Applied Soft Computing Journal

A Compact Meta-Learned Neuro-Fuzzy Technique for Noise-Robust Nonlinear Control

--Manuscript Draft--

Manuscript Number:	ASOC-D-23-04348R2
Article Type:	Full Length Article
Keywords:	Hyper-plane; auto-adaptive; Neuro-fuzzy; Nonlinear; noise-resilient
Corresponding Author:	Md Meftahul Ferdaus University of New Orleans New Orleans, Louisiana UNITED STATES
First Author:	Md Meftahul Ferdaus
Order of Authors:	Md Meftahul Ferdaus
	Ahmad Jobran Al-Mahasneh, PhD
	Sreenatha G. Anavatti
	J. Senthilnath
Abstract:	Neuro-fuzzy systems show promise for adaptive control but can become complex due to the need to learn many parameters. This paper presents a resilient nonlinear controller that combines a simplified neuro-fuzzy system (Simp_NFS) and simplified neural network (Simp_NN) with only two meta-learnable parameters. This architecture enables fast and stable adaptation in uncertain nonlinear discrete-time systems. Simp_NFS utilizes interpretable hyperplane-based rules without antecedent parameters, simplifying the learning process to consequent weights. Simp_NN reduces complexity by replacing hidden-output weights with their mean. The hybrid auto-adaptive controller (HAC) combines the advantages of Simp_NFS and Simp_NN, significantly reducing the number of adaptive parameters compared to standard neuro-fuzzy methods for real-time control with limited resources. Simp_NFS provides structural adaptivity to handle uncertainties, while Simp_NN ensures reliable disturbance attenuation. The stability of HAC is proven using Lyapunov analysis. Extensive testing on challenging single-input single-output (SISO) and multi-input multi-output systems (MIMO) demonstrates that HAC improves performance by up to 82.55% compared to existing techniques. Key innovations include an ultra-compact meta-learned architecture, transparent online evolution of hyperplane clusters, and enhanced modeling capability for nonlinear uncertain systems. This interpretable neuro-fuzzy approach could enhance autonomy and safety by maintaining model transparency. The implementation of HAC will be publicly available on GitHub upon acceptance to promote open research.

Declaration of interests

☒The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Highlights of the research:

- Presents hyperplane neuro-fuzzy controller HAC with just 2 meta-learnable parameters for fast, stable adaptation.
- Achieves transparency through interpretable online evolution of hyperplane fuzzy rules without fixed architectures.
- Uniquely integrates structural and parametric adaptivity for flexible nonlinear uncertain system control.
- Rigorously proves closed-loop stability using Lyapunov analysis for robust disturbance rejection.
- Demonstrates precise tracking, faster convergence, noise resilience, and and statistical significance surpassing existing methods on SISO and MIMO plants.

Mario Köppen
Kyushu Institute of Technology, 680-4, Kawazu, Iizuka, Fukuoka 820-8502 JAPAN, 804-8550, Kitakyushu Japan

Editor-in-Chief
Applied Soft Computing

I'm resubmitting our revised manuscript titled "A Compact Meta-Learned Neuro-Fuzzy Technique for Noise-Robust Nonlinear Control" for your consideration for publication in Applied Soft Computing. We appreciate the constructive feedback from the reviewers, which has enhanced the quality and presentation of this research.

We propose a new neuro-fuzzy controller, the Hybrid Auto-Adaptive Controller (HAC), for controlling nonlinear discrete-time systems. HAC integrates a simplified neuro-fuzzy system and neural network to achieve an ultra-compact architecture with only 2 meta-learnable parameters. This enables fast, stable on-the-fly learning suitable for real-time control applications with limited resources.

In response to the reviewers' comments, we made several important enhancements to the manuscript. We expanded the comparative results across plant types, improved readability and language, incorporated additional studies on fuzzy learning models, added a table of symbols and a flowchart to illustrate the algorithm. The manuscript now includes a detailed discussion of assumptions and their practical implications in the Lyapunov stability analysis. We also analyzed the trade-offs between interpretability and complexity, and discussed limitations.

HAC's interpretable online evolution of hyperplane fuzzy rules without expert knowledge is an innovation. Lyapunov analysis proves closed-loop stability and precise tracking of dynamic references. Experiments on SISO and MIMO platforms show improved tracking accuracy over state-of-the-art methods.

This paper advances intelligent control by combining structural and parametric adaptivity in a human-interpretable design. HAC's innovations in evolving transparent hyperplane neuro-fuzzy systems align with Applied Soft Computing's scope and readership.

Thank you for considering the revised submission. We are open to additional feedback to further improve the work. We hope the enhancements make the manuscript suitable for publication in Applied Soft Computing.

Sincerely,
Md Meftahul Ferdaus, corresponding author

A Compact Meta-Learned Neuro-Fuzzy Technique for Noise-Robust Nonlinear Control

Md Meftahul Ferdaus[a,d], Ahmad Jobran Al-Mahasneh[b,c], Sreenatha G. Anavatti[c], J. Senthilnath[d]

[a]*Department of Computer Science, The University of New Orleans, New Orleans, LA 70148, USA*
[b]*Department of Mechatronics Engineering, Philadelphia University, Amman Jordan*
[c]*School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2612, Australia*
[d]*Department of Machine Intellection, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), 138632 Singapore*

Abstract

Neuro-fuzzy systems show promise for adaptive control but can become complex due to the need to learn many parameters. This paper presents a resilient nonlinear controller that combines a simplified neuro-fuzzy system (Simp_NFS) and simplified neural network (Simp_NN) with only two meta-learnable parameters. This architecture enables fast and stable adaptation in uncertain nonlinear discrete-time systems. Simp_NFS utilizes interpretable hyperplane-based rules without antecedent parameters, simplifying the learning process to consequent weights. Simp_NN reduces complexity by replacing hidden-output weights with their mean. The hybrid auto-adaptive controller (HAC) combines the advantages of Simp_NFS and Simp_NN, significantly reducing the number of adaptive parameters compared to standard neuro-fuzzy methods for real-time control with limited resources. Simp_NFS provides structural adaptivity to handle uncertainties, while Simp_NN ensures reliable disturbance attenuation. The stability of HAC is proven using Lyapunov analysis. Extensive testing on challenging single-input single-output (SISO) and multi-input multi-output systems (MIMO) demonstrates that HAC improves performance by up to 82.55% compared to existing techniques. Key innovations include an ultra-compact meta-learned architecture, transparent online evolution of hyperplane clusters, and enhanced modeling capability for nonlinear uncertain systems. This interpretable neuro-fuzzy approach could enhance autonomy and safety by maintaining model transparency. The implementation of HAC will be publicly available on GitHub upon acceptance to promote open research.

