Unsupervised learning in the bio-inspired online optimization for robust tuning of brushless motor control under dynamic constraints

Miguel Gabriel Villarreal-Cervantes[a,*], Alam Gabriel Rojas López[a], Alejandro Rodríguez-Molina[b,*], Omar Serrano-Pérez[c], Ramón Silva Ortigoza[a], Roberto Castro Medina[d]

[a] *Centro de Innovación y Desarrollo Tecnológico en Cómputo, Instituto Politécnico Nacional, Mexico City, 07700, Mexico*
[b] *Colegio de Ciencia y Tecnología, Universidad Autónoma de la Ciudad de México, Mexico City, 06720, Mexico*
[c] *Unidad Profesional Interdisciplinaria de Ingeniería Campus Guanajuato, Instituto Politécnico Nacional, Guanajuato, 36275, Mexico*
[d] *ESIME U. Zacatenco, Instituto Politécnico Nacional, Mexico City, 07738, Mexico*

Abstract

Adapting the system's controller parameters using computational intelligence is increasingly common in applications with uncertainties and disturbances. Specifically, bio-inspired optimization has demonstrated significant progress and proven efficacy in auto-tuning controllers for Brushless Direct Current (BLDC) motors. However, the stochastic nature of bio-inspired algorithms in dynamic and uncertain environments often leads to solutions in different regions of the search space, resulting in large variations in controller gains and altering closed-loop system performance. To address the multimodality of the dynamic optimization problem in controller auto-tuning, this paper proposes incorporating an unsupervised-learning-based convergence-enhancement mechanism into the differential evolution's variation operator and dynamic constraints. This approach synergistically improves exploration and exploitation in promising regions of the search space, yielding more efficient and reliable solutions for controller tuning and improved closed-loop

*Corresponding authors
Email addresses: mvillarrealc@ipn.mx (Miguel Gabriel Villarreal-Cervantes), alejandro.rodriguez.molina@uacm.edu.mx (Alejandro Rodríguez-Molina)

performance. The proposed method, called the Adaptive Method for Controller Tuning based on Promising Regions (AMCT-PR), is applied to the Proportional-Integral (PI) controller of the BLDC motor. AMCT-PR also includes a way to handle load perturbations by incorporating robust terms into the optimization problem without requiring knowledge of the probability distribution of such perturbations. Comparative statistical results against three state-of-the-art tuning approaches reveal the superiority of the proposed method, achieving improvements of up to 17.33% in closed-loop performance while reducing controller gain variation by up to 37.50%.

Keywords: Adaptive tuning, brushless motor, differential evolution, bio-inspired optimization, meta-heuristics

---

List of acronyms and nomenclature

Acronyms	
ES	Evolution Strategies
DES	(1+1)-Dynamic Evolution Strategy
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AFIN	Adaptive Fuzzy Inference Network
ALO	Ant Lion Optimizer
AMCT	Adaptive Method for Controller Tuning
AMCT-PR	Adaptive Method for Controller Tuning based on Promising Regions
ANFIS	Adaptive Neuro-Fuzzy Inference System
BA	Bat Algorithm
BFO	Bacterial Foraging Optimization
BLDC	Brushless Direct Current
CATSCG	Chaotic Adaptive Tuning Strategy for Controller Gains
CATSCG (R)	Robust version of CATSCG
CODE	Chaotic Online Differential Evolution
CN	Cluster Number
CR	Crossover Rate
CS	Cuckoo Search
DC	Direct Current
DE	Differential Evolution
DE/rand/1/bin	Differential Evolution mutation strategy
DOP	Dynamic Optimization Problem
EA	Evolutionary Algorithm
EMF	Electromotive Force
ES	Evolutionary Strategy
FA	Firefly Algorithm
FL	Fuzzy Logic
FOPD	Fractional-Order Proportional-Derivative
FOPI	Fractional-Order Proportional-Integral
FOPID	Fractional-Order Proportional-Integral-Derivative
FPA	Flower Pollination Algorithm
GA	Genetic Algorithm
GOA	Grasshopper Optimization Algorithm
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
IAE	Integral of Absolute Error
ISE	Integral of Squared Error
ISU	Integral of Squared Control Signal
ITAE	Integral of Time-weighted Absolute Error
ITSE	Integral of Time-weighted Squared Error
Jaya	Jaya Optimization Algorithm
LSO	Lion Swarm Optimization
MMOP	Multi-Modal Optimization Problem
MO	Maximum Overshoot
MSA	Moth Swarm Algorithm
MT	Mersenne Twister
MTRNG	Mersenne Twister Random Number Generator
NCODE	Niching Chaotic Online Differential Evolution
NP	Population Size
ODE	Online Differential Evolution
OPSO	Online Particle Swarm Optimization
OGA	Online Genetic Algorithm
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PD	Proportional-Derivative
PRNG	Pseudo Random Number Generator
PSO	Particle Swarm Optimization
RFNN	Recurrent Fuzzy Neural Network
NN	Neural Network
RMSE	Root Mean Square Error
SCNG	Sine Chaotic Number Generator
SCNR	Sine map-based Chaotic Number Generator Routine
SE	Speed Error
SI	Swarm Intelligence
SL	Speed Loss
ST	Settling Time
TDPC	Time Domain Performance Criteria
TR	Rise Time
WFA-HBVPID	Weighted Firefly Algorithm for Hybrid Bacterial Variant PID
WT2F	Wavelet Type-2 Fuzzy

Nomenclature	
State variables	
$x(t)$	Real state vector of the BLDC motor, $x \in \mathbb{R}^5$.
$\tilde{x}(t)$	Estimated state vector (identification stage).
$\hat{x}(t)$	Predicted state vector (predictive stage).
$\bar{x}(t)$	Desired (reference) state vector.
x_i	i-th component of x.
$\dot{x}$	Time derivative of x.
Motor parameters	
θ	Angular position.
w	Angular velocity.
Θ	Real motor parameter vector, $\Theta \in \mathbb{R}^8$.
$\tilde{\Theta}$	Estimated parameter vector.
$\tilde{\Theta}^r$	Robust estimated parameter vector obtained from robust identification stage.
b_0	Friction coefficient.
J	Rotor inertia.
R	Phase-to-phase resistance.
L	Phase-to-phase inductance.
r	Phase resistance.
l	Phase inductance.
k_m	Torque constant.
k_e	Back-EMF constant.
τ_L	Load torque.
τ_e	Electromagnetic torque.
P	Number of pole pairs.
V_s	Input voltage.
V_{ab}, V_{bc}, V_{ca}	Phase-to-phase voltages.
$\bar{\eta}(\theta)$	Inverter commutation function.
$\bar{e}(\theta)$	Trapezoidal back-EMF shape function.
$\bar{A}, \bar{B}, \bar{C}, \bar{D}$	Auxiliary functions derived from the trapezoidal back-EMF profile.
Controller variables	
$K = [k_p, k_i]^T$	PI gain vector.
K^r	Robust PI gain vector.
$u(t)$	Control signal.
Sensitivity variables	
$S_{\tilde{x}}$	Sensitivity of the estimated states $\partial\tilde{x}/\partial\tau_L$.
$S_{\hat{x}}$	Sensitivity of the predictive states $\partial\hat{x}/\partial\tau_L$.
$\partial J_I/\partial\tau_L$	Identification error sensitivity.
$\partial J_P/\partial\tau_L$	Predictive error sensitivity.
Performance indices	
J_I	Identification performance index.
J_{IR}	Robust identification index.
J_P	Predictive performance index.
J_{PR}	Robust predictive index.
Time and windows	
t	Continuous time.
Δt	Sampling interval.
$\overline{\Delta t}$	Optimization update interval.
Δw_I	Identification window length (samples).
Δw_P	Prediction window length (samples).
$\tilde{\Omega}$	Identification time horizon.
$\hat{\Omega}$	Prediction time horizon.
Optimization (DE/NCODE)	
χ_i^G	i-th individual at generation G.
μ_i^G	Offspring individual.
NP	Population size.
G_{max}	Maximum generations.
F	Scaling factor.
CR	Crossover rate.
CN	Number of clusters.
Π_I, Π_P	Memory (identification/prediction).
ϕ	Constraint violation function.
Chaotic generator	
z_c	Current chaotic state.
z_n	Updated chaotic state.
μ	Sine map parameter.

1. Introduction

In control system design, the engineer must select the controller structure and face the difficult task of adjusting its parameters to fulfill one or a set of conflicting control objectives, such as settling time, maximum overshoot,

steady-state error, or energy consumption. In recent years, there has been an increase in the use of metaheuristic optimization for controller tuning tasks [1, 2, 3, 4, 5] due to their ability to handle dynamic constraints and identify global optimal solutions in complex and uncertain environments [6].

One research direction is to develop methodologies for updating controller gains at predefined time intervals [7] or dynamically during closed-loop operation [8]. This approach to online controller tuning is known as the Adaptive Method for Controller Tuning (AMCT), according to the taxonomy presented in [9, 10, 4]. In the AMCT, Dynamic Optimization Problems (DOPs) are formulated to perform identification and predictive stages. The DOPs are solved using metaheuristic optimization procedures such as Swarm Intelligence (SI) and evolutionary computation [11, 12, 13] to select the most suitable system parameters and controller gains (both of which are referred to as design variables) at their respective stages.

Finding solutions in the DOP is challenging because the fitness landscape and design variables change over time. Thus, the DOP has more than one optimum (which may include acceptable local optima) in each time window across the dynamic environment. Assuming the algorithm finds the global solution, the DOPs in the controller tuning task become even more complex when many global optima are encountered at the predefined time. In this scenario, several system and controller parameters satisfy the selected performance criteria, but traditional evolutionary algorithms (EAs) search for a single global optimal solution rather than multiple ones, thereby losing population diversity and converging to a single basin of attraction. Nevertheless, some global optima can lead to sudden changes in the future behavior of the closed-loop performance, which traditional EAs cannot distinguish. Additionally, over time, promising regions of the search space may become inferior in the dynamic environment, making it difficult for the algorithm to find the most suitable solution.

In this research direction, one unexplored issue in the AMCT approach is that metaheuristic optimization can yield solutions in other regions of the decision space, thereby promoting better solutions in the dynamic optimization framework for controller tuning. However, some regions are not suitable for the controller's closed-loop performance. For instance, when the system reaches the desired behavior during task execution, the AMCT's predictive stage can significantly change the design variables to minimize the predictive function in the simulated environment. However, this abrupt change in the design variables alters the control signal in the current closed-loop system,

leading to deviations from the control objectives.

This issue with changing design variables is attributed to the multimodality of the optimization problem in the AMCT approach. When a problem has multiple global and local optima, it is considered a multimodal optimization problem (MMOP) [14]. Multi-modal optimization problems [15, 16] emerge in the AMCT approach as well as in other real-world applications, such as multi-robot task allocation [17], electromagnetic optimization [18, 19], protein structure prediction [20], project scheduling [21], and others.

Researchers have addressed this problem in controller tuning by implementing low-pass digital filters [10, 22, 23] in the obtained controller gains to handle variations in the design variables. Others have incorporated anti-windup methods into the controller [7, 24] to reduce the increment in the control variable and avoid saturation. Using constant and reduced design-variable bounds in the optimization process is another strategy to maintain the controller gains within a specific region [25, 26]. In these works, the obtained controller gains are over-adjusted or limited to provide a suitable control signal in the current closed-loop system. However, this additional adjustment can degrade the controller's closed-loop performance because these implementations (e.g., filters, anti-windup methods, and gain bounds) limit the metaheuristic optimizer's exploration by excluding them from the optimization process.

On the other hand, given that the electric drive market is among the most rapidly expanding industries and that the controller's flexibility, reconfigurability, and adaptability are essential for high-performance applications such as robots, machine tools, and other systems, the AMCT based on metaheuristics has been applied to the actuators of the system. The AMCT, based on a Genetic Algorithm (GA), is applied to the Proportional-Integral controller of an indirect field-oriented linear induction motor in [25], providing highly effective dynamic properties under changes in plant parameters, frictional forces, and external load disturbances. Fast adaptive memetic and genetic algorithms are incorporated into the AMCT to control a permanent-magnet synchronous motor [27]. These algorithms determine the control parameters for speed, current, the compensator, and the filter.

The AMCT is also applied to the Brushed Direct Current (DC) motor in [24], with a neural network to predict the fitness function of trial solutions and using different variants of metaheuristic algorithms such as Particle Swarm Optimization (PSO), Differential Evolution (DE), Bat Algorithm (BA), and Evolutionary Strategy (ES). Instead of using a neural network, [22] employs

a dynamic DC motor model for prediction using DE variants, GA, and PSO. These works confirm the usefulness and reliability of metaheuristic algorithms for real-time controller tuning in dynamic environments.

One of the most widely used actuators is the Brushless Direct Current (BLDC) motor due to its superior speed versus torque characteristics and high dynamic response, among other advantages, over the conventional DC motor [28]. Several controller tuning strategies based on metaheuristic optimization have emerged in recent years. Table 1 shows the most relevant works in this direction. It is observed that most of the work (80%) implements offline controller tuning, making it more susceptible to unknown uncertainties and disturbances. Online controller tuning based on AMCT (related to the proposal) is used less frequently in BLDC motor controllers, appearing in only 20% of reported works.

Conversely, the most used optimizers are Particle Swarm Optimization (PSO) in 62% of the works, followed by Genetic Algorithm (GA) in 41%, Bat Algorithm (BA) in 25%, Differential Evolution (DE) in 20%, Firefly Algorithm (FA) in 12%, and the rest are used at most once, i.e., in 4%. This result indicates that Swarm Intelligence (SI) algorithms based on collective behavior within self-organizing societies of agents (PSO, BA, and FA) are the most used optimizers, followed by Evolutionary Algorithms (EA) inspired by natural evolution and survival of the fittest (DE and GA). Nevertheless, the literature comparing SI algorithms with EAs, particularly with DE, is scarce [29, 30].

In recent years, learning-based controller tuning strategies have gained significant attention, particularly in expensive and constrained optimization problems arising in controller calibration. In [30], surrogate-assisted adaptive tuning for control systems is explored using response surface models, which significantly reduce the computational cost of the tuning process while maintaining competitive closed-loop performance in BLDC motors (but not superior) to that obtained with model-based or filtering-based adaptive tuning strategies, indicating that their main advantage lies in computational efficiency rather than performance improvement. Similarly, [53] employs Bayesian Optimization to efficiently adjust MPC cost weights in bilateral teleoperation, significantly reducing calibration time while improving position and force tracking. In [54], neural networks approximate the robust MPC control law itself, embedding approximation errors into a robust framework to ensure constraint satisfaction and closed-loop stability with substantial memory reduction.

Table 1: Summary of metaheuristic-based BLDC controller tuning approaches reported in the literature. The table highlights controller types, tuned parameters, objectives, employed optimizers, and whether the tuning process is offline or online.

Ref.	Used controller	Tuning parameters	Design objective*	Employed algorithms	Optimization process
[31]	PID	PID controller gains	SE	PSO	Offline
[32]	FL	Membership function parameters	SE, SL	PSO	Online
[33]	PI	PI controller gains	MO, ST, SE	DE, Modified DE	Online
[34]	Online ANFIS, PID, Fuzzy PID, Adaptive FL	Learning rate, forgetting factor, steepest descent, momentum constant, PID, Fuzzy and FL controller gains	RMSE+IAE+ITAE+ISE	BA, GA, PSO	Offline
[35]	Fuzzy PID supervised on-line RFNN	Learning rate, dynamic factor, node number	RMSE, IAE, ITAE, ISE	GA, PSO, ACO, BA, ALO	Offline
[36]	PI	PI controller gains	IAE, ISU	PSO with five inertia weight adjustment strategies	Offline
[37]	ANFIS	Learning rate, forgetting factor, steepest descent momentum constant	ISE, TDPC	BFO BA, PSO	Offline
[38]	PID-type FL	PID controller gains	IAE	GA	Offline
[39]	PID	PID controller gains	ISE	FPA, PSO, FA Ziegler-Nichols	Offline
[40]	∞ PID	PID controller gains	ISE	GOA	Offline
[41]	Fuzzy PD/PID	PD/PID controller gains, membership function parameters, coefficient of the consequent part of fuzzy PD/PID controller	RMSE, IAE, ITAE, ISE	PSO, CS, BA	Offline
[42]	FOPID	PID controller gains, fractional-orders	RMS	BA, Modified GA, ABC	Offline
[43]	FOPI, PI	PI controller gains, fractional-orders	ITAE	MSA	Offline
[44]	PI	PI controller gains	MO, ST	GA	Offline
[45]	PID	PID controller gains	ITSE	PSO, BFO	Offline
[46]	FOPD	PI controller gains, fractional-orders	TR	Jaya	Offline
[47]	Online ANFIS, PID, offline ANFIS	Learning rate, forgetting factor, steepest descent momentum constant	RMSE, MO	Hybrid GA-PSO	Offline
[48]	FOPID	PID controller gains, fractional-orders	TR	FA, GA	Offline
[29]	PI	PI controller gains	ISE	ODE, OGA, OPSO, CODE	Online
[49]	PI, P-PI, F-PI, PF-PI	PI controller gains	ITAE, ISE, ITSE	PSO	Offline
[30]	PI	PI controller gains	IAE	ODE, OPSO, OGA	Offline and Online
[50]	PI	PI controller gains	IAE	DE, PSO, FA	Offline
[51]	PID	PID controller gains	SE	WFA-HBVPID, WT2F, BA, AFIN	Offline
[52]	fractional-order PI	PI controller gains, fractional-orders	IAE, ISE, ITAE, ITSE	LSO, PSO	Online

* The error is related to the difference between the desired and actual motor velocity.

Recent work has also explored direct learning of controller tuning rules [55]. In [56], an inverse supervised learning framework leverages a digital twin to generate synthetic data and meta-learn a direct mapping from input-output trajectories to controller parameters, bypassing explicit system identification during deployment. In a different direction, [57] applies reinforcement learning to tune PID gains for nonlinear biotechnological systems, modeling the PID controller as an actor network and optimizing gains through trial-and-error interaction with the environment.

It is important to highlight that the role of learning in the proposed method is fundamentally different. In most learning-assisted optimization frameworks described above, learning mechanisms are typically used to approximate the fitness function (e.g., surrogate models), adapt control parameters, or guide the search through predictive models, while overlooking one of the main causes of closed-loop performance degradation, that is, the variation of the controller parameters over time, which may induce abrupt gain changes and consequently deteriorate closed-loop behavior. In contrast, the proposed AMCT-PR does not rely on the approximation of the objective function. Instead, it incorporates an unsupervised learning mechanism that operates at a structural level within the optimization process. Specifically, the Dynamic Memory-Based Niche Identification (DMNI) mechanism is embedded directly into the mutation operator of Differential Evolution. Its purpose is not to estimate solution quality, but to identify and preserve promising regions of the search space across time by exploiting the continuity of previously successful solutions. This memory-driven niching strategy allows the algorithm to maintain temporal coherence in the design variables while still enabling controlled exploration. Then, the DMNI mechanism modifies the selection of the candidate individuals. Therefore, the learning component becomes part of the search dynamics itself.

