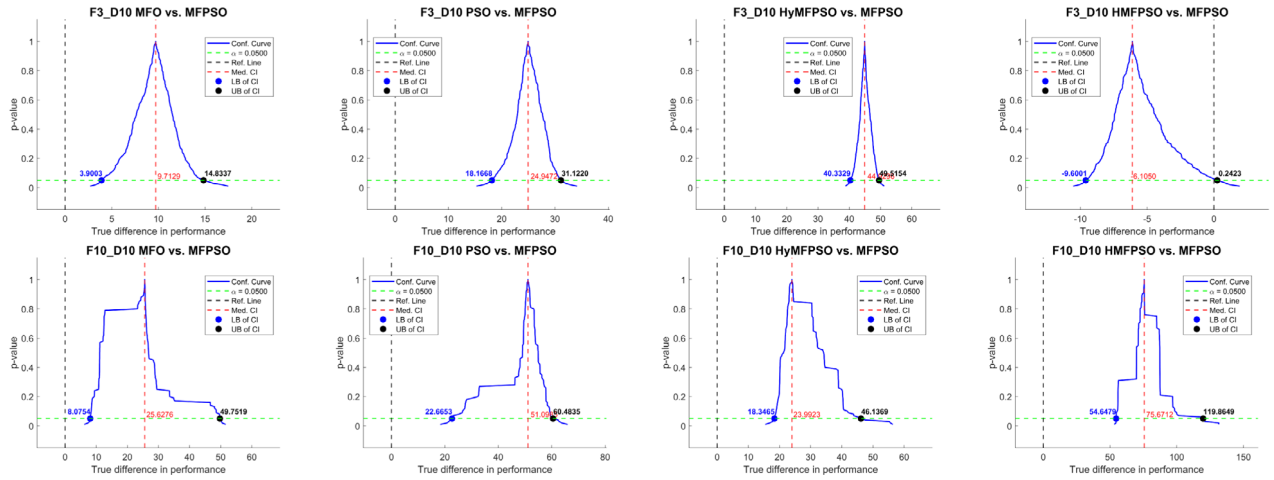
MFPSO serves as the reference for paired comparisons. For the Sign Test, '+' indicates the number of functions where MFPSO performs better, and '=' indicates a draw. In the Wilcoxon Test, $R+ > R-$ indicates MFPSO's superiority. The Friedman Test ranks algorithms, with the lowest mean rank indicating the best performance. Results are significant for p-values ≤ 0.05

Figure 5 illustrates the confidence curves for 10D and 20D functions (F3 and F10), with results for other functions in the Supplementary Materials (Section S3.2). Confidence curves lying entirely on the positive x-axis confirm MFPSO's superior performance, while those on the negative x-axis indicate the other algorithm performs better. MFPSO outperformed MFO, PSO, and HyMFPSO across all 10D and 20D functions with significant positive CIs. MFPSO also outperformed HMFPSO in nine 10D and all 20D functions. For F3-10D, the zero line within the curve indicates no significant

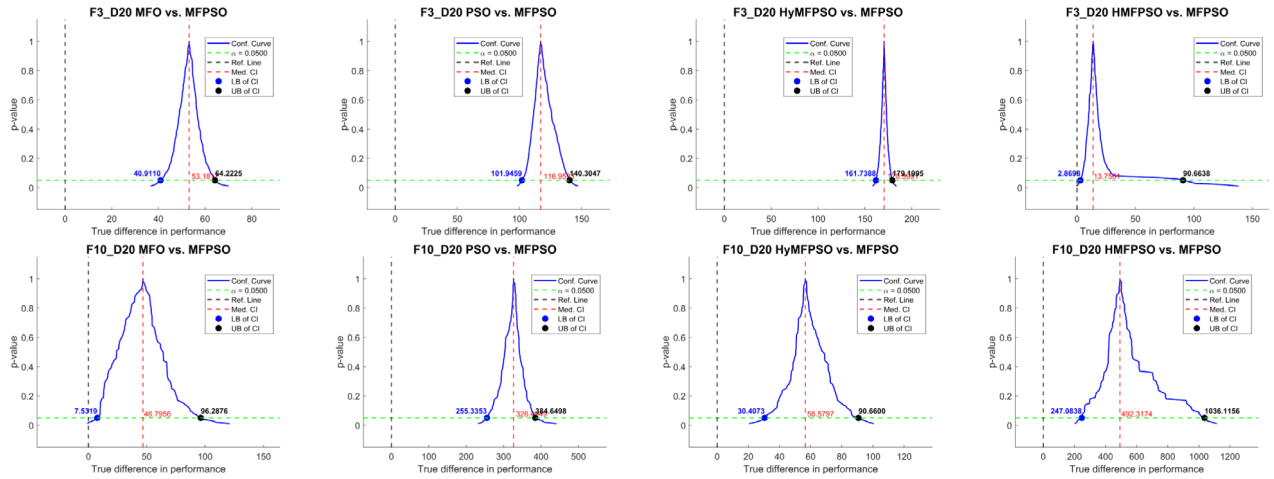
Table 3 Confidence interval (CI) results for 10D and 20D CEC2020/2021 benchmark functions

Dim	Fn.	MFO			PSO			HyMFPSO			HMFPSO		
		LB	UB	pCI	LB	UB	pCI	LB	UB	pCI	LB	UB	pCI
10	F1	4.40E+03	1.13E+04	TRUE	2.09E+08	6.13E+08	TRUE	2.33E+08	3.37E+08	TRUE	1.60E+09	7.21E+09	TRUE
	F2	5.46E+02	8.83E+02	TRUE	6.04E+02	9.63E+02	TRUE	1.02E+03	1.20E+03	TRUE	8.66E+02	1.27E+03	TRUE
	F3	3.90E+00	1.48E+01	TRUE	1.82E+01	3.11E+01	TRUE	4.03E+01	4.95E+01	TRUE	-9.60E+00	2.42E-01	FALSE
	F4	9.42E-01	2.62E+00	TRUE	2.40E+02	1.29E+03	TRUE	1.79E+03	2.06E+04	TRUE	4.84E+03	4.91E+04	TRUE
	F5	2.38E+03	8.88E+03	TRUE	2.67E+05	5.09E+05	TRUE	2.30E+04	3.21E+04	TRUE	7.43E+03	8.53E+03	TRUE
	F6	1.94E+02	2.73E+02	TRUE	2.52E+02	4.16E+02	TRUE	7.00E+01	2.11E+02	TRUE	3.69E+02	4.98E+02	TRUE
	F7	1.63E+03	3.30E+03	TRUE	7.14E+03	5.06E+04	TRUE	3.51E+03	5.47E+03	TRUE	1.61E+03	1.87E+03	TRUE
	F8	4.46E-01	3.28E+00	TRUE	2.17E+01	6.37E+01	TRUE	1.93E+01	4.77E+01	TRUE	6.82E+01	4.59E+02	TRUE
	F9	2.59E+01	3.33E+01	TRUE	1.12E+02	1.33E+02	TRUE	3.99E+01	1.06E+02	TRUE	6.44E+01	1.37E+02	TRUE
	F10	8.08E+00	4.98E+01	TRUE	2.27E+01	6.05E+01	TRUE	1.83E+01	4.61E+01	TRUE	5.46E+01	1.20E+02	TRUE
20	F1	8.74E+08	2.43E+09	TRUE	5.54E+09	7.72E+09	TRUE	1.76E+09	2.03E+09	TRUE	1.24E+10	1.79E+10	TRUE
	F2	9.82E+02	1.57E+03	TRUE	1.39E+03	1.88E+03	TRUE	2.70E+03	3.06E+03	TRUE	3.50E+03	4.03E+03	TRUE
	F3	4.09E+01	6.42E+01	TRUE	1.02E+02	1.40E+02	TRUE	1.62E+02	1.79E+02	TRUE	2.87E+00	9.07E+01	TRUE
	F4	3.69E+02	2.94E+03	TRUE	4.99E+03	1.23E+04	TRUE	3.65E+01	4.68E+02	TRUE	6.16E+05	7.68E+05	TRUE
	F5	2.88E+04	3.75E+05	TRUE	1.68E+06	3.14E+06	TRUE	8.26E+05	1.21E+06	TRUE	4.67E+04	5.24E+05	TRUE
	F6	4.11E+02	5.53E+02	TRUE	7.06E+02	1.02E+03	TRUE	4.31E+02	5.01E+02	TRUE	1.38E+03	1.67E+03	TRUE
	F7	2.34E+04	2.41E+05	TRUE	4.53E+05	8.44E+05	TRUE	2.17E+05	2.72E+05	TRUE	1.34E+06	1.40E+06	TRUE
	F8	8.36E+02	2.82E+03	TRUE	9.11E+02	2.93E+03	TRUE	3.80E+03	4.21E+03	TRUE	5.47E+03	6.04E+03	TRUE
	F9	5.45E+01	7.29E+01	TRUE	4.67E+02	5.28E+02	TRUE	1.03E+02	1.62E+02	TRUE	4.70E+02	5.54E+02	TRUE
	F10	7.53E+00	9.63E+01	TRUE	2.55E+02	3.85E+02	TRUE	3.04E+01	9.07E+01	TRUE	2.47E+02	1.04E+03	TRUE

MFPSO is the reference for unpaired comparisons across 30 runs. LB and UB represent the lower and upper bounds of the CI. pCI is true if the CI excludes zero, indicating a significant difference. Positive CI favors MFPSO, while negative CI favors the other algorithm



(a) 10 Dim.



(b) 20 Dim.

Fig. 5 Confidence curves for F3 and F10 CEC2020/2021 benchmark functions across the 30 runs for all possible significant levels, where MFPSO is compared with the MFO, PSO, HyMFPSO, and HMFPSO algorithms

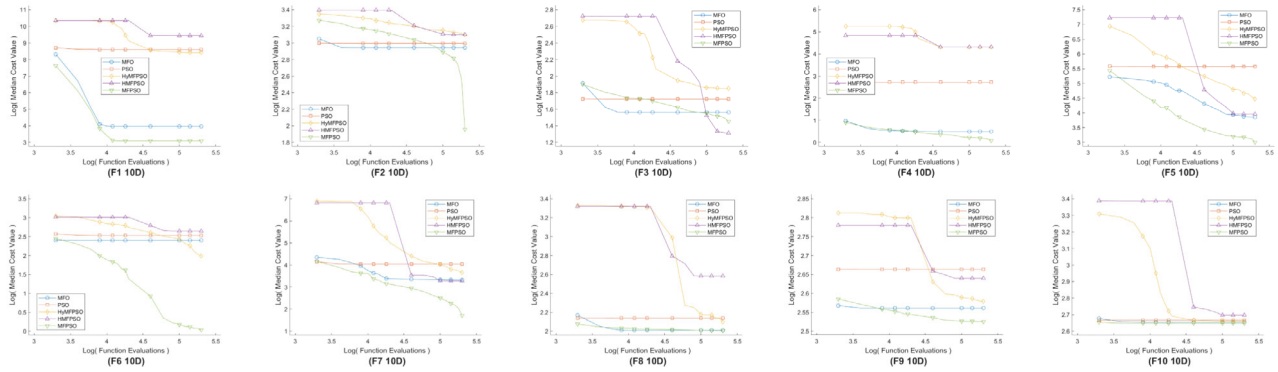
difference between MFPSO and HMFPSO. Across all other functions, MFPSO demonstrated significant superiority, with confidence curves consistently on the positive x-axis.

5.4 Convergence Analysis

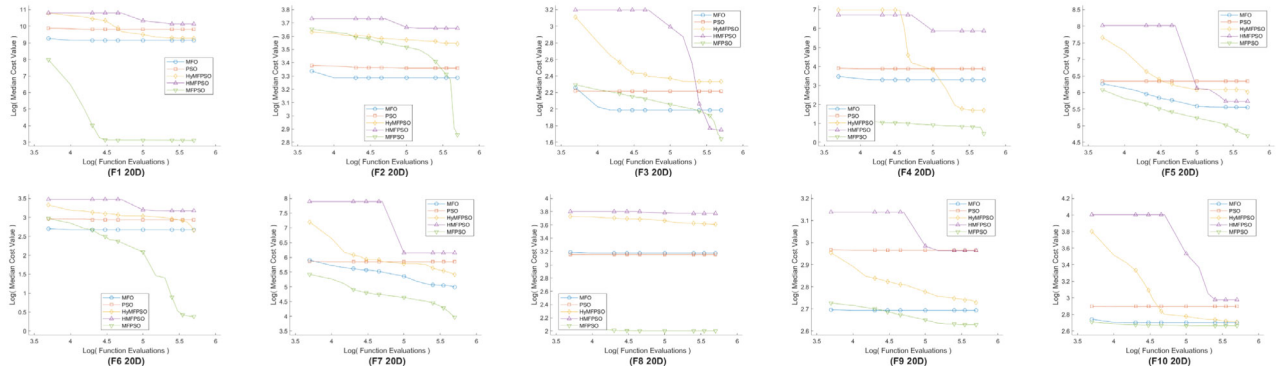
The convergence of the MFPSO algorithm is compared with other algorithms using median error monitored at equidistant cut points across 30 runs for each function and dimension. Figure 6 shows the convergence plots for all algorithms on 10D and 20D benchmark functions. MFPSO demonstrates faster convergence, starting with a higher median error but consistently terminating with the lowest error compared to other algorithms. Statistical analysis confirms that MFPSO outperforms MFO, PSO, and hybrid variants.

Page's test, a non-parametric statistical test, is used to compare convergence trends as detailed in [51]. In this test, $N = 19$ represents cut points across runs, and $k = 10$ denotes the benchmark functions. Median error differences between algorithms A and B are analyzed: a p-value ≤ 0.05 with an increasing trend indicates B converges faster, while a decreasing trend indicates A converges faster. Full details are available in the Supplementary Materials (Section S2.2, S3.3) and in [51].

The trends visualized in Fig. 7 provide a comprehensive analysis of the convergence patterns between MFPSO and other algorithms through Page's test. Notably, the HyMFPSO-MFPSO comparison (Fig. 7c) and the HMFPSO-MFPSO comparison (Fig. 7d) reveal significant increasing trends in both the 10D and 20D cases, indicating that MFPSO

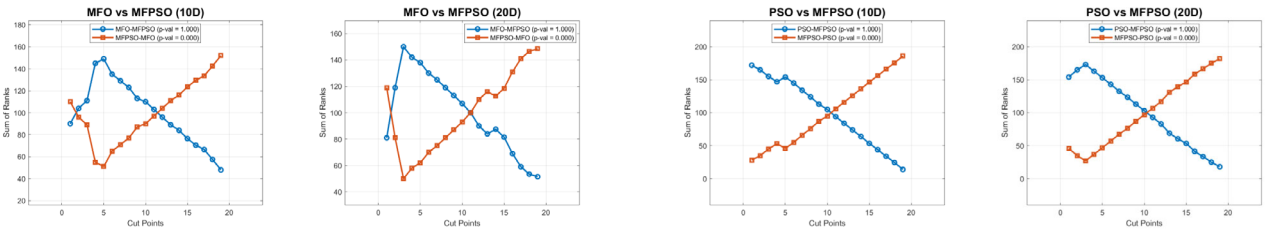


(a) 10 dimensions.

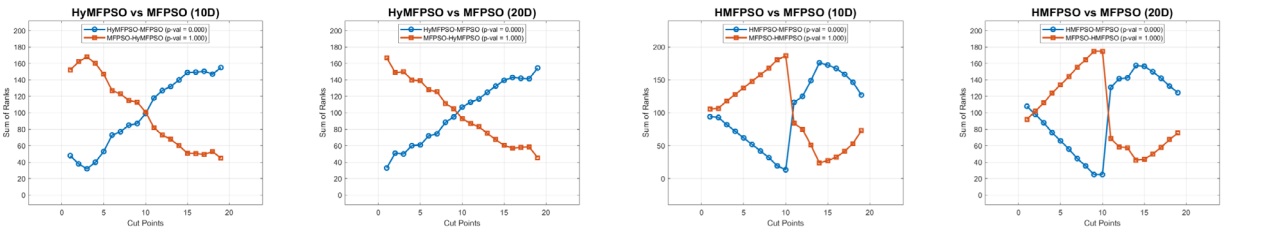


(b) 20 dimensions.

Fig. 6 Convergence graphs for the proposed MFPSO and MFO algorithms for the median error across 30 runs for all CEC2020/2021 functions



(a) MFO vs MFPSO (10D and 20D). (b) PSO vs MFPSO (10D and 20D).



(c) HyMFPSO vs MFPSO (10D and 20D). (d) HMFPSO vs MFPSO (10D and 20D).

Fig. 7 Convergence trends between the algorithms using the Page's test

consistently achieves faster convergence than HyMFPSO and HMFPSO.

Conversely, the MFPSO-PSO comparison (Fig. 7b) shows a significant increasing trend in favor of PSO. Meanwhile, the MFPSO-MFO comparison (Fig. 7a) exhibits a mixed significant trend: MFPSO demonstrates faster convergence during the initial optimization phases, but this trend reverses in later iterations, with MFO showing faster convergence. These results confirm MFPSO's capability to effectively balance convergence speed and accuracy, excelling against hybrid variants while demonstrating competitive trade-offs with simpler algorithms like PSO and MFO.

5.5 Computational Complexity

The space complexity of all the compared algorithms, including the MFPSO, is $\mathcal{O}(n_{\text{Pop}} \times D)$, where n_{Pop} is the population size and D is the problem dimensionality. It arises from the storage of solution parameters such as position, velocity, and fitness values. While PSO, HyMFPSO, HMFPSO, and MFPSO require additional storage for velocity and personal bests, and MFO uses a simpler structure, all algorithms share the same overall complexity, ensuring consistent memory requirements for fair comparison.