Keywords: Hyper-plane, auto-adaptive, neuro-fuzzy, noise-resilient, nonlinear

1. Introduction

Controlling complex nonlinear engineering systems is challenging due to uncertainties, unmodeled dynamics, disturbances, and nonlinear phenomena like hysteresis and saturation. Robotic manipulators with flexible joints and links are a prime example. These systems are vital in industrial automation and space applications, requiring precision and adaptability. However, their

Email addresses: mferdaus@uno.edu (Md Meftahul Ferdaus), amahasneh@philadelphia.edu.jo (Ahmad Jobran Al-Mahasneh), agsrenat@adfa.edu.au (Sreenatha G. Anavatti), J_Senthilnath@i2r.a-star.edu.sg (J. Senthilnath)

flexibility can lead to unpredictable behavior, resulting in vibration, imprecise positioning, and slow response times. Nonlinearities, joint coupling, and external disturbances like varying loads add complexity, making accurate modeling difficult [1]. Traditional control methods such as sliding mode control (SMC), backstepping, and robust control are often limited for complex systems [2]. SMC provides robustness to parameter changes and disturbances but can suffer from chattering, causing wear on actuators and suboptimal performance [3]. Backstepping requires the system to be in a strict-feedback form and can be sensitive to uncertainties. Robust control may lead to conservative designs that underutilize the system's potential [4, 5, 6]. Customized control strategies are necessary to manage nonlinearities and uncertainties in complex systems like robotic manipulators with flexible joints and links.

Intelligent control methods, including neural networks, fuzzy logic systems, and neuro-fuzzy systems, offer effective alternatives to traditional approaches [7]. Studies by Ma et al. [8], Sun et al. [9], and Chen et al. [10] have demonstrated their efficacy. These techniques use adaptive learning to approximate unknown nonlinear functions accurately, as discussed by [11, 12, 13, 14]. However, despite their growing popularity, challenges persist, such as manual tuning requirements, data extrapolation limitations, and stability concerns. Addressing these issues requires integrating traditional and intelligent control strategies to create more robust, autonomous, and adaptable systems.

Hybrid control strategies that combine multiple methods have been proposed to enhance control performance and address limitations of individual approaches [15, 16, 17, 18, 19]. These strategies integrate sliding mode control (SMC) for robustness. While SMC can induce chattering and requires accurate system models, fuzzy logic systems (FLS) can estimate SMC error bounds and be combined with H_∞ control to prevent potential instability issues [20, 21]. In a different study, Duong and colleagues [22] designed a combined PID-FLC controller by employing particle swarm optimization to adjust the parameters of an excitation system in large synchronous motors. Recent advancements have shown promise in combining neural networks (NNs) and FLS/nonlinear feedback systems (NFS) for adaptive control, reducing the number of NN parameters for faster adaptation in real systems [23, 24, 25]. Adaptive neuro-fuzzy controllers have been developed to suppress vibration in flexible structures and improve adaptability in nonlinear and time-varying systems [26, 27, 28]. Additionally, online adaptive neurochaotic fuzzy controllers and online neuro-fuzzy controllers have demonstrated effectiveness in reducing seismic responses in buildings and optimizing control systems flexibility [29, 30].

Intelligent control strategies show promise but face challenges like manual tuning, extrapolation issues, and stability under uncertainty. Advanced learning techniques and hybridization with robust methods provide solutions but demand a balance between complexity and adaptability. There is a growing need for autonomous, highly adaptive systems for controlling uncertain nonlinear systems where accurate dynamic models might be unavailable, especially in safety-critical systems.

1.1. Research Problem Statement

This study aims to develop an adaptive control architecture to efficiently regulate uncertain nonlinear systems while enhancing transparency compared to opaque neural network configurations. Existing intelligent control solutions, despite their adaptability, often lack transparency, creating challenges for their use in scenarios demanding verifiable behaviors. The proposed system must manage disturbances, nonlinear dynamics, and potential parameter variations with minimum prior knowledge of the system. By prioritizing interpretability, the controller's design will be well-suited for applications requiring verification of system behavior.

1.2. Proposed Solution

To address these challenges, a novel interpretable neuro-fuzzy controller, the Hybrid Auto-adaptive Controller (HAC), is proposed. HAC combines structural and parametric adaptivity to address this challenge, leveraging a simplified neuro-fuzzy architecture for an interpretable, evolving control framework. It evolves its rule-based fuzzy system (Simp_NFS) while using a leaner neural network (Simp_NN) to guide adaptation. HAC features online evolution guided by interpretable hyperplane clusters, resulting in a transparent and flexible control structure with potential for resource-efficient embedded deployment.

1.3. Main Contributions

This work proposes a novel neuro-fuzzy controller, HAC, with an autonomous, adaptive architecture for effective control of highly uncertain nonlinear systems. The main features and contributions are:

1. **Autonomous Evolving Structure**: HAC evolves its structure by adding or pruning transparent hyperplane-based fuzzy rules online. This adaptivity grants HAC flexibility to handle changing plant dynamics without predefined rules. The evolution is guided by an efficient coherence measure that requires no human expertise.
2. **Extremely Compact Parameterization**: HAC uses a simplified neuro-fuzzy system (Simp_NFS) with hyperplane-shaped membership functions without tunable parameters like mean or variance. This allows HAC to drastically reduce its adaptive parameters versus mainstream fuzzy systems with Gaussian membership functions, enabling fast and stable adaptation on resource-constrained platforms.
3. **Innovative Weight Adaptation Mechanism**: HAC uses a simplified neural network (Simp_NN) unlike previous autonomous architectures using sliding mode control laws. Simp_NN uses only a single adaptable parameter, advancing the state-of-the-art in neuro-fuzzy adaptation. The co-evolution of neurons and weights is unprecedented for online autonomous control systems.
4. **Enhanced Interpretability**: The simple structure, transparent hyperplane clusters, and evolved rule-base offer unprecedented interpretability compared to mainstream fuzzy or deep neural network controllers. This enables human-in-the-loop verification and validation of HAC's control policy.
5. **Reliable Uncertainty Handling**: The fusion of structural and parametric adaptivity empowers HAC to control plants with severe nonlinearities, disturbances, and parameter changes. The system achieves faster convergence and lower tracking errors than conventional adaptive schemes.