Furthermore, unlike clustering-based DE variants, where clustering is typically used as a static diversity-preservation mechanism, the proposed approach employs dynamic clustering guided by memory evolution, making the notion of "promising regions" adaptive over time. This produces a memory-driven and temporally coherent search process, which is particularly important during the dynamic evolution of controller tuning.

In addition, the proposed method integrates dynamic constraints directly into the optimization problem to explicitly regulate inter-window variation of the decision variables. This contrasts with existing approaches that rely on post-processing strategies such as filtering or bounded parameter updates.

In this way, smooth parameter evolution is enforced inside the optimization loop itself.

It is also worth noting that reinforcement learning approaches to controller tuning, such as [57], learn gain-adjustment policies through trial-and-error interaction with the environment, which typically requires extensive training episodes and does not explicitly address the temporal consistency of the controller gains across consecutive tuning intervals. In contrast, the proposed AMCT-PR enforces smooth gain evolution through dynamic constraints embedded directly in the optimization loop, without relying on policy learning or interaction with the environment.

These differences make the proposed AMCT-PR a structurally integrated learning-enhanced optimization framework specifically designed for dynamic optimization problems in adaptive control, where temporal consistency, robustness, and smooth parameter evolution are critical.

To further clarify the distinctive characteristics of the proposed AMCT-PR framework with respect to existing adaptive controller tuning approaches, a conceptual comparison is presented in Table 2, spanning several dimensions: adaptation mechanism, role of learning, constraint handling, population management, multimodality handling, and gain variability. As shown, unlike approaches in which learning mechanisms are externally coupled or used as auxiliary guidance (e.g., surrogate-assisted or neural-network-based fitness approximation), the proposed framework integrates unsupervised learning directly into the optimization process via the DMNI mechanism, enabling the joint evolution of population structure and constraint handling within a unified scheme. Furthermore, unlike filtering- or bounds-based approaches that regulate gain variability as a post-processing step decoupled from the optimizer, AMCT-PR enforces smooth parameter evolution directly inside the optimization loop through dynamic constraints, preserving closed-loop performance without restricting the optimizer's exploratory freedom. Together, these elements distinguish AMCT-PR as a structurally integrated framework specifically designed for dynamic and multimodal optimization problems in adaptive controller tuning.

Table 2: Conceptual comparison of adaptive controller tuning approaches reviewed in this work, including the proposed AMCT-PR framework.

Feature	DE-based tuning with post-processing [10, 22, 23, 29, 33]	Surrogate/learning-assisted DE tuning [30, 24]	Evolutionary-based tuning [7, 24]	AMCT-PR (proposed)
Optimization strategy	Global stochastic DE search; online DE with post-processing gain adjustment (filters or bounds)	DE guided by surrogate or neural-network fitness approximations	Evolutionary algorithms (DES, ES, DE, PSO, BA), including both single-solution and population-based schemes for online controller tuning	Online DE with niche-based redistribution of individuals via DMNI mechanism
Adaptation mechanism	Gains adjusted via post-processing (filters or bounds) at each window without explicit variation control (gain variation controlled externally).	Adaptation guided by surrogate/model-based objective predictions for computational efficiency	Gains indirectly limited by anti-windup mechanisms to prevent actuator saturation; the adaptation follows the standard evolutionary process	Adaptive niching combined with dynamic constraints that explicitly limit inter-window gain variation
Role of learning	Not included	Supervised or data-driven learning to approximate the fitness function	Supervised learning to predict the objective function and enable online evaluation	Unsupervised learning (clustering) embedded in the mutation operator for niche formation and memory-driven region identification
Population management	Single evolving population with post-processed solutions	Population coupled with surrogate/neural-network evaluation	Single-individual schemes to population-based approaches	Dynamic niches with periodic redistribution guided by memory of previously successful solutions
Constraint handling	Static bounds or penalty functions; gain variation regulated by post-processing	Constraints incorporated via surrogate formulation or problem bounds	Implicitly handled through the control architecture and surrogate-based evaluation to avoid unsafe interactions	Dynamic constraints adjusted during evolution to explicitly bound inter-window design variable variation
Handling of multimodality	Not explicitly addressed; post-processing may mask multimodal structure	Not explicitly addressed; dependent on surrogate accuracy; may miss poorly modeled regions	Not explicitly addressed; limited ability to preserve temporal coherence across multiple optima	Explicitly addressed through memory-driven niching that tracks promising regions across time windows
Controller gain variability	External via filtering or bounds, at the cost of optimizer freedom	Dependent on surrogate/model accuracy; not explicitly targeted	Not explicitly controlled; variability depends on evolutionary operators and changes in the fitness landscape; implicitly bounded with anti-windup mechanisms	Explicitly reduced through dynamic constraints and the DMNI mechanism embedded in the optimization loop, preserving closed-loop performance

Contributions

Based on the fact that the sudden changes in the design variable vector in the AMCT have been addressed by using filters, anti-windup methods, and static gain bounds, this paper proposes a structured adaptive tuning framework for control systems where a Dynamic Memory-Based Niche Identification (DMNI) mechanism and dynamic constraints that explicitly limit inter-window design parameter variation operate synergistically within the identification and predictive stages of the AMCT. Unlike existing learning-assisted and clustering-based approaches, the proposed method incorporates a structurally integrated unsupervised learning mechanism that is not merely used for diversity preservation or offline guidance. Instead, it is embedded directly within the optimization loop, where it operates alongside dynamic constraints to enforce temporal coherence in the solution trajectory. These elements together form a closed-loop adaptive optimization architecture for control tuning called Adaptive Method for Controller Tuning based on Promising Regions (AMCT-PR).

Then, the AMCT-PR includes a mechanism to classify the region of the most suitable solutions via unsupervised learning of design-space niche clustering and applies this to the optimizer's variation operator (mutation) for individual selection. This mechanism is memory-driven, as the promising region is identified relative to the best solution stored from the previous optimization window, allowing the niche to evolve coherently across time intervals while preserving temporal continuity in the design variables. Differential Evolution (DE) is adopted to implement this strategy due to its few parameter requirements, the demonstrated robustness in dynamic optimization contexts [58], and according to the state-of-the-art summarized in Table 1, DE-based optimizers are among the most widely used evolutionary algorithms for controller tuning in electric drives [29]. Moreover, special dynamic constraints are incorporated into the optimization problem to explicitly regulate the inter-window variation of the design variables, maintaining the search within regions close to previously validated solutions. The DMNI mechanism and dynamic constraints transform the adaptive tuning process into a dynamic search strategy that improves the exploration-exploitation balance and accelerates convergence toward promising regions of the multimodal search space, thereby promoting smoother controller gain evolution and improved closed-loop performance. Therefore, the proposed AMCT-PR, which integrates the dynamic memory-based niche identification mechanism and dynamic constraint regulation within an online adaptive framework, con-

stitutes the first main contribution of this paper. This approach is compared with state-of-the-art tuning methods, and inferential statistical analysis confirms the superior performance of the proposed AMCT-PR when applied to the Proportional-Integral (PI) controller of a BLDC motor under parametric uncertainties and load disturbances.

On the other hand, the AMCT inherently handles environmental changes by continuously updating the control gains. Nevertheless, as observed in Table 1, most existing works mitigate variations in control performance objectives by incorporating performance indicators related to task error (91% of the reported studies) and, to a lesser extent, to energy consumption in the control action (12%), typically through the reduction of torque ripple or control signal magnitude. However, these approaches do not explicitly incorporate a mechanism into the optimization formulation to reduce the sensitivity of the performance indices to unknown environmental variations, without assuming prior knowledge of the probability distribution of such uncertainties. In contrast, this paper introduces robust terms directly into the dynamic optimization problems of both the identification and predictive stages of the AMCT-PR. By embedding sensitivity minimization within the objective functions, the proposed method promotes controller parameters that are intrinsically less affected by load disturbances and parametric uncertainties. This integration of robustness at the optimization level, rather than as a post-processing or external compensation strategy, constitutes the paper's second main contribution.

The proposed approach achieves controlled temporal evolution of controller gains in a dynamic multi-modal environment without relying on external filtering, anti-windup adjustments, or restrictive gain bounds. It simultaneously maintains solution continuity across time windows, preserves exploitation in historically promising regions, allows controlled exploration when the niche deteriorates, and embeds robustness against load uncertainties directly into the optimization problem. To the best of our knowledge, current state-of-the-art approaches do not simultaneously integrate these four aspects within an online adaptive controller-tuning framework.

The rest of the paper is organized as follows. Section 2 describes the dynamic equations for the case study that apply the AMCT-PR, i.e., it shows the BLDC motor dynamics. The proposed AMCT-PR and its application to the robust tuning of the PI controller for BLDC motors are presented in Section 3. The discussion of the results is given in Section 4, and the conclusions are drawn in Section 5.

2. BLDC motor dynamics with the PI controller for speed regulation

The schematic diagram of Brushless Direct Current (BLDC) motors in star topology is shown in Fig. 1. The parameters associated with the BLDC motor are the angular position θ, the angular velocity w, the current i_γ in the phase $\gamma \in \{a, b, c\}$, the phase-to-phase resistance R, the phase-to-phase inductance L, the friction coefficient b_0, the rotor inertia J, the torque constant k_m, the back-EMF constant k_e, the load torque τ_L, the total torque delivered by three phases $\tau_e = k_m \left(i_a \bar{e}(\theta) + i_b \bar{e}(\theta - 2\pi/3) + i_c \bar{e}(\theta - 4\pi/3)\right)$, the number of pole pairs P, the input voltage V_s, and the phase-to-phase voltage $V_{\gamma\gamma}$.

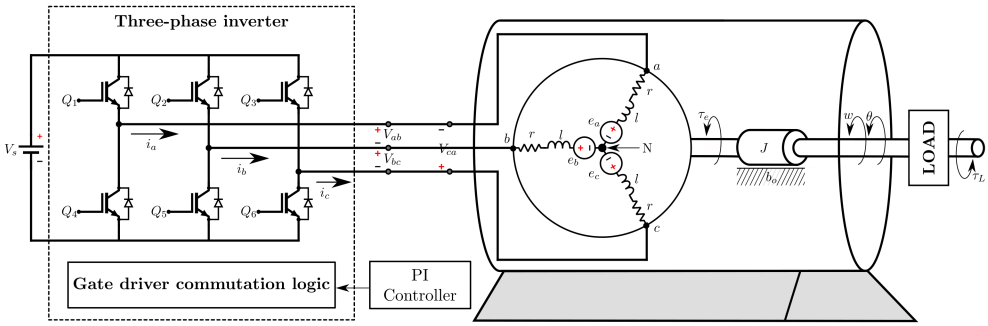


Figure 1: Three-phase BLDC motor and inverter topology used for PI speed regulation. The nonlinear electrical and mechanical dynamics define the plant employed in the identification and predictive stages of the proposed AMCT-PR framework.

The dynamic equations of the BLDC motor comprise electrical and mechanical components [59, 60]. The closed-loop system of the motor dynamics is expressed in (1), where the state vector is related to $x = [\theta,\ w,\ i_a,\ i_b,\ \int(\bar{x}_2 - w)dt]^T \in \mathbb{R}^5$ and it considers the Proportional-Integral (PI) controller as the input signal $u = V_s$. So, this expression represents the state-space dynamics of the BLDC motor under PI control. In simple terms, it describes how the motor states (position, velocity, and current) evolve over time as functions of the system parameters and controller gains. This equation enables emulation of BLDC motor behavior in simulation environments, avoiding the need to collect data from a real platform. Moreover, it facilitates optimization by enabling the identification and prediction of how different controller and system parameter settings affect the closed-loop motor response.

Let $\Theta = [\frac{b_0}{J},\ \frac{k_m}{J},\ \frac{k_e}{L},\ \frac{R}{L},\ \frac{1}{L},\ \frac{1}{J},\ \tau_L, P]^T \in \mathbb{R}^8$ be the BLDC motor parameter vector, $K = [k_p,\ k_i]^T \in \mathbb{R}^2$ be the controller gain vector and $\bar{x}$ is the desired state vector, the resulting equations are as follows [29]:

$$\dot{x} = \begin{bmatrix} 0 & \Theta_8 & 0 & 0 & 0 \\ 0 & -\Theta_1 & \bar{C}\Theta_2 & \bar{D}\Theta_2 & 0 \\ 0 & \frac{\bar{A}}{3}\Theta_3 & -\Theta_4 & 0 & 0 \\ 0 & \frac{\bar{B}}{3}\Theta_3 & 0 & -\Theta_4 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{2}{3}\Theta_5 & \frac{1}{3}\Theta_5 \\ -\frac{1}{3}\Theta_5 & \frac{1}{3}\Theta_5 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{ab} \\ V_{bc} \end{bmatrix} - \begin{bmatrix} 0 \\ \Theta_6\Theta_7 \\ 0 \\ 0 \\ \bar{x}_2 \end{bmatrix}$$

$$\dot{x} = f(x, \Theta, K) \tag{1}$$

The phase-to-phase voltages are $V_{ab} = \frac{u}{2}(\bar{\eta}(\theta) - \bar{\eta}(\theta - 2\pi/3))$ and $V_{bc} = \frac{u}{2}(\bar{\eta}(\theta - 2\pi/3) - \bar{\eta}(\theta - 4\pi/3))$, considering the PI controller $u = k_p(\bar{x}_2 - x_2) + k_i x_5$. The parameters associated with the trapezoidal shape function of the back-EMF force are $\bar{A} = \bar{e}(\theta - 2\pi/3) - 2\bar{e}(\theta) + \bar{e}(\theta - 4\pi/3)$, $\bar{B} = \bar{e}(\theta) - 2\bar{e}(\theta - 2\pi/3) + \bar{e}(\theta - 4\pi/3)$, $\bar{C} = \bar{e}(\theta) - \bar{e}(\theta - 4\pi/3)$, and $\bar{D} = \bar{e}(\theta - 2\pi/3) - \bar{e}(\theta - 4\pi/3)$. The functions associated with the trapezoidal and voltage equations are described as follows:

$$\bar{e}(\theta) = \begin{cases} \frac{6\theta}{\pi}, & \text{If } -\pi/6 \leq \theta \leq \pi/6 \\ 1, & \text{If } \pi/6 \leq \theta \leq 5\pi/6 \\ -\frac{6(\theta-\pi)}{\pi}, & \text{If } 5\pi/6 \leq \theta \leq 7\pi/6 \\ -1, & \text{If } 7\pi/6 \leq \theta \leq 11\pi/6 \end{cases}, \bar{\eta}(\theta) = \begin{cases} 0, & \text{If } -\pi/6 \leq \theta \leq \pi/6 \\ 1, & \text{If } \pi/6 \leq \theta \leq 5\pi/6 \\ 0, & \text{If } 5\pi/6 \leq \theta \leq 7\pi/6 \\ -1, & \text{If } 7\pi/6 \leq \theta \leq 11\pi/6 \end{cases}.$$

One important consideration in the BLDC motor's dynamic simulation is the inclusion of the physical limit on the applied voltage in the control signal. On a real platform, the power source can maintain the nominal voltage. Therefore, to simulate the voltage limit, the control signal is set to its maximum value whenever it exceeds the limit. This limit strategy in the control signal is applied to the motor's dynamics and to the robust predictive stage described in Section 3.2.

3. Adaptive Method for Controller Tuning based on Promising Regions

Two successive steps are necessary for the AMCT of BLDC motors: identification and prediction. The formulation of both stages comprises two dynamic optimization problems that are solved using bio-inspired techniques. The main difference of the proposed Adaptive Method for Controller Tuning

based on Promising Regions (AMCT-PR) is the incorporation of the proposed Niching Chaotic Online DE (NCODE), which, together with the special dynamic constraints in the dynamic optimization problems, promotes a better closed-loop response, reduces design variable changes, and aids the search for solutions in the multi-modal space. Unlike existing learning-assisted and clustering-based DE approaches, the proposed framework integrates an unsupervised learning mechanism (DMNI) directly into the optimization loop in a structural manner. Rather than relying on surrogate models or external guidance, the learning component identifies promising regions from population structure and historical memory and is embedded within the mutation operator, thereby shaping the search dynamics. Furthermore, clustering is dynamically coupled with memory to enforce temporal coherence of the solution trajectory, while dynamic constraints are incorporated into the optimization formulation to explicitly limit inter-window parameter variations. In this section, the proposed AMCT-PR is detailed for robust tuning of the BLDC motor controller.

The first stage estimates the BLDC motor parameters $\tilde{\Theta}(t)$ based on a robust identification process. The second stage determines the controller's parameters $K(t)$ based on a robust predictive process. Both stages are formulated as dynamic optimization problems, and the proposed NCODE provides the corresponding solutions for setting those parameters over different time intervals. One of the main aims in both stages is to make the parameters $\tilde{\Theta}^r(t)$ and $K^r(t)$ as insensitive as possible to load uncertainties, without accounting for the probability distribution of parameter variations in the optimization problem.

The schematic diagram of the proposed AMCT-PR is provided in Fig. 2. Initially, the BLDC motor's initial states $x(0) = \mathbf{0}$, the time $t = 0$, and the proposed controller gains K^r, as well as the estimated motor parameters $\tilde{\Theta}^r$, are established at the beginning of the main loop. Let Δt denote the sampling interval used in numerical integration, while $\overline{\Delta t}$ represents the optimization update interval in the proposed adaptive framework. At each Δt (s), an interruption is generated, and the behavior of the BLDC motor dynamics is computed by integrating the differential equation given in (1) (in practical applications, this behavior is determined by the system sensors and state estimators). This interruption provides the state vector information at the current time, i.e., $x(t + \Delta t)$. The main loop waits until the time period $\overline{\Delta t} >> \Delta t$ is completed. When this occurs, the estimated BLDC motor parameter $\tilde{\Theta}^r$ and the controller parameter vector K^r are updated using

robust identification, followed by the robust predictive stage. Both stages are solved using the proposed NCODE, which, together with the special dynamic constraints described in Section 3.3, promotes the search throughout promising regions. With the robust controller parameter vector K^r found by the NCODE, load uncertainties can have less impact on controller behavior as the task progresses, and smooth behavior in the controller gains is encouraged.

Applying the AMCT-PR has the following requirements:

1. Differential equations must be able to describe and predict BLDC motor dynamics, which means that the physical laws governing the motor's behavior are known.
2. The transcription method [61] is used to transform the initial continuous-time formulation (an infinite-dimensional optimization problem) into a discrete-time formulation (a discrete-dimensional optimization problem), with the goal of casting the associated dynamics into a finite state space through the solution of the differential equations using numerical integration techniques.
3. The associated control gains parameterize the control system.
4. Only the nominal parameters of the uncertainty are known, so these are considered in the solution of the optimization problems.