Computation time is a critical measure of an algorithm's complexity, particularly for real-time applications. Complex algorithms consume more time and computational resources, representing crucial factors for real-time applications. Four-time metrics, T_0, T_1, T_2, and T_3, as introduced in [48], are used to compare the algorithms. Detailed explanations are provided in the Supplementary Materials (Section S2.3).

Table 4 summarizes the time complexity results for all algorithms on the ten CEC2020/2021 benchmark functions for 10D and 20D, evaluated on an Intel Core-i7-6500U CPU (2.60GHz and 2.50GHz), 8GB RAM, Windows 11 23H2, and Matlab R2024a. T_0 is 0.002290, while T_1 equals 0.085644 and 0.170504 for 10D and 20D, respectively.

The PSO algorithm is the fastest with T_3 values of 65.622610 (10D) and 48.749548 (20D). MFPSO ranks second with T_3 values of 69.617994 (10D) and 57.214002 (20D), followed by MFO (third) and HMFPSO (fourth), while HyMFPSO is the slowest. Despite the PSO being the fastest, MFPSO provides a balanced trade-off between computational efficiency and performance accuracy with a competitive time complexity, outperforming more complex algorithms like MFO, HMFPSO, and HyMFPSO, making it suitable for real-time applications requiring both speed and accuracy.

6 Application on Engineering Optimization Benchmark Problems

The MFPSO algorithm is validated against MFO, PSO, HyMFPSO, and HMFPSO using thirteen benchmark engineering optimization problems spanning various real-world applications. Table 5 presents the problem dimensions, global minima, and variable bounds, with detailed problem formulations provided in the Supplementary Materials (Section S4) [52]. All algorithms were tested over 30 independent runs per problem, with a maximum of 150,000 function evaluations (FEs) to ensure consistency and fair comparison.

6.1 Results Tables

The comparative study results across 13 benchmark engineering optimization problems over 30 runs are summarized in Table 6, detailing cost metrics (median, mean, SD), the best solution, and corresponding cost for each algorithm. The success rate (SR), indicating the percentage of runs achieving the global best solution, highlights MFPSO's superiority. MFPSO achieved a 100% SR on seven problems (F1, F8, F10, F12, F13), outperforming all competitors. For F2, F5, F7, and F11, where no algorithm reached the global best,

Table 4 Time complexity for all the algorithms for all the 10 benchmark functions (10D and 20D)

Dim	Algorithm	T0	T1	T2	T3	Rank
10	MFO	0.002290	0.085644	0.256041	74.419029	3
	PSO	0.002290	0.085644	0.235900	65.622610	1
	HyMFPSO	0.002290	0.085644	0.315132	100.226335	5
	HMFPSO	0.002290	0.085644	0.310424	98.170079	4
	MFPSO	0.002290	0.085644	0.245048	69.617994	2
20	MFO	0.002290	0.170504	0.312785	62.139656	3
	PSO	0.002290	0.170504	0.282126	48.749548	1
	HyMFPSO	0.002290	0.170504	0.440347	117.850941	5
	HMFPSO	0.002290	0.170504	0.327062	68.374739	4
	MFPSO	0.002290	0.170504	0.301507	57.214002	2

Maximum function evaluations are 10000

Table 5 Benchmark problems used for validation, along with their details [52, 53]

FNo.	Problem Name	Dim	Bounds	Global Best
F1	Speed Reducer Design	7	LB: [2.6, 0.7, 17, 7.3, 7.3, 2.9, 5] UB: [3.6, 0.8, 28, 8.3, 8.3, 3.9, 5.5]	2994.4245
F2	Tension/Compression Spring Design	3	LB: [0.05, 0.25, 2.0] UB: [2.0, 1.3, 15.0]	0.0127
F3	Pressure Vessel Design	4	LB: [0.51, 0.51, 10, 10] UB: [99.49, 99.49, 200, 200]	6059.7143
F4	Three-Bar Truss Design	2	LB: [0, 0] UB: [1, 1]	263.8958
F5	Gear Train Design	4	LB: [12, 12, 12, 12] UB: [60, 60, 60, 60]	2.701×10^{-12}
F6	Cantilever Beam Design	5	LB: [0.01, 0.01, 0.01, 0.01, 0.01] UB: [100, 100, 100, 100, 100]	1.3400
F7	Minimize I-Beam Deflection	4	LB: [10, 10, 0.9, 0.9] UB: [80, 50, 5.0, 5.0]	0.0131
F8	Tubular Column Design	2	LB: [2, 0.2] UB: [14, 0.8]	26.4864
F9	Piston Lever Design	4	LB: [0.05, 0.05, 0.05, 0.05] UB: [500, 500, 500, 120]	8.4127
F10	Corrugated Bulkhead Design	4	LB: [0, 0, 0, 0] UB: [100, 100, 100, 5]	6.8430
F11	Car Side Impact Design	11	LB: [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0, 0, -30, -30] UB: [1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1, 1, 30, 30]	22.8430
F12	Welded Beam Design	4	LB: [0.1, 0.1, 0.1, 0.1] UB: [2, 10, 10, 2]	1.7249
F13	Reinforced Concrete Beam Design	3	LB: [0, 0, 5] UB: [1, 1, 10]	359.2080

MFPSO obtained the lowest median and mean cost values, showcasing its consistency. Additionally, MFPSO was the only algorithm to achieve a non-zero SR on F3, F4, F6, and F9, demonstrating its robustness on challenging problems. Figure 8 visualizes the solution distributions, further emphasizing MFPSO's superior performance. Additional box plots and metric tables are available in the Supplementary Materials (Section S5.1).

6.2 Statistical Analysis of the Results

The statistical analysis in Table 7 comprehensively evaluates the performance of all algorithms across multiple metrics. MFPSO consistently achieves the lowest mean rank for all metrics in the Friedman test, with significant p-values, reflecting its robust performance compared to MFO, PSO, HyMFPSO, and HMFPSO. Moreover, MFPSO dominates the performance, outperforming its competitors across all metrics, as indicated by the Sign and Wilcoxon tests. The Wilcoxon test further corroborates these findings, with MFPSO demonstrating significantly better performance (dominated by positive ranks) for all metrics with consistently significant p-values (<0.05), confirming the statistical significance of MFPSO's improvements.

The confidence interval (CI) of the mean results difference for the unpaired comparison across the 30 runs, as presented in Table 8, highlights the significant performance of MFPSO in function-by-function evaluations. The results reveal entirely positive CIs (pCI=TRUE) in 37 out of 52 comparisons, indicating statistically significant performance differences favoring MFPSO. Importantly, no entirely negative CIs were observed across the benchmark functions, underscoring that MFPSO did not experience any definitive defeats against its competitors. The consistent presence of positive and statistically significant CIs favoring MFPSO across most problems underscores its robustness, precision, and reliability in solving challenging engineering optimization tasks. Figure 9 visually demonstrates the confidence

Table 6 The results of all the algorithms in 30 runs for all the engineering optimization benchmark problems

FNo.	Alg.	Cost Over 30 Runs				Best Solution		
		Median	Mean	SD	SR	Cost	D	Position from X_1 to X_D
F1	MFO	2.994424E+03	2.994424E+03	4.63E-13	100.00	2.994424E+03	7	3.50, 0.70, 17.00, 7.30, 7.72, 3.35, 5.29,
	PSO	3.006485E+03	3.008281E+03	1.66E+01	0.00	3.000325E+03	7	3.50, 0.70, 17.00, 7.71, 7.80, 3.35, 5.29,
	HyMFPSO	3.002253E+03	3.028013E+03	5.68E+01	0.00	3.000607E+03	7	3.50, 0.70, 17.00, 7.30, 7.93, 3.35, 5.29,
	HMFPSO	3.007390E+03	3.010277E+03	5.53E+00	0.00	3.003760E+03	7	3.50, 0.70, 17.00, 8.30, 7.72, 3.35, 5.29,
	MFPSO	2.994424E+03	2.994424E+03	4.63E-13	100.00	2.994424E+03	7	3.50, 0.70, 17.00, 7.30, 7.72, 3.35, 5.29,
F2	MFO	1.311365E-02	1.332242E-02	6.27E-04	0.00	1.267441E-02	3	0.05, 0.37, 10.34,
	PSO	1.618601E-02	1.620836E-02	2.51E-03	0.00	1.267953E-02	3	0.05, 0.38, 10.11,
	HyMFPSO	1.275396E-02	1.278978E-02	1.11E-04	0.00	1.273373E-02	3	0.05, 0.32, 14.06,
	HMFPSO	1.271906E-02	1.474070E-02	2.52E-03	0.00	1.271905E-02	3	0.05, 0.32, 14.03,
	MFPSO	1.269620E-02	1.269624E-02	1.45E-05	0.00	1.266809E-02	3	0.05, 0.36, 10.90,
F3	MFO	6.069587E+03	6.181671E+03	2.15E+02	0.00	6.069587E+03	4	12.21, 6.48, 40.32, 200.00,
	PSO	9.170091E+03	8.525022E+03	2.14E+03	0.00	6.222400E+03	4	12.81, 6.68, 43.51, 159.85,
	HyMFPSO	6.461508E+03	6.474603E+03	2.24E+02	0.00	6.123560E+03	4	12.70, 6.46, 42.04, 179.26,
	HMFPSO	6.069587E+03	6.530742E+03	7.18E+02	0.00	6.059739E+03	4	12.58, 7.49, 42.10, 176.63,
	MFPSO	6.059714E+03	6.068969E+03	1.44E+01	50.00	6.059714E+03	4	12.63, 7.36, 42.10, 176.64,
F4	MFO	2.638959E+02	2.638961E+02	3.69E-04	0.00	2.638959E+02	2	0.79, 0.41,
	PSO	2.638963E+02	2.638963E+02	7.90E-04	6.67	2.638958E+02	2	0.79, 0.41,
	HyMFPSO	2.639009E+02	2.639008E+02	3.40E-03	0.00	2.638966E+02	2	0.79, 0.41,
	HMFPSO	2.638958E+02	2.638959E+02	1.08E-05	0.00	2.638958E+02	2	0.79, 0.41,
	MFPSO	2.638958E+02	2.638958E+02	3.97E-08	3.33	2.638958E+02	2	0.79, 0.41,
F5	MFO	8.887614E-10	3.655184E-09	5.58E-09	0.00	1.166116E-10	4	54.40, 22.05, 17.32, 48.50,
	PSO	1.117291E-08	2.312508E-06	8.27E-06	0.00	1.545045E-10	4	44.33, 12.51, 21.28, 43.45,
	HyMFPSO	1.166116E-10	3.846974E-10	4.48E-10	0.00	2.700857E-12	4	43.35, 15.59, 19.44, 48.75,
	HMFPSO	9.921580E-10	8.296868E-09	1.18E-08	0.00	2.700857E-12	4	48.88, 16.39, 19.17, 43.21,
	MFPSO	2.700857E-12	1.701122E-10	3.66E-10	0.00	2.700857E-12	4	42.98, 18.55, 15.65, 48.62,
F6	MFO	1.340293E+00	1.340330E+00	2.26E-04	0.00	1.340097E+00	5	5.96, 5.36, 4.49, 3.52, 2.16,
	PSO	3.569919E+00	3.235528E+00	8.73E-01	0.00	1.578109E+00	5	4.94, 6.05, 4.30, 7.59, 2.41,
	HyMFPSO	2.072407E+00	2.072805E+00	1.07E-01	0.00	1.772556E+00	5	7.19, 4.07, 10.01, 5.00, 2.14,
	HMFPSO	1.340301E+00	1.340637E+00	7.55E-04	0.00	1.340089E+00	5	5.96, 5.31, 4.49, 3.55, 2.16,
	MFPSO	1.339958E+00	1.339959E+00	1.34E-06	6.67	1.339958E+00	5	6.01, 5.31, 4.50, 3.50, 2.15,
F7	MFO	1.307412E-02	1.308491E-02	4.20E-05	0.00	1.307412E-02	4	80.00, 50.00, 0.90, 2.32,
	PSO	1.474180E-02	2.019016E-02	1.52E-02	0.00	1.336496E-02	4	80.00, 35.67, 0.90, 3.28,
	HyMFPSO	1.307426E-02	1.307429E-02	2.08E-07	0.00	1.307413E-02	4	80.00, 50.00, 0.90, 2.32,

Table 6 continued

FNo.	Alg.	Cost Over 30 Runs				Best Solution		
		Median	Mean	SD	SR	Cost	D	Position from X_1 to X_D
	HMFPSO	1.307412E-02	1.307412E-02	8.82E-18	0.00	1.307412E-02	4	80.00, 50.00, 0.90, 2.32,
	MFPSO	1.307412E-02	1.307412E-02	8.82E-18	0.00	1.307412E-02	4	80.00, 50.00, 0.90, 2.32,
F8	MFO	2.648636E+01	2.648636E+01	7.23E-15	100.00	2.648636E+01	2	5.45, 0.29,
	PSO	2.686207E+01	2.680505E+01	3.12E-01	36.67	2.648636E+01	2	5.45, 0.29,
	HyMFPSO	2.654395E+01	2.654600E+01	2.72E-02	0.00	2.649364E+01	2	5.45, 0.29,
	HMFPSO	2.648636E+01	2.648636E+01	7.23E-15	100.00	2.648636E+01	2	5.45, 0.29,
	MFPSO	2.648636E+01	2.648636E+01	7.23E-15	100.00	2.648636E+01	2	5.45, 0.29,
F9	MFO	1.674727E+02	9.324472E+01	8.07E+01	46.67	8.412698E+00	4	0.05, 2.04, 4.08, 120.00,
	PSO	3.927654E+02	3.893292E+02	1.10E+02	0.00	2.648809E+02	4	418.59, 340.38, 2.73, 60.00,
	HyMFPSO	1.399820E+01	1.375401E+01	3.24E+00	0.00	9.816044E+00	4	0.05, 2.37, 4.10, 120.00,
	HMFPSO	1.674728E+02	9.854784E+01	8.02E+01	0.00	8.412698E+00	4	0.05, 2.04, 4.08, 120.00,
	MFPSO	8.412698E+00	7.203671E+01	7.93E+01	60.00	8.412698E+00	4	0.05, 2.04, 4.08, 120.00,
F10	MFO	6.842958E+00	6.842958E+00	4.52E-15	100.00	6.842958E+00	4	57.69, 34.15, 57.69, 1.05,
	PSO	6.854685E+00	6.872738E+00	3.92E-02	0.00	6.842958E+00	4	57.69, 34.15, 57.69, 1.05,
	HyMFPSO	6.842958E+00	6.935276E+00	3.20E-01	83.33	6.842958E+00	4	0.00, 0.09, 0.00, 2.62,
	HMFPSO	6.842958E+00	6.843075E+00	4.54E-04	80.00	6.842958E+00	4	0.00, 0.02, 0.00, 3.55,
	MFPSO	6.842958E+00	6.842958E+00	4.52E-15	100.00	6.842958E+00	4	57.69, 34.15, 57.69, 1.05,
F11	MFO	2.318455E+01	2.305257E+01	1.73E-01	0.00	2.284452E+01	11	0.50, 1.11, 0.50, 1.31, 0.50, 1.50, 0.50, 1.00, 1.00, -20.48, -0.00,
	PSO	2.624096E+01	2.601869E+01	1.21E+00	0.00	2.379961E+01	11	0.68, 1.25, 0.50, 1.10, 0.50, 0.75, 0.50, 0.60, 0.71, 18.50, 18.79,
	HyMFPSO	2.295459E+01	2.309015E+01	2.77E-01	0.00	2.288207E+01	11	0.50, 1.12, 0.50, 1.31, 0.50, 1.50, 0.50, 1.00, 1.00, -20.03, 0.23,
	HMFPSO	2.319626E+01	2.320861E+01	1.72E-02	0.00	2.319626E+01	11	0.50, 1.06, 0.50, 0.50, 0.50, 1.50, 0.50, 1.00, 1.00, -30.00, -0.00,
	MFPSO	2.284363E+01	2.294585E+01	1.59E-01	0.00	2.284308E+01	11	0.50, 1.11, 0.50, 1.30, 0.50, 1.50, 0.50, 0.79, 0.62, -19.82, -0.00,
F12	MFO	1.724960E+00	1.766609E+00	7.71E-02	3.33	1.724852E+00	4	0.21, 3.47, 9.04, 0.21,
	PSO	2.349117E+00	2.450153E+00	5.24E-01	0.00	1.881635E+00	4	0.16, 4.98, 8.81, 0.22,
	HyMFPSO	1.916610E+00	1.949934E+00	1.22E-01	0.00	1.869523E+00	4	0.17, 4.06, 9.65, 0.21,
	HMFPSO	2.203959E+00	2.164988E+00	1.30E-01	0.00	1.728616E+00	4	0.21, 3.46, 9.03, 0.21,
	MFPSO	1.724852E+00	1.724852E+00	6.78E-16	100.00	1.724852E+00	4	0.21, 3.47, 9.04, 0.21,
F13	MFO	3.592080E+02	3.594108E+02	7.72E-01	93.33	3.592080E+02	3	0.25, 0.47, 8.50,
	PSO	3.622500E+02	3.614238E+02	1.97E+00	40.00	3.592080E+02	3	0.19, 0.47, 8.50,
	HyMFPSO	3.592080E+02	3.592085E+02	1.84E-03	83.33	3.592080E+02	3	0.22, 0.53, 8.50,
	HMFPSO	3.592080E+02	3.597150E+02	1.15E+00	83.33	3.592080E+02	3	0.22, 0.47, 8.50,
	MFPSO	3.592080E+02	3.592080E+02	5.78E-14	100.00	3.592080E+02	3	0.24, 0.53, 8.50,

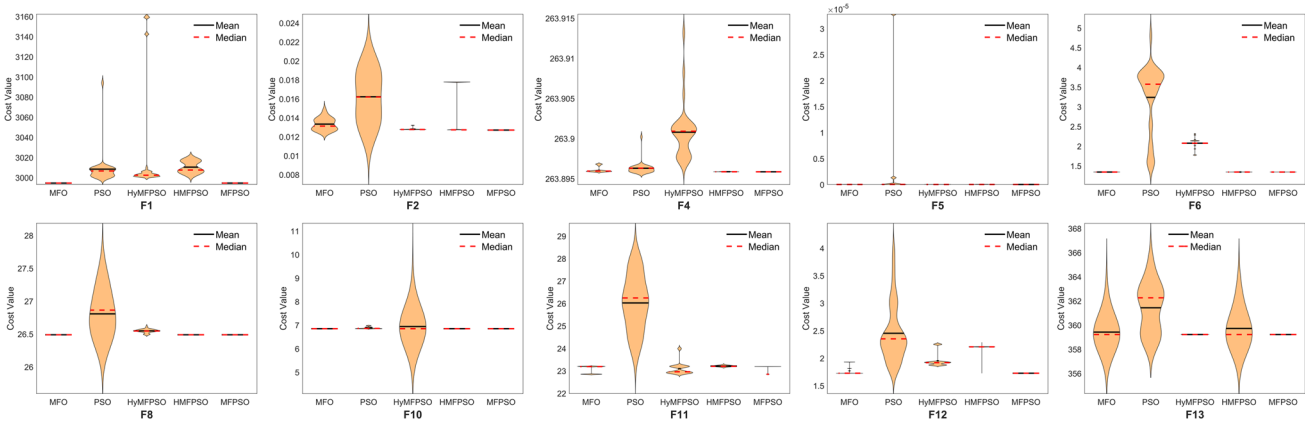


Fig. 8 Violin plots for the Engineering Optimization Problems

curves for a sample of comparisons (F3 and F7), while the complete set of 52 confidence curves is available in the Supplementary Material (Section S5.2).