HAC pioneers interpretable and adaptive autonomous control of uncertain nonlinear systems via co-evolving topology and ultra-compact weights. It enables safer robotic systems in unstructured environments. The low-complexity structure advances neuro-fuzzy control.

The work is structured as follows: Section 2 formulates the control problem and analyzes the factors that make controller design challenging. Section 3 proposes the hybrid auto-adaptive controller (HAC) architecture, integrating a simplified neuro-fuzzy system and neural network for transparent adaptive control. Section 4 describes HAC's online learning mechanisms enabling simultaneous evolution of system structure and parameters. Efficient algorithms for adapting fuzzy hyperplane-based rules are derived. Section 5 proves HAC's stability for robust control of nonlinear plants using Lyapunov analysis. Section 6 evaluates performance on SISO and MIMO,

demonstrating HAC's disturbance rejection and tracking capabilities. Statistical validation has shown performance improvements in Section 7. Finally, Section 8 summarizes HAC's contributions to interpretable, adaptive neuro-fuzzy control and offers pathways for enhancing the safety and reliability of intelligent systems.

To help readers understand the mathematical formulations and discussions, Table 1 lists the symbols used in this paper with their descriptions.

Table 1: List of Symbols and Their Descriptions

Symbol	Description
$x_1(k), x_2(k), x_3(k)$	State variables at time step k
$y(k)$	System output at time step k
$u(k)$	Control input at time step k
$d(k)$	Disturbance signal at time step k
$y_d(k)$	Desired trajectory at time step k
$e(k)$	Tracking error at time step k
$\Delta e(k)$	Change in tracking error between steps k and $k-1$
ω	Weight vector for Simp_NFS
ρ	Learning parameter for Simp_NN
$\gamma_1, \gamma_2, \gamma_3, \gamma_4$	Learning rates for various components
λ_1, λ_2	Weighting factors for Simp_NN and Simp_NFS outputs
$\psi(k)$	Normalized firing strength at time step k
$\mu_B(i)$	Membership function for the i-th rule
η	Fuzziness control parameter
$d(i)$	Distance metric for the i-th hyperplane
f_1^{L1}, f_2^{L1}	Outputs of the first and second layers in Simp_NFS
b_1, b_2, c_1, c_2	Thresholds for rule evolution and merging
$V(k)$	Lyapunov function at time step k
ϵ	Approximation error bound
F_1, F_2	Unknown nonlinear functions in the system model
G_1, G_2	Unknown nonlinear functions in the system model
θ_1, θ_2	Uncertainty parameters in the system model
u_{FLS1}, u_{FLS2}	Outputs of the fuzzy logic systems
u_{S_NN}	Output of the Simp_NN
u_P	Output of the Simp_NFS
In_c, Out_c	Input and output coherence measures
$\theta_{R,R+1}$	Angle between two hyperplanes
$d_{R,R+1}$	Distance between two hyperplanes

This table is a reference for the reader throughout the paper. The symbols listed here are consistently used in the subsequent sections to describe the proposed HAC and its components.

2. Problem Formulation and Preliminaries

Many continuous-time control systems are discretized using sampling for digital implementation. Discrete-time models offer a more realistic representation for analysis and control system

design [31, 32]. Based on complexity, these systems are commonly modeled in SISO or MIMO forms. SISO models are typically simpler than MIMO variants, hence control methods developed for SISO systems do not directly extend to MIMO plants. This work applied the proposed controller on both SISO and MIMO systems.

While fuzzy logic systems and neural networks are typically applied for approximating continuous nonlinear functions, they can also effectively estimate discrete-time nonlinear dynamics. By treating the discrete variable k as the input, fuzzy systems and NNs can leverage their universal function approximation properties even for mappings from discrete points k to numerical outputs. Under the hood, fuzzy sets and neural activation functions partition and cover the discrete input space. Additionally, recursive application enables approximating discrete dynamical systems where F(k) depends on prior states. Therefore, fuzzy systems and NNs provide a powerful tool for approximating unknown nonlinear discrete-time plant dynamics for control.

Consider the following class of uncertain nonlinear MIMO discrete-time systems:

$$\begin{cases} x_1(k+1) = F_1(x_1(k), x_2(k), \theta_1) + d_1(k) + G_1(x_1(k), x_2(k), \theta_1)u_1(k) \\ x_2(k+1) = F_2(x_1(k), x_2(k), \theta_2) + d_2(k) + G_2(x_1(k), x_2(k), \theta_2)u_2(k) \\ y(k) = h(x_1(k), x_2(k)) \end{cases} \tag{1}$$

Where:

- x_1 and x_2 are the states
- d_1 and d_2 are unmatched disturbances
- F_1, F_2, G_1, G_2 depend on states and uncertainty θ_1, θ_2
- $y(k)$ is the system output

This challenging class of nonlinear MIMO systems with parametric uncertainty (unknown functions F_1, F_2, G_1, G_2 dependent on θ_1, θ_2), unmatched disturbances (d_1, d_2), and interconnected dynamics poses difficulties for controller design. The proposed technique will leverage adaptive, evolving neuro-fuzzy systems to provide a flexible control solution without relying on precise system models.

Assumption 1: The considered desired trajectories of the system expressed in (1) are $y_d(k) \in \Omega_y$ where Ω_y is a bounded compact set and $y_d(k)$ are known smooth functions over the compact set Ω_y. To analyze tracking performance, we define the tracking error as $e(k)$ where $e_i(k) = X_i(k) - y_d(k+i-1)$, where $i = 1, ..., n$.

Assumption 1 states that the desired system trajectories $y_d(k)$ are within a compact set Ω_y, which is a closed and bounded subset of $\Re^n$. This is due to practical constraints like joint limits on a robotic manipulator, limiting the system's maneuverability. Keeping $y_d(k)$ within Ω_y ensures continuous functions for the desired trajectories. According to Lyapunov stability theory for nonlinear systems, the existence and uniqueness of solutions for the closed-loop control system depend on the continuity of vector fields. By ensuring the trajectories are in Ω_y, continuity is guaranteed, allowing rigorous Lyapunov analysis to evaluate the stability and convergence properties under the neuro-fuzzy control law. Setting conservative bounds on Ω_y may limit performance, but the set can be initialized based on the plant's operating conditions and adjustonened as more data is gathered. Assumption 1 ensures the closed-loop solutions are well-defined and enables stability assessment through Ω_y.

