Iranian Journal of Fuzzy Systems

Volume 23, Number 5, (2026), pp. 87-105

Original paper

University of Sistan and Baluchestan

A modified Jaya-optimized decomposed fuzzy control approach for self-balancing two-wheel robots

N. K. Long [1] and N. V. Binh [2]

[1,2] *Faculty of Mechatronics Engineering, Lac Hong University, Dong Nai, Vietnam*

ngokimlong@lhu.edu.vn, ngovanbinh@lhu.edu.vn

Abstract

This study develops a decomposed fuzzy controller for a self-balancing two-wheel robot, with the learning rates of the fuzzy modules optimized by a modified Jaya algorithm. The controller is structured into two interacting parts-one regulating the body tilt and the other compensating for wheel dynamics-to improve the handling of nonlinearities and reduce the computational load during real-time operation. The modified Jaya algorithm is incorporated to refine the adaptation process, enabling faster convergence and more reliable parameter updates under varying operating conditions. The proposed approach is assessed through both simulation and hardware experiments. The experimental platform consists of an Arduino Mega 2560 board, an L298N motor driver, and an MPU6050 sensor, which provides real-time attitude measurements for feedback control. Results show that the optimized fuzzy controller stabilizes the robot more quickly and maintains balance more effectively in the presence of disturbances compared with a conventional fuzzy design without optimization. These outcomes confirm the suitability of the modified Jaya–based optimization in enhancing the performance of decomposed fuzzy control for self-balancing robotic systems.

Keywords: Fuzzy controller, decomposed fuzzy controller, self-balancing two-wheel robot, adaptive control, Jaya algorithm.

1 Introduction

Self-balancing two-wheel robots constitute a compact but demanding benchmark for intelligent control because their upright equilibrium is open-loop unstable, the body tilt and wheel displacement are strongly coupled, and the available actuation must regulate two performance variables through a single motor command. Recent mechanical and control-oriented studies have emphasized this difficulty from complementary viewpoints. Chen et al. developed a composite self-balancing robot for stationary operation on unknown terrain, showing that practical balance control must tolerate ground interaction and model mismatch, whereas Papadimitriou et al. demonstrated that evolutionary optimization can improve a two-wheeled robot controller but may increase the tuning burden when many controller parameters are optimized simultaneously [1, 6, 11, 18]. These studies indicate that an effective controller for a low-cost balancing platform should combine nonlinear compensation, robustness, and a compact tuning structure rather than relying only on a nominal model.

Model-based and classical feedback approaches provide an important baseline for this problem. Prescribed-performance tracking and fuzzy navigation studies show that explicit transient specifications and data-aided fuzzy rules can improve mobile-robot behavior, while LQR, PID, fractional-order PID, and ARX-based designs remain attractive because of their transparent structure and low implementation cost [12, 24, 25, 26, 30, 31, 33]. In particular, optimized LQR tuning is effective near the upright equilibrium, and fuzzy fractional-order PID control improves disturbance rejection on inclined surfaces. However, these controllers usually depend on a linearized or manually tuned design, so their

Corresponding Author: N. K. Long
Received: November 2025; Revised: May 2026; Accepted: July 2026.
https://doi.org/10.22111/ijfs.2026.53940.9555

performance can degrade when the robot starts from a larger tilt angle, operates under motor dead zones and friction, or experiences unmodeled load and disturbance variations.

Implementation-oriented robotic studies further suggest that control simplicity alone is insufficient when mobility and disturbance rejection are required. For example, integrating LQRADRC improves the stabilization of wheel-legged robots by combining nominal linear feedback with active disturbance rejection, whereas low-cost Arduino-based balancing platforms demonstrate the importance of computational efficiency for embedded implementation [2, 7, 28, 32]. These works support the use of a controller that is simple enough for real-time microcontroller execution but still flexible enough to compensate the nonlinear coupling between tilt recovery and wheel motion.

Fuzzy, neuro-fuzzy, and type-2 fuzzy controllers are suitable candidates because they can approximate nonlinear control surfaces without requiring a fully accurate analytical model. Bekhiti et al. reported an adaptive neuro-fuzzy MIMO controller for a self-balancing two-wheeled robot, highlighting the benefit of learning-based fuzzy structures for coupled dynamics, while Pham et al. used type-2 fuzzy control to enhance robustness under parameter uncertainties [5, 20]. Related optimized fuzzy PD and optimal fuzzy balancing controllers further confirm the usefulness of fuzzy inference for nonlinear stabilization, while hybrid fuzzy learning models in other nonlinear engineering tasks illustrate the broader relevance of fuzzy approximators [13, 15, 16]. Nevertheless, conventional full-grid fuzzy systems suffer from rule-base growth as the number of inputs increases, and poorly selected learning rates may produce either slow adaptation or oscillatory parameter updates.

Decomposition is therefore a relevant strategy for reducing fuzzy-controller complexity. Evolutionary and time-fractional fuzzy studies motivate bounded and structured fuzzy adaptation under uncertainty, while decomposed fuzzy decision, concept-lattice, multilayer type-2 fuzzy Petri net, and decomposed Z-fuzzy frameworks show that a high-dimensional fuzzy problem can be separated into smaller interpretable components [4, 9, 14, 17, 23, 27]. In the present robot, this idea has a direct physical interpretation: the tilt branch should rapidly recover the upright body angle, whereas the wheel branch should compensate translational drift and support distance regulation. Separating these two roles avoids a monolithic full-grid rule base and makes the resulting controller easier to interpret and implement.

The decomposed structure also creates a favorable setting for optimization-based tuning. Optimizing all fuzzy centers, spreads, and consequents would lead to a high-dimensional search problem and would obscure the role of online adaptation. Instead, the present work optimizes only the learning-rate vector of the adaptive laws. The combination of data decomposition and multiobjective optimization in fuzzy forecasting further supports the value of separating representation complexity from optimizer design [29]. Jaya-type algorithms are attractive for this purpose because they are parameter-light and move candidate solutions toward the best member while avoiding the worst member; recent fractional-order and self-adaptive Jaya variants confirm their usefulness for compact global optimization and parameter-identification problems [8, 19]. Additional studies on fuzzy mobile-robot navigation, control-oriented trajectory optimization, evolving fuzzy and neuro-fuzzy control, and modern optimal-control modeling further motivate a constrained objective that balances tracking error, control effort, disturbance rejection, and admissible learning bounds [3, 10, 21, 22].

Although the above studies provide important foundations, three limitations remain relevant to the present self-balancing robot problem. First, model-based controllers such as SDRE, LQR, and ADRC-enhanced linear designs are effective around their modeled operating regions, but they still require reliable plant parameters and may lose performance under motor dead zones, friction, load variations, and larger initial tilts. Second, conventional fuzzy and neuro-fuzzy controllers improve nonlinear approximation, but full-grid rule bases and manually selected learning gains can increase computational burden and may cause slow or oscillatory adaptation. Third, generic metaheuristic tuning can improve controller parameters, but optimizing all membership centers, spreads, and consequents simultaneously creates a search problem that is unnecessarily large for embedded balancing control.

Building on these observations, this work introduces a Modified-Jaya-optimized decomposed fuzzy control approach for self-balancing two-wheel robots. The controller consists of two separate instances of the same decomposed fuzzy computational structure: a Tilt-DFLC branch that generates the tilt-stabilizing component u_θ and a Wheel-DFLC branch that generates the wheel-motion component u_x. These two components are combined only at the final stage to form the single physically admissible motor command. The Modified-Jaya optimizer tunes only the learning-rate vector of the online adaptation laws, thereby keeping the optimization problem compact while linking the optimized parameters directly to convergence, bounded learning, and stability-oriented constraints.

The main contributions and innovations of this study are summarized as follows. First, a two-branch decomposed fuzzy architecture is formulated specifically for the coupled tilt-position dynamics of a self-balancing two-wheel robot. Compared with a conventional monolithic FLC, the proposed architecture separates tilt stabilization and wheel-motion compensation, thereby reducing the number of active rules and improving interpretability. Second, asymmetric Gaussian membership functions are introduced in both branches. Their independent left and right spreads allow different control sensitivity for positive and negative deviations, which is important because the robot exhibits direction-dependent

behavior during acceleration, motor dead-zone operation, and recovery from external pushes. Third, the Modified-Jaya optimizer is used to tune only the learning rates of the online adaptation laws. This creates a low-dimensional optimization problem directly related to convergence and stability, rather than an oversized offline search over all fuzzy parameters. Fourth, the approach is validated through simulation and hardware experiments, including initial-angle variations, impulse disturbances, sensor/process noise, and real-time implementation on a low-cost microcontroller.