The next subsections describe the components of the proposed AMCT-PR, including the robust identification and predictive stages, as well as the optimizer used to search for solutions in promising regions.

3.1. Robust identification stage in the BLDC motor

In the robust identification stage, the estimated parameter vector $\tilde{\Theta} \in \mathbb{R}^8$ of the BLDC motor dynamics (1) is found through the solution of a dynamic optimization problem. The robust identification stage requires the state information of Δw_I previous states, i.e., $x(t - \Delta w_I \Delta t),\ \ldots,\ x(t - 2\Delta t),\ x(t-\Delta t),\ x(t)$. The dynamic optimization problem consists of finding the robust parameter vector $\tilde{\Theta}^r$ for the estimated BLDC motor dynamics $\tilde{x}(t)$ expressed in (8), which simultaneously minimizes the error between the current states $x(t)$ and the estimated ones $\tilde{x}(t)$ in the time interval $t \in [t - \Delta w_I \Delta t,\ t]$, and also decreases the identification error sensitivity with respect to changes in the load torque. Minimizing the identification objective function improves the model's approximation of BLDC motor states, and decreasing its sensitivity reduces the changes of identification errors due to

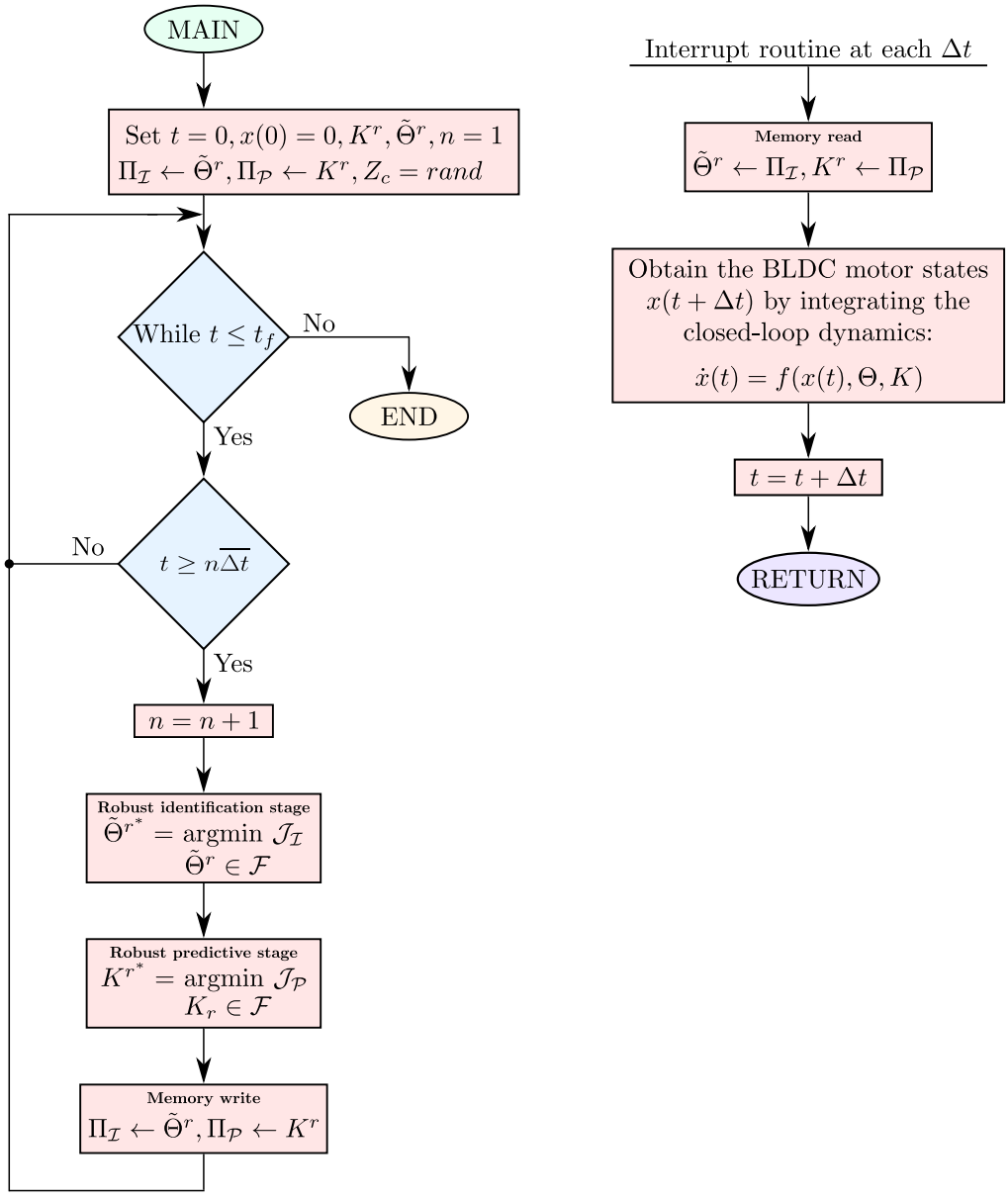


Figure 2: Schematic diagram of the proposed AMCT-PR framework. The architecture integrates robust identification, robust predictive tuning, memory updating, NCODE optimization, and dynamic constraints to regulate inter-window gain variation.

load variations. The main objective of sensitivity minimization is to provide a parameter vector $\tilde{\Theta}^r$ that is less sensitive to load variations for future behavior in the predictive stage of the AMCT-PR. Minimizing sensitivity reduces the extent to which the identification error varies with load changes. This yields parameter estimates that are not only accurate but also less disturbance-dependent.

Considering that the identification error function $J_I \in \mathbb{R}$ given in (2), measures the discrepancy between the real system states and the estimated states, it effectively quantifies the accuracy of the identified model in reproducing the system behavior. Therefore, minimizing this error improves model fidelity.

$$J_I(t) = \left(x(t) - \tilde{x}(t)\right)^T \left(x(t) - \tilde{x}(t)\right) \tag{2}$$

However, as mentioned earlier, the identified model can be affected by variations in load torque. So, the sensitivity terms are obtained to evaluate changes in identification error as the load torque varies, thereby indirectly measuring the model's response to disturbances. The sensitivity of the identification error with respect to the changes in the load torque $\frac{\partial J_I}{\partial \tau_L} \in \mathbb{R}$ can be computed as follows:

$$\begin{aligned} \frac{\partial J_I(t)}{\partial \tau_L} &= \frac{\partial J_I(t)}{\partial \tilde{x}} \frac{\partial \tilde{x}(t)}{\partial \tau_L} \\ &= -2\left(x(t) - \tilde{x}(t)\right)^T \frac{\partial \tilde{x}(t)}{\partial \tau_L} \end{aligned} \tag{3}$$

The sensitivity of the estimated states $S_{\tilde{x}} = \frac{\partial \tilde{x}}{\partial \tau_L} \in \mathbb{R}^7$ is obtained by solving the partial derivative of the estimated BLDC motor dynamics $\dot{\tilde{x}} = \tilde{f}(\tilde{x}, \tilde{\Theta}^r, u)$ (8) with respect to the load torque, and results in:

$$
\begin{aligned}
\frac{d}{dt}\frac{\partial \tilde{x}(t)}{\partial \tau_L} &= \frac{\partial \tilde{f}\left(\tilde{x}, \tilde{\Theta}^r, u\right)}{\partial \tilde{x}}\frac{\partial \tilde{x}}{\partial \tau_L} + \frac{\partial \tilde{f}\left(\tilde{x}, \tilde{\Theta}^r, u\right)}{\partial \tau_L} \\
&= \begin{bmatrix} 0 & \tilde{\Theta}_8^r & 0 & 0 & 0 \\ 0 & -\tilde{\Theta}_1^r & \bar{C}\tilde{\Theta}_2^r & \bar{D}\tilde{\Theta}_2^r & 0 \\ 0 & \frac{1}{3}\bar{A}\tilde{\Theta}_3^r & -\tilde{\Theta}_4^r & 0 & 0 \\ 0 & \frac{1}{3}\bar{B}\tilde{\Theta}_3^r & 0 & -\tilde{\Theta}_4^r & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \frac{\partial \tilde{x}}{\partial \tau_L} - \begin{bmatrix} 0 \\ \tilde{\Theta}_6^r \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
\frac{d}{dt}S_{\tilde{x}} &= \tilde{f}_{R_I}(S_{\tilde{x}}, \tilde{\Theta}^r)
\end{aligned} \tag{4}
$$

The robust identification error $J_{I_R} \in \mathbb{R}$ under the effects of the load variations is expressed as in (5). A low value of J_{I_R} implies that the estimated model remains reliable even when external conditions change, and a high value indicates a model that is highly sensitive to load variations. Therefore, minimizing it promotes parameter estimates that are inherently robust to disturbances.

$$
J_{I_R}(t) = \left(\frac{\partial J_I(t)}{\partial \tau_L}\right)^2 \tag{5}
$$

Then, the robust identification problem can be interpreted as a trade-off between accuracy and robustness. On one hand, the model must reproduce the observed system behavior; on the other hand, it must remain insensitive to load disturbances. The mathematical formulation of the dynamic optimization problem for the robust identification stage is presented in (6)-(11). The time interval for the robust identification stage is $t \in \tilde{\Omega} \subseteq [t - \Delta w_I \Delta t,\ t] = [t_i,\ t_f]$, bounded by its initial time t_i and final time t_f. At the end of the final time t_f, the estimated robust BLDC motor parameter vector $\tilde{\Theta}^r \in \mathbb{R}^8$ is obtained by considering a weighted product method [62] in the objective function. The first design objective is the identification error, and the second is the corresponding sensitivity. The problem is constrained by both the estimated DC motor dynamics (8) and the state sensitivity dynamics (9) with their initial conditions (10), and box constraints (bounds) on the estimated robust BLDC motor parameter vector (11).

$$\min_{\tilde{\Theta}^r \in \mathbb{R}^7} \mathcal{J}_{\mathcal{I}} \tag{6}$$

$$\mathcal{J}_{\mathcal{I}} = \int_{t \in \tilde{\Omega}} J_I(t)\ J_{I_R}(t) dt \tag{7}$$

Subject to :

$$\dot{\tilde{x}} = \begin{bmatrix} 0 & \tilde{\Theta}_8^r & 0 & 0 & 0 \\ 0 & -\tilde{\Theta}_1^r & \bar{C}\tilde{\Theta}_2^r & \bar{D}\tilde{\Theta}_2^r & 0 \\ 0 & \frac{\bar{A}}{3}\tilde{\Theta}_3^r & -\tilde{\Theta}_4^r & 0 & 0 \\ 0 & \frac{\bar{B}}{3}\tilde{\Theta}_3^r & 0 & -\tilde{\Theta}_4^r & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{2}{3}\tilde{\Theta}_5^r & \frac{1}{3}\tilde{\Theta}_5^r \\ -\frac{1}{3}\tilde{\Theta}_5^r & \frac{1}{3}\tilde{\Theta}_5^r \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{ab} \\ V_{bc} \end{bmatrix} - \begin{bmatrix} 0 \\ \tilde{\Theta}_6^r \tilde{\Theta}_7^r \\ 0 \\ 0 \\ \bar{x}_2 \end{bmatrix} \tag{8}$$

$$\frac{d}{dt} S_{\tilde{x}} = \tilde{f}_{R_I}(S_{\tilde{x}}, \tilde{\Theta}^r) \tag{9}$$

$$\tilde{x}(t_i) = x(t_i), S_{\tilde{x}}(t_i) = \frac{\partial x(t_i)}{\partial \tau_L} \tag{10}$$

$$\tilde{\Theta}_{min} \leq \tilde{\Theta} \leq \tilde{\Theta}_{max} \tag{11}$$

3.2. Robust predictive stage in the BLDC motor

In the robust predictive stage, the goal is not only to minimize future tracking error but also to ensure that the selected controller gains remain robust to potential load variations. In simple terms, the optimizer searches for controller parameters that perform well in the future while being insensitive to uncertainties.

In particular, in the robust predictive stage, the controller gains $K \in \mathbb{R}^2$ of the PI speed controller are obtained by solving a dynamic optimization problem. This stage requires the information of the predictor $\hat{x}$ for the BLDC motor states in the time interval $t \in [t,\ t + \Delta w_P \Delta t]$ to compute the future system behavior, i.e., the state information of Δw_P future states $x(t),\ x(t + \Delta t), x(t + 2\Delta t),\ \ldots,\ x(t + \Delta w_P \Delta t)$. It also uses the estimated robust BLDC motor parameter vector $\tilde{\Theta}^r$ found in the robust identification stage. The dynamic optimization problem consists of finding the controller parameter vector K for the state predictor dynamics $\hat{x}(t)$ expressed in (18), which simultaneously minimizes the error between the predictive state of the motor speed and the reference speed and also reduces the error sensitivity with respect to variations in the load torque. Reducing predictive error yields PI control gains that meet the regulation task within the prediction time horizon, while minimizing sensitivity to predictive error yields control gains that

accommodate future load changes in the BLDC motor. The main objective of sensitivity minimization is to provide a PI control parameter vector K^r that is less affected by load variations in the next time period. Minimizing predictive sensitivity promotes PI gains that inherently attenuate the impact of future load disturbances, improving robustness at the optimization level.

Considering that the predictive tracking error $J_P \in \mathbb{R}$ defined in (12), measures the tracking performance within the prediction horizon, its minimization improves tracking fidelity. It measures how accurately the controller is expected to follow the reference trajectory in the future

$$J_P(t) = (\bar{x}_2(t) - \hat{x}_2(t))^2 \tag{12}$$

However, the predictive tracking error can be perturbed by the variations in the load torque. So, the sensitivity terms are obtained to evaluate changes in tracking error as the load torque varies, thereby indirectly measuring the controller performance to external disturbances. The sensitivity of the predictive error with respect to the variations in the load torque $\frac{\partial J_P}{\partial \tau_L} \in \mathbb{R}$ can be obtained as follows:

$$\begin{aligned} \frac{\partial J_P(t)}{\partial \tau_L} &= \frac{\partial J_P}{\partial \hat{x}} \frac{\partial \hat{x}(t)}{\partial \tau_L} \\ &= -2(\bar{x}_2(t) - \hat{x}_2(t)) \frac{\partial \hat{x}_2(t)}{\partial \tau_L} \end{aligned} \tag{13}$$

The sensitivity in the predictive states $S_{\hat{x}} = \frac{\partial \hat{x}}{\partial \tau_L} \in \mathbb{R}^7$ is computed by solving the partial derivative of the predictive BLDC motor dynamics $\dot{\hat{x}} = \hat{f}(\hat{x}, \tilde{\Theta}^r, K^r)$ (18) with respect to the load torque, and results in

$$\begin{aligned} \frac{d}{dt}\frac{\partial \hat{x}(t)}{\partial \tau_L} &= \left(\frac{\partial \hat{f}\left(\hat{x}, \tilde{\Theta}^r, K^r\right)}{\partial \hat{x}} \right) \frac{\partial \hat{x}}{\partial \tau_L} + \frac{\partial \hat{f}\left(\hat{x}, \tilde{\Theta}^r, K^r\right)}{\partial \tau_L} \\ &= \begin{bmatrix} 0 & \tilde{\Theta}_8^r & 0 & 0 & 0 \\ 0 & -\tilde{\Theta}_1^r & \bar{C}\tilde{\Theta}_2^r & \bar{D}\tilde{\Theta}_2^r & 0 \\ 0 & -\frac{k_p}{3}(Q+2W)\tilde{\Theta}_5^r + \frac{\bar{A}}{3}\tilde{\Theta}_3^r & -\tilde{\Theta}_4^r & 0 & \frac{k_i}{3L}(Q+2W) \\ 0 & \frac{k_p}{3}(-Q+W)\tilde{\Theta}_5^r + \frac{\bar{B}}{3}\tilde{\Theta}_3^r & 0 & -\tilde{\Theta}_4^r & \frac{k_i}{3L}(Q-W) \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \frac{\partial \hat{x}}{\partial \tau_L} - \begin{bmatrix} 0 \\ \tilde{\Theta}_6^r \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ \frac{d}{dt} S_{\hat{x}} &= \hat{f}_{R_P}(S_{\hat{x}}, \tilde{\Theta}^r, K^r) \end{aligned} \tag{14}$$

where $W = (\bar{\eta}(\theta) - \bar{\eta}(\theta - 2\pi/3))/2$ and $Q = (\bar{\eta}(\theta - 2\pi/3) - \bar{\eta}(\theta - 4\pi/3))/2$ are related to the switching sequence functions of the inverter.

Then, the robust predictive error $J_{P_R} \in \mathbb{R}$ under the effects of the load variations is expressed as in (15). A high value indicates that small load changes can significantly degrade tracking accuracy, whereas a low value implies robust performance.

$$J_{P_R}(t) = \left(\frac{\partial J_P(t)}{\partial \tau_L} \right)^2 \tag{15}$$

Then, the robust predictive problem can be interpreted as a trade-off between tracking performance and robustness. On one hand, the controller must minimize the predicted tracking error to accurately follow the reference trajectory within the prediction horizon. On the other hand, it must remain insensitive to load disturbances, ensuring that the selected control gains provide consistent performance under varying operating conditions. The formulation of the dynamic optimization problem for the robust predictive stage is shown in (16)-(21) for the time interval $t \in \hat{\Omega} \subseteq [t,\ t + \Delta w_P \Delta t] = [t_i,\ t_f]$. The aim is to find the controller robust gains $K^r \in \mathbb{R}^2$ based on the simultaneous minimization of the predictive error and its sensitivity, subject to the predictive state equations (18), the predictive state sensitivity dynamics (19) with their corresponding initial conditions (20), and box constraints (bounds) on the controller robust gains (21).

$$\min_{K^r \in \mathbb{R}^2} \mathcal{J}_{\mathcal{P}} \tag{16}$$

$$\mathcal{J}_{\mathcal{P}} = \int_{t \in \hat{\Omega}} J_P(t)\ J_{P_R}(t)\ dt \tag{17}$$

Subject to :

$$\dot{\hat{x}} = \begin{bmatrix} 0 & \tilde{\Theta}_8^r & 0 & 0 & 0 \\ 0 & -\tilde{\Theta}_1^r & \bar{C}\tilde{\Theta}_2^r & \bar{D}\tilde{\Theta}_2^r & 0 \\ 0 & \frac{\bar{A}}{3}\tilde{\Theta}_3^r & -\tilde{\Theta}_4^r & 0 & 0 \\ 0 & \frac{\bar{B}}{3}\tilde{\Theta}_3^r & 0 & -\tilde{\Theta}_4^r & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \hat{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{2}{3}\tilde{\Theta}_5^r & \frac{1}{3}\tilde{\Theta}_5^r \\ -\frac{1}{3}\tilde{\Theta}_5^r & \frac{1}{3}\tilde{\Theta}_5^r \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{ab} \\ V_{bc} \end{bmatrix} - \begin{bmatrix} 0 \\ \tilde{\Theta}_6^r\tilde{\Theta}_7^r \\ 0 \\ 0 \\ \bar{x}_2 \end{bmatrix} \tag{18}$$

$$\frac{d}{dt} S_{\hat{x}} = \hat{f}_{R_P}(S_{\hat{x}}, \tilde{\Theta}^r, K^r)| \tag{19}$$

$$\hat{x}(t_i) = x(t_i),\ S_{\hat{x}}(t_i) = \frac{\partial x(t_i)}{\partial \tau_L} \tag{20}$$

$$K^r_{min} \leq K^r \leq K^r_{max} \tag{21}$$

The dynamic optimization problem described in (16)-(21) is transformed from a continuous-time formulation into a discrete-time one [61] by converting the constraints associated with the BLDC motor (18) and sensitivity (19) dynamics (nonlinear differential equations) into a finite set of states through their solutions by the numerical methods for Ordinary Differential Equations (ODE solvers), and using the control signal parameterized by the PI controller. It is important to note that the feasibility of the dynamic constraints given by the system dynamics is ensured when they are solved by ODE solvers.