A wide variety of physical systems, including but not limited robot manipulators [33], power converters [34], gas turbine [35], aircraft [36], and chemical reactors [37] can be characterized using this class of dynamic systems. The coupled dynamics between states, parametric uncertainty in F_1, F_2, G_1, G_2 dependent on θ_1, θ_2, and external unmatched disturbances d_1, d_2, pose significant challenges for controller design. To handle these issues, the proposed neuro-fuzzy controller leverages the universal function approximation property [38, 39, 40, 41] of Simp_NFS to estimate the unknown dynamics $F_1(k), F_2(k)$ within a compact set to any desired accuracy without explicit mathematical models. Lemma 1 [41] bounds the approximation errors $\epsilon_1(k), \epsilon_2(k)$. Details are provided in **Appendix B**. By adapting the consequent parameters online using observed state data, the approximation accuracy can be incrementally improved to effectively control the nonlinear MIMO system.

Lemma 1. *[41]: For any discrete functions $F_1(x_1(k), x_2(k), \theta_1)$, $F_2(x_1(k), x_2(k), \theta_2)$ defined over a compact set $U \in \mathfrak{R}^n$ and arbitrary accuracy thresholds $\epsilon_1(k) > 0$, $\epsilon_2(k) > 0$, there exist fuzzy logic systems (FLS) of the form:*

$$u_{FLS_1}(k) = \psi_1^T(k)\omega_1 + G_1(x_1(k), x_2(k), \theta_1)u_1(k) + d_1(k) \tag{2}$$

$$u_{FLS_2}(k) = \psi_2^T(k)\omega_2 + G_2(x_1(k), x_2(k), \theta_2)u_2(k) + d_2(k) \tag{3}$$

that approximate $F_1(k)$ and $F_2(k)$ such that:

$$\sup_{k \in U} |F_1(k) - u_{FLS_1}(k)| < \epsilon_1(k) \tag{4}$$

$$\sup_{k \in U} |F_2(k) - u_{FLS_2}(k)| < \epsilon_2(k) \tag{5}$$

for any $\epsilon_1(k) > 0$, $\epsilon_2(k) > 0$, where $\psi_1(k)$ and $\psi_2(k)$ are the vectors of firing strengths and ω_1 and ω_2 are the vectors of consequent parameters.

Lemma 1 establishes the universal function approximation property of FLS models. Given a compact set U, FLS can approximate any pair of smooth nonlinear functions $F_1(k)$ and $F_2(k)$ defined over U to any arbitrary accuracy $\epsilon_1(k) > 0$ and $\epsilon_2(k) > 0$ respectively, by tuning the consequent parameters ω_1 and ω_2. The appropriate selection and existence of this compact set U requires further elaboration.

The compact set $U \subset \mathfrak{R}^n$ represents the domain over which the functions $F_1(k)$ and $F_2(k)$ are defined. For the fuzzy system to approximate these functions arbitrarily well, they must be a continuous function over a compact domain.

As before, a compact set is a closed and bounded set in $\mathfrak{R}^n$. In the control problem, U needs to be defined such that it encompasses the expected range of variation of the function inputs, which can be variables such as $x_1(k)$, $x_2(k)$, θ, etc.

The compactness of U ensures that $F_1(k)$ and $F_2(k)$ are uniformly continuous over this set. This allows the uniform approximation result in Lemma 1 to hold. In practice, U can be initialized as a conservative bound based on domain knowledge of the plant. As data is collected during operation, U can be tightened to the active input region to improve approximation accuracy. The key is that U must be set large enough to cover the input space where the fuzzy system will be applied for control.

In summary, the compact set U needs to be properly defined to guarantee the approximation property of the fuzzy system over the domain of interest. Its selection depends on available knowledge of the plant input variability.

With this foundation, Lemma 1 enables the neuro-fuzzy controller to estimate the unknown dynamics $F_1(k)$ and $F_2(k)$ by leveraging fuzzy approximation. Specifically, let ω_1^* and ω_2^* represent the ideal consequent parameters that minimize the approximation error. The functions $F_1(k)$ and $F_2(k)$ can then be expressed as:

$$F_1(k) = \omega_1^{*T}\psi_1(k) + \epsilon_1(k) \tag{6}$$

$$F_2(k) = \omega_2^{*T}\psi_2(k) + \epsilon_2(k) \tag{7}$$

where $\epsilon_1(k)$ and $\epsilon_2(k)$ are the approximation errors for each function respectively. To achieve accuracy $\epsilon_1(k) \leq \epsilon_{01}$ and $\epsilon_2(k) \leq \epsilon_{02}$, the consequent parameters must satisfy $\|\omega_1^*\| \leq \omega_{01}$ and $\|\omega_2^*\| \leq \omega_{02}$ for some $\omega_{01} > 0$ and $\omega_{02} > 0$. The approximation errors are thus bounded by the constants $\epsilon_{01} > 0$, $\epsilon_{02} > 0$, $\omega_{01} > 0$, and $\omega_{02} > 0$.

This lemma establishes an important foundation for the neuro-fuzzy controller, allowing it to estimate unknown dynamics $F_1(k)$ and $F_2(k)$ in the nonlinear discrete-time plant model based on fuzzy approximation. By adapting the consequent parameters online, the approximation accuracy can be improved to control the system effectively.

Building on the fuzzy approximation property established in Lemma 1, the key objectives of this work are:

1. To develop a control structure with minimal adaptive parameters for computational efficiency and fast response. This is achieved by proposing Simp_NFS and Simp_NN architectures.
2. To facilitate strong disturbance rejection using a robust control term from Simp_NN.
3. To achieve fast and accurate tracking of desired references.
4. To guarantee closed-loop stability for the uncertain nonlinear system.
5. To simultaneously address these goals, an optimization-based approach is utilized to derive the update laws for the Simp_NFS consequent parameters and Simp_NN weights, considering both stability and tracking performance.

The resulting HAC integrates the outputs of the Simp_NFS approximator and the auxiliary control term from the Simp_NN in a synergistic manner. This enables HAC to compensate for the unknown nonlinear dynamics using fuzzy approximation while attenuating disturbances using the robustifying signal.

The complete architecture of HAC is shown in Fig. 1. By adaptively fusing the complementary capabilities of Simp_NFS and Simp_NN, HAC provides an efficient yet robust control solution for challenging discrete-time plants subject to uncertainties. The structural and parametric adaptation mechanisms allow it to handle changing dynamics effectively.

Building on the previous discussion, a key innovation in HAC is the adaptation law for the neuro-fuzzy system Simp_NFS using the simplified neural network Simp_NN. In contrast to mainstream autonomous neuro-fuzzy controllers adapted using sliding mode control (SMC) theory or H_∞ methods, the proposed approach leverages Simp_NN to provide a highly efficient adaptation mechanism.