The median convergence trends across all the engineering optimization problems have been validated using the Page test. As shown in Fig. 10, the convergence trends align closely with those observed in the CEC2020/2021 benchmark functions (7), where MFPSO consistently excels against hybrid variants while maintaining competitive trade-offs with simpler algorithms like PSO. Detailed Page's test

Table 7 Statistical Analysis for Engineering Optimization Problems for all the algorithms, where MFPSO is the reference in all paired comparisons

Metric	Alg.	Friedman Test				Sign Test	Wilcoxon Test			
		SumRank	MeanRank	Rank	p-value	+/=/-	R+	R-	p-value	H
Best	MFO	32.5	2.5000	2	6.75E-05	+6/=7/-0	21	0	0.031250	TRUE
	PSO	52	4.0000	5		+10/=3/-0	55	0	0.001953	TRUE
	HyMFPSO	50.5	3.8846	4		+10/=3/-0	55	0	0.001953	TRUE
	HMFPSO	38	2.9231	3		+8/=5/-0	36	0	0.007813	TRUE
	MFPSO	22	1.6923	1		NA	NA	NA	NA	NA
Worst	MFO	34	2.6154	2	6.94E-07	+10/=3/-0	55	0	0.001953	TRUE
	PSO	62	4.7692	5		+13/=0/-0	91	0	0.000244	TRUE
	HyMFPSO	42	3.2308	4		+12/=0/-1	80	11	0.013428	TRUE
	HMFPSO	40.5	3.1154	3		+11/=2/-0	66	0	0.000977	TRUE
	MFPSO	16.5	1.2692	1		NA	NA	NA	NA	NA
Median	MFO	33	2.5385	2	2.82E-07	+8/=5/-0	36	0	0.007813	TRUE
	PSO	63	4.8462	5		+13/=0/-0	91	0	0.000244	TRUE
	HyMFPSO	41	3.1538	4		+11/=2/-0	66	0	0.000977	TRUE
	HMFPSO	39.5	3.0385	3		+9/=4/-0	45	0	0.003906	TRUE
	MFPSO	18.5	1.4231	1		NA	NA	NA	NA	NA
Mean	MFO	32	2.4615	2	8.24E-07	+10/=3/-0	55	0	0.001953	TRUE
	PSO	61	4.6923	5		+13/=0/-0	91	0	0.000244	TRUE
	HyMFPSO	42	3.2308	3		+12/=0/-1	79	12	0.017090	TRUE
	HMFPSO	43.5	3.3462	4		+11/=2/-0	66	0	0.000977	TRUE
	MFPSO	16.5	1.2692	1		NA	NA	NA	NA	NA
SD	MFO	34	2.6154	2	2.75E-06	+10/=3/-0	55	0	0.001953	TRUE
	PSO	61	4.6923	5		+13/=0/-0	91	0	0.000244	TRUE
	HyMFPSO	43	3.3077	4		+12/=0/-1	79	12	0.017090	TRUE
	HMFPSO	39.5	3.0385	3		+10/=2/-1	59	7	0.018555	TRUE
	MFPSO	17.5	1.3462	1		NA	NA	NA	NA	NA

Table 8 The confidence interval (CI) of the mean results difference for all the engineering optimization problems with MFPSO as a reference in the unpaired comparisons

Fn.	MFO			PSO			HyMFPSO			HMFPSO		
	LB	UB	pCI	LB	UB	pCI	LB	UB	pCI	LB	UB	pCI
F1	0.00E+00	0.00E+00	FALSE	7.50E+00	1.45E+01	TRUE	7.83E+00	1.13E+01	TRUE	1.30E+01	2.23E+01	TRUE
F2	5.33E-05	1.01E-03	TRUE	2.02E-03	5.19E-03	TRUE	5.34E-05	7.38E-05	TRUE	2.55E-05	5.06E-03	TRUE
F3	9.87E+00	3.07E+01	TRUE	6.85E+02	3.41E+03	TRUE	3.24E+02	4.02E+02	TRUE	9.87E+00	9.87E+00	TRUE
F4	8.81E-05	8.88E-05	TRUE	1.46E-05	4.52E-04	TRUE	3.83E-03	5.06E-03	TRUE	2.96E-06	1.58E-05	TRUE
F5	8.86E-10	1.36E-09	TRUE	1.26E-09	7.13E-08	TRUE	1.76E-11	1.14E-10	TRUE	9.69E-10	1.34E-09	TRUE
F6	2.12E-04	3.83E-04	TRUE	2.05E+00	2.46E+00	TRUE	7.32E-01	7.35E-01	TRUE	1.98E-04	4.61E-04	TRUE
F7	0.00E+00	1.92E-06	FALSE	1.08E-03	3.19E-03	TRUE	3.89E-08	1.78E-07	TRUE	0.00E+00	0.00E+00	FALSE
F8	0.00E+00	0.00E+00	FALSE	0.00E+00	3.76E-01	FALSE	5.70E-02	6.78E-02	TRUE	0.00E+00	0.00E+00	FALSE
F9	0.00E+00	3.69E-13	FALSE	2.56E+02	3.84E+02	TRUE	-1.51E+02	2.49E+00	FALSE	5.51E-05	5.00E-03	TRUE
F10	0.00E+00	0.00E+00	FALSE	4.15E-03	3.22E-02	TRUE	0.00E+00	0.00E+00	FALSE	0.00E+00	0.00E+00	FALSE
F11	1.43E-03	3.41E-01	TRUE	2.76E+00	3.40E+00	TRUE	4.80E-02	1.11E-01	TRUE	3.53E-01	3.53E-01	TRUE
F12	3.58E-07	4.95E-04	TRUE	5.55E-01	6.73E-01	TRUE	1.81E-01	2.04E-01	TRUE	4.79E-01	4.79E-01	TRUE
F13	0.00E+00	0.00E+00	FALSE	0.00E+00	3.43E+00	FALSE	0.00E+00	0.00E+00	FALSE	0.00E+00	0.00E+00	FALSE

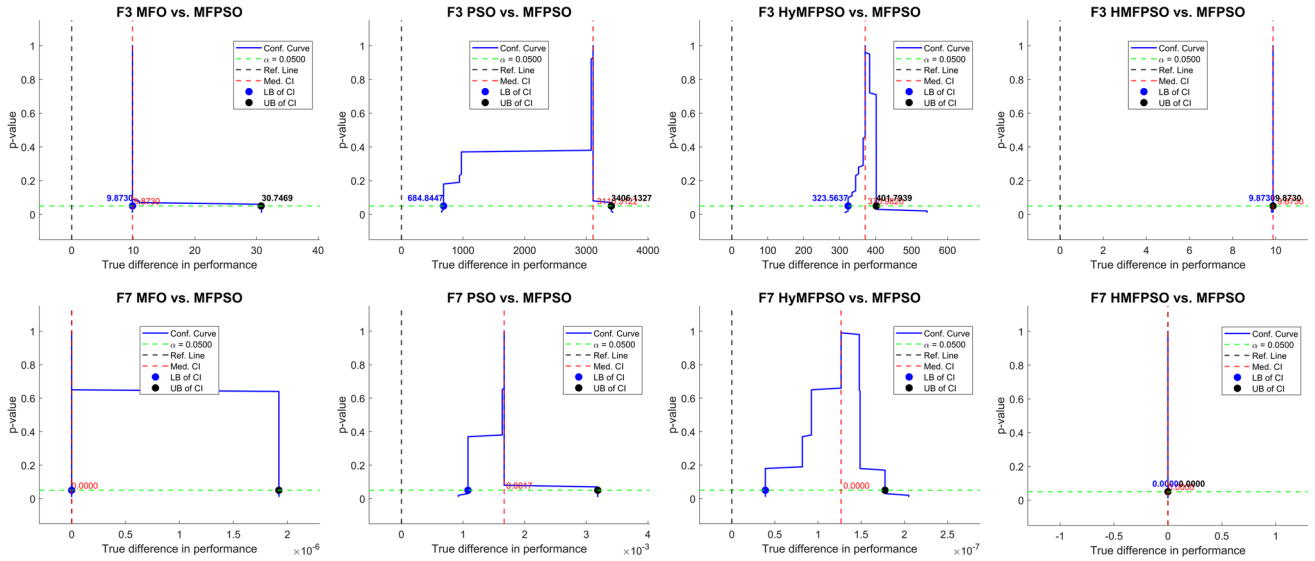


Fig. 9 Confidence curves for the engineering optimization problems (F3 and F7) across the 30 runs, where MFPSO is compared with all algorithms

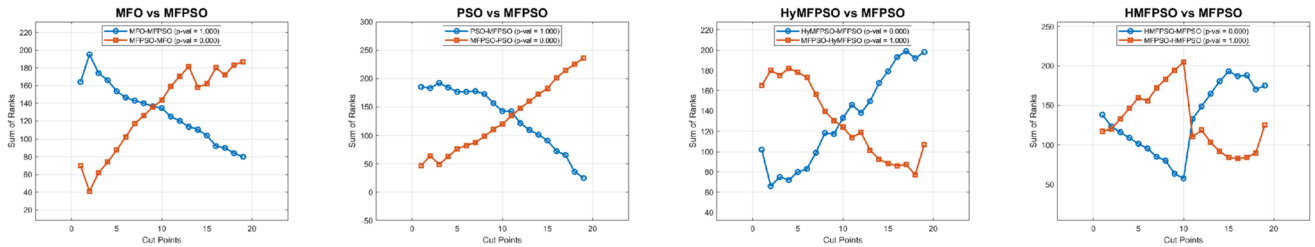


Fig. 10 Convergence trends between the algorithms using the Page's test

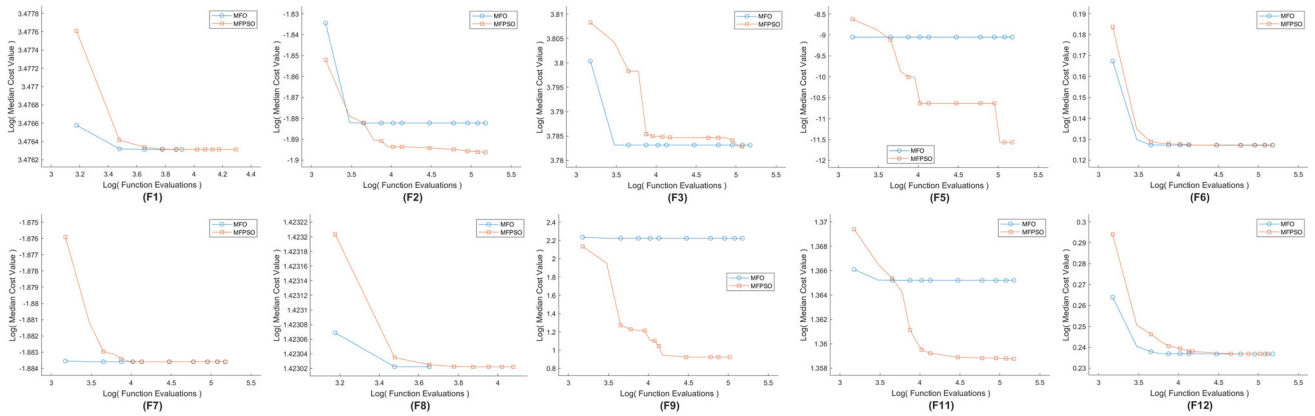


Fig. 11 Convergence graphs for the proposed MFPSO and MFO algorithms for the median error across 30 runs for Engineering Optimization Problems

table is available in the Supplementary Material (Section S5.3).

Figure 11 highlights the convergence comparison between the top two algorithms, MFO and MFPSO. The MFPSO algorithm demonstrates faster convergence during the initial iterations, while both algorithms achieve similar performance toward the end, with no significant progress observed in later stages. These results underscore MFPSO's capability to reach global optima across diverse and complex problem landscapes consistently. It solidifies its role as a robust and efficient solution for engineering optimization tasks while outperforming state-of-the-art alternatives.

7 Hardware Design and Assembly of the 4WD Car

7.1 Overall View of the Modified Elegoo Car

This section presents the overall hardware block diagram of the proposed 4WD ADS, as shown in Fig. 12. The system includes three main hardware categories: sensors (four rotary encoders), controllers (Arduino Mega and Raspberry Pi), and actuators (four DC motors with motor drivers). The Raspberry Pi is the primary controller, running ROS to implement the proposed real-time PID tuning algorithm. It receives speed data from the Arduino and sends PID parameters to maintain the desired motor speed [54]. The Arduino acts as a secondary controller, driving the motors and providing electric isolation. It applies PID control, receives encoder feedback, and sends speed data to the Raspberry Pi, ensuring a closed-loop system.

7.2 Modification 1: Adding Four Rotary Encoder

The car's original design lacked encoders, preventing speed monitoring and position estimation. Rotary encoders are added to each motor to measure current speed and enable closed-loop control. Each encoder comprises a 20-slot disc, an LED light source, and a photodetector, which generates pulses as the disc rotates [55, 56]. After mounting, friction was observed between the rotary disc and the sensor body due to the motor's proximity to the base. This problem was resolved by lifting the motor slightly using a 3mm nut on the

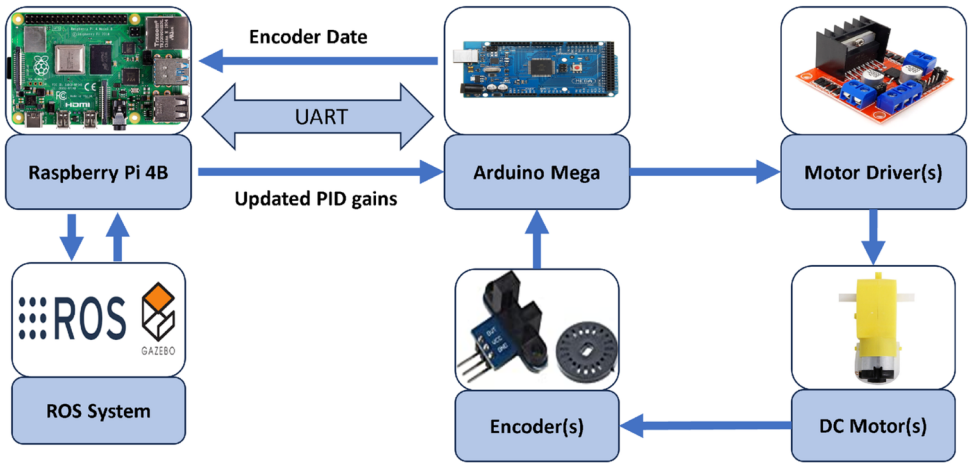


Fig. 12 The block diagram of the overall hardware view of the proposed system

(a) Lifting the motor by 3mm nut. (b) The encoders alignment top view.

Fig. 13 Mounting and aligning the four encoders

vertical screw, allowing the disc to rotate freely (Fig. 13a). Figure 13b shows the final encoder setup secured with 3M double-sided adhesive tape.

7.3 Modification 2: Adding the Four Motor Drivers

The Elegoo car initially used two TB6612 motor drivers, each controlling two motors, limiting the car to a two-wheel drive setup and causing kinematic conflicts [32]. This limitation was resolved by replacing the two TB6612 drivers with four L298N drivers, one for each motor (14). The L298N drivers, with a higher voltage range of 4.5V to 46V, were better suited to handle the increased power demands of heavier components like the Raspberry Pi and additional batteries [57, 58] (Fig. 14).