It is important to note that the computational time required to solve the resulting discrete-dimensional optimization problem is directly related to the integration horizon and the number of discretization points used in the numerical solution of the differential equations. Specifically, for a given integration time interval $[t_i, t_f]$ and integration step size Δt, the number of samples is given by Δw_I and Δw_P for the robust identification and prediction stages. Consequently, the computational burden increases proportionally with the number of integration steps, since the system dynamics and sensitivity equations must be evaluated at each discretization point during the optimization process.

3.3. Handling the design variable's dynamic variation as dynamic constraints

Controller tuning using evolutionary techniques requires solving dynamic optimization problems at each period $\overline{\Delta t}$ for both stages. Evolutionary techniques can yield different alternatives (solutions) for the design variable vector across time intervals due to the multi-modal performance function. When the system to be controlled is in steady state, the controller gains may shift at each $\overline{\Delta t}$, causing the system to deviate from the reference. Low-pass filters, as proposed in [10], are among the most commonly used strategies for handling variation in the design parameter vector obtained by the optimization technique during task execution. The filter is applied to the solution obtained from the optimization process, and the filtered solution is used as the controller for the real system. One drawback of using the filter strategy is that the filtered solution yields a delayed response and, consequently, may not be the most efficient.

In this work, the maximum variations of the design variable vector in the current time t with respect to the previous time $t - \overline{\Delta t}$ are set as constraints in both optimization stages. The design variable changes are selected to be at most twice the current design variable vector between the two time intervals ($t - \overline{\Delta t}$ and t). These dynamic constraints are formulated in (22) and (23), and are included in the identification (6)-(11) and prediction (16)-(21) stages, respectively. These constraints limit how much the system parameters and controller gains can change between consecutive time intervals. Intuitively, they prevent abrupt variations in the design variable vectors, ensuring smooth evolution and avoiding undesirable closed-loop behavior.

$$g_A(\tilde{\Theta}^r(t), t) : \frac{|\tilde{\Theta}^r(t) - \tilde{\Theta}^r(t - \overline{\Delta t})|}{\tilde{\Theta}^r(t)} - 1 \leq 0 \tag{22}$$

$$g_B(K^r(t), t) : \frac{|K^r(t) - K^r(t - \overline{\Delta t})|}{K^r(t)} - 1 \leq 0 \tag{23}$$

It is important to point out that in the constraints (22) and (23), the design variable vector associated with the current time t ($\tilde{\Theta}^r(t)$ or $K^r(t)$ is provided by the solution in the current generation. On the other hand, the design variable vector in the previous time ($\tilde{\Theta}^r(t - \overline{\Delta t})$ or $K^r(t - \overline{\Delta t})$) is obtained with the best results obtained in the optimization process of the previous time interval, i.e., in $t \in [t - \overline{\Delta t}, t]$. It is important to point out that the design variable vector $\tilde{\Theta}^r(t - \overline{\Delta t})$ *or* $K^r(t - \overline{\Delta t})$ is acquired through a

memory. The memory is explained in Section 3.4.1. The implementation of the evaluation of such constraints is detailed in lines 4 and 16 of Algorithm 1. The additional computational burden introduced by these constraints is minimal, since their evaluation requires only straightforward algebraic comparisons. It is important to mention that these constraints directly limit the relative change of the design variables between consecutive windows. Unlike post-processing filters, smoothness is enforced within the optimization process itself, preventing abrupt gain transitions.

3.4. Searching for solutions in promising regions

The robust identification and predictive stages of the AMCT-PR, formulated as a dynamic optimization problem, are solved at each $\overline{\Delta t}$ using the Niching Chaotic Online Differential Evolution (NCODE) algorithm. The proposed NCODE is based on the DE/rand/1/bin variant, incorporating niching in the mutation process, memory, and a chaotic number generator instead of a general-purpose pseudo-random number generator such as Mersenne Twister (MT). Conceptually, NCODE preserves temporal coherence in controller gains by biasing the search toward historically successful regions stored in memory. Instead of re-optimizing from scratch at each window, the algorithm refines previously validated solutions while allowing controlled exploration when necessary. The details of the NCODE are presented next.

3.4.1. Differential Evolution

As evidenced by the published specialized literature, Differential Evolution (DE), presented by Storn and Price in 1997 [63], is a well-known bio-inspired, population-based optimizer that has proven to be successful in solving complex optimization problems, particularly those involving real-world applications [64, 65, 66]. Systematically, DE perturbs candidate solutions using scaled differences between individuals. As population diversity decreases, update magnitudes naturally decrease, favoring smoother parameter evolution in adaptive controller tuning.

DE works as follows: An initial random population $\mathbf{X}^{G=1} \in \mathbb{R}^{NP \times D}$ of NP vectors, called individuals $\chi \in \mathbb{R}^D$, is established. Each vector comprises the design-variable set and represents a possible solution to the optimization problem. Once the initial population is created, the population (called the parent population) evolves by emulating natural selection and the survival of the fittest. This emulation creates an offspring population through mutation

and crossover, and the offspring compete with the parent population according to the selection process. An elitism mechanism based on the feasibility rules proposed by Deb [67] is employed in the selection process, in which the fittest individual between the parent χ_i^G and the offspring μ_i^G is selected to survive to the next generation.

Considering a minimization problem, the fittest individual between a pair is selected according to the following rules:

- Between two feasible solutions, the fittest solution is preferred (the one with the lowest value in the objective function).
- A feasible solution is always preferred over an unfeasible one.
- Between two unfeasible solutions, the one with the fewer violated constraints is selected. If they meet the same number of violated constraints, the fittest solution is randomly selected.

The constraint function ϕ is related to the number of violated constraints. This function is given in (24), where the $k-th$ inequality constraint is represented by g_k, and the total number of inequality constraints is given by n_g.

$$\phi = \sum_{k=1}^{n_g} \max(0, sign(g_k(\chi_i^G))) \tag{24}$$

The above processes are repeated until the maximum number of generations G_{max} is reached. The last population must contain the most suitable design variable vector, and the best among them, according to the feasibility rules, is selected for the next time interval $\overline{\Delta t}$.

The pseudo-code of the NCODE is given in Algorithm 1, and the particulars related to the incorporation of niching into the mutation process, the memory, and the chaotic number generator are detailed next.

Algorithm 1 ensures parameter continuity in time by inserting the previous best solution into the population. Clustering identifies distinct regions in the search space, and mutation is preferentially performed within the niche containing the memory. Moreover, dynamic constraints regulate inter-window variation, while feasibility-based selection preserves the best consistent solution.

Algorithm 1: Pseudo-code of the proposed NCODE optimizer. The algorithm incorporates dynamic memory-based niche identification, chaotic number generation, and constraint handling to solve the dynamic optimization problems in both tuning stages.

Data: Maximum generations G_{max}, population size NP, crossover rate CR, scaling factor F, cluster number CN, memory Π_α and the current state of the chaotic dynamics z_c.
Result: The best solution (Π_α).

$G = 1$

Chaotically generate an initial population $\mathbf{X}^G$ with $NP-1$ candidate vectors within the search space.

foreach $i = 1$ *to* $NP-1$ $\wedge$ $j = 1$ *to* D **do**

$\chi_{i,j}^G = \chi_{min,j} + Sine(z^c)(\chi_{max,j} - \chi_{min,j})$

Insert the memory Π_α into the last member of the population, corresponding to the individual NP.

Evaluate the performance function $\mathcal{J}_\alpha(\chi_i^G) \mid \alpha \in \mathcal{I}, \mathcal{P}$ and the dynamic constraints (22)-(23), requiring $\tilde{\Theta}^r(t - \overline{\Delta t}) \leftarrow \Pi_\mathcal{I}$ or $K^r(t - \overline{\Delta t}) \leftarrow \Pi_\mathcal{P}$ with $\tilde{\Theta}^r(t) \leftarrow \chi_i^G$ or $K^r(t) \leftarrow \chi_i^G$. Then, compute the constraint function $\phi(\chi_i^G)$ of the population $\mathbf{X}^G$.

while $G \leq G_{max}$ **do**

Group the population $\mathbf{X}^G$ in CN cluster by using Kmeans algorithm.

Select the cluster in the most promising region (it is related to the closeness of the individual NP) and store its member in $\boldsymbol{\Upsilon}$.

$F = F_{min} + Sine(z^c)(F_{max} - F_{min})$

foreach $\chi_i^G \in \mathbf{X}^G$ **do**

Generate a mutant vector ν_i^G with three chaotically picked individuals from the selected cluster $\boldsymbol{\Upsilon}$, according to (25). If the cluster does not contain enough members, individuals are selected chaotically from the entire population.

$$\nu_i^G = \overbrace{\chi_{r_1}^G}^{\text{Base vector}} + \underbrace{F(\chi_{r_2}^G - \chi_{r_3}^G)}_{\text{Scaled difference vector}} \quad \mid r_1 \neq r_2 \neq r_3 \neq \chi_i^G \in \boldsymbol{\Upsilon} \tag{25}$$

$jrand = \lceil D\ Sine(z^c) \rceil$

Generate an offspring vector μ_i^G using (26).

foreach $j = 1$ *to* D **do**

$$\mu_{i,j}^G = \begin{cases} \nu_{i,j}^G & \text{if } Sine(z^c) < CR \textbf{ or } j = j_{rand} \\ \chi_{i,j}^G & \text{otherwise} \end{cases} \tag{26}$$

Apply the bound constraint-handling method using random technique [68] in the offspring population $\mathbf{v}^G$.

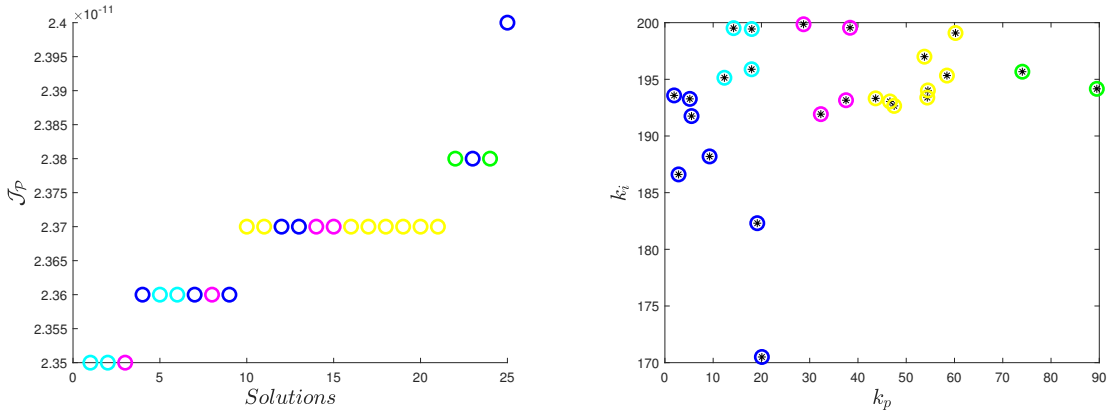
Evaluate $\mathcal{J}_\alpha(\mu_i^G) \mid \alpha \in \mathcal{I}, \mathcal{P}$ and the dynamic constraints (22)-(23), requiring $\tilde{\Theta}^r(t - \overline{\Delta t}) \leftarrow \Pi_\mathcal{I}$ or $K^r(t - \overline{\Delta t}) \leftarrow \Pi_\mathcal{P}$ with $\tilde{\Theta}^r(t) \leftarrow \mu_i^G$ or $K^r(t) \leftarrow \mu_i^G$. Then, compute the constraint function $\phi(\mu_i^G)$.

Select the vector between χ_i^G and μ_i^G to incorporate it in $\mathbf{X}^{G+1}$ based on the feasibility rules.

$G = G + 1$

Get the best vector from $\mathbf{X}^G$ and update the memory $\Pi_\alpha \leftarrow best(\mathbf{X}^G) \mid \alpha \in \mathcal{I}, \mathcal{P}$.

return Π_α



(a) Performance function space. (b) Design variable space.

Figure 3: Distribution of candidate solutions in a representative optimization window. The objective function and design variable spaces illustrate the multi-modal nature of the AMCT formulation and motivate the use of niche identification.

Niching mutation. It is important to note that both the robust identification and predictive optimization problems given above present multimodal performance functions in the adaptive method for PI controller tuning of the BLDC motor. To illustrate this, Figure 3 represents the population taken from the last generation for the time $t \in \hat{\Omega}$ in the predictive stage provided by the DE algorithm. Visually, four local solutions and one global solution are identified (five valleys in total in the objective function space). Some valleys contain more than one solution. The population could then be partitioned into five clusters. K-means is applied in the design-variable space for this purpose. K-means is an unsupervised learning technique widely used in many applications to partition data into CN clusters whose members share similar features [69]. The colors in the figure represent five different clusters in the population. It is observed that the obtained valleys (Fig. 3a) contain more than one cluster despite belonging to the same performance function. This confirms that the AMCT is a multimodal optimization problem [70], i.e., it admits multiple optimal solutions (global and local) associated with a single objective function. Thus, the selection of the solution may significantly alter the design variable (design-variable variation), which in turn influences the closed-loop system performance, as mentioned in the introduction.

Based on confirmation that the AMCT in the BLDC motor is a multimodal optimization problem, the Dynamic Memory-Based Niche Identifica-

tion (DMNI) mechanism is incorporated to find the individuals in the most promising regions of the population, which are randomly selected in the mutation process. To identify the promising region at the current time in the DMNI, the information (design vector) previously found in the past time window, along with its evolution, is taken into account. In the first generation, $G = 1$, the promising region corresponds to the niche that includes the memory (the previously found design vector). In subsequent generations ($G > 1$), the promising region is allowed to evolve dynamically. Such evolution is determined by whether the parent individual from the previous generation is related to the memory (i.e., the best solution obtained in the previous optimization window) or its corresponding offspring exhibits greater fitness (or its evolution) and, therefore, a higher probability of surviving the selection stage. So, the promising region always corresponds to the niche that contains the memory evolution.

Using information from the DMNI mechanism, a mutant population is created using the differential mutation operator DE/rand/1 [71], except that it uses individuals from the promising niche. The mutation operator is denoted by DE/Nrand/1 (25). The term Nrand denotes the random selection of individuals within the niche, and the last term denotes a unique difference vector. This operator consists of adding a scaled difference vector between two randomly picked individuals ($\chi_{r_2}^G$ and $\chi_{r_3}^G$) from the niche to the base vector $\chi_{r_1}^G$ (individual) randomly obtained from the same niche. The individuals $\chi_{r_1}^G$, $\chi_{r_2}^G$, $\chi_{r_3}^G$ must be different from each other and different from the current individual χ_i^G. If the individuals in the niche lack sufficient information to perform the mutation, they are selected from the entire population, thereby promoting exploration of the search space. The proposed niching mutation is detailed in lines 6, 7, and 10 of Algorithm 1, where the unsupervised learning algorithm K-means++ [72], a variant of the original K-means with an improved initialization, is used to partition the population into CN clusters based on the similarity of the design variable vector. For small datasets with NP elements, the empirical rule $CN = \sqrt{NP/2}$ is adopted to estimate an appropriate number of clusters [73]. This heuristic mitigates over-segmentation while preserving the ability to distinguish distinct promising regions in multimodal search spaces, thereby maintaining an adequate exploration-exploitation balance.

It is important to note that the DMNI mechanism is executed at each generation within each optimization interval $\overline{\Delta t}$. Since the adaptive framework performs optimization every $\overline{\Delta t} = 5.0 \times 10^{-3}$ s, the niching procedure

is naturally embedded in the online loop and dynamically updated as the population evolves. An empirical timing analysis shows that the clustering stage in the DMNI mechanism represents approximately 13.95% of the total computational time during identification and 8.89% during controller tuning. Therefore, the niching procedure incurs a moderate overhead while remaining compatible with the adaptive framework's real-time constraints.

Memory. One important key in the AMCT based on evolutionary computation is the use of a memory Π_α in the identification and predictive stages ($\Pi_{\alpha=\mathcal{I}}$ and $\Pi_{\alpha=\mathcal{P}}$, respectively), through the dynamic environment [10]. This memory helps achieve the optimum within a short, acceptable time interval given by $\overline{\Delta t}$, facilitating the initial search and avoiding starting from scratch [74]. This memory also identifies the promising region, since this region is selected according to this memory, allowing the niche to evolve coherently across time intervals while preserving temporal continuity in the design variables. The memory is incorporated at the beginning of the optimization process and updated with the fittest individual at the end of the process, as suggested in [29]. These steps are detailed in lines 3 and 19 of Algorithm 1.

Incorporation of chaotic sequences. The Pseudo Random Number Generator (PRNG) used in the DE/Nrand/1 algorithm is based on a single-state Sine chaotic map with parameter $\mu = 4$, as indicated in [75]. The initial condition of the Sine chaotic map is randomly chosen with the Merssene Twister Random Number Generator (MTRNG) in the interval $[0, 1]$ at the beginning of the closed-loop system evolution (when the simulation time is zero), and its dynamics evolve during each function call of the Sine map-based Chaotic Number Generator (SCNR). The chaotic behavior of the number generator in the evolution of the DE algorithm increases the performance of the search process of the optimizer and reduces the computational burden according to [75]. The Sine chaotic function is given in Algorithm 2.

3.5. Implementation of the AMCT-PR

The closed-loop diagram of the AMCT-PR applied to the BLDC motor is given in Fig. 4. The pseudocode of the complete implementation of the proposal in the closed-loop system is given in Algorithm 3 with the interrupt routine provided in Algorithm 4.