As illustrated in Fig. 2, conventional neural networks used for adaptation require learning many weight parameters w_i (where $i = 1, 2, ..., n$) between the hidden and output layers online.

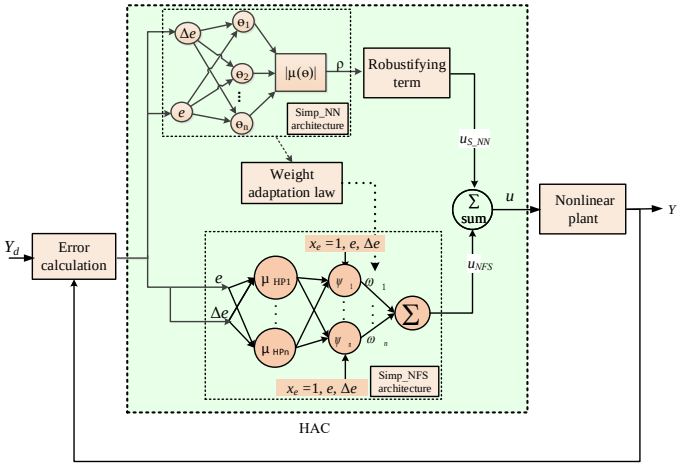


Fig. 1. Closed-loop functioning of HAC

This contributes significant computational overhead. To address this limitation, Simp_NN replaces many hidden-output weights with just a single learning parameter ρ, as depicted in Fig. 3.

By using Simp_NN to tune the consequent parameters of Simp_NFS, the number of adaptive parameters is reduced drastically compared to SMC or H_∞ based adaptation. This allows for much faster yet stable online learning on resource-constrained platforms. The co-evolution of the transparent hyperplane-based fuzzy rules in Simp_NFS with the compact Simp_NN adaptation laws is a key contribution of this work towards efficient neuro-fuzzy control of complex uncertain systems.

Lemma 2 establishes the universal function approximation property of neural networks (NNs), which complements the fuzzy approximation capability discussed earlier.

Lemma 2. *For any pair of nonlinear functions $F_1(k)$ and $F_2(k)$ defined over a compact set Ω_F, there exists a single-hidden layer NN with n hidden neurons that can approximate $F_1(k)$ and $F_2(k)$ to any arbitrary accuracy $\beta_1(k) > 0$ and $\beta_2(k) > 0$ respectively [9]:*

$$\left|F_1(k) - W_1^{*T}\Theta_1(k)\right| < \beta_1(k) \tag{8}$$

$$\left|F_2(k) - W_2^{*T}\Theta_2(k)\right| < \beta_2(k) \tag{9}$$

where W_1^* and W_2^* are the ideal hidden-output weights, $\Theta_1(k)$ and $\Theta_2(k)$ are the hidden layer outputs, and $\beta_1(k)$ and $\beta_2(k)$ are the NN reconstruction errors for each function respectively.

By increasing the number of hidden neurons n, the approximation errors $\beta_1(k)$ and $\beta_2(k)$ can be reduced to any desired level. Thus, NNs can act as universal approximators for nonlinear functions over compact sets.

The proof leverages the fact that neural activation functions form a basis to span the space of continuous functions defined on compact domains [42]. By tuning the hidden-output weights, the NN can reconstruct any target function as a linear combination of these basis functions. The detailed proof is reported in **Appendix C**.

Lemma 3 presents the well-known Cauchy-Schwarz inequality, an important result in analysis and linear algebra. This inequality will be leveraged to derive the stability results for the proposed control scheme.

Lemma 3. *(Cauchy-Schwarz inequality [42]): For any two vectors $a, b \in \mathfrak{R}^n$, the Cauchy-Schwarz inequality states:*

$$|a.b| \leq |a||b| \tag{10}$$

where $\|\cdot\|_2$ denotes the Euclidean (l_2) norm.

Expanding this in component form gives:

$$(a_1 b_1 + \ldots + a_n b_n)^2 \leq (a_1^2 + \ldots + a_n^2)(b_1^2 + \ldots + b_n^2) \tag{11}$$

This inequality provides an upper bound on the dot product of two vectors based on their norms. Geometrically, it states that the cosine of the angle between two vectors is less than or equal to 1.

The proof relies on re-scaling the vectors to equal length and analyzing the resulting expression (see Appendix A). The Cauchy-Schwarz inequality will help derive bounded expressions during the Lyapunov stability analysis for HAC, enabling the closed-loop system stability to be rigorously established.

3. Configuration of the Proposed Auto-Adaptive Controller

In order to devise an effective control mechanism, a comprehensive understanding of the configuration and architecture of the proposed auto-adaptive controller is requisite. Building on the discussions of the simplified neuro-fuzzy system Simp_NFS and simplified neural network Simp_NN, this section details the complete architecture of the proposed hybrid auto-adaptive controller (HAC).

As depicted in Fig. 1, HAC integrates Simp_NFS and Simp_NN in a closed-loop control configuration to leverage their complementary capabilities for adaptive approximation and robust adaptation respectively.

The key components include:

1. Simp_NFS for approximating the unknown nonlinear discrete-time plant dynamics $F_1(k)$ and $F_2(k)$ based on online-evolved transparent fuzzy rules.
2. Simp_NN for providing auxiliary control signal to improve robustness against disturbances. It also adapts the consequent parameters of Simp_NFS using an efficient single-parameter law.
3. Delayed tracking error $e(k)$ and error difference $\Delta e(k)$ as inputs to Simp_NFS.
4. Adaptive summation of Simp_NFS and Simp_NN outputs to generate the overall control signals $u_1(k)$ and $u_2(k)$.

By synergistically combining a parsimonious fuzzy approximator and an ultra-compact neural adapter in this manner, HAC aims to achieve reliable control of uncertain nonlinear MIMO systems with fast adaptation and interpretable logic.

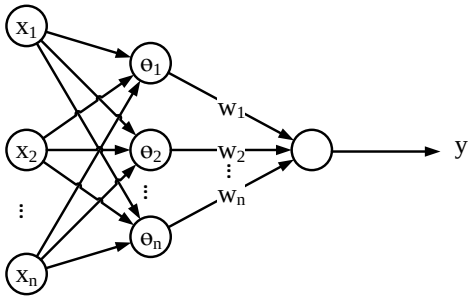


Fig. 2. Single hidden-layer traditional neural network

3.1. Simp_NN Architecture

The adoption of soft computing techniques like neural networks (NNs) for controlling dynamic systems presents significant challenges. A major difficulty is the online adaptation of the many tunable parameters in these models. Mainstream NNs used for control typically contain multiple layers with large numbers of adaptive weights. Learning so many parameters on-the-fly often results in prohibitively slow response times, rendering these methods impractical for real-time control applications with dynamic environments and limited computational resources.