The remainder of the paper is organized as follows. Section 2 presents the dynamic model and the linearized state-space representation of the self-balancing robot. Section 3 describes the proposed decomposed fuzzy controller, the asymmetric membership functions, the online adaptation laws, the Modified-Jaya learning-rate optimization problem, and the stability analysis. Section 4 reports the simulation and experimental results, including robustness tests and comparative performance indices. Section 5 concludes the paper and discusses future extensions.

2 Mathematical modeling of the two-wheel self-balancing robot

2.1 System definition and kinematics

The two-wheel self-balancing robot is fundamentally modeled as an **Inverted Pendulum on a Cart (IPOC)** system, where the horizontal motion of the cart (x) is replaced by the rotational motion of the wheels (ϕ). The system has two degrees of freedom (DoF) in the sagittal plane: the tilt angle of the body (θ) and the angular displacement of the wheels (ϕ).

The key parameters used for modeling are defined as follows:

- M: Mass of the robot body (chassis).
- m: Mass of one wheel/motor assembly.
- J_B: Moment of inertia of the body about its center of mass (CoM).
- J_W: Moment of inertia of one wheel about its rotational axis.
- R: Radius of the wheels.
- L: Distance from the wheel axis to the CoM of the body.
- θ: Tilt angle of the body with respect to the vertical axis (0 is upright).
- ϕ: Angular displacement of the wheels (related to the horizontal position x by $x = R\phi$).
- τ: Input torque applied by the motors to the wheels.
- g: Acceleration due to gravity.

2.2 Dynamic equations of motion

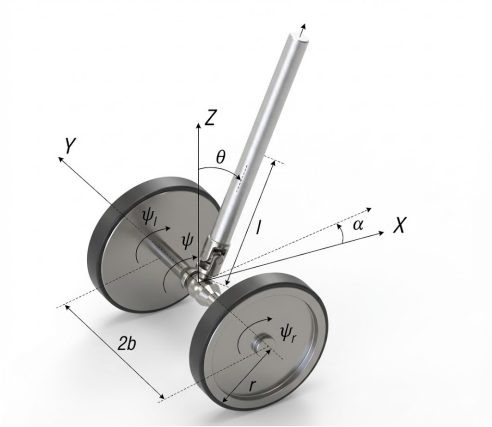

Figure 1: The dynamic model of the two-wheel self-balancing robot.

The dynamic equations of motion are typically derived using the Euler-Lagrange formulation, assuming pure rolling without slippage and rigid body dynamics. The coupling between the body tilt (θ) and the wheel rotation (ϕ) results in two primary non-linear differential equations:

(A) Body Tilt Dynamics (Governing $\ddot{\theta}$) The equation governing the rotational acceleration of the body $\ddot{\theta}$ is:

$$(J_{eq} + ML^2)\ddot{\theta} - MLR\cos(\theta)\ddot{x} - MLg\sin(\theta) = -MLR\sin(\theta)\dot{x}^2 + D_\theta\dot{\theta}, \tag{1}$$

where $J_{eq} = J_B + 2J_W + 2mR^2$.

(B) Wheel Rotational Dynamics (Governing $\ddot{x}$) The equation governing the angular acceleration of the wheels $\ddot{\phi}$ is:

$$(2J_W + 2mR^2)\ddot{x} - MLR\cos(\theta)\ddot{\theta} + MLR\sin(\theta)\dot{\theta}^2 = \tau - D_x\dot{x}, \tag{2}$$

where D_θ and D_x represent viscous damping coefficients.

2.3 Linearized state-space model

For control design purposes, the non-linear equations are typically linearized around the equilibrium point $\theta = 0$ (upright position) and $\dot{\theta} = 0$. We use the small-angle approximations $\sin(\theta) \approx \theta$ and $\cos(\theta) \approx 1$. The state vector is defined as $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]^T$. The resulting linearized system can be expressed in the standard state-space form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$, where the input u is the motor torque τ.

The linearized equations are derived by isolating $\ddot{\theta}$ and $\ddot{x}$ from (1) and (2) and applying the small-angle approximation.

Final Linearized Equations (Simplified Form) The dynamics can be simplified and expressed in matrix form, leading to the state-space representation. The key relationships derived from the linearized equations are:

$$\begin{bmatrix} I_{xx} & I_{x\theta} \\ I_{\theta x} & I_{\theta\theta} \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} B_x \\ B_\theta \end{bmatrix} \tau + \begin{bmatrix} D_{xx} & D_{x\theta} \\ D_{\theta x} & D_{\theta\theta} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} G_x \\ G_\theta \end{bmatrix} \theta. \tag{3}$$

By defining appropriate constants, the system is converted into the 4^{th}-order state-space representation required for LQR or adaptive control design. In (1)–(3), all variables are defined as follows. The state variable $x = R\phi$ denotes the horizontal wheel-axis displacement, $\dot{x}$ and $\ddot{x}$ are its velocity and acceleration, θ is the body tilt angle measured from the vertical upright position, and $\dot{\theta}$ and $\ddot{\theta}$ are the corresponding angular velocity and acceleration. The control input $u \equiv \tau$ is the motor torque applied to the wheels. The parameters M, m, J_B, J_W, R, L, D_x, D_θ, and g denote, respectively, body mass, wheel/motor mass, body moment of inertia, wheel moment of inertia, wheel radius, distance from the wheel axis to the body center of mass, translational damping, tilt damping, and gravitational acceleration. In this work, distance tracking means regulation of the horizontal displacement $x(t)$ toward a bounded reference $x_{\text{ref}}(t)$; for pure self-balancing tests, $x_{\text{ref}}(t)$ is set to a constant so that the wheel branch mainly suppresses drift while the tilt branch maintains $\theta(t) \to 0$.

3 The structure of proposed MJaya-DFLC

3.1 Decomposed fuzzy control architecture

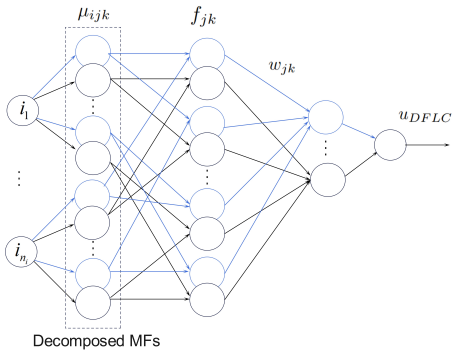

Figure 2: Structure of the DFLC architecture with representative parallel decomposed fuzzy pathways.

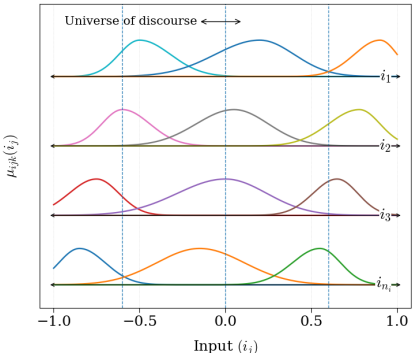

Figure 3: The structure of decomposed membership functions.

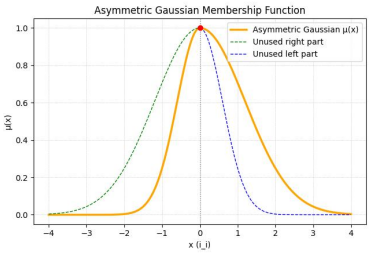

Figure 4: Asymmetric Gaussian membership function.

The structure of the DFLC controller is shown in Fig. 2. The control system comprises two parallel Decomposed Fuzzy Logic Controller (DFLC) branches:

- **Tilt-DFLC branch:** Takes the input $\mathbf{i}^{(\theta)} = [\theta, \dot{\theta}]^T$ (from MPU6050); it outputs the compensating torque u_θ to stabilize the system along the vertical axis.
- **Wheel-DFLC branch:** Takes the input $\mathbf{i}^{(x)} = [x, \dot{x}]^T$; it outputs the wheel-motion component u_x for displacement regulation and drift suppression.

The final control signal u fed to the L298N motor driver is a weighted sum:

$$u = \alpha u_\theta + (1 - \alpha)u_x, \quad 0 \leq \alpha \leq 1. \tag{4}$$

In Fig. 2, the blue and black links are used to visually distinguish representative parallel decomposed fuzzy pathways associated with different input channels and different fuzzy-parameter sets within the same DFLC architecture. Each pathway follows the same computational procedure, including membership evaluation, firinge-strength calculation, consequent weighting, and contribution to the final DFLC output. Therefore, the two colors indicate parallel decomposed pathways with different inputs and parameters, while their mathematical role in the DFLC inference process is identical. In the proposed controller, this DFLC architecture is instantiated separately for the tilt and wheel branches. The tilt-related instance generates u_θ, whereas the wheel-motion instance generates u_x. These two branch outputs are available before fusion and are combined only at the final stage as $u = \alpha u_\theta + (1 - \alpha)u_x$. The proposed control architecture employs the decomposed membership-function structure shown in Fig. 3. The asymmetric Gaussian membership function shown in Fig. 4 is adopted to increase local shape flexibility. It is uniquely defined by two separate spread parameters: the left spread ($\bar{\sigma}$) and the right spread ($\underline{\sigma}$). Compared with a symmetric Gaussian function, this form can assign a different sensitivity to positive and negative deviations around the center. This property is useful for a self-balancing robot because the recovery torque required after a forward fall is not necessarily symmetric with the torque required after a backward fall, especially in the presence of motor saturation, dead-zone, friction, and non-identical acceleration/deceleration dynamics.