In each execution, the behavior of the BLDC motor is simulated at $t \in [0.0, 3.0]$ (s), considering a sampling interval of $\Delta t = 5.0e-6$ (s) given by the

Algorithm 2: Sine map-based chaotic number generator (SCNG) used in NCODE. The procedure updates the chaotic state and generates pseudo-random values used for population initialization, mutation, and crossover operations.

Data: Current state vector of the chaotic dynamics $z^c \in \mathbb{R}^2$.
Result: A chaotic number in the interval $(0, 1)$ and the current state of chaotic dynamics z^c.

1 Evaluate the next chaotic dynamics z^n by using (27) and setting the Sine map parameter as $\mu = 4$.

$$z_1^n = \tfrac{1}{4}\mu \, \sin(\pi \, z_1^c) \tag{27}$$

2 Set $z^c = z^n$ as global variable.
3 **return** z^n

interrupt routine and the controller gains provided by the memory $\Pi_{\mathcal{I}}$. The *ode1* method is used to numerically integrate the motor's dynamic equations during the simulation. It is important to mention that, before the first $\overline{\Delta t}$, the fixed gains $k_p = 85.9045$ and $k_i = 131.79$ of the PI controller are used in the proposed strategy to induce initial control signals to the motor and gather enough information to carry out the optimization processes at each $\overline{\Delta t}$. These fixed gains are obtained by a robust optimization process, which is explained in Section 4.2.4.

The optimization processes involved in the proposed adaptive control strategy are carried out every $\overline{\Delta t} = 5.0e-3$ (s). Thus, for each $\overline{\Delta t}$, the optimization process is first performed to identify the motor parameters using $\Delta w_I = 10$ past samples of the acquired states and sensitivities, with sampling interval Δt. The obtained motor parameter vector is stored in the memory $\Pi_{\mathcal{I}}$. Immediately after, the optimization is performed for tuning the PI controller in the robust predictive stage, in which $\Delta w_p = 10$ future samples are predicted, considering the dynamic parameters stored in the memory $\Pi_{\mathcal{I}}$ and the sampling interval Δt. The controller parameters are stored in the memory $\Pi_{\mathcal{P}}$, which is used later at each interrupt routine (Δt).

Both problems are independently solved by the Niching Chaotic Online DE presented in Algorithm 1. The search space of the robust identification problem is constrained by $\tilde{\Theta}_{min} = [\frac{\bar{b}_0}{2\bar{J}_0}, \frac{\bar{k}_m}{2\bar{J}_0}, \frac{\bar{k}_e}{2\bar{L}_a}, \frac{\bar{R}_a}{2\bar{L}_a}, \frac{1}{2\bar{L}_a}, \frac{1}{2\bar{J}_0}, 0, \frac{P}{2}]$ and $\tilde{\Theta}_{max} = [\frac{2\bar{b}_0}{\bar{J}_0}, \frac{2\bar{k}_m}{\bar{J}_0}, \frac{2\bar{k}_e}{\bar{L}_a}, \frac{2\bar{R}_a}{\bar{L}_a}, \frac{2}{\bar{L}_a}, \frac{2}{\bar{J}_0}, 2, 2P]$, which are calculated with the nominal values of the motor dynamic parameters in Table 3. The nominal val-

Table 3: BLDC motor nominal parameters.

Parameter	Nominal value
b_0	$1.8144e-04$ ($\mathrm{kg\,m^2}$)
$\bar{J}_0$	$5.0600e-04$ ($\mathrm{kg\,m^2}$)
$\bar{L}_a$	$1.0700e-03$ (H)
$\bar{R}_a$	$8.4400e-01$ (Ω)
$\bar{k}_m$	$2.3100e-01$ ($\mathrm{N\,m/A}$)
$\bar{k}_e$	$2.0700e-01$ ($\mathrm{V\,s/rad}$)

ues were established based on the manufacturer's specifications. On the other hand, the box constraints for the robust predictive problem are given by $K^r_{min} = [0, 0]$ and $K^r_{max} = [200, 200]$. The values of the NCODE hyperparameters used in this work are the number of generations per optimization process of $G_{max} = 10$, the population size $NP = 25$, the fixed crossover rate $CR = 0.5$, and the scaling factor $F \in [0.3, 0.9]$, randomly calculated at the beginning of each generation. These hyperparameters were established based on the suggestions in [29, 76]. In addition, three clusters $CN = \lfloor\sqrt{NP/2}\rfloor = 3$. are selected in the K-means++ algorithm with a maximum number of iterations $epochs = 100$ (the latter is used in case the convergence of groups is not achieved before). All pseudo-random numbers used in NCODE are generated from the SCNG, except for the initial state of the SCNG, which in each experiment uses the MTRNG, as can be seen in line 2 of Algorithm 3.

4. Results

4.1. Experimental conditions

The experiments presented in this section evaluate the proposed AMCT-PR's performance in BLDC motor speed regulation over time, considering adverse conditions that may occur in real operating environments.

The simulation scenario is designed to approximate real operating conditions. To conduct these experiments, a test scenario is considered in which the brushless motor is subject to time-varying parametric uncertainties, load torque disturbances, abrupt variations in the desired speed profile, actuator saturation constraints, and measurement noise in the acquired states and their sensitivities. The behavior of the dynamic parameters of the BLDC motor, subject to uncertainties, is shown in Table 4. This table lists the

Algorithm 3: Complete implementation of the Adaptive Method for Controller Tuning based on Promising Regions (AMCT-PR) in the closed-loop BLDC motor system. The algorithm integrates the robust identification and predictive stages, NCODE optimization, dynamic memory updating, and constraint handling within the online adaptive control loop.

Data: Final execution time t_f, integration time (sampling interval) Δt, optimization interval $\overline{\Delta t}$, identification/prediction time window Δw_I, Δw_P, desired speed profile $\bar{x}_2$, NCODE parameters G_{max}, NP, CR, F and CN.

Result: Closed-loop simulation.

Initialize the time $t = 0$, the real BLDC motor parameter vector Θ, the estimated BLDC motor parameter vector $\tilde{\Theta}^r$, the controller parameter vector K^r and $n = 1$.

Set the chaotic state $z^c = rand(0, 1)$ as a global variable.

Update the memory $\Pi_\alpha \mid \alpha \in \{\mathcal{I}, \mathcal{P}\}$ with the parameter vectors $\tilde{\Theta}^r$, K^r.

while $t < t_f$ **do**

  if $t \geq n\overline{\Delta t}$ **then**

    n=n+1

    Acquire the states $x(t)$.

    Robust identification stage:

    Solve (6)-(11) with (22)-(23) by using NCODE (Algorithm 1).

    Robust predictive stage:

    Solve (16)-(21) with (22)-(23) by using NCODE (Algorithm 1).

    Update the memory $\Pi_\alpha \mid \alpha \in \{\mathcal{I}, \mathcal{P}\}$ with the parameter vectors $\tilde{\Theta}^r$, K^r.

Algorithm 4: Interrupt routine executed at each sampling interval Δt within the AMCT-PR framework. The routine updates the BLDC motor states, applies the current robust controller gains, and triggers the adaptive optimization process when the predefined optimization update interval is reached.

Data: Current state vector $x(t)$, current time t and integration time Δt.

Result: Next state vector $x(t + \Delta t)$ and next time $t + \Delta t$.

Acquire the vector K^r from the memory $\Pi_{\mathcal{P}}$.

Solve the nonlinear differential equation of the BLDC motor dynamics $\dot{x} = f(x, \Theta, K^r)$ (1) with the Euler's method for the next time $t + \Delta t$.
$x(t + \Delta t) = x(t) + \Delta t \frac{dx(t)}{dt}$

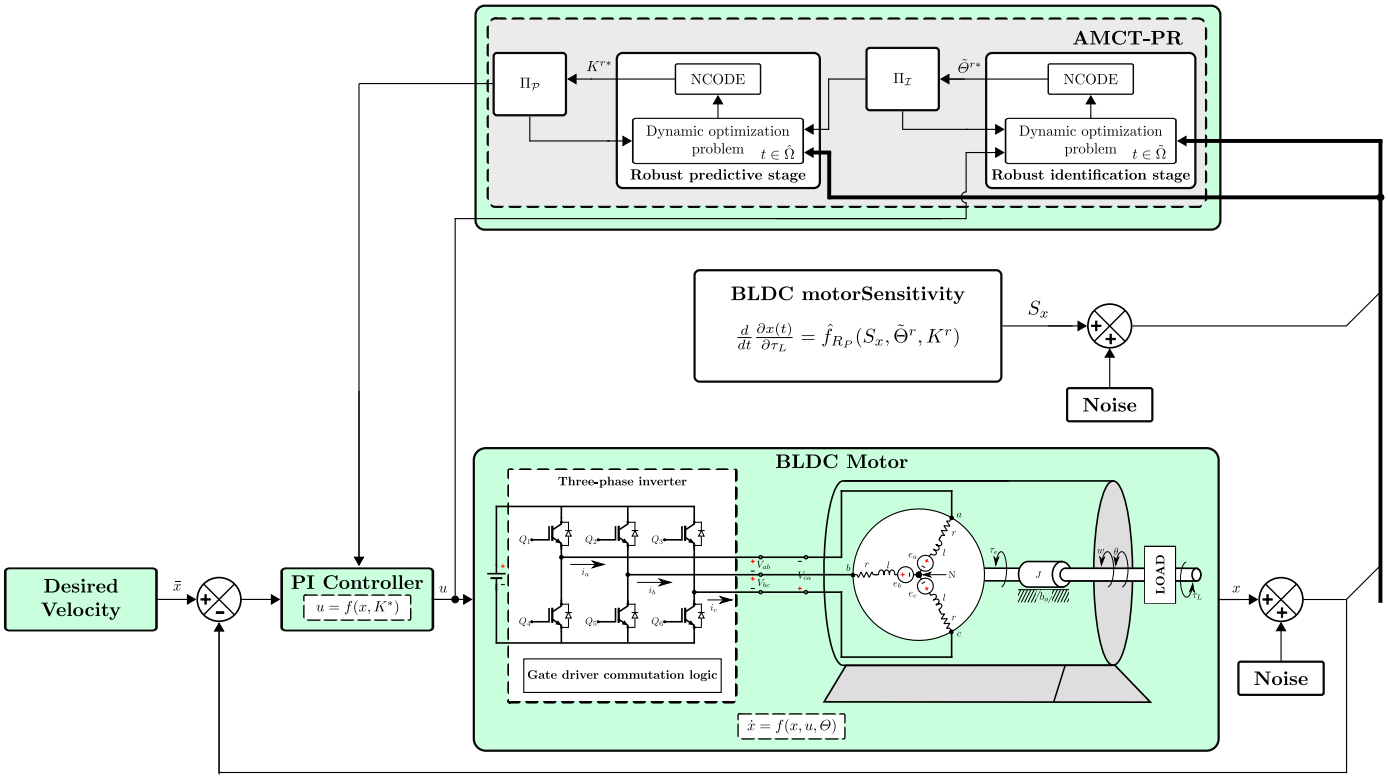


Figure 4: Schematic diagram of the proposed Adaptive Method for Controller Tuning based on Promising Regions (AMCT-PR). The diagram illustrates the interaction between the robust identification stage, the robust predictive stage, the NCODE optimizer with dynamic memory-based niche identification, and the closed-loop BLDC motor system within the online adaptive framework.

motor parameters with their uncertain values used in the experiments, updated from the nominal values as a function of time t. The uncertainties were chosen assuming an adverse case in which the motor parameters vary periodically and rapidly by $\pm$ 10% of their nominal values. As for the rest of the motor parameters, the number of poles is considered fixed at $P = 11$, and the maximum voltage supported is $V_{max} = 250$ (V). On the other hand, the load torque τ_L is disturbed according to (28) and the desired speed profile varies as indicated by (29). Finally, random noise is included in the acquired states of ± 0.01 for x_1; ± 0.1 for x_2; and ± 0.001 for x_3, x_4, and x_5. Likewise, the sensitivities include a random noise of ± 0.01 in Sx_1; ± 0.1 in Sx_2; and ± 0.001 in Sx_3, Sx_4, and Sx_5.

Table 4: BLDC motor parameters used during experimentation.

Parameter	Experimental value
b_0	$\bar{b}_0 + 0.1 b_0 \cos(\pi t)$
J_0	$\bar{J}_0 + 0.1 \bar{J}_0 \cos(2\pi t/3)$
L_a	$\bar{L}_a + 0.1 \bar{L}_a \cos(\pi t)$
R_a	$\bar{R}_a + 0.1 \bar{R}_a \cos(2\pi t/3)$
k_m	$\bar{k}_m + 0.1 \bar{k}_m \cos(2\pi t)$
k_e	$\bar{k}_e + 0.1 \bar{k}_e \cos(2\pi t)$

$$\tau_L = \begin{cases} 1.0 \text{ (N m)}, & 0.5 \text{ (s)} \leq t \leq 2.5 \text{ (s)} \\ 0.0 \text{ (N m)}, & otherwise \end{cases} \tag{28}$$

$$x_{2,d} = \begin{cases} 150.0 \text{ (rad/s)}, & t < 1.0 \text{ (s)} \\ 100.0 \text{ (rad/s)}, & 1.0 \text{ (s)} \leq t < 2.0 \text{ (s)} \\ 125.0 \text{ (rad/s)}, & t \geq 2.0 \text{ (s)} \end{cases} \tag{29}$$

4.2. Comparative results

Due to the stochastic nature of the NCODE algorithm, performance will vary across independent executions of AMCT-PR. To assess the behavior of the proposed strategy, 30 independent runs are performed under the same experimental conditions described in the previous section. The performances obtained with AMCT-PR are compared with other tuning methods:

- A robust Chaotic Adaptive Tuning Strategy for Controller Gains (CATSCG (R)): This method online tunes the controller with the Chaotic Online Differential Evolution (CODE). CODE operates in the same way as NCODE but without incorporating the dynamic memory-based niche-identification mechanism. Also, to handle abrupt variations in the design variables, CATSCG (R) uses digital filters [10] rather than incorporating the dynamic constraints and the DMNI mechanism adopted by AMCT-PR.

- A non-robust version of AMCT-PR (AMCT-PR (NR)): The only difference between the AMCT-PR (NR) and the proposed AMCT-PR is the performance functions of the optimization problems in the identification and predictive stages. The AMCT-PR (NR) omits the robust objective functions J_{I_R} (5) and J_{P_R} (15) during the computation of the

identification and predictive performance indexes $\mathcal{J}_{\mathcal{I}}$ (6) and $\mathcal{J}_{\mathcal{P}}$ (17), i.e., the values of the robust objective functions remain as unit values.

- A variant of AMCT-PR using Hierarchical Density-Based Spatial Clustering of Applications with Noise (AMCT-PR (HDBSCAN)): This behaves similarly to AMCT-PR; however, instead of using K-means++ as a niching strategy, it employs HDBSCAN [77], a widely used state-of-the-art density-based clustering algorithm. HDBSCAN constructs a hierarchical density-based clustering structure and extracts stable clusters without requiring the number of clusters to be specified a priori. Unlike K-means++, it does not rely on an iterative centroid-update procedure. In this tuning method, HDBSCAN is configured to generate clusters containing at least four solutions to ensure their proper exploitation within the NCODE mutation operator. At each generation, the cluster containing the current best individual is selected as the promising region, rather than the niche containing the memory evolution.

- A Robust Tuning Approach for Controller Gains (RTACG): The RTACG [78] is an offline controller tuning method based on optimization [9] used to optimize the PI controller gains for the BLDC motor, i.e., once these gains are optimized, they remain fixed. In RTACG, the robust performance function $\mathcal{J}_{\mathcal{P}}$ and the same optimization problem presented in (16)-(21) are considered in the interval $[0, 3]$ (s). The optimizer used in RTACG is also CODE.

It is important to emphasize that CATSCG (R), AMCT-PR (NR), and AMCT-PR (HDBSCAN) are executed under the same computational budget and stopping criteria as AMCT-PR to ensure a fair comparison. Furthermore, identical optimizer settings are adopted across these online methods, including the optimization interval, population size, number of generations per call, crossover rate, scaling factor, and number of independent runs. In the case of RTACG, unlike the other methods, the tuning process is performed offline and is therefore not subject to strict time constraints. Consequently, a more generous evaluation budget is allocated to CODE to ensure high-quality tuning results. Specifically, the maximum number of generations is set to $G_{max} = 1000$ and the population size to $NP = 50$, while the remaining parameters are kept unchanged. Although RTACG benefits from a larger evaluation budget, this does not compromise fairness, as it operates

offline and is not subject to the real-time computational constraints imposed on adaptive online methods. Under these settings, CODE is executed 30 times, and the best set of PI gains obtained is selected for the comparative analysis, as RTACG is designed as a one-time offline tuning procedure. The resulting gains are $k_p = 85.9045$ and $k_i = 131.79$, which are also employed as the initial gains for all online methods.

Table 5 provides a concise overview of the five tuning approaches considered in the comparative study, summarizing their main features, key assumptions, and computational complexity drivers to facilitate a quick understanding of their differences and trade-offs.

Table 5: Summary comparison of the tuning approaches considered in the experimental evaluation, including main features, assumptions, and computational complexity drivers.

Feature	AMCT-PR	CATSCG (R)	AMCT-PR (NR)	AMCT-PR (HDBSCAN)	RTACG
Tuning mode	Online adaptive	Online adaptive	Online adaptive	Online adaptive	Offline
Optimizer	NCODE (CODE with K-means++ niching)	CODE (without niching)	NCODE (CODE with K-means++ niching)	NCODE (CODE with HDBSCAN niching)	CODE
Niching mechanism	K-means++ memory-driven (DMNI)	None	K-means++ memory-driven (DMNI)	HDBSCAN density-based	None
Robust objective	Yes (sensitivity-based)	Yes (sensitivity-based)	No	Yes (sensitivity-based)	Yes (sensitivity-based)
Gain variation control	Dynamic constraints embedded in the optimization loop	Digital filter applied as post-processing	Dynamic constraints embedded in the optimization loop	Dynamic constraints embedded in the optimization loop	Not required (fixed gains)
Key assumption	Accurate plant model required; sufficient computational budget per $\overline{\Delta t}$	Accurate plant model required; filter parameters must be set a priori	Accurate plant model required; sufficient computational budget per $\overline{\Delta t}$	Accurate plant model required; sufficient computational budget per $\overline{\Delta t}$	Accurate plant model required; operating conditions assumed stationary
Computational complexity drivers	Function evaluations, K-means++ clustering, dynamic constraint updates per window	Function evaluations and digital filtering per window	Function evaluations and dynamic constraint updates per window	Function evaluations, HDBSCAN clustering, dynamic constraint updates per window	Function evaluations only (offline, larger budget)

The results obtained by comparing AMCT-PR with the aforementioned controller tuning schemes after 30 executions are shown in Table 6. These results are given in terms of the Integral Squared Error (ISE) control performance index and its standard deviation (Std.) of the speed regulation task, calculated from the first stationary state $\overline{\Delta t}$, i.e., after 5 (ms). The best results are highlighted in gray.