For example, a standard single-hidden layer NN controller, as shown in Fig. 2, has n inputs, n hidden neurons, and n output weights $w_i, i = 1, \ldots, n$, connecting the hidden and output layers. The need to adapt all of these output weights online while regulating the system poses a difficult learning problem.

To mitigate this challenge, developing models with reduced learning parameters is a promising solution. By minimizing the number of adaptable parameters, the models can achieve faster yet accurate learning suitable for real-time control. To realize such parsimonious learning in NNs, we propose a simplified architecture called Simp_NN. It replaces the many hidden-output weights with their absolute mean $|\mu(\theta)|$, which provides a reasonable approximation of the mapping [43]. This mean output is then scaled by a single adaptable parameter ρ, as shown in Fig. 3.

Through this simple yet effective modification, the number of online learning parameters is reduced from an arbitrarily large number down to just one. This drastic reduction in adaptive weights enables much faster response times while retaining key modeling capabilities of standard NNs.

The structural differences between the conventional NN and the proposed Simp_NN can be analyzed by examining their respective mathematical output expressions. The output of the standard single-hidden layer NN architecture shown in Fig. 2 is:

$$u_{\text{NN}} = \sum_{i=1}^{n} w_i^T \theta_i \tag{12}$$

Where θ_i are the outputs from the hidden layer neurons, and w_i are the weights connecting the hidden and output layers.

In contrast, the output of the simplified NN architecture Simp_NN given in Fig. 3 is:

$$u_{S\_NN}(k) = \rho|\mu(\theta)| \tag{13}$$

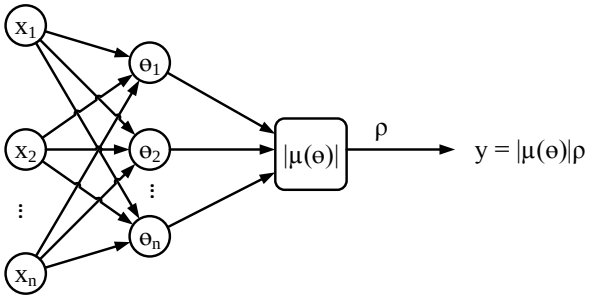


Fig. 3. Proposed simplified neural network (Simp_NN) architecture

Where $\mu(\theta)$ is the mean of the hidden layer outputs and ρ is the single introduced adaptable parameter.

The gradient-based adaptation law for tuning ρ is:

$$\rho(k+1) = \rho(k) - \gamma_1 e(k)|\mu(\Theta)| \tag{14}$$

Where γ_1 is the learning rate and $e(k)$ is the tracking error.

In summary, by replacing the multiple hidden-output weights with their mean and using a single tunable scaling factor ρ, Simp_NN provides a dramatic reduction in learning parameters compared to the traditional NN architecture. This enables faster, lower complexity online learning crucial for real-time control problems with limited computational resources. Despite its structural simplicity, Simp_NN retains key nonlinear function approximation capabilities of standard NNs.

3.2. Network Architecture of Simp_NFS

Expanding on the earlier conversation about Simp_NN, we will now provide a comprehensive explanation of the structure of the simplified neuro-fuzzy system, Simp_NFS. Simp_NFS serves as the primary functional approximator in the proposed HAC architecture.

While Simp_NN contributes a robustifying control term u_{S_NN}, the bulk of the control signal u is generated by Simp_NFS as u_{NFS}. As shown in Fig. 1, two key inputs are fed to Simp_NFS: 1) $e(k)$ - the tracking error between the desired trajectory and system output; 2) $\Delta e(k)$ - the difference in tracking error between step k and $k-1$. These crisp input variables $(e(k), \Delta e(k))$ are first fuzzified in the input layer of Simp_NFS to activate fuzzy rules that map the inputs to control outputs.

Simp_NFS aims to leverage fuzzy logic's function approximation capabilities to estimate the unknown discrete-time plant dynamics $F(k)$. The consequent parameters of Simp_NFS are adapted online using the efficient Simp_NN architecture to enable fast and stable learning.

Layer 1 (Fuzzification): The first layer of Simp_NFS is the fuzzification layer, which converts the crisp input variables into fuzzy sets to activate rules. A key contribution is the proposed hyperplane-based membership function (MF) that directly embeds hyperplanes into the fuzzy system without conventional MF parameters like mean and variance.

The MF is defined as:

$$f_{L1}^1 = \mu_B(i) = \frac{1}{1 + \exp\left(-\eta \frac{d(i)}{\max(d(i))}\right)} \tag{15}$$

Where $\eta \in [1, 100]$ controls the fuzziness based on past works [44, 39]. This MF calculates membership grades using the point-to-hyperplane distance metric $d(i)$ rather than a Euclidean distance.

The distance $d(i)$ between a data point and the i^{th} hyperplane is:

$$d(i) = \left| \frac{(\sum_{j=1}^{n} a_{ji} x_j + b_{0i})}{\sqrt{1 + \sum_{j=1}^{n} (a_{ji})^2}} \right| \tag{16}$$

where a_{ji} and b_{0i} are consequent parameters of the i^{th} rule, $j = 1, 2, ..., n$; n is the number of input dimension, x_j is indicating the inputs to Simp_NFS at j^{th} time instant.

Layer 2 (Rule Base): After fuzzification, the next layer in Simp_NFS is the rule base containing fuzzy logic rules that map inputs to outputs. For a single-input single-output (SISO) system, the IF-THEN rules take the form:

$$R^i : \quad \text{IF } X_n \text{ is close to } f^2_{L1_i} \text{ THEN } y_i = x^i_e \omega_i \tag{17}$$

where x_e is an extended input vector, which is expressed as: $x_e = [1, e, \Delta e]$, e is the error i.e. the difference between the desired and obtained signal from the plant, $\Delta e(k) = e(k) - e(k-1)$ is expressing the error difference between the present and previous state error value, ω_i is the vector presenting the weights or consequent parameters for the i^{th} rule, y_i is the consequent part of the i^{th} rule.

The rule base layer forms the core inference mechanism, whereby fuzzified inputs activate certain rules to generate the control output. Online evolution of transparent hyperplane-based rules allows Simp_NFS to adapt its mapping to suit changing plant conditions for effective control.