Rationale for the Decomposed Structure: The DFS approach *decomposes* each input variable into fuzzy sets, which are then combined to form independent component fuzzy systems. This strategy minimizes the undesirable

cross-learning effect (adhering to the *minimum-disturbance* principle), ensuring that an increase in the number of rules minimally impacts the convergence time. This feature is critical for achieving real-time performance on a microcontroller. Furthermore, the Simplified DFS (SDFS) significantly reduces the total number of component fuzzy systems (from N_s^n to N_s).

3.2 Input normalization and constraints

At every sampling instant t_k:

1. Noise filtering and bias compensation are performed (using a Kalman filter).
2. Input variables i_i are normalized to the predefined range $[i_i^{min}, i_i^{max}]$ and bounded (clipped) according to:

$$\tilde{i}_i = \text{clip}\left(\frac{i_i - i_i^{min}}{i_i^{max} - i_i^{min}}, 0, 1\right). \tag{5}$$

3.3 Fuzzy decomposition and membership functions

Consider a branch with n input variables, where each variable is partitioned into N_s fuzzy sets. Utilizing the SDFS architecture, N_s independent component fuzzy systems are established by grouping the 'same-rank' fuzzy sets across all input variables. The asymmetric Gaussian function is employed for each membership function:

$$\mu_{ijk}(\tilde{i}_i) = \begin{cases} \exp\left(-\frac{(\tilde{i}_i - c_{ijk})^2}{2(\bar{\sigma}_{ijk})^2}\right), & \text{if } \tilde{i}_i < c_{ijk} \\ 1, & \text{if } \tilde{i}_i = c_{ijk} \\ \exp\left(-\frac{(\tilde{i}_i - c_{ijk})^2}{2(\underline{\sigma}_{ijk})^2}\right), & \text{if } \tilde{i}_i > c_{ijk} \end{cases} \tag{6}$$

Here, c_{ijk} (center), $\bar{\sigma}_{ijk}$ (left spread), and $\underline{\sigma}_{ijk}$ (right spread) denote the parameters for the k-th fuzzy set related to the i-th input variable. In Fig. 3, the three fuzzy sets assigned to each normalized input correspond to negative/small, zero/medium, and positive/large operating regions. The same label x_1 in the preliminary figure has been corrected so that the horizontal axes are consistently denoted by the corresponding normalized inputs $\tilde{i}_1$ and $\tilde{i}_2$. Three membership functions are selected as a compromise between approximation capability and embedded computational cost: increasing the number of sets would improve local resolution but would also increase the number of firing-strength calculations and online parameter updates.

The firing strength in this Zero-Order TSK model is calculated as the fuzzy product (t-norm):

$$f_{jk} = \prod_{i=1}^{n} \mu_{ijk}(\tilde{i}_i). \tag{7}$$

This firing strength is then normalized:

$$\bar{f}_{jk} = \frac{f_{jk}}{\sum_{l=1}^{N_s} f_l}. \tag{8}$$

3.4 Inference and defuzzification (zero-order TSK)

Each Rule k possesses a singleton consequent, w_k, referred to as the 'rule weight'. The output u_{DFLC} of the respective DFLC branch is calculated using the center-average defuzzification approach:

$$u_{DFLC} = \frac{\sum_{k=1}^{N_s} \bar{f}_{jk} w_{jk}}{\sum_{k=1}^{N_s} \bar{f}_{jk}}, \tag{9}$$

where the rule weights, denoted w_{jk}, represent the singleton consequents in the Zero-Order TSK fuzzy inference system. These weights are the most critical parameters that directly shape the control surface.

Initially, the weights for both the Tilt-DFLC ($\mathbf{W}^{(\theta)}$) and Wheel-DFLC ($\mathbf{W}^{(x)}$) branches are set to small random values near zero to avoid system saturation upon startup, while ensuring sufficient variance for the initial learning phase.

$$w_{jk}(0) = \text{rand}(\pm\epsilon), \quad \epsilon \ll 1. \tag{10}$$

3.5 Online parameter adaptation

The tracking error is defined as $e(t) = y_{ref}(t) - y(t)$. The quadratic loss function is $E(t) = \frac{1}{2}e^2(t)$. The parameters w_{jk}, c_{ijk}, $\bar{\sigma}_{ijk}$, and $\underline{\sigma}_{jk}$ are updated using the gradient descent algorithm based on the normalized rule firing strength $\bar{f}_k$, yielding the following adaptation laws:

(a) Rule Weight Update: The weight update law remains unchanged:

$$w_{jk}(t+1) = w_{jk}(t) - \frac{1}{2}\eta_w e(t)\bar{f}_{jk}. \tag{11}$$

(b) Gaussian Center Update The center update law is modified to account for the asymmetric spread parameters:

$$c_{ijk}(t+1) = c_{ijk}(t) - \frac{1}{2}\eta_c e(t)\bar{f}_{jk}(w_{jk} - u_{DFLC}) \times \begin{cases} \frac{(\tilde{i}_j - c_{ijk})}{(\bar{\sigma}_{ijk})^2}, & \text{if } \tilde{i}_j < c_{ijk} \\ 0, & \text{if } \tilde{i}_j = c_{ijk} \\ \frac{(\tilde{i}_j - c_{ijk})}{(\underline{\sigma}_{ijk})^2}, & \text{if } \tilde{i}_j > c_{ijk} \end{cases} \tag{12}$$

(c) Gaussian Spread Updates: The single spread update is replaced by two separate laws for the left ($\bar{\sigma}_{ijk}$) and right ($\underline{\sigma}_{ijk}$) spreads, executed conditionally.

Left Spread Update ($\bar{\sigma}_{ijk}$) Executed only if $\tilde{i}_j < c_{ijk}$:

$$\bar{\sigma}_{ijk}(t+1) = \bar{\sigma}_{ijk}(t) - \frac{1}{2}\eta_\sigma e(t)\bar{f}_{jk}(w_{jk} - u_{DFLC})\frac{(\tilde{i}_j - c_{ijk})^2}{(\bar{\sigma}_{ijk})^3}. \tag{13}$$

Right Spread Update ($\underline{\sigma}_{ijk}$) Executed only if $\tilde{i}_j > c_{ijk}$:

$$\underline{\sigma}_{ijk}(t+1) = \underline{\sigma}_{ijk}(t) - \frac{1}{2}\eta_\sigma e(t)\bar{f}_{jk}(w_{jk} - u_{DFLC})\frac{(\tilde{i}_j - c_{ijk})^2}{(\underline{\sigma}_{ijk})^3}. \tag{14}$$

The positive learning rates, $\eta_w, \eta_c, \eta_\sigma > 0$, are optimized using the Modified-Jaya algorithm to ensure rapid convergence while preventing parameter oscillation.

3.6 Cyclic implementation algorithm

The control and adaptation process is executed cyclically for each branch following these steps:

- **Step 1:** Read or estimate the branch inputs, namely $(\theta, \dot{\theta})$ for the tilt branch and $(x, \dot{x})$ for the wheel branch. Apply filtering and normalize the inputs to $\tilde{i}_j$.
- **Step 2:** For each rule $k = 1$ to N_s, compute the membership functions $\mu_{jk}(\tilde{i}_j)$ using (6), the firing strength f_k, and the normalized firing strength $\bar{f}_k$.
- **Step 3:** Calculate the DFLC branch output u_{DFLC} using Equation (9). Apply saturation limits based on the motor's power constraints.
- **Step 4:** Calculate the tracking error $e(t)$; update w_k using (11); update c_{jk} using (12); conditionally update $\bar{\sigma}_{jk}$ using (13) (if $\tilde{i}_j < c_{jk}$) and $\underline{\sigma}_{jk}$ using (14) (if $\tilde{i}_j > c_{jk}$).
- **Step 5:** Integrate the two branch outputs to generate the final control signal $u = \alpha u_\theta + (1-\alpha)u_x$, which is then implemented as a PWM signal via the L298N driver.