On the other hand, to draw strong conclusions about the distributions of the results of the five compared methods, this paper uses the Wilcoxon pairwise test with a statistical significance level of 5% [79]. The results of the Wilcoxon tests are shown in Table 7, where the first column denotes the comparative test performed, and the $p - value$ column indicates the probability value. If the $p - value$ of two test samples shows no significant differences, a symbol ($\approx$) is included at the beginning of the corresponding $p-value$. In cases where two samples show significant differences, i.e., when the $p-value$ is less than the significance level of 5%, the $p-value$ is shown with (+), indicating that the first sample performs better than the second one. The inclusion of (−) in the $p - value$ means the second sample outperforms the first. The winning alternatives are highlighted in gray in this table. As a supplement to Table 7, Table 8 summarizes each alternative's wins obtained in the Wilcoxon tests, and the best one is indicated in gray. Additionally, the best and worst closed-loop behavior of the proposed AMCT-PR and the compared adaptive methods for controller tuning are presented in Figures 5 and 6.

In the following subsections, the comparative results of the proposal with other previously mentioned AMCT strategies are discussed to highlight the advantages and limitations of the proposed tuning strategy for each aspect.

4.2.1. Comparing the proposal with the AMCT with filters

It is interesting to examine the advantages of the proposed AMCT-PR, which derive from its mechanism for identifying the most suitable region of solutions: incorporating the dynamic memory-based niche identification mechanism into the optimizer and addressing special dynamic constraints in robust identification and prediction problems. In the first instance, the comparison is with respect to CATSCG (R).

Thus, the performance obtained from 30 runs of AMCT-PR is compared with that achieved from 30 runs of CATSCG (R). This comparison evaluates the proposal against a similar state-of-the-art approach that solves the same optimization problems without incorporating the DMNI mechanism in

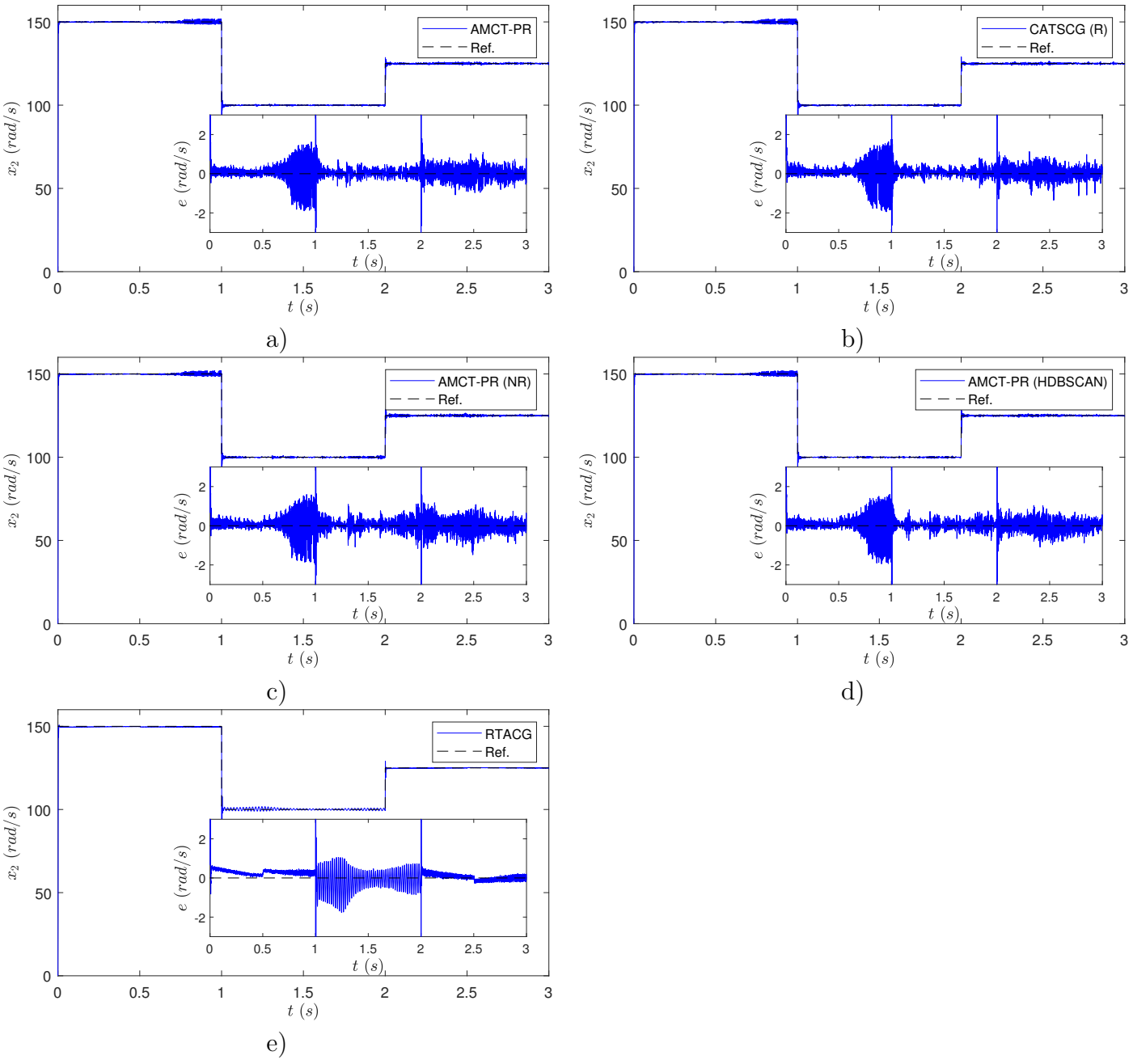


Figure 5: Best closed-loop speed responses obtained over 30 independent runs for each tuning approach.

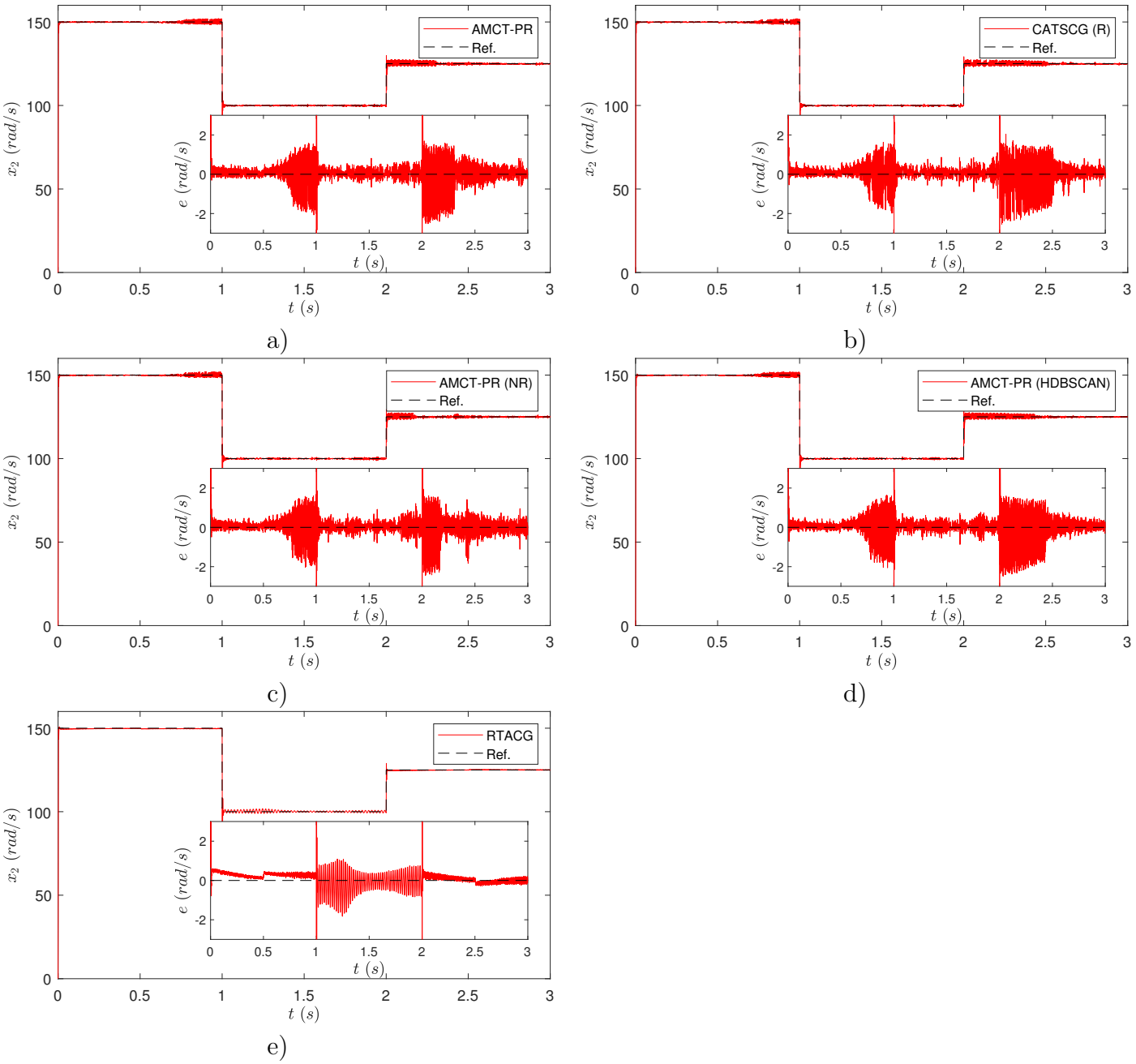


Figure 6: Worst closed-loop speed responses obtained over 30 independent runs, illustrating robustness differences among methods.

the optimizer and the special dynamic constraints to handle design-variable variations. Instead, CATSCG (R) applies a digital filter [10] at the end of each optimization process to reduce variations in controller parameters.

Table 6 shows that the average ISE value obtained by AMCT-PR is approximately 17.33% lower than that of CATSCG (R). On the other hand, although both alternatives exhibit comparable variability, AMCT-PR presents a lower standard deviation. The results of the Wilcoxon tests in Table 7 confirm that AMCT-PR is significantly better than CATSCG (R).

The best and worst closed-loop behaviors of AMCT-PR and CATSCG (R) are shown in Figs. 5a) and 5b), and Figs. 6a) and 6b). The figures illustrate noticeable differences in CATSCG (R) between the velocity profiles from the best and worst runs. On the other hand, the proposed AMCT-PR shows more consistent behavior.

The observed performance differences can be associated with two main design features of AMCT-PR. First, the special dynamic constraints limit excessive deviations of the design variables from previously optimized values (beyond 100%) in previous processes, thereby preventing larger oscillations in the system due to abrupt changes in their values. Additionally, the dynamic constraints do not excessively condition the design variables as the filter does, so the variables do not tend to smooth out or become overly similar to their previous values. This is useful since values that are too far from the previous ones are not lost but can help compensate for uncertainties or disturbances.

Secondly, using the DMNI mechanism with the K-means++ clustering-based technique facilitates the exploration and exploitation of design space regions in the NCODE, capturing characteristics similar to both the best found in a previous optimization process and its possible evolution during optimization. The special dynamic constraints and DMNI mechanism yield solutions that preserve some of the benefits of the previous best alternatives

Table 6: Descriptive statistical performance results over 30 independent runs, including central tendency and dispersion metrics for all compared tuning methods.

Scheme	Mean ISE	Std. ISE
AMCT-PR	2.3335	2.0011e-01
CATSCG (R)	2.8228	2.7095e-01
AMCT-PR (NR)	2.5473	1.8672e-01
AMCT-PR (HDBSCAN)	2.3609	2.2316e-01
RTACG	2.4167	5.0235e-03

Table 7: Wilcoxon pairwise test results over performances of different tuning methods after 30 independent runs.

Test			$p-value$
AMCT-PR	vs	CATSCG (R)	(+) 1.8626e-08
AMCT-PR	vs	AMCT-PR (NR)	(+) 7.9790e-04
AMCT-PR	vs	AMCT-PR (HDBSCAN)	(≈) 5.9781e-01
AMCT-PR	vs	RTACG	(+) 1.9661e-02
CATSCG (R)	vs	AMCT-PR (NR)	(−) 3.4496e-04
CATSCG (R)	vs	AMCT-PR (HDBSCAN)	(−) 1.9912e-06
CATSCG (R)	vs	RTACG	(−) 2.5518e-07
AMCT-PR (NR)	vs	AMCT-PR (HDBSCAN)	(−) 5.5483e-04
AMCT-PR (NR)	vs	RTACG	(−) 8.7182e-04
AMCT-PR (HDBSCAN)	vs	RTACG	(≈) 9.6102e-02

Table 8: Summary of the Wilcoxon pairwise test results over performances of different tuning methods after 30 independent runs.

Scheme	Wins
AMCT-PR	3
RTACG	2
AMCT-PR (NR)	1
AMCT-PR (HDBSCAN)	2
CATSCG (R)	0

while introducing variations that improve control performance by managing uncertainties and perturbations. These mechanisms operate within the limited computational budget imposed by the online framework, enabling solution updates within the interval $\overline{\Delta t}$.

4.2.2. Comparing the proposal with the non-robust AMCT

The comparison between the proposed robust approach and its non-robust counterpart named AMCT-PR (NR) aims to assess whether incorporating robustness into the identification and predictive optimization problems improves performance under adverse experimental conditions, particularly load torque variations (τ_L) and other sources of uncertainty and noise.

Across 30 independent runs, Table 6 shows that AMCT-PR outperforms

AMCT-PR (NR), with an average ISE of the speed regulation task that is 8.39% lower. The superiority of AMCT-PR over AMCT-PR (NR) is confirmed by the Wilcoxon test results in Table 7. These results suggest that incorporating robustness into the identification and predictive optimization stages yields solutions that are less sensitive to load variations and better suited for operation under adverse conditions.

It is also observed in Table 6 that the standard deviation of the results in AMCT-PR is larger than that of AMCT-PR (NR), indicating slightly higher dispersion across runs. This difference is attributed to the fact that the optimization problems solved with the non-robust tuning approach are considerably less complex than those addressed by the robust formulation (AMCT-PR). Although the same number of objective function evaluations is performed in both approaches, the additional sensitivity-based terms in the robust formulation increase the complexity of the optimization landscape, potentially leading to slightly greater variability in the solutions obtained across independent runs. Although the standard deviation of the proposed AMCT-PR increases, it outperforms AMCT-PR (NR) according to the Wilcoxon test. Table 7 also indicates that the results obtained with the non-robust tuning approach (AMCT-PR (NR)) are statistically similar to those achieved by the controller tuning method that incorporates robust optimization and filtering (CATSCG (R)), as no significant differences were detected according to the Wilcoxon test.

Figs. 5c) and 6c) show the best and worst performance of AMCT-PR (NR). In these figures, AMCT-PR (NR) presents a higher error amplitude when there are load disturbances in the time interval $[0.5, 2.5]$ (s) (see equation (28)), compared to the robust strategies AMCT-PR (Figs. 5a) and 6a)) and CATSCG (R) (Figs. 5b) and 6b)), which exhibit smaller error amplitudes during the load disturbance interval.

From the results presented above, it is important to note that the robustness of the proposed AMCT-PR is achieved primarily through the explicit incorporation of sensitivity minimization into the optimization formulation. In both the identification and predictive stages, the objective functions include sensitivity terms with respect to load torque variations, promoting controller parameters that maintain stable closed-loop performance even under disturbances and parametric uncertainty in the state and sensitivity vectors. Additionally, the Dynamic Memory-Based Niche Identification mechanism and the dynamic constraints that regulate the variation of design variables between consecutive optimization windows prevent abrupt changes in the

controller gains. These strategies promote smoother evolution of controller gains, mitigate the closed-loop regulation error commonly observed in adaptive tuning approaches, and yield controller gains that are less sensitive to load variations.

4.2.3. Comparing the proposal with the AMCT-PR based on HDBSCAN

The comparison between the proposed AMCT-PR and its variant using HDBSCAN as the niching strategy allows evaluation of whether replacing K-means++ with a density-based clustering mechanism provides a practical advantage in the adaptive tuning process.

Across 30 independent runs, Table 6 shows that AMCT-PR achieves a slightly lower average ISE than AMCT-PR (HDBSCAN), with a difference of approximately 1.16%. However, the Wilcoxon test results in Table 7 indicate that there are no statistically significant differences between the two approaches. This suggests that the overall closed-loop performance obtained with either niching strategy is comparable under the experimental conditions evaluated.

The best and worst closed-loop responses for AMCT-PR and AMCT-PR (HDBSCAN) are presented in Figs. 5a) and d), and Figs. 6a) and d), respectively. In both cases, the transient and steady-state responses are very similar. Although slight differences in error amplitude are observed during load disturbances, neither strategy shows a clear superiority in handling abrupt perturbations.

From these results, it can be concluded that replacing K-means++ with HDBSCAN does not significantly modify the overall control performance. Both niching mechanisms can identify promising regions in the design space and promote adaptive gain updates within the limited computational budget. Therefore, the main advantage of the proposed AMCT-PR lies not only in the specific clustering algorithm employed but in the integration of the niching mechanism with the special dynamic constraints and the robust optimization formulation.

Although both clustering strategies yield statistically comparable control performance, K-means++ provides a simpler, more transparent mechanism for identifying promising regions in the design space. In the present low-dimensional optimization problem, K-means++ provides stable partitioning with explicit control over the number of clusters, facilitating its integration into the NCODE framework. Therefore, K-means++ is a suitable and reliable choice for the adaptive tuning scheme, while HDBSCAN is a valuable

alternative, confirming the robustness of the proposed approach.

4.2.4. Comparing the proposal vs a fixed-gain controller tuning

The comparison between AMCT-PR and RTACG allows us to contrast the adaptive tuning strategy with an optimization-based offline method that provides fixed controller gains for the control system.

Tables 6-8 show the descriptive and inferential statistics of the 30 independent executions of RTACG. The same column organization previously described is followed. The results indicate that AMCT-PR achieves a lower average ISE than RTACG, with a relative improvement of approximately 3.44%. In particular, the average ISE value for RTACG ranks second in the comparisons and indicates that its standard deviation is significantly lower than those of the other control alternatives. The low standard deviation observed in RTACG is explained by the deterministic behavior of the fixed-gain PI controller, in which performance variations are primarily due to measurement noise rather than changes in controller parameters.

Figures 5e) and 6e) show the behavior of the best and worst performance with the RTACG. Here, it can be seen that the PI controller has difficulty handling load disturbances with the RTACG, i.e., discontinuous perturbations (see the load disturbances in the interval 0.5 (s)-2.5 (s). These observations suggest that under more persistent load disturbances, a fixed-gain strategy such as RTACG may exhibit greater performance degradation than adaptive tuning approaches. Overall, these results emphasize the advantages of adaptive tuning over fixed-gain optimization when operating under varying disturbance conditions.