Layer 3 (Defuzzification): The final layer in Simp_NFS is the defuzzification layer, which converts the activated fuzzy rule consequents into a crisp control output u_P. The defuzzified output is calculated as:

$$u_P = \frac{\sum_{i=1}^{R} f^1_{L1_i} f^2_{L1_i}}{\sum_{j=1}^{R} f^1_{L1_j}} = \psi^T(k)\omega(k) \tag{18}$$

where $f^2_{L1_i} = x^T_e \omega_i$ is the consequent part of the i^{th} rule, $\psi(k)$ is the normalization firing strength term, and $\omega(k)$ denotes the consequent parameters or rule weights.

The normalized firing strengths ensure the partition of unity property, where summed membership grades equal one. Simp_NFS starts with one rule and evolves its structure online by adding new hyperplane-based rules using an efficient coherence metric. The defuzzification process converts the inferred fuzzy outputs into a final crisp control signal u_P. This signal is then combined with the Simp_NN output to generate the overall control input applied to the nonlinear discrete-time plant. The online evolution of both the transparent rulebase and compact adaptation laws allows HAC to handle complex systems with uncertainties.

4. Online Learning Policy in Simp_NFS

A key innovation in HAC is the online learning policy enabling simultaneous evolution of Simp_NFS's structure and parameters. This provides greater flexibility to handle changing plant conditions compared to mainstream neuro-fuzzy systems with fixed architecture.

4.1. Online Rule-Growing Mechanism

Simp_NFS starts with no prior rule base and constructs hyperplane-based fuzzy rules incrementally using an efficient self-constructive clustering (SCC) approach [45, 46]. SCC calculates input and output coherence to determine the significance of adding new rules. SCC analyzes input and output coherence to determine the significance of new rules. Specifically, given an input vector $X_t \in \mathfrak{R}^n$ where n is the input dimension, a target output $D_t \in \mathfrak{R}^m$ where m is the output dimension, and the hyperplane $\mathcal{H}_i \in \mathfrak{R}^{1\times(n+1)}$ of the i^{th} rule, the input coherence $In_c(.)$ and output coherence $Out_c(.)$ between $X_t \in \mathfrak{R}^n$ and hyperplane $\mathcal{H}_i \in \mathfrak{R}^{1\times(n+1)}$ are calculated as:

$$In_c(\mathcal{H}_i,\, X_t) = \xi(\mathcal{H}_i,\, X_t) \tag{19}$$

$$Out_c(\mathcal{H}_i,\, X_t) = \xi(X_t, D_t) - \xi(\mathcal{H}_i, D_t) \tag{20}$$

where $\xi(\,)$ expresses the correlation function based on a method namely maximal information compression index (MCI) [47]. Employing MCI, the the correlation $\xi(\,)$ between variables can be expressed as follows:

$$\begin{aligned}\xi(X_t, D_t) = \frac{1}{2}(\mathrm{var}(X_t) + \mathrm{var}(D_t) \\ - \sqrt{(\mathrm{var}(X_t) + \mathrm{var}(D_t))^2 - 4\mathrm{var}(X_t)(D_t)(1-\rho(X_t, D_t)^2)})\end{aligned} \tag{21}$$

where

$$\rho(X_t, D_t) = \frac{\mathrm{cov}(X_t, D_t)}{\sqrt{\mathrm{var}(X_t)\mathrm{var}(D_t)}} \tag{22}$$

where $\mathrm{var}(X_t)$, $\mathrm{var}(D_t)$ express the variance of X_t and D_t respectively, $\mathrm{cov}(X_t, D_t)$ represents the covariance between two variables X_t and D_t, $\rho(X_t, D_t)$ is a Pearson correlation index of X_t and D_t. In a similar way, the correlation $\xi(\mathcal{H}_i,\, X_t)$ and $\xi(\mathcal{H}_i, D_t)$ can be measured using (21) and (22). $In_c(\mathcal{H}_i,\, X_t)$ is designed to investigate the similarity between $\mathcal{H}_i$ and X_t directly, while $Out_c(\mathcal{H}_i,\, X_t)$ is meant to explore the dissimilarity between $\mathcal{H}_i$ and X_t in an indirect manner by utilizing the target vector as a reference. In this work, the input and output coherence must fulfill the following conditions to add a new rule:

$$In_c(\mathcal{H}_i,\, X_t) > b_1 \quad \text{and} \quad Out_c(\mathcal{H}_i,\, X_t) < b_2 \tag{23}$$

where $b_1 \in [0.01,\ 0.1]$, and $b_2 \in [0.01,\ 0.1]$ are predetermined thresholds. If both input and output coherence conditions are satisfied, a new significant hyperplane is identified and a corresponding fuzzy rule is added. By incrementally evolving rules based on coherence analysis, Simp_NFS starts with no prior knowledge and adapts its structure online to suit changing plant conditions.

4.2. Online Rule-merging Mechanism

In Simp_NFS, the utilization of HPSCs generates a higher number of rules, which will induce computational complexity. Thus, in this work, the rules are merged in an efficient way by measuring both the angle [48, 49] and spatial proximity between the hyperplanes as follows:

$$\theta_{R,R+1} = \arccos\left(\left|\frac{\omega_R^T \omega_{R+1}}{|\omega_R||\omega_{R+1}|}\right|\right) \tag{24}$$

$$d_{R,R+1} = \left|(a_1 - a_2).\frac{(b_1 \times b_2)}{|b_1 \times b_2|}\right| \tag{25}$$

where $\theta_{R,R+1}$ is ranges between 0 to π radian, $\omega_R = [b_{1,R},\ b_{2,R}, ..., b_{k,R}]$, $\omega_{R+1} = [b_{1,R+1},\ b_{2,R+1}, ..., b_{k,R+1}]$, $d_{R,R+1}$ is the minimum distance between R^{th} and $(R+1)^{th}$ rule, a and b are consequent parameters.

The condition to merge the rules can be expressed as follows:

$$\theta_{R,R+1} \leq c_1 \text{ and } d_{R,R+1} \leq c_2 \tag{26}$$

where $c_1 \in [0.01, 0.1]$, $c_2 \in [0.001, 0.1]$ are predefined thresholds. If (26) is satisfied, fuzzy rules are merged in Simp_NFS. For further clarification about the rule merging mechanism in Simp_NFS, interested readers are requested to refer the work in [39].