7.4 Modification 3: Changing Arduino UNO to MEGA 2560

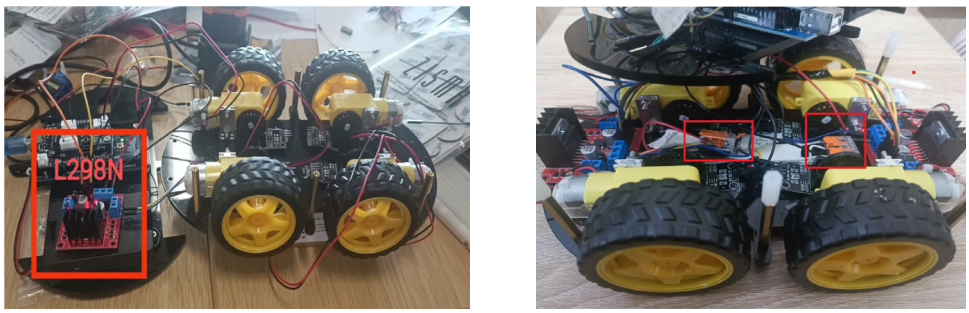
The Arduino UNO included with the Elegoo car lacks sufficient IO pins for the additional hardware: four encoders (requiring four interrupt pins) and four L298N drivers (needing four PWM and eight digital pins). With only 14 digital pins (six PWM-capable) and two interrupt pins, the UNO is unsuitable for the proposed system [59]. It is replaced by the Arduino Mega 2560, which provides 54 digital pins (15 PWM-capable) and six interrupt pins, meeting all hardware requirements [60].

7.5 Modification 4: Adding Raspberry Pi on a New Third Layer

A custom third layer is designed using Nano CAD software and laser-cut from an acrylic sheet to accommodate the Raspberry Pi and its batteries. The layer is mounted atop the second layer using 3mm threaded bars and male-female spacers as foundation piles (Fig. 15a). The Raspberry Pi 4B is secured on the new layer and powered by an X728 V2.3 smart UPS with two rechargeable 18650 lithium-ion batteries, ensuring stable voltage and surge protection [61]. Additionally, an ultra-thin ice tower with a PWM-controlled fan is installed to maintain the Raspberry Pi's temperature within a safe range (Fig. 15b).

8 Troubleshooting Speed and PWM Synchronization

This section discusses the challenges encountered during the implementation process and the steps taken to debug, test, and resolve them. The 4WD architecture demands precise synchronization of all four motors, as even minor speed



(a) Testing the L298N motor driver. (b) Wires after using Terminal Block.

Fig. 14 Mounting motor drivers

Fig. 15 The third layer assembly for the Raspberry Pi

(a) Mounting the third layer. (b) Raspberry Pi before and after the cooling fan.

discrepancies can lead to trajectory deviations. The following subsections detail four experiments conducted to identify and address these issues.

8.1 Experiment 1: Arduino PWM Output Synchronization

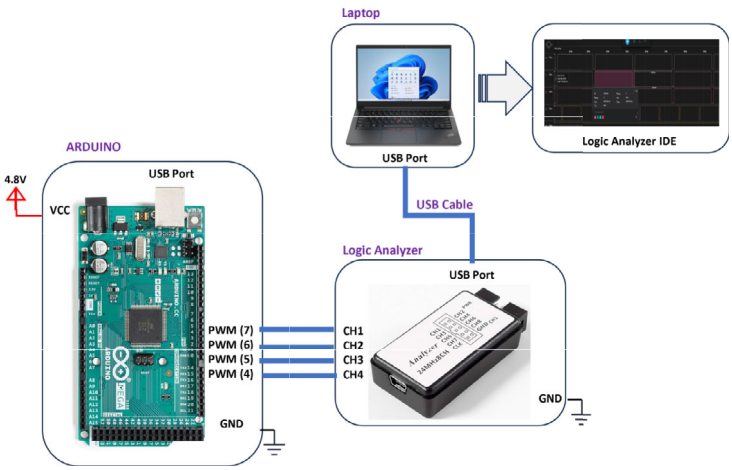
Motor speed is controlled via PWM, where the duty cycle (0-100%) regulates the average voltage delivered to the motors. Synchronization across all four PWM signals is essential, as frequency mismatches can cause speed inconsistencies, mechanical resonance, and instability [62, 63]. To verify synchronization, the PWM frequencies of Arduino pins (7, 6, 5, 4) were monitored using a logic analyzer and a Tektronix oscilloscope (Fig. 16). Pins 7, 6, and 5 produced consistent frequencies (490.196 Hz), while pin 4 operated at 980.392 Hz (Fig. 17), revealing a synchronization issue.

The discrepancy was traced to Timer0, which controls pin 4 and defaults to Fast PWM mode at 976.5625 Hz. Synchronization was achieved by adjusting Timer0's prescaler and TOP values using Eq. 11. Setting the prescaler to 256 and TOP to 127 resulted in a PWM frequency of 490.169 Hz. The modifications involved updating the CR0A register to 127 (TOP value) and configuring the TCCR0B bits (CS02, CS01, CS00) to (1, 0, 0). After these adjustments, all PWM signals synchronized at 490.169 Hz with consistent duty cycles (Fig. 18).

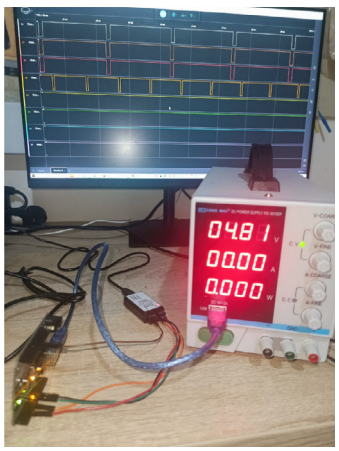
$$f_{\text{PWM}} = \frac{f_{\text{CPU}}}{\text{Prescaler} \times (1 + \text{TOP})} \tag{11}$$

8.2 Experiment 2: Motor Driver PWM Output Test

This experiment validates that the PWM signal output from the motor driver matches the duty cycle and frequency of the signal received from the Arduino. As shown in Fig. 19, the Arduino is connected to a motor driver powered by a 7.4V input. The motor driver's output is monitored using a signal analyzer (CH1), with results visualized on a laptop. The PWM signal at Arduino Pin 7 is set to a 50% duty cycle and 490.196 Hz frequency for the test. Figure 20 confirms that the motor driver's output signal matches these specifications, verifying the proper functionality of the L298N H-bridge IC. Any deviation would indicate a fault in the motor driver.

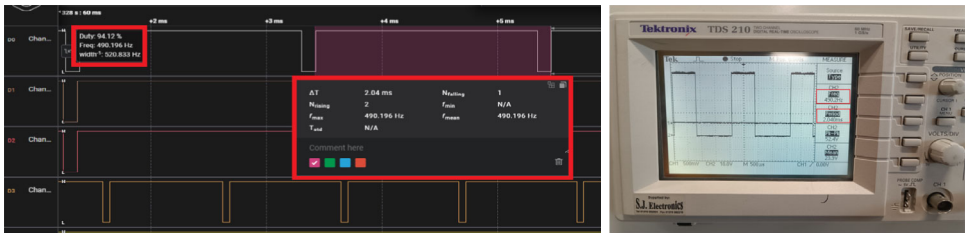


(a) Schematic diagram.

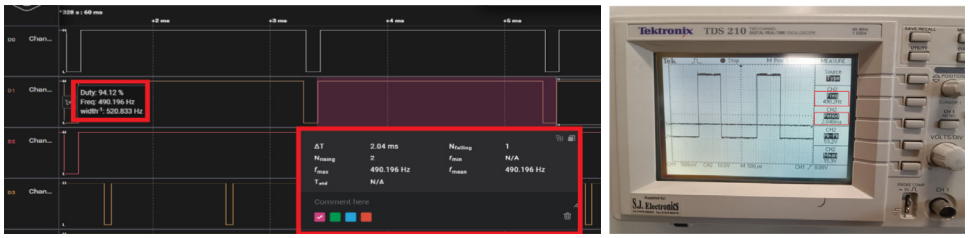


(b) Hardware implementation.

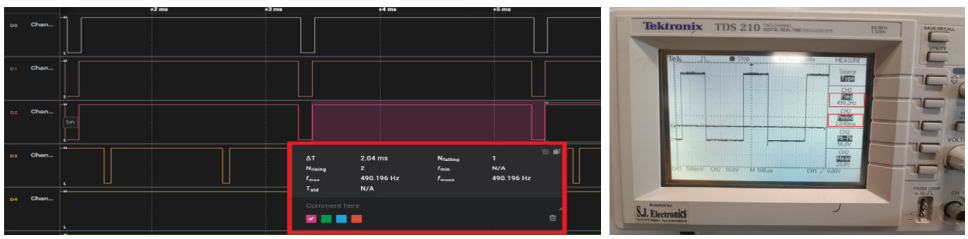
Fig. 16 The PWM Synchronization test circuit setup



(a) Channel 1 for PWM Pin Number 7.



(b) Channel 2 for the PWM Pin Number 6.



(c) Channel 3 for PWM Pin Number 5.

(d) Channel 4 for the PWM Pin Number 4.

Fig. 17 The results of the PWM Synchronization test

8.3 Experiment 3: Threshold Test of the Input Voltage of the Motor Driver

This experiment evaluates the minimum input voltage required for the L298N motor drivers to operate effectively. The L298N driver, powered by a 7.4V lithium battery, modulates the motor voltage via PWM signals applied to its ENA/ENB pins [58]. While the motor operates optimally between 6-8V and draws a maximum current of 200mA, voltage drops during battery discharge may impact performance. The test involved varying the input voltage from 1.91V to 7.7V while monitoring the PWM output signal (Fig. 19), with results displayed in Fig. 21.

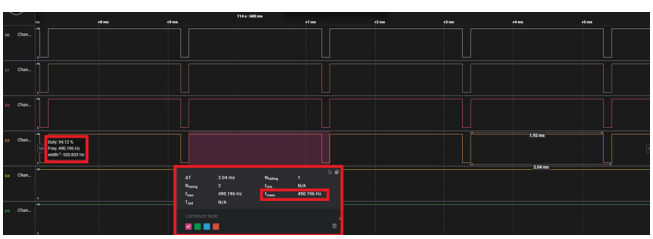
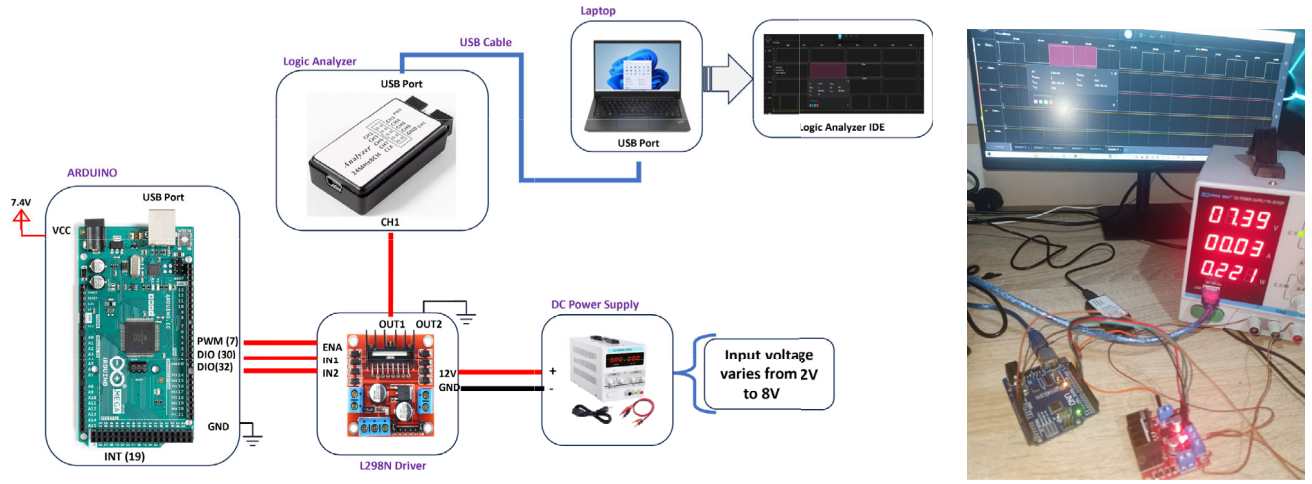


Fig. 18 The PWM signal for pin 4 after the modifications



(a) Schematic diagram. (b) Hardware setup.

Fig. 19 Schematic diagram of motor drivers signal test setup

At input voltages below 3.58V, no signal was produced, indicating a "dead zone" (Fig. 21a and b). Between 3.58V and 4.57V, the output signal displayed a near 100% duty cycle, bypassing PWM control and applying maximum voltage to the motor (Fig. 21c and d). From 4.57V to 5.68V, the output exhibited instability, with oscillations and irregular pulses caused by brownout conditions (Fig. 21e to f) [64]. Stable operation with accurate PWM output was achieved only above 5.7V, as shown in Fig. 21g and h. Therefore, maintaining an input voltage of at least 5.7V is essential for reliable motor driver performance, ensuring stable and predictable motor operation.

8.4 Experiment 4: Encoder Disc Alignment

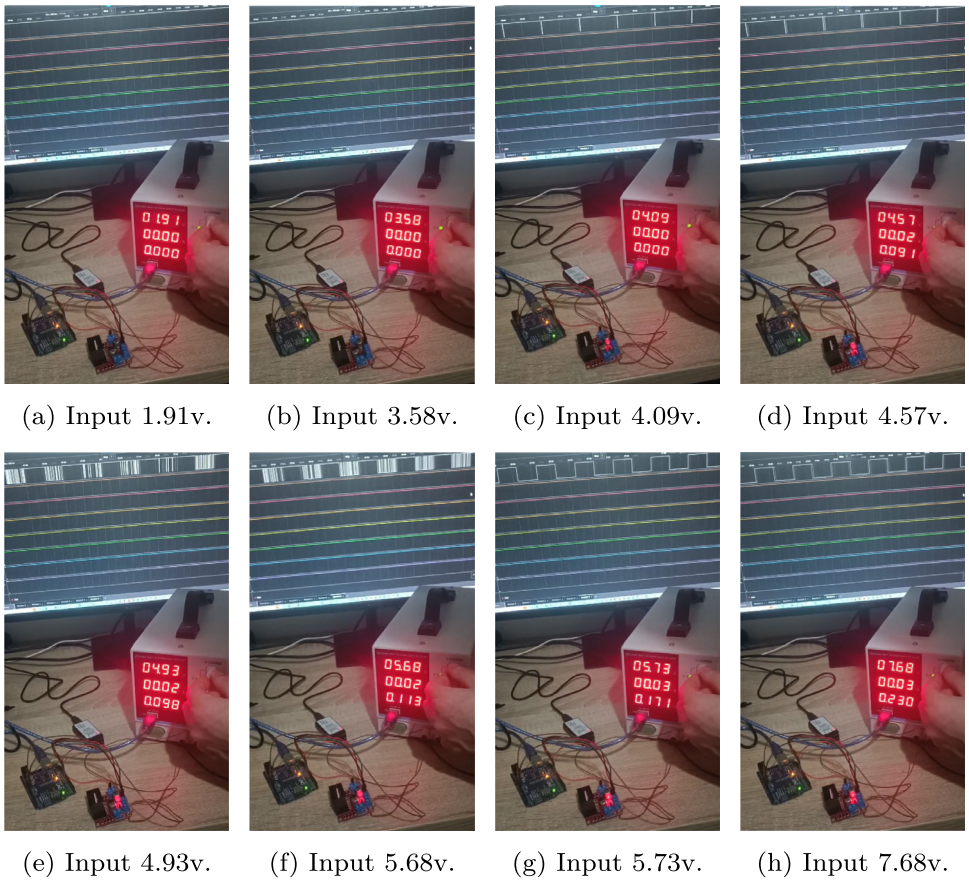
Proper alignment of the encoder disc ensures that all 20 slots on the rotary disc pass the light beam between the sender and receiver, generating 20 pulses per revolution. The test setup (Fig. 22) involves sending a PWM signal (30% duty cycle) from the Arduino (Pin 7) to the motor driver, with the encoder output connected to the Arduino's interrupt pin (INT19). Encoder readings are displayed on the Arduino IDE's serial monitor via UART communication.

A reference mark is placed on the wheel and car body as the starting position. The motor rotates clockwise until the encoder registers 20 pulses. If the reference marks realign after one revolution, the encoder is correctly aligned, while misalignment indicates the need for adjustment. This process is repeated for all four wheels to ensure accurate encoder readings.

9 PID Tuning Application for Speed Control of DC Motor

This section details the application of MFPSO for tuning the PID controller, which regulates the speed of the 4WD car's DC motors. Accurate speed control is essential in this system,

Fig. 20 The output PWM signal of a motor driver under 50% duty cycle



(a) Input 1.91v. (b) Input 3.58v. (c) Input 4.09v. (d) Input 4.57v.

(e) Input 4.93v. (f) Input 5.68v. (g) Input 5.73v. (h) Input 7.68v.

Fig. 21 The results of threshold input voltage test of the motor driver

as steering is achieved by adjusting wheel speeds rather than using a traditional steering mechanism.

9.1 PID Controller Structure

The PID controller minimizes the difference between the desired and actual motor speeds. The closed-loop system, shown in Fig. 23, consists of the motor as the plant, the motor driver as the actuator, and the encoder providing feedback. The encoder measures the actual motor speed, which is compared to the desired speed to generate an error signal, e(t), as described in Eq. 12. The PID controller processes this error and generates a control signal, u(t), using proportional, integral, and derivative terms, as defined in Eq. 13. These terms work together to ensure accurate and stable speed control by minimizing errors and improving response time. The controller outputs the PWM signal to the motor driver, which adjusts the motor speed to match the desired value.

$$e(t) = \text{Desired Speed} - \text{Actual Speed} \tag{12}$$

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \tag{13}$$

9.2 Interactive PID Tuning Using the Proposed MFPSO Algorithm

The MFPSO is implemented for real-time PID tuning in the 4WD vehicle's speed control system. During the tuning process, the MFPSO algorithm interacts dynamically with the motors, using live feedback from the Arduino to continuously refine the PID parameters (K_p, K_i, and K_d). This real-time approach ensures adaptive adjustments to account for any discrepancies in motor performance.