3.7 Controller parameterization and link to the optimization variables

The parameters of the proposed controller are divided into two groups. The first group contains the online fuzzy parameters, namely the singleton consequents $\mathbf{W}_\theta$ and $\mathbf{W}_x$, the membership centers $\mathbf{C}_\theta$ and $\mathbf{C}_x$, and the left/right spreads $\overline{\mathbf{\Sigma}}_\theta$, $\underline{\mathbf{\Sigma}}_\theta$, $\overline{\mathbf{\Sigma}}_x$, and $\underline{\mathbf{\Sigma}}_x$. These parameters are updated at each sampling instant using (11)–(14). The second group contains the learning-rate vector $\boldsymbol{\eta} = [\eta_w, \eta_c, \eta_\sigma]^T$, which determines the update speed of the consequent, center, and spread parameters. Only $\boldsymbol{\eta}$ is optimized by the Modified-Jaya algorithm. This separation keeps the metaheuristic search low-dimensional and prevents the optimizer from replacing the physically interpretable online adaptation laws.

All comparative controllers are evaluated under the same initial conditions, sampling time, disturbance profile, sensor/process noise levels, and actuator saturation limits. The fixed controller and optimizer settings used in the comparisons are summarized in Table 1. This table is included to ensure that performance differences are attributable to the control architecture and learning-rate optimization rather than to inconsistent test conditions.

Table 1: Fixed settings used for controller implementation and fair comparison.

Item	**Setting**	**Purpose**
Sampling time	$T_s = 0.01\,\mathrm{s}$	Digital implementation
DFLC inputs	$[\theta, \dot{\theta}]^T$ and $[x, \dot{x}]^T$	Tilt and wheel branches
Membership sets per input	$N_s = 3$	Negative, zero, and positive operating regions
Fusion coefficient	$\alpha = 0.7$	Tilt-dominant torque fusion
Consequent initialization	$w_{jk}(0) \in [-10^{-3}, 10^{-3}]$	Avoid startup saturation
Learning-rate bounds	$\eta_w \in [10^{-4}, 10^{-2}]$, $\eta_c \in [10^{-4}, 5 \times 10^{-3}]$, $\eta_\sigma \in [10^{-5}, 10^{-3}]$	Stability-admissible search interval
Disturbance for robustness test	3 Nm impulse at $t = 2\,\mathrm{s}$ for $0.02\,\mathrm{s}$	Recovery assessment
Baseline controllers	T1FNN and FCMAC with the same plant, noise, and disturbance settings	Fair comparison

3.8 Modified-Jaya algorithm for hyperparameter optimization

The robustness and convergence speed of the DFLC adaptation process are critically dependent on the selection of optimal learning rates $\boldsymbol{\eta} = [\eta_w, \eta_c, \eta_\sigma]^T$. Setting these hyperparameters manually often results in sub-optimal performance—a high rate may cause parameter oscillation, while a low rate leads to slow convergence. To address this, the **Modified-Jaya (MJaya)** optimization algorithm is employed to determine the near-optimal values of $\boldsymbol{\eta}$ in an offline or slow-adaptive manner.

The core principle of the Jaya algorithm is "Jaya" (meaning victory in Sanskrit), where the solution attempts to move directly toward the best solution found so far while simultaneously moving away from the worst solution. The algorithm maintains a population of candidate solutions (learning rate sets) and iteratively updates them based on this winning tendency, requiring only the two extreme candidates (best and worst) without needing any algorithm-specific control parameters.

The objective function to be minimized is the system's performance metric over a predefined testing trajectory. To better connect the optimizer with closed-loop performance and stability requirements, the optimization problem is formulated as

$$\min_{\boldsymbol{\eta}} \; J(\boldsymbol{\eta}) = \sum_{k=0}^{N-1} \left(q_\theta |e_{\theta,k}| + q_x |e_{x,k}| + r_u |u_k| + q_d |e_{\theta,k}| \mathbb{1}_{k \geq k_d} \right), \tag{15}$$

$$\text{subject to } \eta_w^{\min} \leq \eta_w \leq \eta_w^{\max},\; \eta_c^{\min} \leq \eta_c \leq \eta_c^{\max},\; \eta_\sigma^{\min} \leq \eta_\sigma \leq \eta_\sigma^{\max},\; 0 < \lambda_{\max}(\Gamma) < \frac{2}{L_J}. \tag{15a}$$

In this formulation, q_θ, q_x, and r_u weight the tilt-tracking error, displacement-tracking error, and control effort, respectively, while the last term penalizes the post-disturbance tilt error after the impulse time k_d. Therefore, the optimization explicitly includes robustness through disturbance rejection, bounded control effort, and admissible learning-rate constraints.

The update equation for the j-th learning rate in the i-th candidate solution, $\eta_{i,j}$, at iteration $k+1$ is calculated based on the previous value $\eta_{i,j}^k$, the best value η_j^{best}, and the worst value η_j^{worst} in the population:

$$\eta_{i,j}^{k+1} = \eta_{i,j}^k + r_{1,j}^k (\eta_j^{\text{best}(d)} - |\eta_{i,j}^k|) - r_{2,j}^k (\eta_j^{\text{worst}(d)} - |\eta_{i,j}^k|), \tag{16}$$

where $r_{1,j}^k$ and $r_{2,j}^k$ are uniformly distributed random numbers in $[0, 1]$. The index $d \in \{1, 2, 3\}$ selects the candidate used in the modified repulsion term. In this study, the second-worst candidate is used with a prescribed probability to avoid excessively large jumps generated by the worst candidate when outliers occur. After the update, each candidate is projected onto the admissible set in (15a). This projection step explicitly connects the optimizer with the stability and actuator constraints.

By iteratively applying Equation (16), the entire population of learning rate sets converges towards the optimum, providing a highly effective mechanism for tuning the critical hyperparameters $\boldsymbol{\eta}$ and thus maximizing the overall control system performance.

3.9 Stability analysis

Because the controller is implemented digitally on a microcontroller, the stability analysis is formulated in discrete time. Let T_s denote the sampling period. The continuous-time state-space model adopted from the standard two-wheeled self-balancing robot formulation in [31], is discretized as

$$\mathbf{x}_{k+1} = A_d\mathbf{x}_k + B_d u_k + \mathbf{d}_k, \qquad \mathbf{y}_k = C\mathbf{x}_k, \tag{17}$$

where $\mathbf{x}_k = [x_k, \dot{x}_k, \theta_k, \dot{\theta}_k]^T$, u_k is the motor torque/PWM-equivalent input, and $\mathbf{d}_k$ represents bounded discretization error, unmodeled dynamics, friction, and external disturbances. The matrices A_d and B_d are obtained from the continuous model in Section 2 by zero-order-hold discretization, i.e., $A_d = e^{AT_s}$ and $B_d = \int_0^{T_s} e^{A\tau}B\,d\tau$.

Define the two-output tracking vector

$$\mathbf{e}_k = \begin{bmatrix} e_{x,k} \\ e_{\theta,k} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{ref,k}} - x_k \\ \theta_{\mathrm{ref,k}} - \theta_k \end{bmatrix}. \tag{18}$$

The fuzzy controller model can be written explicitly as

$$u_k = \alpha\,\boldsymbol{\psi}_\theta^T(\mathbf{i}_k^{(\theta)})\mathbf{W}_{\theta,k} + (1-\alpha)\,\boldsymbol{\psi}_x^T(\mathbf{i}_k^{(x)})\mathbf{W}_{x,k}, \tag{19}$$

where $\boldsymbol{\psi}_\theta$ and $\boldsymbol{\psi}_x$ are the normalized firing-strength vectors generated by the asymmetric Gaussian membership functions. Since the inputs are normalized and clipped and the spreads satisfy $\sigma_{ijk} \geq \sigma_{\min}$, the firing strengths and their gradients are bounded. Therefore, there exist finite constants $\psi_{\max}$ and $g_{\max}$ such that $\|\boldsymbol{\psi}\| \leq \psi_{\max}$ and $\|\nabla_{\boldsymbol{\phi}} u_k\| \leq g_{\max}$.

Consider the Lyapunov candidate

$$V_k = \frac{1}{2}\mathbf{e}_k^T Q\mathbf{e}_k + \frac{1}{2}\tilde{\boldsymbol{\phi}}_k^T\Gamma^{-1}\tilde{\boldsymbol{\phi}}_k, \qquad Q = Q^T \succ 0, \quad \Gamma = \mathrm{diag}(\eta_w I, \eta_c I, \eta_\sigma I) \succ 0, \tag{20}$$

where $\tilde{\boldsymbol{\phi}}_k$ denotes the fuzzy-parameter estimation error. The update laws (11)–(14) can be represented compactly as

$$\boldsymbol{\phi}_{k+1} = \Pi_\Omega\left[\boldsymbol{\phi}_k - \Gamma\nabla_{\boldsymbol{\phi}} J_k\right], \tag{21}$$

where $\Pi_\Omega[\cdot]$ is a projection operator onto the compact admissible set Ω defined by bounded consequents, bounded centers, positive spreads, and the learning-rate constraints in (15a). This projection prevents parameter drift during learning.