4.2.5. Evolution of the controller parameters

This section analyzes the evolution of the controller parameters under different tuning approaches, with particular attention to their variability.

Fig. 7 illustrates the dispersion of the optimized proportional and integral gains in the design variable space during the closed-loop execution of the median run. Each point represents the gains obtained by the tuning approach at each tuning time interval $\overline{\Delta t}$. Since both the robust proposal AMCT-PR and the non-robust AMCT-PR (NR) rely on the same NCODE framework, which is based on K-means++ with special dynamic constraints, the resulting sets of gains exhibit similar dispersion patterns. In the case of AMCT-PR (HDBSCAN), the gains appear more structurally grouped, with

Table 9: Descriptive statistics of consecutive gain distance, quantifying gain smoothness across 30 independent runs.

Scheme	Mean $\overline{k_p}$	Std $\overline{k_p}$	Var $\overline{k_p}$	Mean $\overline{k_i}$	Std $\overline{k_i}$	Var $\overline{k_i}$	Total Var
AMCT-PR	190.73	1.0272	1.0551	157.12	2.1735	4.7243	5.7794
CATSCG (R)	189.72	1.4682	2.1557	126.38	2.6632	7.0928	9.2485
AMCT-PR (NR)	188.02	0.9802	0.9608	158.46	2.2678	5.1431	6.1039
AMCT-PR (HDBSCAN)	192.05	0.7788	0.6065	156.49	2.6347	6.9418	7.5483
RTACG	85.9045	0	0	131.79	0	0	0

fewer intermediate solutions dispersed across the central region of the parameter space. This behavior may be related to HDBSCAN's density-based clustering mechanism, which does not require a predefined number of clusters. It is also observed that the filter mechanism in CATSCG modifies the distribution and evolution of the computed controller gains.

To further analyze the temporal variation of the controller gains, Fig. 8 shows the design variable vector magnitude during the closed-loop execution of the median of the 30 runs. To quantitatively assess the variability of the controller gains across independent executions, a statistical analysis was performed on the 30 runs obtained for each tuning method. For every run, the average gains $\overline{k_p}$ and $\overline{k_i}$ were computed over the entire closed-loop simulation. This provided 30 representative gain pairs per method. The global mean, variance, standard deviation, and total variance of the controller gain vector were then calculated to quantify the dispersion of the optimized controller parameters across runs. The corresponding results are reported in Table 9.

Table 9 shows that AMCT-PR reduces the dispersion of the controller gains, as reflected by the lowest total variance across runs, followed by AMCT-PR (NR), AMCT-PR (HDBSCAN), in that order. The AMCT-PR (HDBSCAN) variant shows slightly higher dispersion, indicating larger gain adjustments between consecutive tuning intervals in Fig. 8. The CATSCG (R) exhibits the largest variation in the controller gains. As expected, the offline RTACG method yields constant gains, resulting in zero dispersion. These variations in controller gains are also quantified as the percentage reduction in gain variability of the comparative approaches relative to the proposed AMCT-PR method. The corresponding reductions are 37.50%, 23.43%, and 5.31% for CATSCG (R), AMCT-PR (HDBSCAN), and AMCT-PR (NR), respectively.

It is important to clarify the conceptual difference between CATSCG (R)

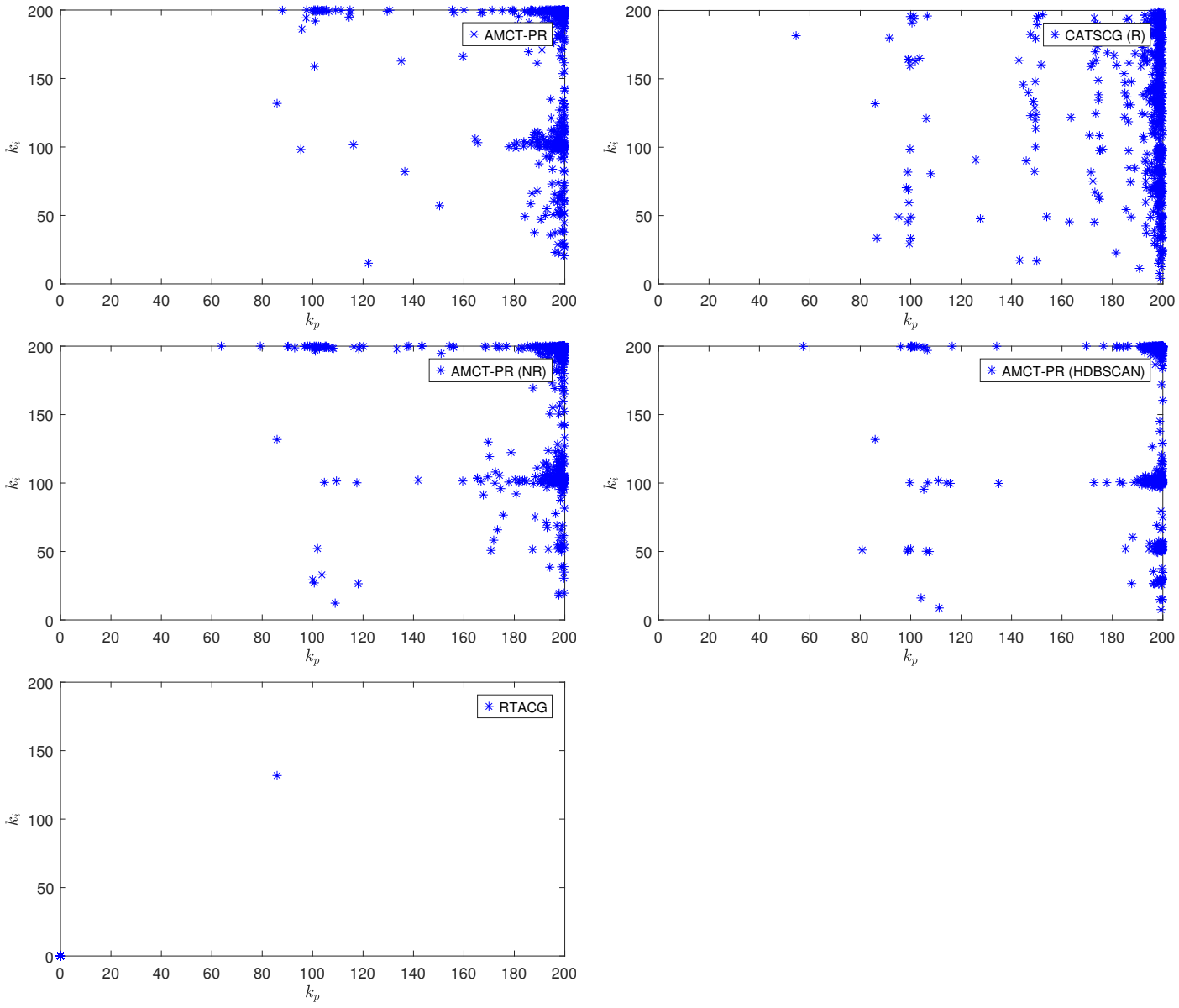


Figure 7: Time evolution of controller gain dispersion for the median-performance run of each tuning approach, illustrating gain variability throughout the adaptive process.

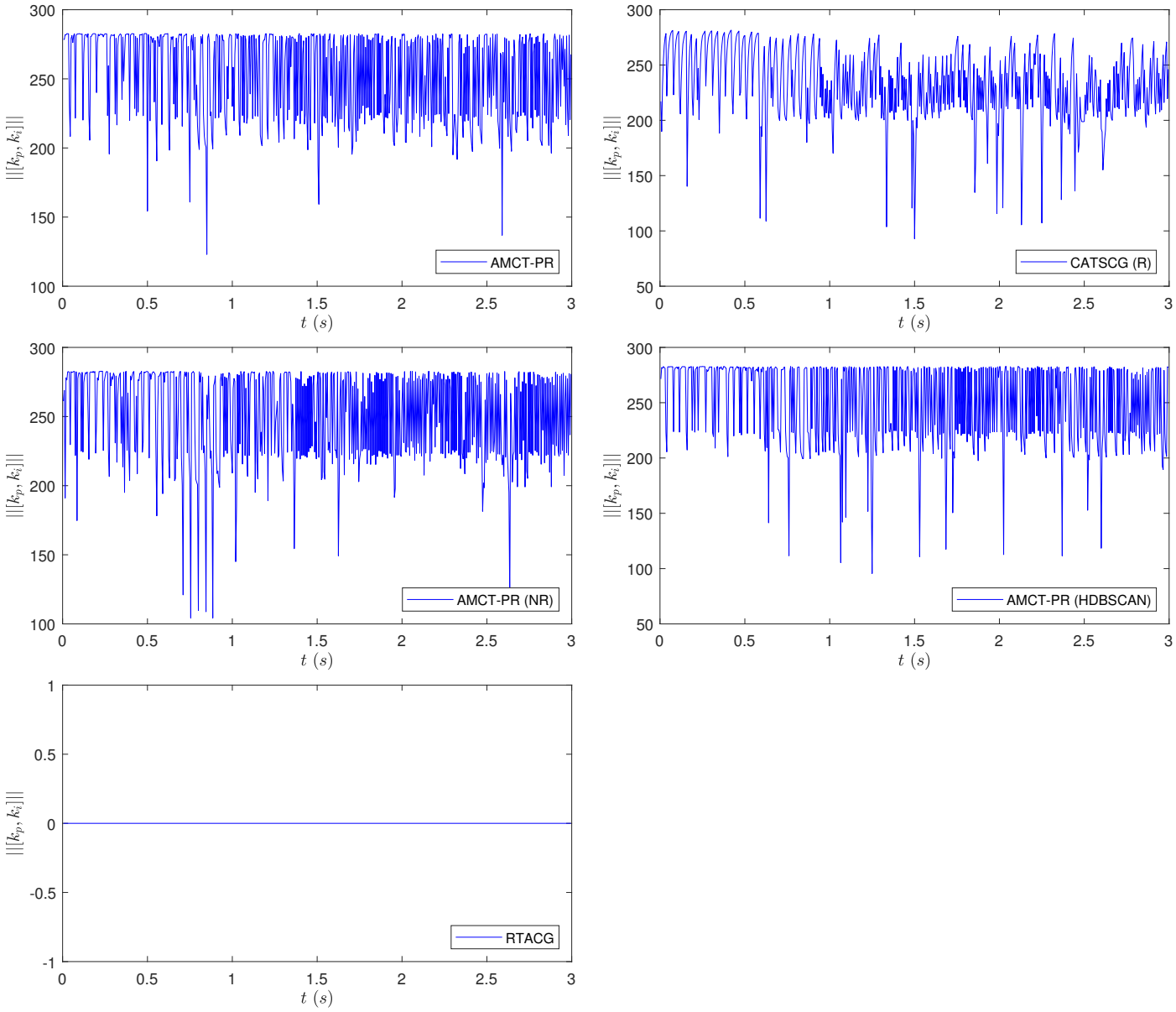


Figure 8: Time evolution of the norm of the optimized controller gains for the median-performance run of each approach, illustrating the magnitude and stability of gain adaptation throughout the tuning process.

and the proposed AMCT-PR. In CATSCG (R), the optimization process is exclusively directed toward minimizing the objective function. Once the optimal design variable vector is obtained, a digital filtering stage is applied a posteriori to smooth abrupt variations in the controller gains. However, this filtering mechanism does not explicitly consider the closed-loop performance of the filtered gains. In other words, the optimization and filtering stages are decoupled: the optimizer minimizes the objective function without accounting for the effects of subsequent filtering on the actual system performance. As a consequence, and as observed in the ISE results reported in Table 6, the final filtered gains are not necessarily those that best improve closed-loop behavior, despite also enhancing robustness to load disturbances in their formulation.

In contrast, AMCT-PR addresses the gain selection problem in an integrated manner. By incorporating the proposed special dynamic constraints directly into the dynamic optimization problems, together with the Niching Chaotic Online Differential Evolution (NCODE) and the DMNI mechanism employed to solve them, the method promotes the selection of the most suitable gain vector within promising regions of the search space. In this control tuning, the dynamic constraints regulate the admissible gain variations during the optimization process, rather than modifying the solution afterward, as in the filtering mechanism. Moreover, the inclusion of sensitivity-based robust terms in the identification and predictive objective functions explicitly drives the optimizer toward gain configurations that reduce the closed-loop system's sensitivity to load variations. Consequently, this unified formulation allows AMCT-PR to simultaneously improve control performance, enforce smooth gain evolution within the optimization loop, and enhance robustness against disturbances.

4.3. Limitations and practical considerations

The results demonstrate that the proposed AMCT-PR improves closed-loop performance while promoting smooth controller-gain evolution and robustness to load disturbances by incorporating realistic conditions, including measurement noise, time-varying parametric uncertainties, and actuator saturation limits. Nevertheless, the proposed framework assumes that a reasonably accurate dynamic model of the plant can be obtained to support the identification and predictive stages. So, the efficiency of the proposed approach relies on the availability of such a model, since both the identification and predictive stages require solving the system differential equations

during the optimization process. Consequently, the applicability of the proposed framework may be limited in systems where the dynamics are poorly known or difficult to model, potentially degrading the predictive stage's performance and leading to suboptimal controller updates. In such cases, the niche identification performed by the DMNI mechanism may become less reliable, since the promising regions are defined relative to solutions obtained from an inaccurate model, potentially increasing gain variability and reducing the consistency of closed-loop performance.

On the other hand, the computational cost is directly influenced by the numerical integration of the system dynamics and sensitivity equations within each optimization iteration. As described in Section 3.2, the computational burden increases proportionally with the number of discretization points and the prediction horizon, since the system dynamics must be evaluated at each step of the optimization process. In this work, it is assumed that the available computational capacity is sufficient to handle this process, a condition that is increasingly feasible given current technological advances in computing hardware. However, this assumption may not always hold in practice, and the resulting computational demand may limit real-time applicability in systems with fast dynamics or limited computational resources. Among the possible strategies to improve computational efficiency, parallelization of function evaluations is particularly promising for the proposed framework, since the population-based structure of DE allows independent evaluation of individuals across time windows.

To mitigate these limitations, future work may explore data-driven or surrogate modeling as complementary alternatives to physics-based models, particularly in scenarios where explicit system identification is impractical. The choice among these approaches should be guided by the trade-off among model fidelity, data availability, and computational cost [80]. Regarding computational efficiency, additional improvements can be achieved through reduced population size, adaptive prediction horizons, and more efficient numerical solvers.

5. Conclusions and future work

This paper presents the Adaptive Method for Controller Tuning based on Promising Regions (AMCT-PR), which synergistically incorporates an unsupervised learning-based convergence enhancement mechanism (DMNI) into the differential evolution's variation operator and dynamic constraints

to improve the explorative-exploitative capability in promising regions of the search space. This approach encourages more efficient and reliable controller tuning, thereby improving closed-loop performance. The proposed AMCT-PR is applied to the BLDC motor PI controller to account for the performance function's sensitivity to load variations during the identification and prediction stages.

The inferential statistics show that the proposed AMCT-PR outperforms the three comparative control tuning approaches across 30 executions. In particular, the proposed method evidences an improvement of up to 17.33% in the mean ISE control performance index for the speed regulation task. The proposal can also efficiently obtain solutions in promising regions of the search space without requiring a filtering process on the design variables at the end of the optimization. The convergence of the solution toward the most promising regions, driven by the DMNI mechanism and the dynamic constraints, allows the controller gains to evolve smoothly while preserving suitable closed-loop performance. In particular, the proposal reduces the variation in controller gains by up to 37.5% compared with state-of-the-art controller tuning approaches. Moreover, the proposed method improves handling of load uncertainties (discontinuous perturbations) by incorporating sensitivity-based terms into the performance function. Therefore, the AMCT-PR achieves a balanced trade-off between adaptability and smooth evolution of gains, ensuring reliable online controller tuning in multi-modal and dynamically changing optimization landscapes. Consequently, the obtained controller gains exhibit improved robustness against load disturbances, contributing to more consistent closed-loop performance under varying operating conditions.

On the other hand, although the proposed method is evaluated on the PI speed controller of a BLDC motor, the AMCT-PR can be extended to other nonlinear dynamic systems and other controller structures, provided that the mathematical model of the plant dynamics is available.

Future work involves the application of the AMCT-PR on a testbed platform. This may be considered experimental validation on physical BLDC platforms and other classes of electric drives to assess robustness under real measurement noise, actuator saturation, and hardware limitations. Another direction for future research is to extend the proposed AMCT-PR framework into a preference-driven multiobjective dynamic optimization formulation, in which interactive or learning-based preference-handling approaches may be incorporated to infer priority structures from historical performance. This

would provide a structured and decision-oriented mechanism for selecting solutions in the multimodal and time-varying search landscape.

References

[1] P. Fleming, R. Purshouse, Evolutionary algorithms in control systems engineering: a survey, Control Engineering Practice 10 (11) (2002) 1223–1241. doi:https://doi.org/10.1016/S0967-0661(02)00081-3.
URL https://www.sciencedirect.com/science/article/pii/S0967066102000813

[2] D. R. Lewin, Evolutionary algorithms in control system engineering, IFAC Proceedings Volumes 38 (1) (2005) 45–50, 16th IFAC World Congress. doi:https://doi.org/10.3182/20050703-6-CZ-1902.00868.
URL https://www.sciencedirect.com/science/article/pii/S147466701636880X

[3] G. Reynoso-Meza, X. B. Ferragud, J. S. Saez, J. M. H. Durá, Controller tuning with evolutionary multiobjective optimization: A holistic multiobjective optimization design procedure, Springer, Switzerland, 2017.

[4] A. Rodríguez-Molina, E. Mezura-Montes, M. G. Villarreal-Cervantes, M. Aldape-Pérez, Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem, Applied Soft Computing 93 (2020) 106342. doi:https://doi.org/10.1016/j.asoc.2020.106342.
URL https://www.sciencedirect.com/science/article/pii/S1568494620302829

[5] S. Joseph, E. Dada, A. Abidemi, D. Oyewola, B. Khammas, Metaheuristic algorithms for pid controller parameters tuning: Review, approaches and open problems, Heliyon 8 (5) (2022) e09399.

[6] S. R. Nekoo, J. Ángel Acosta, A. Ollero, A search algorithm for constrained engineering optimization and tuning the gains of controllers, Expert Systems with Applications 206 (2022) 117866. doi:https://doi.org/10.1016/j.eswa.2022.117866.
URL https://www.sciencedirect.com/science/article/pii/S0957417422011216

[7] D. Fister, J. Šafarič, I. Fister, R. Šafarič, I. Fister, Online adaptive controller based on dynamic evolution strategies, Applied Sciences 8 (11) (2018). doi:10.3390/app8112076.
URL https://www.mdpi.com/2076-3417/8/11/2076

[8] M. G. Villarreal-Cervantes, A. Rodríguez-Molina, O. Serrano-Pérez, Novel asynchronous activation of the bio-inspired adaptive tuning in the speed controller: Study case in dc motors, IEEE Access 9 (2021) 138976–138993. doi:10.1109/ACCESS.2021.3118658.