In the MFPSO algorithm, the position of each moth represents a candidate solution for the PID parameters. The algorithm evaluates these solutions using a fitness function, defined as the sum of squared errors (SSE) over a given time period T (Eq. 14). This fitness function quantifies the deviation between the desired and actual motor speeds, penalizing larger errors more heavily to prioritize accurate speed control.

$$\text{SSE} = \sum_{t=1}^{T} e(t)^2 \tag{14}$$

Figure 24 illustrates the system setup. The MFPSO runs as a ROS node on the Raspberry Pi, which receives encoder-

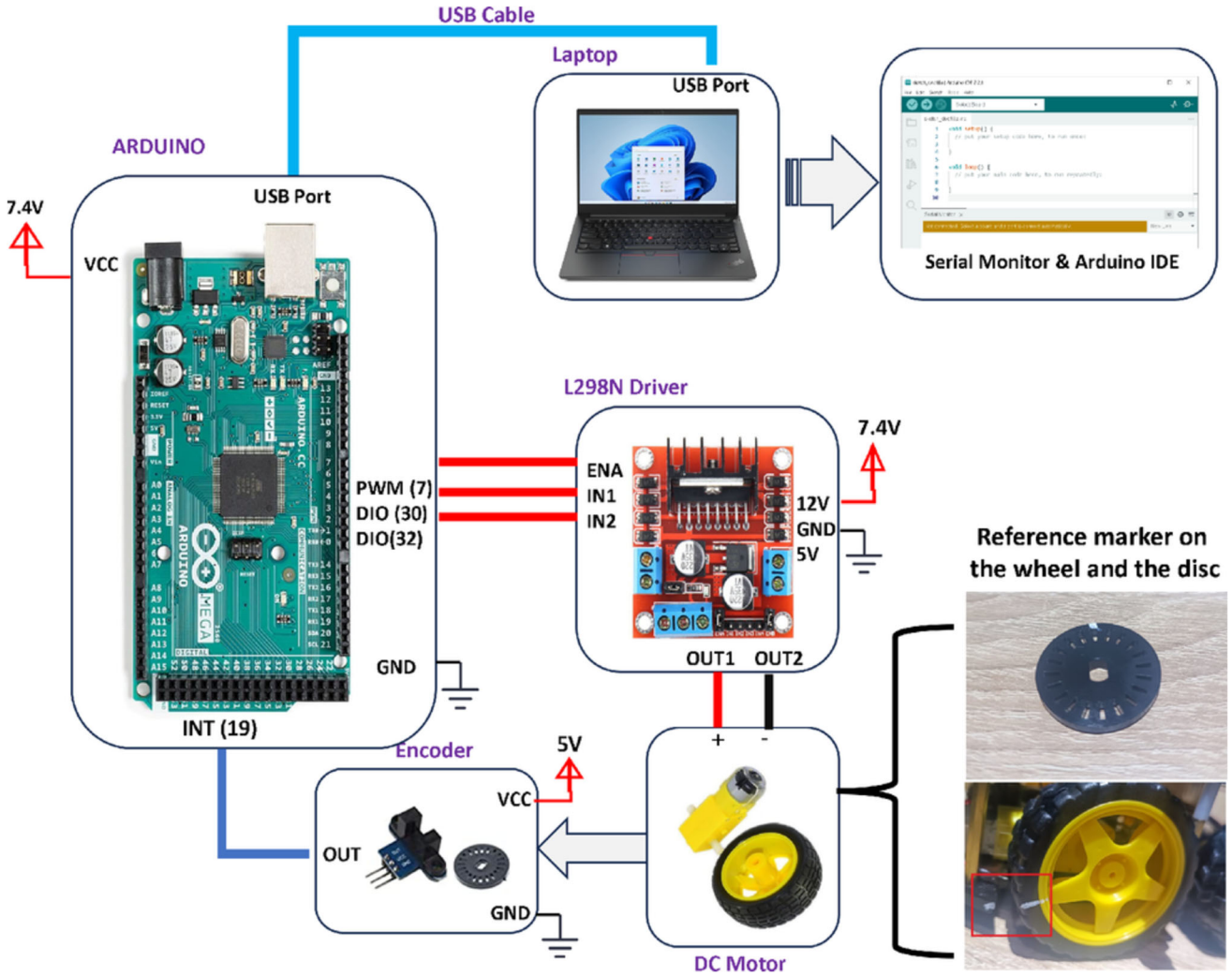


Fig. 22 Encoder Alignment test circuit

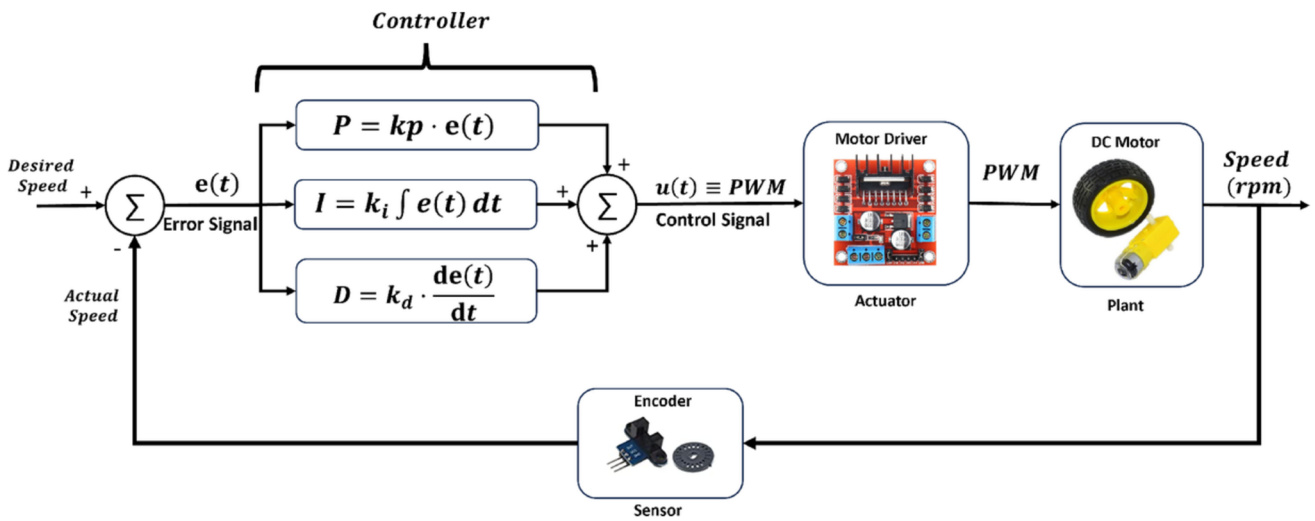


Fig. 23 Block diagram of the PID speed control for 4WD car

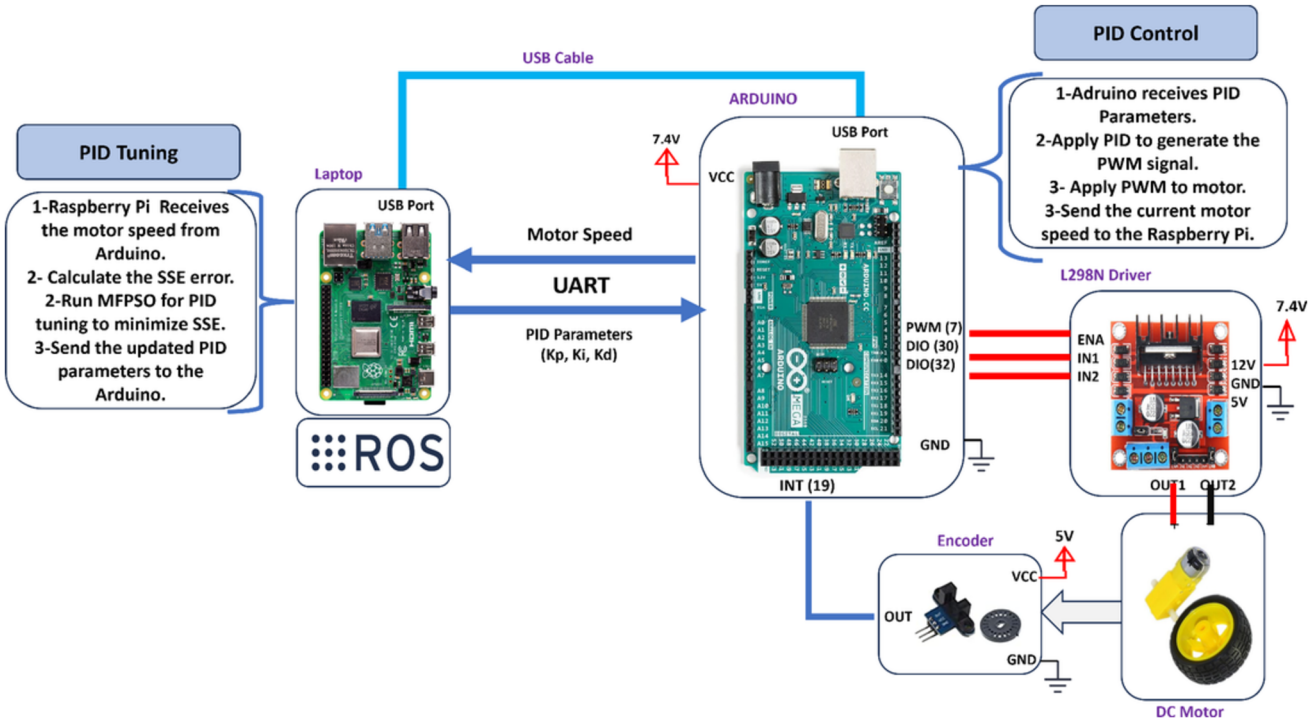


Fig. 24 Schematic diagram of PID tuning using the proposed MFPSO algorithm

based speed data from the Arduino. The SSE is calculated using Eq. 14 to update the fitness value and guide the algorithm's iterations, ultimately generating optimal PID gains that minimize the error. These gains are sent back to the Arduino, which updates the PWM duty cycle controlling the DC motor. The encoder readings are then transmitted to the Raspberry Pi, completing the feedback loop. This process continues until stable PID parameters are achieved. The final optimized PID gains obtained using the MFPSO algorithm are $K_p = 2.6897$, $K_i = 1.7789$, and $K_d = 0.3910$, ensuring precise speed control.

9.3 Testing Four Motors of the 4WD Using MFPSO-PID

The PID controller, tuned using the MFPSO algorithm, is tested on all four motors of the 4WD vehicle to evaluate synchronization and transient responses across various duty cycles. During these tests, all motors are set to the same PWM duty cycle, ranging from 40% to 100% in 5% increments, to assess performance during straight-line movement. Key performance metrics-rise time (t_r), peak time (t_p), maximum overshoot (M_p), settling time (t_s), and steady-state error (ess)-are collected for each duty cycle [65].

The primary metrics, M_p and ess, are critical for ensuring accuracy and safety. The steady-state error (ess) is particularly important to maintain accurate positioning, crucial for path planning, while minimizing M_p is essential to avoid excessive speeds or overshooting, which could disrupt navigation or compromise safety. Although response speed metrics like t_r and t_s are considered, maintaining accuracy and safety are prioritized.

The experiment was conducted over approximately two hours and 13 minutes (8034 seconds) to ensure reliable long-term performance while monitoring transient and steady-state responses. Throughout this period, the ess remained consistently below 10^{-8} for all motors, effectively treated as zero, underscoring the system's precision and stability. Table 9 summarizes the results for t_r, t_p, t_s, and M_p across various duty cycles, while Fig. 25a highlights the uniformity in steady-state performance.

At a 100% duty cycle, Fig. 25b illustrates the transient responses of the four motors, revealing negligible differences in performance metrics. The low standard deviation (SD) across all metrics, consistently below 0.05, further demonstrates high synchronization and uniform performance among the motors, validating the MFPSO-PID controller's effectiveness in maintaining consistent and synchronized operation.

10 Transient Response Benchmark Testing for all the PID Controllers

The section compares the transient response of the MFPSO controller with other controllers to evaluate its performance

Table 9 Performance metrics of the four DC motors in the 4WD using MFPSO

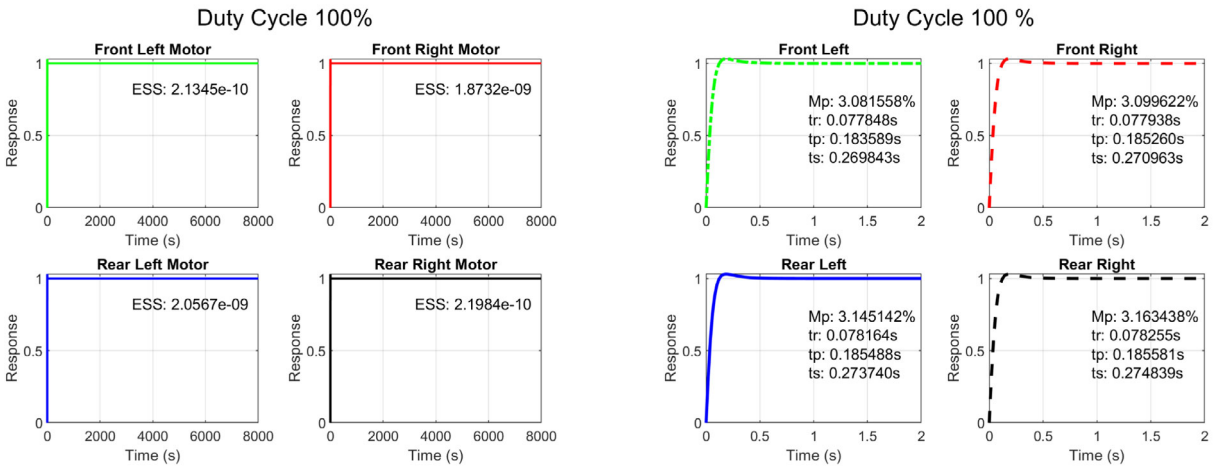
SP	Wheel	RiseTime	PeakTime	SettlingTime	M_p (%)
40	RL	0.179231	0.414442	0.482988	2.285791
	FR	0.179305	0.415099	0.485327	2.305023
	RL	0.179487	0.410707	0.490883	2.354356
	RR	0.179559	0.411358	0.492999	2.374227
	Mean	0.179395	0.412902	0.488049	2.329849
	SD	0.000153	0.002191	0.004675	0.041341
45	RL	0.160903	0.375486	0.457542	2.453315
	FR	0.160989	0.370608	0.459405	2.473344
	RL	0.161202	0.372222	0.463892	2.524726
	RR	0.161286	0.372869	0.465633	2.545182
	Mean	0.161095	0.372796	0.461618	2.499142
	SD	0.000179	0.002030	0.003777	0.042973
50	RL	0.146073	0.341456	0.432245	2.583205
	FR	0.146171	0.342108	0.433881	2.603732
	RL	0.146415	0.338609	0.437853	2.654932
	RR	0.146512	0.339253	0.439401	2.676118
	Mean	0.146293	0.340356	0.435845	2.629497
	SD	0.000205	0.001688	0.003341	0.043310
55	RL	0.133909	0.316104	0.408732	2.683747
	FR	0.134007	0.316766	0.410232	2.704251
	RL	0.134249	0.313670	0.413893	2.757256
	RR	0.134345	0.314324	0.415324	2.778410
	Mean	0.134128	0.315216	0.412045	2.730916
	SD	0.000203	0.001458	0.003079	0.044292
60	RL	0.123681	0.293270	0.387255	2.763480
	FR	0.123779	0.293948	0.388664	2.784055
	RL	0.124020	0.291229	0.392114	2.837210
	RR	0.124116	0.291897	0.393465	2.858425
	Mean	0.123899	0.292586	0.390374	2.810793
	SD	0.000203	0.001243	0.002900	0.044423
65	RL	0.114978	0.274198	0.367710	2.827112
	FR	0.115071	0.272309	0.369054	2.848654
	RL	0.115315	0.270770	0.372353	2.900483
	RR	0.115411	0.271459	0.373647	2.921878
	Mean	0.115194	0.272184	0.370691	2.874532
	SD	0.000203	0.001483	0.002773	0.044096
70	RL	0.107449	0.255629	0.349906	2.880085
	FR	0.107544	0.255142	0.351202	2.900833
	RL	0.107792	0.256413	0.354384	2.952669
	RR	0.107894	0.255940	0.355636	2.973707
	Mean	0.107670	0.255781	0.352782	2.926824
	SD	0.000208	0.000534	0.002676	0.043689
75	RL	0.100925	0.240329	0.333648	2.923950
	FR	0.101020	0.239736	0.334904	2.944378
	RL	0.101258	0.240827	0.337993	2.995729
	RR	0.101362	0.240259	0.339209	3.016360
	Mean	0.101141	0.240288	0.336438	2.970104
	SD	0.000203	0.000447	0.002597	0.043160
80	RL	0.095193	0.225914	0.318746	2.961607
	FR	0.095284	0.227787	0.319970	2.981460
	RL	0.095527	0.225905	0.322982	3.031920
	RR	0.095628	0.227801	0.324170	3.052478
	Mean	0.095408	0.226852	0.321467	3.006866
	SD	0.000204	0.001088	0.002533	0.042434
85	RL	0.090120	0.214089	0.305045	2.995076
	FR	0.090210	0.215655	0.306240	3.014423
	RL	0.090449	0.215549	0.309184	3.064128
	RR	0.090547	0.214475	0.310347	3.083946
	Mean	0.090331	0.214942	0.307704	3.039393
	SD	0.000200	0.000779	0.002476	0.041570
90	RL	0.085580	0.202634	0.292401	3.025417
	FR	0.085673	0.204386	0.293570	3.044576
	RL	0.085903	0.204023	0.296455	3.093004
	RR	0.085994	0.204921	0.297595	3.112290
	Mean	0.085788	0.203991	0.295005	3.068822
	SD	0.000193	0.000977	0.002425	0.040604
95	RL	0.081518	0.192925	0.280701	3.054134
	FR	0.081609	0.193026	0.281845	3.072727
	RL	0.081837	0.194894	0.284675	3.119562
	RR	0.081929	0.195001	0.285794	3.138473
	Mean	0.081723	0.193962	0.283254	3.096224
	SD	0.000192	0.001140	0.002379	0.039384
100	RL	0.077848	0.183589	0.269843	3.081558
	FR	0.077938	0.185260	0.270963	3.099622
	RL	0.078164	0.185488	0.273740	3.145142
	RR	0.078255	0.185581	0.274839	3.163438
	Mean	0.078051	0.184980	0.272346	3.122440
	SD	0.000190	0.000937	0.002333	0.038246

FL, FR, RL, and RR are the front left, front right, rear left, and rear right motors. The SD tolerance level is 0.05. If SD $\leq$ 0.05, this means there is no significant difference, and the synchronization is achieved. SP is the setpoint (duty cycle)

significance. The comparison includes the traditional Ziegler-Nichols (ZN) method, PSO, MFO, hybrid HMFPSO, and hybrid HyMFPSO algorithms, where each algorithm's PID parameters are computed.