Using the standard descent lemma for a differentiable cost function with Lipschitz constant L_J, the change in the instantaneous cost satisfies

$$J_{k+1} - J_k \leq -\left(\lambda_{\min}(\Gamma) - \frac{L_J}{2}\lambda_{\max}^2(\Gamma)\right)\|\nabla_{\boldsymbol{\phi}} J_k\|^2 + \rho\|\mathbf{d}_k\|^2, \tag{22}$$

where $\rho > 0$ depends on the plant and output matrices. Therefore, if the Modified-Jaya search is constrained by

$$0 < \lambda_{\max}(\Gamma) < \frac{2}{L_J}, \tag{23}$$

then the nominal learning dynamics are non-increasing in the Lyapunov sense. In the presence of bounded disturbances $\|\mathbf{d}_k\| \leq d_{\max}$, (22) implies practical stability: $\mathbf{e}_k$ and $\boldsymbol{\phi}_k$ remain uniformly ultimately bounded, and the ultimate bound decreases with smaller disturbance magnitude and smoother control action.

The stability-oriented role of MJaya is thus not to prove asymptotic stability for arbitrary learning rates, but to search inside the admissible set (15a) and (23) for learning rates that minimize the finite-horizon control objective. This explains why the optimization problem is constrained and why the reported controller remains stable during learning mode.

4 Simulation and experimental results

This section evaluates the proposed MJaya-DFLC on the self-balancing two-wheel robot introduced in Section 2, using both numerical simulations and hardware experiments.

The initial conditions used in simulation are explicitly set as follows: $\mathbf{x}_0 = [0, 0, 8^\circ, 0]^T$ for Case 1 and $\mathbf{x}_0 = [0, 0, 15^\circ, 0]^T$ for Case 2. The sampling time is $T_s = 0.01\,\mathrm{s}$ in simulation and in the embedded implementation unless

otherwise stated. The nominal plant parameters are those listed in Table 2; they were obtained from direct weighing and geometric measurement of the prototype, while the damping coefficients were identified from free-response experiments by fitting the decay envelope of the wheel and body motions. The continuous-time model was then validated by comparing the open-loop and low-gain closed-loop responses before being used in the controller-design simulations.

For reproducibility of the controller configuration, each DFLC branch uses two inputs and three asymmetric Gaussian membership functions per input, which yields $N_s = 3$ same-rank component fuzzy subsystems in the SDFS implementation. The normalization ranges are chosen as $\theta \in [-20^\circ, 20^\circ]$, $\dot{\theta} \in [-150, 150]^\circ/\mathrm{s}$, $x \in [-0.20, 0.20]\,\mathrm{m}$, and $\dot{x} \in [-0.60, 0.60]\,\mathrm{m/s}$. The fusion coefficient is set to $\alpha = 0.7$, the initial consequent weights are sampled in $[-10^{-3}, 10^{-3}]$, and the Modified-Jaya search bounds are $\eta_w \in [10^{-4}, 10^{-2}]$, $\eta_c \in [10^{-4}, 5\times10^{-3}]$, and $\eta_\sigma \in [10^{-5}, 10^{-3}]$.

Three controllers are compared: a type-1 fuzzy neural network (T1FNN) controller, a fuzzy CMAC controller (FCMAC), and the proposed MJaya-DFLC with decomposed fuzzy structure and Modified-Jaya–tuned learning rates. The T1FNN and FCMAC serve as strong learning-based baselines for nonlinear control and function approximation In contrast, the MJaya-DFLC integrates a decomposed fuzzy architecture, asymmetric Gaussian membership functions and online gradient adaptation, utilizing the Modified-Jaya algorithm to tune its learning rates. The physical parameters of the two-wheeled self-balancing robot are given in Table 2. The RMSE comparison of the tilt angle (in degrees) is given in Table 3.

Table 2: Physical parameters of the two-wheeled self-balancing robot.

Parameter	**Value**	**Unit**
Pendulum/body mass (M_p)	0.52	kg
Chassis/train mass (M_c)	1.80	kg
Distance from wheel axle to body CoG (l)	0.25	m
Wheel radius (r)	0.04	m
Distance between wheels (D)	0.14	m
Gravitational acceleration (g)	9.80	$\mathrm{m/s^2}$
Wheel mass (M_w)	0.10	kg
Moment of inertia of body ($J_p = M_p l^2$)	0.0325	$\mathrm{kg{\cdot}m^2}$
Moment of inertia of wheel ($J_w = M_w r^2/2$)	8.0×10^{-5}	$\mathrm{kg{\cdot}m^2}$

The closed-loop configuration of the control system is illustrated in Fig. 5. The two-wheel robot is modeled as an IPOC system, with its states measured and passed through filtering, normalization, and saturation blocks as described in Section 3. The proposed MJaya-DFLC block is decomposed into two branches: Tilt-DFLC and Wheel-DFLC. The former regulates the body tilt angle using the state variables $[\theta, \dot{\theta}]$, whereas the latter compensates for wheel dynamics utilizing translational velocity and/or acceleration. Although the plant has one actuator input, this single torque simultaneously influences both θ and x through the coupled dynamics in (1)–(3). The two DFLC branches do not generate two independent motor commands; instead, they compute two complementary torque components that are combined into one physically admissible input u_k.

In simulation, the nonlinear plant is subjected to zero-mean Gaussian process noise with standard deviation $\sigma_{\text{process}} = 0.05$ (rad or rad/s), zero-mean Gaussian sensor noise with standard deviation $\sigma_{\text{sensor}} = 0.03$, and an impulse disturbance torque of 3 Nm applied at $t = 2\,\mathrm{s}$ for 0.02 s. Two initial tilt conditions are investigated: Case 1, with initial tilt angle $\theta_0 = 8^\circ$ (Figs. 6–10), and Case 2, with initial tilt angle $\theta_0 = 15^\circ$ (Figs. 11–15).

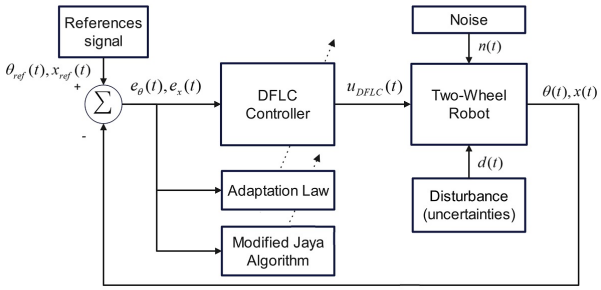

Figure 5: The closed-loop configuration of the control system.

4.1 Case 1: Initial tilt angle $\theta_0 = 8°$

In the first scenario, the robot is released from a moderate tilt of 8°. Figure 6 shows that all three controllers eventually stabilize the robot, but their transient behaviours differ. The MJaya-DFLC yields the shortest settling time and the smallest overshoot. After the disturbance at $t = 2$ s, the MJaya-DFLC response exhibits only a single, smooth deviation before returning quickly to the upright region, whereas both T1FNN and CMAC display larger excursions and longer oscillatory tails. The control signals in Fig. 7 indicate that MJaya-DFLC achieves this improved tilt response without excessive actuation. The torque generated by MJaya-DFLC is more compact and smoother over time, with visibly reduced peaks around the initial transient and the impulse instant. The T1FNN and CMAC commands show higher spikes and more abrupt changes, suggesting less efficient use of the available actuation. The error plot in Fig. 8 provides a concise quantitative comparison. Under MJaya-DFLC, the error magnitude decays rapidly and remains within a narrow band after the initial transient, with only a modest bump at the disturbance time. Both T1FNN and CMAC exhibit larger peak errors and a slower return to small values. The trajectories of the learning rates, optimized via the Modified-Jaya algorithm, are depicted in Fig. 10. Taken together, Figs. 6–10 show that, for $\theta_0 = 8°$, the proposed controller achieves faster stabilization and smaller steady fluctuations with lower control effort than the two baselines.

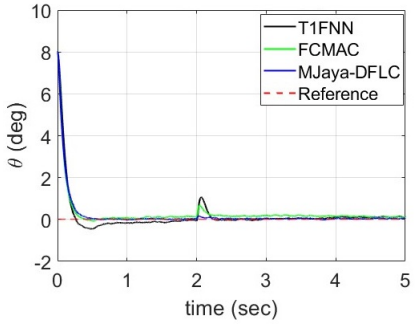

Figure 6: Evolution of the tilt angle (θ in degrees) for simulation Case 1.