[9] M. G. Villarreal-Cervantes, J. Alvarez-Gallegos, Off-line pid control tuning for a planar parallel robot using de variants, Expert Systems with Applications 64 (2016) 444–454. doi:https://doi.org/10.1016/j.eswa.2016.08.013.
URL https://www.sciencedirect.com/science/article/pii/S0957417416304080

[10] M. G. Villarreal-Cervantes, A. Rodríguez-Molina, C.-V. García-Mendoza, O. Peñaloza-Mejía, G. Sepúlveda-Cervantes, Multi-objective on-line optimization approach for the dc motor controller tuning using differential evolution, IEEE Access 5 (2017) 20393–20407. doi:10.1109/ACCESS.2017.2757959.

[11] C. Cruz, J. R. González, D. A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Computing 15 (2011) 1427–1448.

[12] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24. doi:https://doi.org/10.1016/j.swevo.2012.05.001.
URL https://www.sciencedirect.com/science/article/pii/S2210650212000363

[13] M. Mavrovouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: Algorithms and applications, Swarm and Evolutionary Computation 33 (2017) 1–17. doi:https://doi.org/10.1016/j.swevo.2016.12.005.
URL https://www.sciencedirect.com/science/article/pii/S2210650216302541

[14] X.-Y. Chen, H. Zhao, J. Liu, A network community-based differential evolution for multimodal optimization problems, Information Sciences 645 (2023) 119359. doi:https://doi.org/10.1016/j.ins.2023.119359.
URL https://www.sciencedirect.com/science/article/pii/S0020025523009441

[15] S. Das, S. Maity, B.-Y. Qu, P. Suganthan, Real-parameter evolutionary multimodal optimization — a survey of the state-of-the-art, Swarm and Evolutionary Computation 1 (2) (2011) 71–88.

doi:https://doi.org/10.1016/j.swevo.2011.05.005.
URL https://www.sciencedirect.com/science/article/pii/S221065021100023X

[16] S. Cheng, X. Wang, M. Zhang, X. Lei, H. Lu, Y. Shi, Solving multimodal optimization problems by a knowledge-driven brain storm optimization algorithm, Applied Soft Computing 150 (2024) 111105. doi:https://doi.org/10.1016/j.asoc.2023.111105.
URL https://www.sciencedirect.com/science/article/pii/S1568494623011237

[17] L. Huang, Y. Ding, M. Zhou, Y. Jin, K. Hao, Multiple-solution optimization strategy for multirobot task allocation, IEEE Transactions on Systems, Man, and Cybernetics: Systems 50 (2020) 4283–4294.
URL https://api.semanticscholar.org/CorpusID:53703179

[18] S. Pearson, S. V. Hum, Optimization of electromagnetic metasurface parameters satisfying far-field criteria, IEEE Transactions on Antennas and Propagation 70 (5) (2022) 3477–3488. doi:10.1109/TAP.2021.3137272.

[19] D.-K. Woo, J.-H. Choi, M. Ali, H.-K. Jung, A novel multimodal optimization algorithm applied to electromagnetic optimization, IEEE Transactions on Magnetics 47 (6) (2011) 1667–1673. doi:10.1109/TMAG.2011.2106218.

[20] Z. Lei, S. Gao, Z. Zhang, M. Zhou, J. Cheng, Mo4: A many-objective evolutionary algorithm for protein structure prediction, IEEE Transactions on Evolutionary Computation 26 (3) (2022) 417–430. doi:10.1109/TEVC.2021.3095481.

[21] S. Elsayed, R. Sarker, T. Ray, C. C. Coello, Consolidated optimization algorithm for resource-constrained project scheduling problems, Information Sciences 418-419 (2017) 346–362. doi:https://doi.org/10.1016/j.ins.2017.08.023.
URL https://www.sciencedirect.com/science/article/pii/S0020025517308745

[22] A. Rodríguez-Molina, M. G. Villarreal-Cervantes, J. Álvarez Gallegos, M. Aldape-Pérez, Bio-inspired adaptive control strategy for the highly efficient speed regulation of the dc motor under parametric uncertainty, Applied Soft Computing 75 (2019) 29–45.

doi:https://doi.org/10.1016/j.asoc.2018.11.002.
URL https://www.sciencedirect.com/science/article/pii/S1568494618306288

[23] A. Rodríguez-Molina, M. G. Villarreal-Cervantes, M. Aldape-Pérez, Indirect adaptive control using the novel online hypervolume-based differential evolution for the four-bar mechanism, Mechatronics 69 (2020) 102384. doi:https://doi.org/10.1016/j.mechatronics.2020.102384.
URL https://www.sciencedirect.com/science/article/pii/S0957415820300647

[24] J. Šafarič, P. Bencak, D. Fister, R. Šafarič, I. Fister, Use of stochastic nature-inspired population-based algorithms within an online adaptive controller for mechatronic devices, Applied Soft Computing 95 (2020) 106559. doi:https://doi.org/10.1016/j.asoc.2020.106559.
URL https://www.sciencedirect.com/science/article/pii/S1568494620304981

[25] F.-J. Lin, H.-J. Shieh, K.-K. Shyu, P.-K. Huang, On-line gain-tuning ip controller using real-coded genetic algorithm, Electric Power Systems Research 72 (2) (2004) 157–169. doi:https://doi.org/10.1016/j.epsr.2004.03.013.
URL https://www.sciencedirect.com/science/article/pii/S037877960400104X

[26] D. Blanck-Kahan, G. Ortiz-Cervantes, V. Martínez-Gama, H. Cervantes-Culebro, J. E. Chong-Quero, C. A. Cruz-Villar, Neural-optimal tuning of a controller for a parallel robot, Expert Systems with Applications 236 (2024) 121184. doi:https://doi.org/10.1016/j.eswa.2023.121184.
URL https://www.sciencedirect.com/science/article/pii/S095741742301686X

[27] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, M. Sumner, A fast adaptive memetic algorithm for online and offline control design of pmsm drives, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37 (1) (2007) 28–41.

[28] D. Mohanraj, R. Aruldavid, R. Verma, K. Sathiyasekar, A. B. Barnawi, B. Chokkalingam, L. Mihet-Popa, A review of bldc motor: State of art, advanced control techniques, and applications, IEEE Access 10 (2022) 54833–54869. doi:10.1109/ACCESS.2022.3175011.

[29] A. Rodríguez-Molina, M. G. Villarreal-Cervantes, O. Serrano-Pérez, J. Solís-Romero, R. Silva-Ortigoza, Optimal tuning of the speed control

for brushless dc motor based on chaotic online differential evolution, Mathematics 10 (12) (2022).
URL https://www.mdpi.com/2227-7390/10/12/1977

[30] A. G. Rojas-López, M. G. Villarreal-Cervantes, A. Rodríguez-Molina, Surrogate indirect adaptive controller tuning based on polynomial response surface method and bioinspired optimization: Application to the brushless direct current motor controller, Expert Systems with Applications 245 (2024) 123070.
URL https://www.sciencedirect.com/science/article/pii/S0957417423035728

[31] M. K. Merugumalla, P. K. Navuri, Inertia weight strategies in pso for bldc motor drive control, in: G. Panda, S. C. Satapathy, B. Biswal, R. Bansal (Eds.), Microelectronics, Electromagnetics and Telecommunications, Springer Singapore, Singapore, 2019, pp. 475–484.

[32] M. García, P. Ponce, L. A. Soriano, A. Molina, B. MacCleery, D. Romero, Lifetime improved in power electronics for bldc drives using fuzzy logic and pso, IFAC-PapersOnLine 52 (13) (2019) 2372–2377.

[33] H. Jigang, F. Hui, W. Jie, A pi controller optimized with modified differential evolution algorithm for speed control of bldc motor, Automatika 60 (2) (2019) 135–148.

[34] K. Premkumar, B. Manikandan, Speed control of brushless dc motor using bat algorithm optimized adaptive neuro-fuzzy inference system, Applied Soft Computing 32 (2015) 403–419.

[35] K. Premkumar, B. V. Manikandan, C. A. Kumar, Antlion algorithm optimized fuzzy pid supervised on-line recurrent fuzzy neural network based controller for brushless dc motor, Electric Power Components and Systems 45 (20) (2017) 2304–2317.

[36] W. Xie, J.-S. Wang, H.-B. Wang, Pi controller of speed regulation of brushless dc motor based on particle swarm optimization algorithm with improved inertia weights, Mathematical Problems in Engineering 2019 (2019).

[37] T. Sivarani, S. J. Jawhar, C. A. Kumar, et al., Novel bacterial foraging-based anfis for speed control of matrix converter-fed industrial bldc mo-

tors operated under low speed and high torque, Neural Computing and Applications 29 (12) (2018) 1411–1434.

[38] H. Hu, T. Wang, S. Zhao, C. Wang, Speed control of brushless direct current motor using a genetic algorithm–optimized fuzzy proportional integral differential controller, Advances in Mechanical Engineering 11 (11) (2019) 1687814019890199.

[39] D. Potnuru, K. Alice Mary, C. Sai Babu, Experimental implementation of flower pollination algorithm for speed controller of a bldc motor, Ain Shams Engineering Journal 10 (2) (2019) 287–295.

[40] D. Potnuru, A. S. L. V. Tummala, Implementation of grasshopper optimization algorithm for controlling a bldc motor drive, in: J. Nayak, A. Abraham, B. M. Krishna, G. T. Chandra Sekhar, A. K. Das (Eds.), Soft Computing in Data Analytics, Springer Singapore, Singapore, 2019, pp. 369–376.

[41] K. Premkumar, B. Manikandan, Bat algorithm optimized fuzzy pd based speed controller for brushless direct current motor, Engineering Science and Technology, an International Journal 19 (2) (2016) 818–840.

[42] K. Vanchinathan, K. Valluvan, A metaheuristic optimization approach for tuning of fractional-order pid controller for speed control of sensorless bldc motor, Journal of circuits, systems and computers 27 (08) (2018) 1850123.

[43] T. Yigit, H. Celik, Speed controlling of the pem fuel cell powered bldc motor with fopi optimized by msa, International Journal of Hydrogen Energy 45 (60) (2020) 35097–35107, 4th International HYDROGEN TECHNOLOGIES Congress.

[44] M. Demirtas, Off-line tuning of a pi speed controller for a permanent magnet brushless dc motor using dsp, Energy Conversion and Management 52 (1) (2011) 264–273.

[45] H. Ibrahim, F. Hassan, A. O. Shomer, Optimal pid control of a brushless dc motor using pso and bf techniques, Ain Shams Engineering Journal 5 (2) (2014) 391–398.

[46] K. Prathibanandhi, C. Yaashuwanth, A. R. Basha, Improved torque performance in bldc-motor-drive through jaya optimization implemented on xilinx platform, Microprocessors and Microsystems 81 (2021) 103681.

[47] K. Premkumar, B. Manikandan, Ga-pso optimized online anfis based speed controller for brushless dc motor, Journal of Intelligent & Fuzzy Systems 28 (6) (2015) 2839–2850.

[48] B. N. Kommula, V. R. Kota, Direct instantaneous torque control of brushless dc motor using firefly algorithm based fractional order pid controller, Journal of King Saud University - Engineering Sciences 32 (2) (2020) 133–140.

[49] J. Shi, Q. Mi, W. Cao, L. Zhou, Optimizing bldc motor drive performance using particle swarm algorithm-tuned fuzzy logic controller, SN Applied Sciences 4 (2022) 293.
URL https://doi.org/10.1007/s42452-022-05179-6

[50] A. G. Rojas-López, M. G. Villarreal-Cervantes, A. Rodríguez-Molina, C. V. García-Mendoza, Offline optimum tuning of the proportional integral controller for speed regulation of a bldc motor through bio-inspired algorithms, in: Telematics and Computing, Springer International Publishing, Cham, 2020, pp. 169–184.

[51] A. Kanungo, C. Choubey, V. Gupta, P. Kumar, N. Kumar, Design of an intelligent wavelet-based fuzzy adaptive pid control for brushless motor, Multimedia Tools and Applications 82 (2023) 33203–33223.
URL https://doi.org/10.1007/s11042-023-14872-6

[52] T. Wang, H. Wang, H. Hu, J. Qing, C. Wang, Lion swarm optimisation-based tuning method for generalised predictive fractional-order pi to control the speed of brushless direct current motor, IET Electric Power Applications 16 (8) (2022) 879–895.

[53] F. A. Almasalmah, H. Omran, C. Liu, T. Poignonec, B. Bayle, Auto-tuning of model predictive control for bilateral teleoperation with bayesian optimization, IFAC-PapersOnLine 58 (30) (2024) 85–90, 5th IFAC Workshop on Cyber-Physical Human Systems. doi:https://doi.org/10.1016/j.ifacol.2025.01.161.
URL https://www.sciencedirect.com/science/article/pii/S2405896325001612

[54] M. Horváthová, K. Kiš, M. Klaučo, J. Oravec, Supervised learning for robust predictive control: Safe and tunable approach, Neurocomputing 671 (2026) 132637. doi:https://doi.org/10.1016/j.neucom.2026.132637.
URL https://www.sciencedirect.com/science/article/pii/S0925231226000342

[55] X. Zhao, Y. Sun, Y. Li, N. Jia, J. Xu, Applications of machine learning in real-time control systems: a review, Measurement Science and Technology 36 (1) (2024) 012003. doi:10.1088/1361-6501/ad8947.
URL https://doi.org/10.1088/1361-6501/ad8947

[56] B. Lakshminarayanan, F. Dettú, C. R. Rojas, S. Formentin, Inverse supervised learning of controller tuning rules, Automatica 178 (2025) 112356. doi:https://doi.org/10.1016/j.automatica.2025.112356.
URL https://www.sciencedirect.com/science/article/pii/S0005109825002493

[57] G. Bujgoi, D. Sendrescu, Tuning of pid controllers using reinforcement learning for nonlinear system control, Processes 13 (3) (2025). doi:10.3390/pr13030735.
URL https://www.mdpi.com/2227-9717/13/3/735

[58] V. Kenneth, Price. an introduction to differential evolution, New ideas in optimization (1999) 79–108.

[59] J. Chiasson, Modeling and High-Performance Control of Electric Machines, Wiley, 2005.

[60] R. Krishnan, Electric motor drives: modeling, analysis, and control, Vol. 626, Prentice Hall New Jersey, 2001.

[61] J. T. Betts, Practical methods for optimal control and estimation using nonlinear programming, 2nd Edition, Advances in Design and Control, SIAM, 2010.

[62] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering, Structural and multidisciplinary optimization 26 (6) (2004) 369–395.

[63] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.
URL https://doi.org/10.1023/A:1008202821328

[64] C.-Y. Lee, C.-H. Hung, Feature ranking and differential evolution for feature selection in brushless dc motor fault diagnosis, Symmetry 13 (7) (2021).

[65] L. Feng, J. Peng, Z. Huang, Gas system scheduling strategy for steel metallurgical process based on multi-objective differential evolution, Information Sciences 654 (2024) 119817. doi:https://doi.org/10.1016/j.ins.2023.119817.
URL https://www.sciencedirect.com/science/article/pii/S0020025523014020

[66] A. Hafiz Al Hariri, A. E. Khalifa, M. Talha, Y. Awda, A. Hasan, S. M. Alawad, Artificial neural network and differential evolution optimization of a circulated permeate gap membrane distillation unit, Separation and Purification Technology (2024) 126517doi:https://doi.org/10.1016/j.seppur.2024.126517.
URL https://www.sciencedirect.com/science/article/pii/S1383586624002569

[67] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2) (2000) 311–338.

[68] E. Juárez-Castillo, H.-G. Acosta-Mesa, E. Mezura-Montes, Empirical study of bound constraint-handling methods in particle swarm optimization for constrained search spaces, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 604–611. doi:10.1109/CEC.2017.7969366.

[69] M. Ahmed, R. Seraj, S. M. S. Islam, The k-means algorithm: A comprehensive survey and performance evaluation, Electronics 9 (8) (2020) 1295.

[70] K. Deb, A. Saha, Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 447–454. doi:10.1145/1830483.1830568.
URL https://doi.org/10.1145/1830483.1830568

[71] K. Opara, J. Arabas, Comparison of mutation strategies in differential evolution – a probabilistic perspective,

Swarm and Evolutionary Computation 39 (2018) 53–69. doi:https://doi.org/10.1016/j.swevo.2017.12.007.
URL https://www.sciencedirect.com/science/article/pii/S2210650217303310

[72] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, Tech. rep., Stanford (2006).

[73] P. Nerurkar, A. Shirke, M. Chandane, S. Bhirud, Empirical analysis of data clustering algorithms, Procedia Computer Science 125 (2018) 770–779.

[74] X. Peng, X. Gao, S. Yang, Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments, Soft Computing 15 (2) (2011) 311–326. doi:10.1007/s00500-010-0547-5.
URL https://doi.org/10.1007/s00500-010-0547-5

[75] M. F. Parra-Ocampo, O. Serrano-Pérez, A. Rodríguez-Molina, M. G. Villarreal-Cervantes, G. Hernández, M. E. Sánchez-Gutiérrez, V. M. Silva-García, Enhancing the performance in the offline controller tuning of robotic manipulators with chaos: a comparative study with differential evolution, International Journal of Dynamics and Control (2024) 1–38.

[76] E. Mezura-Montes, J. Velázquez-Reyes, C. A. Coello Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 485–492. doi:10.1145/1143997.1144086.
URL https://doi.org/10.1145/1143997.1144086

[77] R. J. G. B. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, in: J. Pei, V. S. Tseng, L. Cao, H. Motoda, G. Xu (Eds.), Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 160–172.

[78] O. Serrano-Pérez, M. G. Villarreal-Cervantes, A. Rodríguez-Molina, J. Serrano-Pérez, Offline robust tuning of the motion control for omnidirectional mobile robots, Applied Soft Computing 110 (2021)

107648. doi:https://doi.org/10.1016/j.asoc.2021.107648.
URL https://www.sciencedirect.com/science/article/pii/S156849462100569X

[79] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18. doi:https://doi.org/10.1016/j.swevo.2011.02.002.
URL https://www.sciencedirect.com/science/article/pii/S2210650211000034

[80] A. Aloisio, A. Contento, R. Alaggio, G. Quaranta, Physics-based models, surrogate models and experimental assessment of the vehicle–bridge interaction in braking conditions, Mechanical Systems and Signal Processing 194 (2023) 110276. doi:https://doi.org/10.1016/j.ymssp.2023.110276.
URL https://www.sciencedirect.com/science/article/pii/S0888327023001838