10.1 PID Parameter Setting

In the Ziegler-Nichols (ZN) method, the initial step involves setting the gains $K_i = 0$ and $K_d = 0$, and then gradually



(a) Steady-state response. (b) Transient response.

Fig. 25 Transient and steady-state responses of four DC motors in the 4WD using MFPSO algorithm

increasing K_p until the output speed begins to oscillate [66]. The gain value at this point is referred to as the ultimate gain, K_u, which in this case equals 2.9, while the period of oscillation, P_u, is 1.4. These two values are then used to calculate the PID parameters using Eq. 15 [67], resulting in $K_p = 1.75$, $K_i = 2.5$, and $K_d = 0.3063$. The PID parameters for the MFO, PSO, HMFPSO, HyMFPSO, and MFPSO algorithms are calculated as described in Section 9.2. Table 10 summarizes the PID gains for all the algorithms.

$$K_p = 0.6 \times K_u, \quad K_i = \frac{2 \times K_p}{P_u}, \quad K_d = \frac{K_p \times P_u}{8} \tag{15}$$

10.2 Transient Response Results

Different setpoints corresponding to duty cycles ranging from 30% to 100% in 5% increments are applied to evaluate the transient response performance. The starting point is set at 30% because the motor driver cannot overcome inertia to rotate the motor below this threshold. Table 11 presents the performance metrics at different speeds for all algorithms, while Fig. 26 illustrates the transient response curves for the minimum and maximum duty cycles (additional figures are provided in the Supplementary Materials (Section S6)). The steady-state error remains zero across all algorithms due to the integral action, which effectively eliminates this error [65].

Table 10 PID parameters for each algorithm

Algorithm name	K_p	K_i	K_d
Ziegler-Nichols	1.7500	2.5000	0.3063
MFO Algorithm	2.1495	2.0172	0.2970
PSO Algorithm	1.9831	1.9593	0.2198
HMFPSO Algorithm	1.9437	2.2411	0.3790
HyMFPSO Algorithm	2.6897	1.9143	0.5180
MFPSO Algorithm	2.6897	1.7789	0.3910

The M_p in all algorithms is directly proportional to the duty cycle; as the motor rotates at higher speeds, it is expected to overshoot more. One of the standout features of the MFPSO is its superior control over the M_p for all duty cycles compared to the other methods. The M_p starts from 1.851% at the lowest duty cycle to 3.160% at 100% duty cycle. The MFPSO improves the overshoot by 86.11%, 64.99%, 71.02%, 74.37%, and 60.58% compared with the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms, respectively. The algorithm's ability to control and minimize the overshoot makes it the best choice in the path planning application.

Regarding t_r, all the controllers' rise time is inversely proportional to the duty cycle; they give a high response time at small duty cycles and a low response time at higher speeds and duty cycles. In all cases, the MFPSO algorithm has a better t_r than HyMFPSO. MFPSO maintains competitive rise times, slightly lagging between 30% and 60% duty cycles compared to ZN, MFO, PSO, and HMFPSO. However, it achieves the lowest rise time at 100%, demonstrating adaptive behavior that balances caution at lower speeds with rapid response at higher speeds.

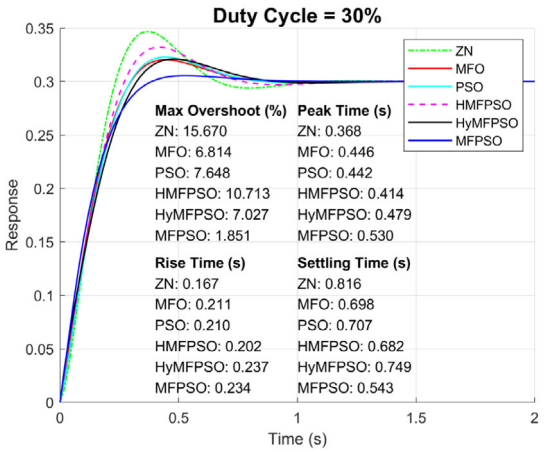
Regarding peak time, all the controllers' peak time is inversely proportional to the duty cycle. The MFPSO gives a peak time higher than all the algorithms, especially at small duty cycles; It is a trade-off for the significant overshoot performance. The peak time of the MFPSO improves starting from 50% duty cycle compared with HyMFPSO. The ZN-tuned controller gives the lowest peak time compared with

Table 11 PID Tuning results and performance metrics

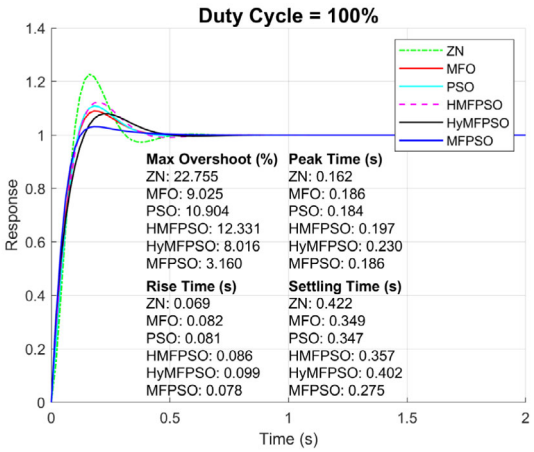
SP	Alg.	RiseTime	PeakTime	SettlingTime	M_p (%)
30%	ZN	0.167	0.368	0.816	15.670
	MFO	0.211	0.446	0.698	6.814
	PSO	0.210	0.442	0.707	7.648
	HMFPSO	0.202	0.414	0.682	10.713
	HyMFPSO	0.237	0.479	0.749	7.027
	MFPSO	0.234	0.530	0.543	1.851
35%	ZN	0.148	0.334	0.783	16.887
	MFO	0.187	0.396	0.639	7.435
	PSO	0.186	0.392	0.646	8.386
	HMFPSO	0.181	0.379	0.625	11.439
	HyMFPSO	0.211	0.432	0.689	7.601
	MFPSO	0.203	0.462	0.515	2.147
40%	ZN	0.134	0.304	0.737	17.853
	MFO	0.168	0.361	0.592	7.889
	PSO	0.167	0.357	0.598	8.949
	HMFPSO	0.164	0.348	0.580	11.944
	HyMFPSO	0.191	0.399	0.642	7.991
	MFPSO	0.180	0.411	0.493	2.371
45%	ZN	0.123	0.280	0.694	18.638
	MFO	0.153	0.330	0.554	8.224
	PSO	0.152	0.333	0.558	9.377
	HMFPSO	0.151	0.321	0.544	12.294
	HyMFPSO	0.176	0.368	0.603	8.252
	MFPSO	0.161	0.373	0.465	2.542
50%	ZN	0.114	0.263	0.655	19.291
	MFO	0.141	0.309	0.521	8.470
	PSO	0.140	0.305	0.524	9.715
	HMFPSO	0.140	0.304	0.513	12.520
	HyMFPSO	0.163	0.345	0.571	8.415
	MFPSO	0.146	0.339	0.439	2.672
55%	ZN	0.106	0.245	0.621	19.845
	MFO	0.131	0.284	0.493	8.651
	PSO	0.130	0.286	0.495	9.978
	HMFPSO	0.131	0.282	0.488	12.674
	HyMFPSO	0.152	0.325	0.543	8.509
	MFPSO	0.134	0.314	0.415	2.775
60%	ZN	0.099	0.232	0.590	20.322
	MFO	0.122	0.267	0.469	8.786
	PSO	0.121	0.268	0.471	10.185
	HMFPSO	0.124	0.269	0.465	12.758
	HyMFPSO	0.143	0.309	0.519	8.549
	MFPSO	0.124	0.292	0.393	2.855
65%	ZN	0.094	0.219	0.562	20.739
	MFO	0.115	0.252	0.448	8.882
	PSO	0.114	0.253	0.449	10.351
	HMFPSO	0.117	0.257	0.446	12.791
	HyMFPSO	0.135	0.294	0.498	8.550
	MFPSO	0.115	0.271	0.373	2.918
70%	ZN	0.089	0.207	0.537	21.108
	MFO	0.108	0.240	0.429	8.951
	PSO	0.107	0.240	0.430	10.484
	HMFPSO	0.111	0.246	0.429	12.786
	HyMFPSO	0.128	0.282	0.480	8.520
	MFPSO	0.108	0.256	0.355	2.970
75%	ZN	0.085	0.198	0.514	21.444
	MFO	0.102	0.228	0.412	8.996
	PSO	0.102	0.229	0.412	10.591
	HMFPSO	0.106	0.236	0.414	12.751
	HyMFPSO	0.122	0.270	0.463	8.468
	MFPSO	0.101	0.240	0.339	3.013
80%	ZN	0.081	0.191	0.492	21.744
	MFO	0.097	0.219	0.397	9.024
	PSO	0.097	0.217	0.397	10.680
	HMFPSO	0.101	0.227	0.400	12.695
	HyMFPSO	0.116	0.260	0.449	8.398
	MFPSO	0.096	0.228	0.324	3.049
85%	ZN	0.077	0.182	0.473	22.026
	MFO	0.093	0.210	0.383	9.038
	PSO	0.092	0.208	0.383	10.751
	HMFPSO	0.097	0.219	0.387	12.621
	HyMFPSO	0.111	0.252	0.435	8.315
	MFPSO	0.091	0.215	0.310	3.081
90%	ZN	0.074	0.175	0.455	22.284
	MFO	0.089	0.200	0.371	9.042
	PSO	0.088	0.200	0.370	10.812
	HMFPSO	0.093	0.211	0.376	12.534
	HyMFPSO	0.107	0.244	0.423	8.222
	MFPSO	0.086	0.204	0.297	3.109
95%	ZN	0.071	0.170	0.438	22.524
	MFO	0.085	0.193	0.360	9.037
	PSO	0.085	0.191	0.358	10.863
	HMFPSO	0.089	0.204	0.366	12.437
	HyMFPSO	0.102	0.236	0.412	8.122
	MFPSO	0.082	0.195	0.286	3.135
100%	ZN	0.069	0.162	0.422	22.755
	MFO	0.082	0.186	0.349	9.025
	PSO	0.081	0.184	0.347	10.904
	HMFPSO	0.086	0.197	0.357	12.331
	HyMFPSO	0.099	0.230	0.402	8.016
	MFPSO	0.078	0.186	0.275	3.160

Table 11 continued

SP	Wheel	RiseTime	PeakTime	SettlingTime	M_p (%)



(a) 30% duty cycle.



(b) 100% duty cycle.

Fig. 26 Transient response of the speed control using PID controller

the other algorithms. Still, the MFPSO gives a competitive peak time, especially at high-duty cycles, considering the outstanding performance in the other metrics.

Concerning the settling time, all the controllers' settling time is inversely proportional to the duty cycle. The MFPSO gives the best settling time in all duty cycles compared with the other algorithms, showing fast stability and convergence towards the setpoint. At the maximum rating, the MFPSO reduced the settling time by 34.83%, 21.20%, 20.75%, 22.97%, and 31.59% compared with the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms, respectively. This advantage and the MPFSO's best overshoot prove the MFPSO algorithm's superiority.

10.3 Statistical Analysis of the Results

Table 12 shows the statistical analysis of the transient response results. Regarding the rise time t_r, the p-value of the Friedman test is significant and equals $3.49E-11$. The ZN method has the best rise time with a minimum mean rank of 1, while the MFPSO algorithm comes third after the PSO algorithm with a mean rank of 3.6. The ZN outperforms the MFPSO in the paired comparison in the rise time metric with a significant p-value of 0.00065496. On the other hand, the HMFPSO outperforms the HyMFPSO with a significant p-value of 0.00065496. However, when comparing the MFPSO with the MFO, PSO, and HMFPSO algorithms, the rise time is insignificant, with p-values greater than 0.5, suggesting that the MFPSO algorithm performs similarly to these other algorithms in terms of rise time.

Figure 27a presents the confidence curves for the t_r metric. The zero-reference line falls within the confidence curves for the comparisons with the MFO, PSO, and HMFPSO algorithms, showing insignificant results consistentwith the Wilcoxon test results. For the ZN algorithm, the confidence curve lies on the negative side, indicating a significantly faster rise time for ZN compared to HMFPSO, with a confidence interval ranging from 0.016686 to 0.038023s across all duty cycles. Conversely, for the HyMFPSO algorithm, the confidence curve lies on the positive side, demonstrating a significantly faster rise time for MFPSO compared to HyMFPSO, with a confidence interval of [0.014173s to 0.020149s] across all duty cycles.

Regarding the peak time t_p, the ZN method comes first in the Friedman test with a significant p-value of $1.18E-11$, while PSO comes second and the MFPSO comes fifth. The paired comparisons show that the peak time of the ZN, MFO, PSO, and HMFPSO algorithms outperforms the MFPSO with significant p-values. The peak time is insignificant when comparing the MFPSO and the HyMFPSO algorithms. Figure 27b presents the confidence curves for the t_p metric. The zero-reference line falls within the confidence curves for the HyMFPSO comparison, showing insignificant peak time. The confidence curves for ZN, MFO, PSO, and HMFPSO algorithms lie on the negative side, showing better peak time than the MFPSO algorithm, consistent with Wilcoxon test results.

Concerning the settling time t_s, the MFPSO algorithm emerges as the clear winner in the Friedman test, with a significant p-value of $4.47E-13$. The HMFPSO and MFO algorithms follow the MFPSO closely in second and third places. In contrast, the ZN algorithm exhibits the longest settling time among all the algorithms. In the paired comparisons, the MFPSO consistently outperforms the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms in all 15 trials of different duty cycles, as evidenced by significant p-values in the Wilcoxon test. Figure 27c presents the confidence curves for the t_s metric. All confidence curves lie on the positive side,

Table 12 Statistical Analysis for all the algorithms

Metric	Alg.	Friedman Test			Sign Test	Wilcoxon Test				Confidence Interval		
		SumRank	MeanRank	p-value	+/=/-	R+	R-	p-value	H	LB	UB	pCI
t_r	ZN	15	1.000	3.49E-11	+0/=0/-15	0	120	0.00065496	TRUE	−0.038023	−0.016686	TRUE
	MFO	57	3.800		+7/=0/-8	38	82	0.21147639	FALSE	−0.008275	0.001028	FALSE
	PSO	40	2.667		+6/=0/-9	31	89	0.09953969	FALSE	−0.009245	0.000368	FALSE
	HMFPSO	59	3.933		+8/=0/-7	54	66	0.73327139	FALSE	−0.010077	0.004271	FALSE
	HyMFPSO	90	6.000		+15/=0/-0	120	0	0.00065496	TRUE	0.014173	0.020149	TRUE
	MFPSO	54	3.600		NA	NA	NA	NA	NA	NA	NA	NA
t_p	ZN	15	1.000	1.18E-11	+0/=0/-15	0	120	0.00065496	TRUE	−0.090626	−0.041163	TRUE
	MFO	47	3.133		+1/=0/-14	1	119	0.00080528	TRUE	−0.041191	−0.011507	TRUE
	PSO	41	2.733		+0/=0/-15	0	120	0.00065496	TRUE	−0.043299	−0.011585	TRUE
	HMFPSO	52	3.467		+4/=0/-11	18	102	0.01705872	TRUE	−0.049090	−0.003996	TRUE
	HyMFPSO	86	5.733		+11/=0/-4	91	29	0.07829229	FALSE	−0.003237	0.031060	FALSE
	MFPSO	74	4.933		NA	NA	NA	NA	NA	NA	NA	NA
t_s	ZN	90	6.000	4.47E-13	+15/=0/-0	120	0	0.00065496	TRUE	0.172012	0.220250	TRUE
	MFO	44	2.933		+15/=0/-0	120	0	0.00065496	TRUE	0.073872	0.099061	TRUE
	PSO	49	3.267		+15/=0/-0	120	0	0.00065496	TRUE	0.073500	0.101892	TRUE
	HMFPSO	42	2.800		+15/=0/-0	120	0	0.00065496	TRUE	0.074711	0.091367	TRUE
	HyMFPSO	75	5.000		+15/=0/-0	120	0	0.00065496	TRUE	0.125578	0.149625	TRUE
	MFPSO	15	1.000		NA	NA	NA	NA	NA	NA	NA	NA
M_p	ZN	90	6.000	2.08E-14	+15/=0/-0	120	0	0.00065496	TRUE	16.439093	18.530644	TRUE
	MFO	41	2.733		+15/=0/-0	120	0	0.00065496	TRUE	5.610579	5.941262	TRUE
	PSO	60	4.000		+15/=0/-0	120	0	0.00065496	TRUE	6.836199	7.578400	TRUE
	HMFPSO	75	5.000		+15/=0/-0	120	0	0.00065496	TRUE	9.382883	9.772483	TRUE
	HyMFPSO	34	2.267		+15/=0/-0	120	0	0.00065496	TRUE	5.262601	5.598712	TRUE
	MFPSO	15	1.000		NA	NA	NA	NA	NA	NA	NA	NA

MFPSO is the reference in all paired comparisons. For the Sign Test: '+' means the number of functions in which the MFPSO is better, and '=' means draw. For the Wilcoxon: R+ > R- means the MFPSO is better. For the Friedman test, the best algorithm is the one with the minimum mean rank. The significance level α equals 0.05. The results are significant if p-value $< \alpha$

showing a significantly faster setting time for the MFPSO than ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms with positive confidence intervals.