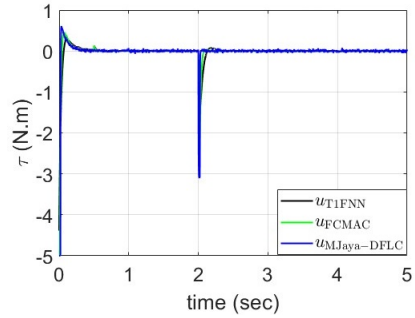

Figure 7: The control signal for simulation Case 1.

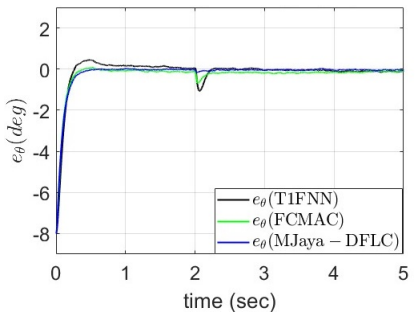

Figure 8: The tracking error of the tilt angle for simulation Case 1.

The corresponding wheel displacement response $x(t)$ is shown in Fig. 9. This response shows that the wheel branch suppresses translational drift while the tilt branch stabilizes the body angle. For self-balancing tests, the desired displacement is constant; hence the objective is bounded drift suppression rather than aggressive trajectory following.

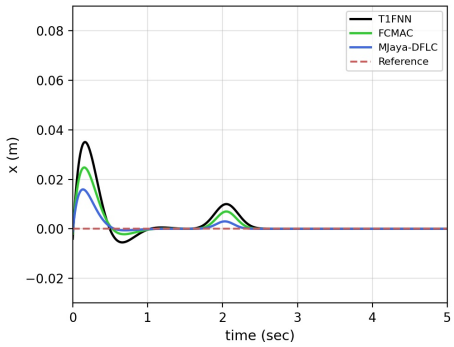

Figure 9: Wheel displacement response $x(t)$ for simulation Case 1.

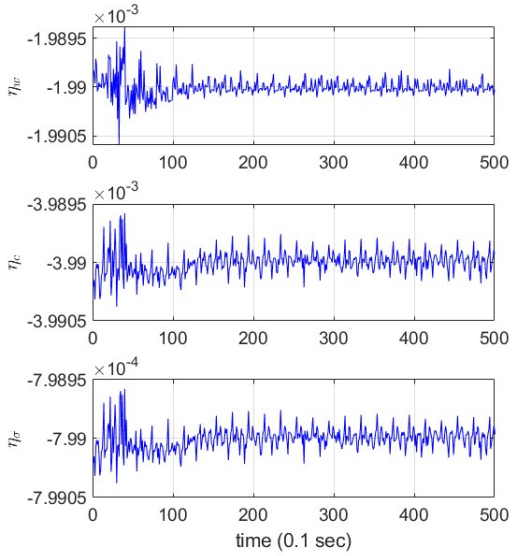

Figure 10: The trajectories of the learning rates using M-Jaya algorithm for simulation Case 1.

4.2 Case 2: Initial tilt angle $\theta_0 = 15°$

The second scenario is more challenging, with the robot starting from a larger tilt of 15°. The same noise and disturbance settings are used as in Case 1. Figure 10 indicates that the MJaya-DFLC maintains its advantage as the operating condition becomes more severe. The proposed controller still brings the tilt angle back to the upright region in a shorter time and with less overshoot than T1FNN and CMAC. The impulse at $t = 2\,\mathrm{s}$ causes a pronounced deviation for all three controllers, but the MJaya-DFLC trajectory re-enters the small-angle neighbourhood more quickly and with fewer residual oscillations. As shown in Fig. 11, all controllers require larger control actions than in Case 1, which is expected given the larger initial deviation. Nevertheless, the MJaya-DFLC torque profile remains comparatively smooth, with reduced duration and amplitude of saturation-like peaks. The T1FNN and CMAC efforts are more aggressive, featuring higher and more frequent peaks. This behaviour is important for practical implementation, as it directly relates to motor stress and energy consumption. The error trajectories in Fig. 12 corroborate the previous observations. The MJaya-DFLC produces the smallest peak error and the fastest error decay, both after the initial release and after the disturbance. T1FNN and CMAC show larger error excursions and slower convergence. The trajectories of the learning rates, optimized via the Modified-Jaya algorithm, are depicted in Fig. 15. Overall, Figs. 11–15 demonstrate

that the proposed controller preserves its performance advantage for larger tilt angles, confirming its robustness to operating-point variations and stronger nonlinear effects.

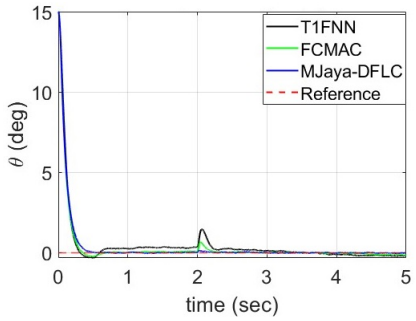

Figure 11: Evolution of the tilt angle (θ in degrees) for simulation Case 2.

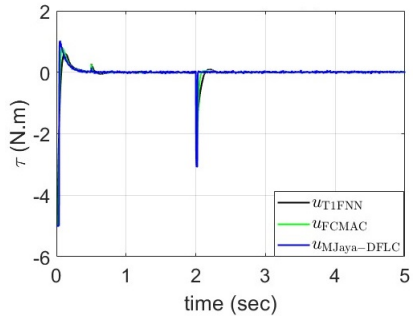

Figure 12: The control signal for simulation Case 2.

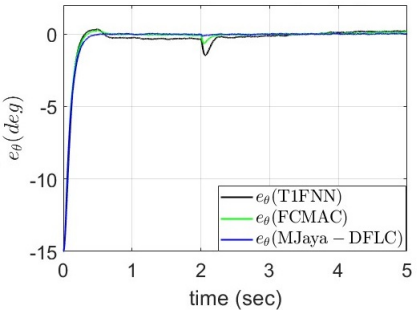

Figure 13: The tracking error of the tilt angle for simulation Case 2.

The wheel displacement response for the larger initial tilt is shown in Fig. 14. The increased initial angle produces a larger transient displacement, but the proposed controller still bounds the drift and recovers the upright posture without sustained oscillation.

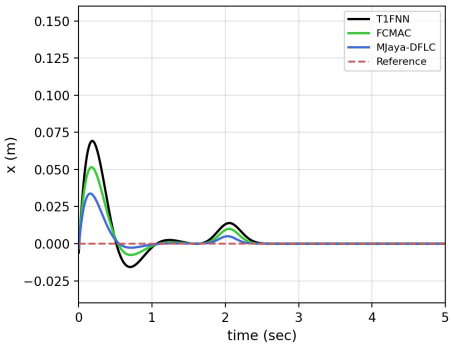

Figure 14: Wheel displacement response $x(t)$ for simulation Case 2.

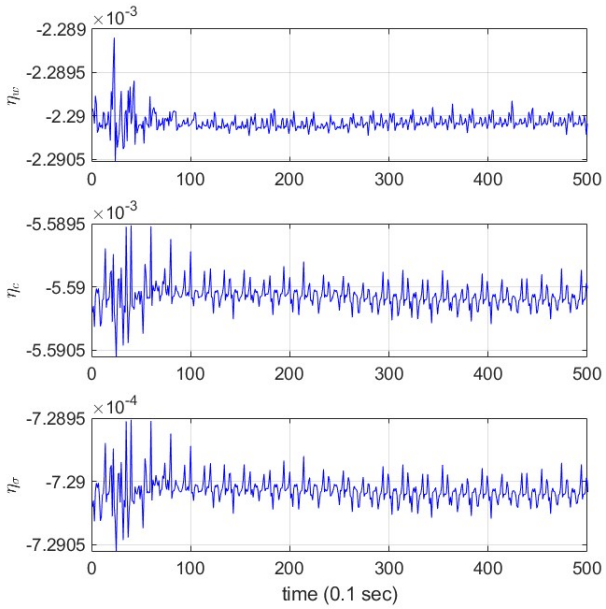

Figure 15: The trajectories of the learning rates using M-Jaya algorithm for simulation Case 2.