Similarly, the MFPSO outperforms the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms in the maximum overshoot criteria. It comes first in the Friedman test with a significant p-value of $2.08E-14$. The HyMFPSO and MFO algorithms come in second and third places, while ZN comes last with the largest maximum overshoot. The MFPSO consistently delivers the best results in all 15 trials in the paired comparison test, as indicated by significant p-values in the Wilcoxon test. Figure 27d presents the confidence curves for the M_p (maximum overshoot) metric. The confidence curves for all algorithms (ZN, MFO, PSO, HMFPSO, and HyMFPSO) consistently lie on the positive side, indicating a significantly higher overshoot than the MFPSO algorithm across all duty cycles. Specifically, the difference in overshoot (as a percentage) ranges from 16.44% to 18.53% for ZN, 5.61% to 5.94% for MFO, 6.84% to 7.58% for PSO, 9.38% to 9.77% for HMFPSO, and 5.26% to 5.60% for HyMFPSO. On average, MFPSO improves overshoot by 86.11% over ZN, 64.99% over MFO, 71.02% over PSO, 74.37% over HMFPSO, and 60.58% over HyMFPSO. This substantial reduction in overshoot highlights MFPSO's superior ability to control and minimize overshoot.

The MFPSO achieved the minimum overshoot with the fastest settling response and an acceptable rise time. The results show a multi-objective optimization problem and represent a trade-off between speed, accuracy, and safety. The balanced performance between accuracy and speed of the MFPSO algorithm suggests that MFPSO is the best choice for applications whose primary goal is maintaining system stability, safety, and precision. Its ability to provide the best overshoot control while maintaining competitive response times and adaptive response with high speed underlines its effective performance as a PID tuning algorithm and a versatile choice for speed control.

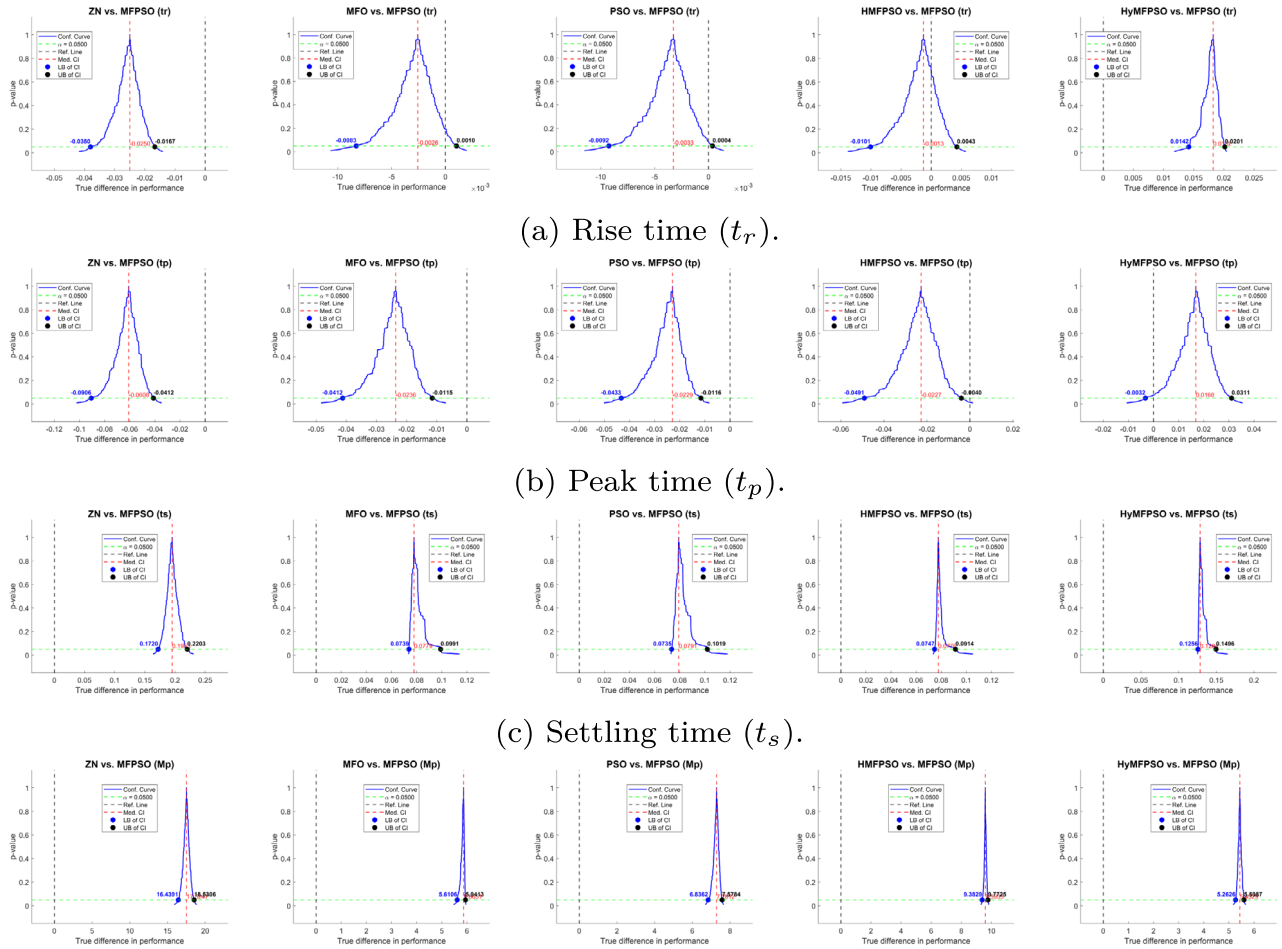


Fig. 27 Confidence curves for all the paired comparisons between MFPSO and all the other algorithms for all metrics

10.4 MFPSO Relationship Deductions of the Performance Metrics

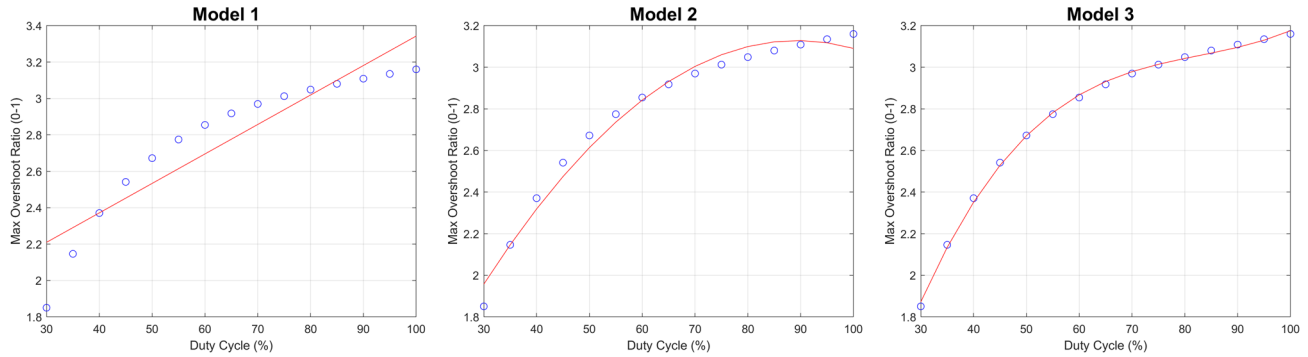
Supervised learning is applied to determine the relationship between performance metrics and the duty cycle for the MFPSO algorithm using ordinary least squares (OLS) regression. The OLS method minimizes the squared error between observed and predicted values, fitting linear, quadratic, and cubic models [68]. The general polynomial form is represented in Eq. 16, where x is the independent variable (duty cycle), y is the dependent variable (performance metric), and $(\beta_0, \beta_1, \beta_2, \beta_3)$ are model coefficients.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \tag{16}$$

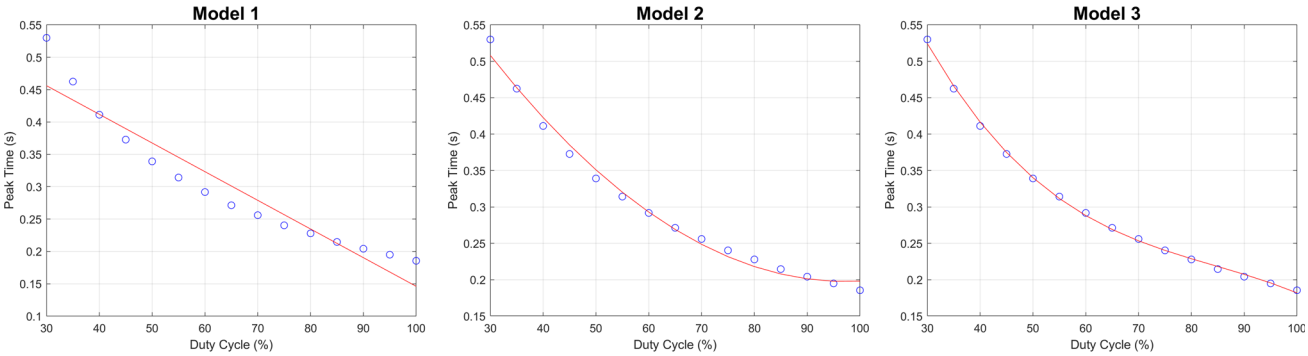
The analysis shows that quadratic models generally offer the best balance between accuracy and complexity for most metrics, avoiding underfitting seen in linear models and overfitting in cubic models (Fig. 28). Table 13 provides detailed coefficients, R^2 values, p-values, and RMSE for each model. For maximum overshoot (M_p), the quadratic model parameters ($\beta_0 = 0.474149$, $\beta_1 = 0.059425$, and $\beta_2 = -3.3258E-04$) explain 98.28% of the variance in M_p with an RMSE of 0.0497. The positive slope of β_1 confirms that M_p increases with the duty cycle, while the negative β_2 accounts for curvature.

The rise time (t_r) relationship is best described by the quadratic model with ($\beta_0 = 0.381766$, $\beta_1 = -0.006237$, and $\beta_2 = 3.2589E-05$), achieving an RMSE of 0.0043 and an R^2 of 99.09%. The negative slope of β_1 indicates t_r decreases as the duty cycle increases. For peak time (t_p), the quadratic model parameters ($\beta_0 = 0.846839$, $\beta_1 = -0.013348$, $\beta_2 = 6.86 \times 10^{-5}$) explain 99.08% of the variance in t_p and yield an RMSE of 0.0096. The relationship shows t_p decreases with increasing duty cycle.

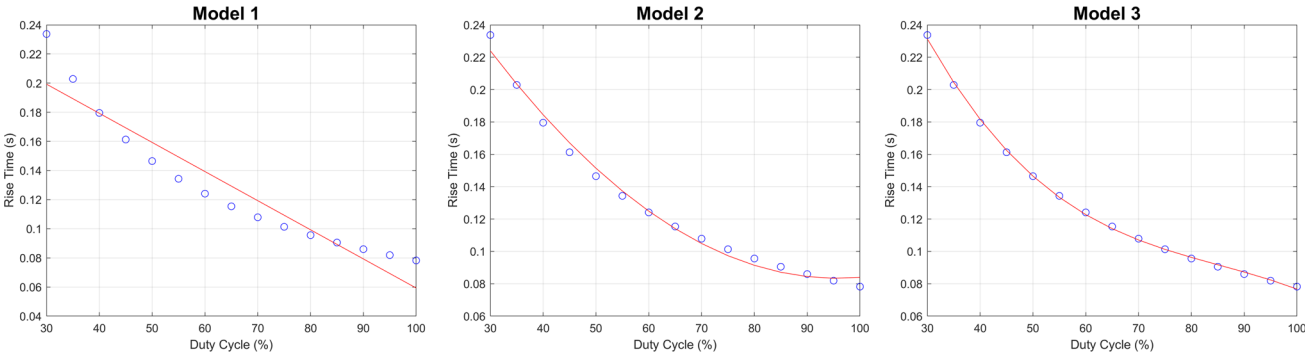
Settling time (t_s) is best described by a linear model, with parameters ($\beta_0 = 0.638315$, $\beta_1 = -0.003848$) explaining 97.94% of the variance with an RMSE of 0.0121. The



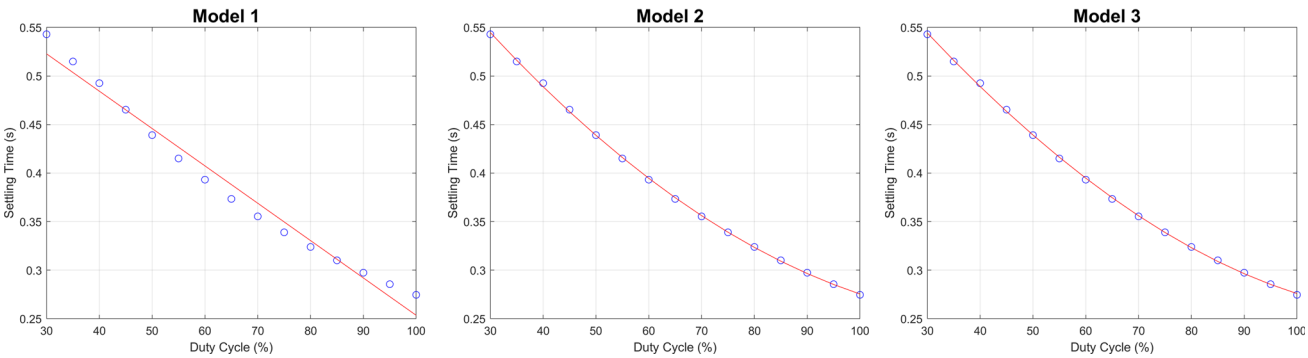
(a) Maximum Overshoot vs Duty Cycle for the three models.



(b) Rise time vs Duty Cycle for the three models.



(c) Peak time vs Duty Cycle for the three models.



(d) Settling time vs Duty Cycle for the three models.

Fig. 28 Regression results of the MFPSO performance metrics results

Table 13 OLS Regression Results for MFPSO Performance Metrics vs Duty Cycle

Metric	Model	β_0	β_1	β_2	β_3	$R\_squared$	$p-val$	RMSE
M_p	1	1.724113	0.016189	0.0000E+00	0.0000E+00	0.8506	9.9958E−07	0.1466
	2	0.474149	0.059425	−3.3258E−04	0.0000E+00	0.9828	2.5538E−11	0.0497
	3	−0.898670	0.133191	−1.5475E−03	6.2302E−06	0.9989	1.1868E−16	0.0123
t_r	1	0.259285	−0.002000	0.0000E+00	0.0000E+00	0.9027	6.0197E−08	0.0142
	2	0.381766	−0.006237	3.2589E−05	0.0000E+00	0.9909	5.5213E−13	0.0043
	3	0.499478	−0.012562	1.3676E−04	−5.3420E−07	0.9992	3.0602E−17	0.0013
t_p	1	0.589003	−0.004429	0.0000E+00	0.0000E+00	0.9103	3.5228E−08	0.0300
	2	0.846839	−0.013348	6.8604E−05	0.0000E+00	0.9908	6.2205E−13	0.0096
	3	1.106813	−0.027317	2.9867E−04	−1.1798E−06	0.9990	7.9783E−17	0.0031
t_s	1	0.638315	−0.003848	0.0000E+00	0.0000E+00	0.9794	2.4350E−12	0.0121
	2	0.746958	−0.007606	2.8907E−05	0.0000E+00	0.9997	5.3320E−22	0.0014
	3	0.739352	−0.007197	2.2177E−05	3.4514E−08	0.9997	7.3135E−20	0.0014

negative slope demonstrates that t_s decreases as the duty cycle increases. All p-values are significant (< 0.05), showing a strong relationship between the performance metrics and the motor's speed, represented in a continuous numerical relationship of the performance over time. These results establish strong relationships between the duty cycle and performance metrics, with quadratic models generally providing the best representation. For t_s, a linear model suffices. This analysis captures the continuous variation of performance with motor speed, offering an accurate model for system behavior across operating ranges.

11 Conclusion

This paper proposed a novel PID controller optimized by the MFPSO algorithm to control the speed of DC motors in a four-wheel differential drive (4WD) car, known for its high maneuverability. The assembly and hardware design of a modified Elegoo Smart Car V4 were discussed, including multiple modifications such as adding encoders for feedback, a Raspberry Pi controller for the ROS system, and upgraded motor drivers to make the four motors independent. A practical procedure, structured as four experiments, was proposed to ensure the alignment of the encoders and the synchronization of the four motors.

The MFPSO algorithm, a novel hybrid that combines the strengths of the PSO and MFO algorithms, is introduced to address the PSO's premature convergence and the MFO's slow convergence. The MFPSO outperformed the traditional MFO, PSO, and two other recent hybrid variants on CEC2020/2021 and engineering optimization benchmark functions, ranking first in the Friedman test with significant p-values and confidence intervals, while the MFO ranked second.