4.3 Experimental validation

The proposed controller is further validated on a physical prototype comprising an Arduino Mega 2560, an L298N motor driver, DC motors, and an MPU6050 IMU. In the experiments, the robot is released from an initial tilt similar to the simulated cases, and two disturbances are applied: continuous airflow from a fan and a manual hand push at about $t = 12\,\mathrm{s}$. The experimental setup of the two-wheeled self-balancing robot platform is shown in Fig. 14. Figure 15 shows that the tilt angle returns quickly to a small neighbourhood of the upright position and remains there despite the disturbances. The control effort in Fig. 16 is consistent with the simulated MJaya-DFLC profile: torque remains bounded and free of sustained chattering. The tracking error in this case stays small during steady operation and exhibits only short-lived spikes at the disturbance instants. It should be noted that the experimental prototype used in this study is equipped with an MPU6050 IMU but does not have an external motion-capture camera, or another absolute-position sensor. Therefore, the experimental validation reports directly measured tilt angle and control effort, whereas the translational displacement $x(t)$ is reported for the simulation cases where the plant state is available. No unmeasured experimental position trajectory is reconstructed from the IMU alone, because double integration of acceleration would produce unacceptable drift. Adding wheel encoders or vision-based localization is identified as a future extension for full experimental position tracking. From Fig. 15, the steady balancing fluctuation is mainly confined within about $\pm 3^\circ$, while the manual push at approximately $t = 12$ s produces a short peak of about 12°. The

control signal in Fig. 16 correspondingly reaches a short peak of about 15 in normalized control units and then returns to its low-amplitude operating range, indicating bounded actuation without sustained chattering.

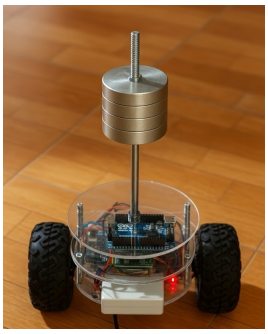

Figure 16: Experimental setup of the two-wheeled self-balancing robot platform.

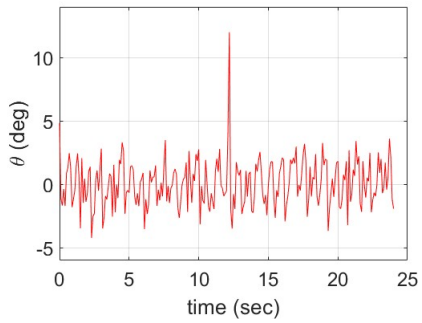

Figure 17: Evolution of the tilt angle (θ in degrees) for experimental validation.

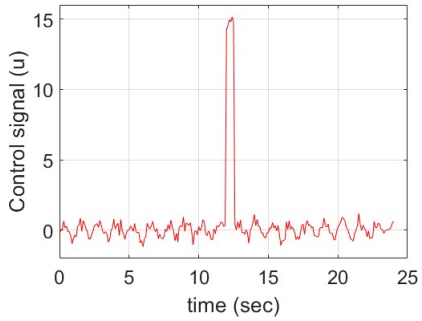

Figure 18: The control signal for experimental validation.

Table 3: RMSE comparison of the tilt angle (in degrees) for Case 1 and Case 2.

Controller	**Computation Time (ms)**	**Case 1**	**Case 2**
T1FNN	0.013	0.9221	1.7599
FCMAC	0.017	0.8525	1.7238
MJaya-DFLC (*proposed controller*)	0.021	0.8391	1.6969

In addition to RMSE, multi-criteria transient and robustness indices are reported in Table 4. The settling time

T_s is evaluated using a small-angle neighborhood around the upright equilibrium after the initial release, the undershoot/overshoot index denotes the largest excursion around the zero reference after the first zero crossing, and the disturbance peak and recovery time are measured after the impulse disturbance applied at $t = 2$ s. The values were extracted from the plotted time histories in Figs. 6, 8, 11, and 13.

Table 4: Multi-criteria transient and robustness comparison extracted from the simulation responses.

Controller	**Case**	T_s **(s)**	**Undershoot/overshoot (deg)**	**Peak after impulse (deg)**	**RMSE (deg)**
T1FNN	1	≈ 0.55	≈ 0.35	≈ 1.10	0.9221
FCMAC	1	≈ 0.35	≈ 0.12	≈ 0.85	0.8525
MJaya-DFLC	1	≈ 0.25	< 0.10	≈ 0.20	0.8391
T1FNN	2	≈ 0.75	≈ 0.25	≈ 1.45	1.7599
FCMAC	2	≈ 0.55	≈ 0.18	≈ 0.95	1.7238
MJaya-DFLC	2	≈ 0.38	< 0.10	≈ 0.35	1.6969

The additional indices confirm the trend observed from the RMSE values: the proposed MJaya-DFLC gives the shortest settling time in both initial-angle cases and substantially reduces the angular deviation caused by the impulse disturbance. The improvement is particularly visible in Case 2, where the larger initial angle activates the nonlinear region of the robot dynamics.

4.4 Computational complexity and implementation cost

For each DFLC branch with n inputs and N_s fuzzy sets per input under the SDFS structure, the dominant operations at each sampling instant are the evaluation of nN_s membership grades, N_s firing strengths, N_s normalized firing strengths, and $O(nN_s)$ gradient terms for parameter adaptation. Therefore, the online complexity is $O(nN_s)$ per branch. Because the proposed controller uses two branches, the overall online complexity is $O(n_\theta N_s + n_x N_s)$, which remains linear in the number of fuzzy sets. A conventional full-grid FLC with n inputs and N_s sets per input would require $O(N_s^n)$ rule evaluations; thus, the decomposed structure substantially reduces the computational load. The Modified-Jaya optimization is performed offline or in a slow-adaptive design loop. If P candidates, K iterations, and N simulation samples are used, its design-stage complexity is $O(PKN)$ times the cost of one closed-loop simulation. This cost does not affect the real-time sampling burden after the optimized learning rates are fixed.

5 Conclusions

This paper has introduced a Modified-Jaya-optimized decomposed fuzzy controller (MJaya-DFLC) for self-balancing two-wheel robots. The controller employs two parallel DFLC branches—one regulating the body tilt angle and the other compensating wheel dynamics—implemented with asymmetric Gaussian membership functions and online gradient-descent parameter adaptation. The learning rates of these adaptive laws are tuned by a Modified-Jaya algorithm operating on a low-dimensional search space, thus avoiding manual gain selection.

A Lyapunov-based analysis shows that, for learning rates constrained to an admissible region, the closed-loop error remains bounded and converges to a small neighbourhood of the origin in the presence of bounded disturbances and uncertainties. Simulation studies with Gaussian process and sensor noise and a 3 Nm impulse disturbance at $t = 2$ s demonstrate that the proposed MJaya-DFLC consistently outperforms two strong learning-based baselines T1FNN controller and a FCMAC controller in terms of: faster tilt-angle stabilization, reduced overshoot and oscillations, lower and smoother control effort, smaller and more rapidly decaying tracking errors, for both moderate (8°) and large (15°) initial tilts. Experimental validation on a low-cost Arduino-based platform further confirms that MJaya-DFLC can be implemented in real time and maintains robust balancing under fan-induced airflow and manual pushes.

These results suggest that combining a decomposed fuzzy architecture with a parameter-free metaheuristic such as Modified-Jaya is an effective strategy for improving the robustness and performance of learning-based controllers on underactuated, nonlinear systems. Future work will extend this framework to full trajectory tracking and more complex manoeuvres, investigate multi-objective formulations of the Modified-Jaya optimization (e.g., jointly minimizing settling time, energy consumption, and control smoothness), and explore integration with additional sensory modalities such as wheel encoders and vision for advanced autonomous navigation.

Acknowledgement

This work was supported by Lac Hong University, Vietnam under grant 2025-L-0419-0039.

References

[1] H. Ahmadi Jeyed, A. Ghaffari, *A nonlinear optimal control based on the SDRE technique for the two-wheeled self-balancing robot*, Australian Journal of Mechanical Engineering, **20**(3) (2022), 722-730. https://doi.org/10.1080/14484846.2020.1745733

[2] A. Aldhalemi, A. Chlaihawi, A. Al-Ghanimi, *Design and implementation of a remotely controlled two-wheel self-balancing robot*, IOP Conference Series: Materials Science and Engineering, **1067**(1) (2021), 012132. https://doi.org/10.1088/1757-899X/1067/1/012132

[3] G. Andonovski, D. Leite, R. E. Precup, F. Gomide, M. Pratama, I. Škrjanc, *Advancements in data-driven evolving fuzzy and neuro-fuzzy control: A comprehensive survey*, Applied Soft Computing, **186**(Part A) (2026), 114058. https://doi.org/10.1016/j.asoc.2025.114058

[4] R. G. Aragón, J. Medina, E. Ramírez-Poussa, *Independent subcontexts and blocks of concept lattices. Definitions and relationships to decompose fuzzy contexts*, Fuzzy Sets and Systems, **509** (2025), 109345. https://doi.org/10.1016/j.fss.2025.109345

[5] B. Bekhiti, R. Al-Sabur, M. Roudane, J. A. Younis, A. N. Sharkawy, *Intelligent neuro-fuzzy adaptive MIMO control for a self-balancing two wheeled autonomous robot via recursive resolution of the matrix diophantine equation*, Discover Robotics, **1** (2025), 1-28. https://doi.org/10.1007/s44430-025-00011-3