The proposed MFPSO algorithm was deployed as an interactive PID controller for the speed control of the 4WD car's DC motors. The PID-MFPSO controller demonstrated superior performance compared to the traditional Ziegler-Nichols (ZN) method, MFO, PSO, and other hybrid algorithms, achieving the minimum overshoot and settling time across different duty cycles. The MFPSO algorithm achieved significant improvements, reducing settling time by 34.83%, 21.20%, 20.75%, 22.97%, and 31.59%, and decreasing overshoot by 86.11%, 64.99%, 71.02%, 74.37%, and 60.58% when compared to the ZN, MFO, PSO, HMFPSO, and HyMFPSO algorithms, respectively.

The MFPSO algorithm shows excellent potential for future applications in various PID control scenarios, such as steering control, and in solving other engineering optimization problems. Future work could further enhance the performance of the MFPSO by introducing adaptive parameters or by integrating it with other optimization methods.

Supplementary Materials

The following supporting information can be downloaded at: Supplementary Materials or at https://github.com/MohamedRedaMu/MFPSO-Algorithm. The supplementary materials include Section S1: sensitivity analysis heatmaps and bar charts for the MFPSO parameters (c_1, c_2, and w) on the 10D and 20D CEC2020/2021 benchmark functions. Section S2 covers statistical analysis background and preliminaries, including confidence interval and confidence curves (Section S2.1), convergence trends using Page Test (Section S2.2), and computational time complexity analysis (Section S2.3). Section S3 provides additional CEC2020/2021 benchmark results, including detailed results and violin plots (Section

S3.1), full sets of confidence curves (Section S3.2), and a detailed Page Test table (Section S3.3). Section S4 outlines engineering optimization benchmark problem definitions. Section S5 contains additional engineering optimization benchmark results, such as detailed results and box plots (Section S5.1), full sets of confidence curves (Section S5.2), and a detailed Page Test table (Section S5.3). Section S6 includes additional transient response curves for PID speed control testing at different duty cycles (30%-100%).

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10846-025-02228-1.

Author Contributions Conceptualization, M.R.; taxonomy, M.R.; methodology, M.R.; software, M.R.; hardware, M.R.; visualization, M.R.; formal analysis, M.R.; validation, M.R.; data curation, M.R.; investigation, M.R.; resources, A.O.; writing—original draft preparation, M.R; writing—review and editing, M.R., A.O, A.G., and A.Y.H.; supervision, A.O, A.G., and A.Y.H.; project administration, A.O, A.G., and A.Y.H.; funding acquisition, A.O., All authors have read and agreed to the published version of the manuscript.

Funding Not applicable.

Data Availability No datasets were used during the current study.

Code Availability The MATLAB source code and the result files for the proposed algorithm are now available at the following GitHub repository: https://github.com/MohamedRedaMu/MFPSO-Algorithm.

Declarations

Ethical Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

Conflict of Interest There is no conflict of interest between the authors to publish this manuscript.

References

1. Gao, H., Zhu, J., Zhang, T., Xie, G., Kan, Z., Hao, Z., Liu, K.: Situational assessment for intelligent vehicles based on stochastic model and gaussian distributions in typical traffic scenarios. IEEE Trans. Syst. Man Cybern. Syst. **52**(3), 1426 (2020)
2. Wei, S., Pfeffer, P.E., Edelmann, J.: State of the art: Ongoing research in assessment methods for lane keeping assistance systems. IEEE Trans. Intell. Veh. (2023)
3. Reda, M., Onsy, A., Haikal, A.Y., Ghanbari, A.: Optimizing the steering of driverless personal mobility pods with a novel differential Harris hawks optimization algorithm (dhho) and encoder modeling. Sensors (Basel, Switzerland) **24**(14) (2024)
4. Gao, H., Kan, Z., Li, K.: Robust lateral trajectory following control of unmanned vehicle based on model predictive control. IEEE/ASME Trans. Mechatron. **27**(3), 1278 (2021)
5. Reda, M., Onsy, A., Haikal, A.Y., Ghanbari, A.: Path planning algorithms in the autonomous driving system: A comprehensive review. Robot. Auton. Syst. **174**, 104630 (2024)
6. Hang, P., Chen, X.: In: Actuators, vol. 10-8. MDPI, p. 184 (2021)
7. Toliyat, H.A., Kliman, G.B.: Handbook of Electric Motors, vol. 120. CRC press (2018)
8. Tian, J., Tong, J., Luo, S.: Differential steering control of four-wheel independent-drive electric vehicles. Energies **11**(11), 2892 (2018)
9. Galasso, F., Rizzini, D.L., Oleari, F., Caselli, S.: Efficient calibration of four wheel industrial agvs. Robot. Comput. Integr. Manuf. **57**, 116 (2019)
10. Borase, R.P., Maghade, D., Sondkar, S., Pawar, S.: A review of pid control, tuning methods and applications. Int. J. Dyn. Control **9**, 818 (2021)
11. Purnama, H.S., Sutikno, T., Alavandar, S., Subrata, A.C.: In: 2019 IEEE Conference on Energy Conversion (CENCON). IEEE, pp. 24–30 (2019)
12. Sridhar, H., Hemanth, P., Soumya, H., Joshi, B.G., et al.: In: 2020 International Conference on Smart Electronics and Communication (ICOSEC). IEEE, pp. 1162–1168 (2020)
13. Rahayu, E.S., Ma'arif, A., Cakan, A.: Particle swarm optimization (pso) tuning of pid control on dc motor. Int. J. Robot. Control Syst. (2022)
14. Shaikh, M.S., Raj, S., Babu, R., Kumar, S., Sagrolikar, K.: A hybrid moth-flame algorithm with particle swarm optimization with application in power transmission and distribution. Decis. Anal. J. **6**, 100182 (2023)
15. Sahoo, S.K., Saha, A.K.: A hybrid moth flame optimization algorithm for global optimization. J. Bionic Eng. **19**(5), 1522 (2022)
16. Qi, Z., Shi, Q., Zhang, H.: Tuning of digital pid controllers using particle swarm optimization algorithm for a can-based dc motor subject to stochastic delays. IEEE Trans. Ind. Electron. **67**(7), 5637 (2019)
17. Xie, W., Wang, J.S., Wang, H.B., et al.: Pi controller of speed regulation of brushless dc motor based on particle swarm optimization algorithm with improved inertia weights. Math. Probl. Eng. **2019** (2019)
18. Garba, S., Ntuen, E., Salawudeen, A., Zubairu, A., Abubakar, A., Adebiyi, B.: Design of an optimized controller for dc motor speed control using abc and pso: A comparative study. Bayero J. Eng. Technol. (2019)
19. Yazgan, H., Yener, F., Soysal, S., Ahmet, G.: Comparison performances of pso and ga to tuning pid controller for the dc motor. Sakarya Univ. J. Sci. **23**(2), 162 (2019)

20. Ramya, M., Jadhav, S.P., Pawar, S.N.: In: 2020 International Conference for Emerging Technology (INCET). IEEE, pp. 1–6 (2020)
21. Acharyulu, B., Mohanty, B., Hota, P.: In: Applications of Artificial Intelligence Techniques in Engineering: SIGMA 2018, vol. 1. Springer, pp. 509–518 (2019)
22. Bennaoui, A., Saadi, S., Ameur, A.: In: 2020 International Conference on Electrical Engineering (ICEE). IEEE, pp. 1–5 (2020)
23. Mustafa, N., Hashim, F.H.: Design of a predictive pid controller using particle swarm optimization. Int. J. Electron. Telecommun. **66**(4), 737 (2020)
24. Sultan, G.A., Sheet, A.F., Ibrahim, S.M., Farej, Z.K.: Speed control of dc motor using fractional order pid controller based on particle swarm optimization. Indones. J. Electr. Eng. Comput. Sci. **22**(3), 1345 (2021)
25. Valluru, S.K., Sehgal, K., Thareja, H.: In: 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). IEEE, pp. 1–6 (2021)
26. Safarzadeh, O., Noori-kalkhoran, O.: A fractional pid controller based on fractional point kinetic model and particle swarm optimization for power regulation of smart reactor. Nucl. Eng. Des. **377**, 111137 (2021)
27. Vishnoi, V., Tiwari, S., Singla, R.: Performance analysis of moth flame optimization-based split-range pid controller. Mapan **36**, 67 (2021)
28. Yusubov, E., Bekirova, L.: A moth-flame optimized robust pid controller for a sepic in photovoltaic applications. IFAC-PapersOnLine **55**(11), 120 (2022)
29. Naik, K.P., Pradhan, R., Majhi, S.K.: In: International Conference on Communications and Cyber Physical Engineering 2018. Springer, pp. 847–859 (2023)
30. Sharma, A., Sharma, V., Rahi, O.: In: Control and Measurement Applications for Smart Grid: Select Proceedings of SGESC 2021. Springer, pp. 79–89 (2022)
31. Shary, D.K., Nekad, H.J., Alawan, M.A.: Speed control of brushless dc motors using (conventional, heuristic, and intelligent) methods-based pid controllers. Indones. J. Electr. Eng. Comput. Sci. **30**(3), 1359 (2023)
32. ELEGOO Inc.: Smart robot car V4.0 with camera assembly tutorial. https://www.elegoo.com/en-gb/blogs/arduino-projects/elegoo-smart-robot-car-kit-v4-0-tutorial (2021). Available from ELEGOO Inc. Website
33. Febbo, R., Flood, B., Halloy, J., Lau, P., Wong, K., Ayala, A.: In: Practice and Experience in Advanced Research Computing. ACM, pp. 333–338 (2020)
34. Latoui, A., Daachi, M.E.H.: In: 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET). IEEE, pp. 1–5 (2021)
35. Singh, S., Weeber, M., Birke, K.P.: Advancing digital twin implementation: A toolbox for modelling and simulation. Procedia CIRP **99**, 567 (2021)
36. Farrugia, S.: Autonomous Robot Path Planning and Obstacle Avoidance in a Dynamic Environment. B.S. thesis, University of Malta (2022)
37. Williams, N.L., Rewkowski, N., Li, J., Lin, M.C.: In: 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 12478–12485 (2023)
38. Huynh, T., Walter, J., Aveta, F.: In: 2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). IEEE, pp. 584–589 (2023)
39. Borg, G., Montebello, M.: In: 2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA). IEEE, pp. 1–7 (2023)
40. Kennedy, J., Eberhart, R.: In: Proceedings of ICNN'95-international Conference on Neural Networks. IEEE, vol. 4, pp. 1942–1948 (1995)
41. Mirjalili, S.: Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl.-based Syst. **89**, 228 (2015)
42. Bingi, K., Kulkarni, R.R., Mantri, R.: In: 2021 IEEE Madras Section Conference (MASCON). IEEE, pp. 1–6 (2021)
43. Yang, Z., Shi, K., Wu, A., Qiu, M., Hu, Y.: In: 2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). IEEE, vol. 2, pp. 207–210 (2019)
44. Jain, M., Saihjpal, V., Singh, N., Singh, S.B.: An overview of variants and advancements of pso algorithm. Appl. Sci. **12**(17), 8392 (2022)
45. Harrison, K.R., Ombuki-Berman, B.M., Engelbrecht, A.P.: In: International Conference on Swarm Intelligence. Springer, pp. 93–105 (2019)
46. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M.: Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm. Swarm Evol. Comput. **41**, 20 (2018)
47. Isiet, M., Gadala, M.: Sensitivity analysis of control parameters in particle swarm optimization. J. Comput. Sci. **41**, 101086 (2020)
48. Yue, C.T., Price, K.V., Suganthan, P.N., Liang, J.J., Ali, M.Z., Qu, B.Y., Awad, N.H., Biswas, P.P.: Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical Report, Nanyang Technological University. Singapore (2019). https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark
49. Mohamed, A.W., Hadi, A.A., Mohamed, A.K., Agrawal, P., Kumar, A., Suganthan, P.N.: Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical Report. Nanyang Technological University (2020). https://github.com/P-N-Suganthan/2021-SO-BCO
50. Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H.: Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. **97**, 849 (2019)
51. Carrasco, J., García, S., Rueda, M., Das, S., Herrera, F.: Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. Swarm Evol. Comput. **54**, 100665 (2020)
52. Bayzidi, H.: Social network search for solving engineering problems (2024). Available: https://www.mathworks.com/matlabcentral/fileexchange/97577-social-network-search-for-solving-engineering-problems. Last Accessed 18 Nov 2024
53. Bayzidi, H., Talatahari, S., Saraee, M., Lamarche, C.P.: Social network search for solving engineering optimization problems. Comput. Intell. Neurosci. **2021**(1), 8548639 (2021)
54. Raspberry Pi Foundation: Raspberry Pi 4 Model B specifications. (n.d.). Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. Last Accessed 14 Dec 2023
55. JOY-IT: KY-040 Rotary Encoder Datasheet. (n.d.). https://www.alldatasheet.com/datasheet-pdf/pdf/1648739/JOY-IT/KY-040.html. Accessed 14 Dec 2023
56. Ellin, A., Dolsak, G.: The design and application of rotary encoders. Sensor Rev. **28**(2), 150 (2008)
57. AI Williams: Fet based motor driver is better than l298n. (2019). Available online: https://hackaday.com/2019/12/29/fet-based-motor-driver-is-better-than-l298n/
58. STMicroelectronics: L298N Dual Full-Bridge Driver. (n.d.). https://www.alldatasheet.com/datasheet-pdf/pdf/22440/STMICROELECTRONICS/L298N.html. Accessed 14 Dec 2023
59. Arduino: Arduino UNO R3. (n.d.). https://www.arduino.cc/en/Main/ArduinoBoardUno. Accessed 14 Dec 2023

60. Arduino: Arduino Mega 2560 Datasheet. (n.d.). Available: https://docs.arduino.cc/hardware/mega-2560/. Last Accessed 14 Dec 2023
61. GEEKWORM: X728 V2.3 Raspberry Pi UPS Manual. (2022). https://wiki.geekworm.com/X728#X728_V2.3. Accessed 23 Dec 2023
62. Tatenda Katsambe, C., Luckose, V., Shahabuddin, N.S.: Effect of pulse width modulation on dc motor speed. Int. J. Stud. Res. Technol. Manag. **5**(2), 42 (2017). https://doi.org/10.18510/ijsrtm.2017.522. https://mgesjournals.com/ijsrtm/article/view/ijsrtm.2017.522
63. Raza, K.M., Mohd, K., Kumar, P.: Speed control of dc motor by using pwm. Int. J. Adv. Res. Comput. Commun. Eng. **5**(4) (2016)
64. Yung, C.: How to save your motors during a brownout, EC&M (2005). https://www.ecmweb.com/content/article/20894433/how-to-save-your-motors-during-a-brownout. Accessed 19 Dec 2023
65. Ogata, K.: Modern Control Engineering Fifth Edition. Prentice Hall PTR. (2010)
66. Kumar, V., Patra, A.: Application of ziegler-nichols method for tuning of pid controller. Int. J. Electr. Electron. Eng. **8**(2), 559 (2016)
67. Patel, V.V.: Ziegler-nichols tuning method: Understanding the pid controller. Resonance **25**(10), 1385 (2020)
68. Boyko, A., Kukartsev, V., Tynchenko, V., Korpacheva, L., Dzhioeva, N., Rozhkova, A., Aponasenko, S.: In: Journal of Physics: Conference Series, vol. 1582. IOP Publishing, vol. 1582, p. 012016 (2020)

Mohamed Reda received his B.Sc. and M.Sc. degrees in Computers and Control Systems Engineering from the Faculty of Engineering at Mansoura University, Egypt. He is currently pursuing his Ph.D. at the School of Engineering, University of Central Lancashire (UCLan) in the UK. Since 2016, he has been an Assistant Lecturer at the Faculty of Engineering, Mansoura University, and an Associate Lecturer at UCLan's School of Engineering since 2022. His primary research areas include artificial intelligence, meta-heuristic optimization techniques, evolutionary computation, and control engineering. Additionally, Mohamed has a strong interest in the application of AI and deep learning to embedded systems, biomedical systems, and robotics, with a particular focus on automated vehicles.

Ahmed Onsy awarded his Ph.D. from the School of Mechanical and Systems Engineering, Design Unit and Mechatronics Group, Newcastle University, UK. His main research interests are intelligent diagnostics and health management systems, intelligent maintenance systems, advanced mechatronics, and embedded systems, which can be directly applied to Intelligent Diagnostic and Health Management (DHM) and Predictive Health Monitoring (PHM) systems for oil well, wind turbine, aerospace (SHM & HUM), marine and automotive applications. He is a member of the Jost Institute for Tribotechnology. Ahmed has taken different roles at the University of Central Lancashire School of Engineering since 2014: Lecturer, Senior Lecturer, Principal Lecturer, and Academic Lead for the Mechanical and Maintenance Engineering area. Ahmed is currently the Associate Dean School of Engineering and Computing for Business Development and Partnerships, driving the School of Engineering UK and International Partnerships. Ahmed is a member of the University Research and Innovation Committee. He successfully placed the Mechanical Engineering Programmes in the top 15 universities in the UK in 2020 NSS results in overall student satisfaction.

Amira Y. Haikal received the B.Sc., M.Sc., and Ph.D. degrees from the Computers and Control Systems Engineering Department, Mansoura University, Egypt. She is the vice dean of environmental and community affairs and the head of the Computers and Control Systems Engineering Department, Faculty of Engineering, Mansoura University. Her major research interests include artificial intelligence fields: meta-heuristic optimization techniques, machine learning, deep learning and smart systems engineering.

Ali Ghanbari received his Ph.D. in Mechanical Engineering from Amirkabir University of Technology in 2011. His research focuses on control of microrobots for biomedical applications. Dr Ghanbari is interested in how to make and model intelligent devices at small scale. He was a researcher in Robotics Engineering Department at Daegu Gyeongbuk Institute of Science and Technology (DGIST) and University of Leeds working on micro-/nanorobots and soft robotics. Dr Ghanbari was also a guest researcher at Multi-Scale Robotics Lab in ETH Zurich. Currently, he is a lecturer in the School of Engineering and Computing at the University of Central Lancashire.