[6] J. Chen, N. He, Z. Xu, M. Dou, L. He, *Design and implementation of a novel two-wheeled composite self-balancing robot for stationary operations in unknown terrain*, IEEE Access, **13** (2025), 86032-86045. https://doi.org/10.1109/ACCESS.2025.3569325

[7] X. Feng, S. Liu, Q. Yuan, J. Xiao, D. Zhao, *Research on wheel-legged robot based on LQR and ADRC*, Scientific Reports, **13** (2023), 15122. https://doi.org/10.1038/s41598-023-41462-1

[8] Z. Feng, D. Zhu, H. Guo, J. Xue, C. Zhou, *A Jaya algorithm based on self-adaptive method for parameters identification of photovoltaic cell and module*, Cluster Computing, **28** (2025), 145. https://doi.org/10.1007/s10586-024-04877-7

[9] H. Hashemi, R. Ezzati, N. M. Mikaeilvand, M. Nazari, *Fuzzy time-fractional advection-dispersion and Navier-Stokes equations: A comprehensive approach*, Iranian Journal of Fuzzy Systems, **21**(5) (2024), 105-119. https://doi.org/10.22111/ijfs.2024.48569.8574

[10] A. Hentout, A. Maoudj, A. Kouider, *Shortest path planning and efficient fuzzy logic control of mobile robots in indoor static and dynamic environments*, Romanian Journal of Information Science and Technology, **27**(1) (2024), 21-36. https://doi.org/10.59277/ROMJIST.2024.1.02

[11] A. Hess, O. Medina, *A self-balancing robot with redundant actuators for obstacle avoidance*, IEEE Access, **13** (2025), 134376–134384. https://doi.org/10.1109/ACCESS.2025.3590812

[12] C. F. Juang, C. Y. Chou, C. T. Lin, *Navigation of a fuzzy-controlled wheeled robot through the combination of expert knowledge and data-driven multiobjective evolutionary learning*, IEEE Transactions on Cybernetics, **52**(8) (2022), 7388-7401. https://doi.org/10.1109/TCYB.2020.3041269

[13] L. Kakinada, K. Singh, *WCA optimized fuzzy PD controller for stabilizing the two wheel self-balancing robot*, in Proc. 2021 Asian Conference on Innovation in Technology (ASIANCON), IEEE, (2021), 1-6. https://doi.org/10.1109/ASIANCON51346.2021.9544711

[14] T. L. Le, N. H. Hung, *Trajectory tracking control of a line-following quadcopter using multilayer type-2 fuzzy Petri nets controller*, Neural Computing and Applications, **36** (2024), 13617-13627. https://doi.org/10.1007/s00521-024-09750-7

[15] T. A. Mai, T. S. Dang, H. C. Ta, S. P. Ho, *Comprehensive optimal fuzzy control for a two-wheeled balancing mobile robot*, Journal of Ambient Intelligence and Humanized Computing, **14** (2023), 9451-9467. https://doi.org/10.1007/s12652-023-04613-w

[16] U. Mishra, D. Gupta, A. Sarkar, B. B. Hazarika, *A hybrid approach for plant leaf detection using ResNet50-intuitionistic fuzzy RVFL (ResNet50-IFRVFLC) classifier*, Computers and Electrical Engineering, **123**(Part B) (2025), 110135. https://doi.org/10.1016/j.compeleceng.2025.110135

[17] S. Moslem, F. K. Gündoğdu, S. Saylam, F. Pilla, *A hybrid decomposed fuzzy multi-criteria decision-making model for optimizing parcel lockers location in the last-mile delivery landscape*, Applied Soft Computing, **154** (2024), 111321. https://doi.org/10.1016/j.asoc.2024.111321

[18] K. D. Papadimitriou, N. Murasovs, M. E. Giannaccini, S. S. Aphale, *Genetic algorithm-based control of a two-wheeled self-balancing robot*, Journal of Intelligent and Robotic Systems, **111** (2025), 34. https://doi.org/10.1007/s10846-025-02236-1

[19] Y. Peng, S. Sun, S. He, J. Zou, Y. Liu, Y. Xia, *A fractional-order JAYA algorithm with memory effect for solving global optimization problem*, Expert Systems with Applications, **270** (2025), 126539. https://doi.org/10.1016/j.eswa.2025.126539

[20] D. H. Pham, M. K. Pham, M. L. Nguyen, T. V. A. Nguyen, *Type-2 fuzzy control for stabilizing two-wheeled mobile robot under parameter uncertainties*, Engineering Research Express, **7**(4) (2025), 045314. https://doi.org/10.1088/2631-8695/ae0d2e

[21] R. E. Precup, C. A. Bojan Dragos, K. Gao, S. Cui, *Problem setting for trajectory planning and cruise control of a connected autonomous electric bus in intersection scenarios with human-driven vehicles to optimize energy, comfort and tracking*, Romanian Journal of Information Science and Technology, **28**(3) (2025), 299-312. https://doi.org/10.59277/ROMJIST.2025.3.05

[22] R. E. Precup, R. C. Roman, E. L. Hedrea, A. I. Szedlak-Stinean, I. A. Zamfirache, *Classical and modern optimization techniques applied to control and modeling*, CRC Press, Boca Raton, FL, 2025. https://doi.org/10.1201/9781003488279

[23] H. Rafiei, A. Salehi, F. Baghbani, M. R. Akbarzadeh-T., *Evolutionary fuzzy control of pandemics with vaccination and isolation constraints*, Iranian Journal of Fuzzy Systems, **22**(5) (2025), 69-96. https://doi.org/10.22111/ijfs.2025.50369.8889

[24] H. Ren, C. Zhou, *Control system of two-wheel self-balancing vehicle*, Journal of Shanghai Jiaotong University (Science), **26** (2021), 713-721. https://doi.org/10.1007/s12204-021-2361-x

[25] J. R. Rivera-Ruiz, R. Rojas-Galván, J. R. García-Martínez, E. E. Cruz-Miguel, O. A. Barra-Vázquez, J. E. Escalante-Martínez, *Performance evaluation of metaheuristics for LQR controller optimization: A two-wheel balancing robot case study*, IEEE Access, **13** (2025), 97870-97888. https://doi.org/10.1109/ACCESS.2025.3576343

[26] C. Savithri, R. Roopesh, N. Lavanya Devi, P. Shanthakumar, P. Thirumurugan, *Self-balancing robot using Arduino and PID controller*, in Computer Vision and Machine Intelligence Paradigms for SDGs, Lecture Notes in Electrical Engineering, **967**, Springer, (2023), 201-208. https://doi.org/10.1007/978-981-19-7169-3_18

[27] N. Tüysüz, C. Kahraman, *A novel decomposed Z-fuzzy TOPSIS method with functional and dysfunctional judgments: An application to transfer center location selection*, Engineering Applications of Artificial Intelligence, **127**(Part A) (2024), 107221. https://doi.org/10.1016/j.engappai.2023.107221

[28] B. Vignesh, D. Jose, P. Nirmal Kumar, *Implementation of indoor navigation control for two-wheeled self-balancing robot*, in Smart Sensors Measurement and Instrumentation, Lecture Notes in Electrical Engineering, **957**, Springer, (2023), 461-474. https://doi.org/10.1007/978-981-19-6913-3_31

[29] Y. Xia, J. Wang, Z. Zhang, D. Wei, Z. Cao, Z. Li, *A wind speed point-interval fuzzy forecasting system based on data decomposition and multiobjective optimizer*, Applied Soft Computing, **165** (2024), 112084. https://doi.org/10.1016/j.asoc.2024.112084

[30] S. Yang, H. Pang, L. Liu, L. Zheng, L. Wang, M. Liu, *A RISE-based asymptotic prescribed performance trajectory tracking control of two-wheeled self-balancing mobile robot*, Nonlinear Dynamics, **112** (2024), 15327-15348. https://doi.org/10.1007/s11071-024-09569-w

[31] J. Zhang, T. Zhao, B. Guo, S. Dian, *Fuzzy fractional-order PID control for two-wheeled self-balancing robots on inclined road surface*, Systems Science and Control Engineering, **10**(1) (2022), 289-299. https://doi.org/10.1080/21642583.2021.2001768

[32] Z. Zheng, Q. Cai, J. Wang, X. Xu, M. Cao, H. Yu, J. Li, J. Meng, G. Lu, *CapsuleBot: A novel hybrid aerial-ground bi-copter robot with two actuated-wheel-rotors*, IEEE Robotics and Automation Letters, **10**(1) (2025), 120-127. https://doi.org/10.1109/LRA.2024.3518103

[33] W. Zhou, J. Wu, Z. Yin, X. Liu, *LQR control of two-wheeled self-balancing vehicle based on ARX model*, in Proc. 2024 6th International Conference on Electronic Engineering and Informatics (EEI), IEEE, (2024), 816-820. https://doi.org/10.1109/EEI63073.2024.10696340