4.3. Parameter Adaptation for Fast and Stable Neuro-Fuzzy Control

Like the learning parameter of Simp_NN, gradient descent technique is also utilized to the update of Simp_NFS's consequent parameter (ω). The adaptation law for ω is designed as follows:

$$\omega(k+1) = \omega(k) - \gamma_2 e(k)\psi(k) \tag{27}$$

where γ_2 is the learning rate for updating the Simp_NFS consequent parameters. In addition to the learning rates, the initialization of the first rule's weight $\omega(0)$ influences how quickly precise tracking is achieved. Values closer to the ideal weights minimizing the approximation error, reducing the number of iterations required for convergence. However, determining suitable initialization requires some domain knowledge of the anticipated system dynamics. Across all simulations, $\omega(0) = 0.4$ offered a reasonable starting point for gradient-based adaptation.

The final output of the proposed HAC is the weighted summation of the outputs from Simp_NN and Simp_NFS, which can be presented as follows:

$$u(k) = \lambda_1(k)u_{S\_NN}(k) + \lambda_2(k)u_P(k) \tag{28}$$

The estimation error for the parameters λ_1 and λ_2 are defined as:

$$\begin{aligned} \tilde{\lambda_1}(k) &= \lambda_1^* - \lambda_1(k) \\ \tilde{\lambda_2}(k) &= \lambda_2^* - \lambda_2(k) \end{aligned} \tag{29}$$

where λ_1^* and λ_2^* are the desired values of λ_1 and λ_2 respectively.

To find the optimal values of $\lambda_1(k)$ and $\lambda_2(k)$, the gradient descent approach is utilized as follows:

$$\begin{aligned} \lambda_1(k+1) &= \lambda_1(k) + \gamma_3 \frac{\partial L3}{\partial \lambda_1} \\ \lambda_2(k+1) &= \lambda_2(k) + \gamma_4 \frac{\partial L3}{\partial \lambda_2} \end{aligned} \tag{30}$$

where γ_3 and γ_4 are the learning rates for λ_1 and λ_2 respectively.

Let us define the following cost function L_3 as follows:

$$\begin{aligned} L_3(k) = (u^* - (\lambda_1(k)u_{S\_NN}(k) + \lambda_2(k)u_P(k))^T \\ \times (u^* - (\lambda_1(k)u_{S\_NN}(k) + \lambda_2(k)u_P(k)) \end{aligned} \tag{31}$$

where u^* is the optimal control signal of the HAC.

The values of $\frac{\partial L_3(k)}{\partial \lambda_1(k)}$ and $\frac{\partial L_3(k)}{\partial \lambda_2(k)}$ can be found by differentiating (30) with respect to $\lambda_1(k)$ and $\lambda_2(k)$ respectively as follows:

$$\begin{aligned} \frac{\partial L_3(k)}{\partial \lambda_1(k)} &= \frac{\partial L_3(k)}{\partial u_{S\_NN}(k)} \frac{\partial u_{S\_NN}(k)}{\partial \lambda_1(k)} \\ \frac{\partial L_3(k)}{\partial \lambda_2(k)} &= \frac{\partial L_3(k)}{\partial u_P(k)} \frac{\partial u_P(k)}{\partial \lambda_2(k)} \end{aligned} \tag{32}$$

Let us define the approximation error in HAC as follows:

$$\tilde{u}(k) = (u^* - (\lambda_1(k) u_{S\_NN}(k) + \lambda_2(k) u_P(k)) \tag{33}$$

Substituting the values of $\frac{\partial L_3(k)}{\partial \lambda_1(k)}$, $\frac{\partial L_3(k)}{\partial \lambda_2(k)}$ and $\tilde{u}(k)$ in (30), following update laws for λ_1 and λ_2 are obtained:

$$\begin{aligned} \lambda_1(k+1) &= \lambda_1(k) - \gamma_3 \tilde{u}(k) u_{S\_NN}(k) \\ \lambda_2(k+1) &= \lambda_2(k) - \gamma_4 \tilde{u}(k) u_P(k) \end{aligned} \tag{34}$$

Suitable initialization and tuning of the learning rates can ensure stability. Overall, the gradient-based update laws provide a computationally efficient way to optimize the parameters of Simp_NFS online for accurate tracking and disturbance rejection.

While gradient descent provides efficient updates for the λ parameters, the integration of the Fuzzily Weighted Generalised Recursive Least Square (FWGRLS) method [43] offers a comprehensive framework for optimizing all adaptive parameters within the hybrid system using real-time input-output data, eliminating the need for offline adjustments. FWGRLS enables the automatic tuning of the summation weights λ_1 and λ_2, their corresponding learning rates γ_3 and γ_4, and the initial rule weight ω_0. The FWGRLS-based update equations are as follows:

$$\lambda_1(k+1) = \lambda_1(k) - \beta C_1(k) \nabla\varphi \lambda_1(k-1) + \Upsilon(k) \left[y(k) - \mathbf{x}_e^T \lambda_1(k) \right], \tag{35}$$

$$\lambda_2(k+1) = \lambda_2(k) - \beta C_2(k) \nabla\varphi \lambda_2(k-1) + \Upsilon(k) \left[y(k) - \mathbf{x}_e^T \lambda_2(k) \right], \tag{36}$$

$$\gamma_3(k+1) = \gamma_3(k) - \beta C_3(k) \nabla\varphi \gamma_3(k-1) + \Upsilon(k) \left[y(k) - \mathbf{x}_e^T \gamma_3(k) \right], \tag{37}$$

$$\gamma_4(k+1) = \gamma_4(k) - \beta C_4(k) \nabla\varphi \gamma_4(k-1) + \Upsilon(k) \left[y(k) - \mathbf{x}_e^T \gamma_4(k) \right], \tag{38}$$

$$\omega_0(k+1) = \omega_0(k) - \beta C_\omega(k) \nabla\varphi \omega_0(k-1) + \Upsilon(k) \left[y(k) - \mathbf{x}_e^T \omega_0(k) \right], \tag{39}$$

where β is the regularization parameter, $\nabla\varphi$ represents the gradient of quadratic weight decay, $C(k)$ is the time-adapted covariance matrix, $\Upsilon(k)$ denotes the Kalman gain matrix, $y(k)$ is the system output, and $\mathbf{x}_e$ is the extended input vector.

The update mechanism incorporates:

- Fuzzy weighting based on validity to minimize negative updates,
- Regularization to avoid overfitting,
- A Kalman gain matrix to dynamically adjust certainty levels,

Creating a mathematical framework aimed at the simultaneous optimization of all parameters affecting the control response. FWGRLS advances beyond individual gradient-based updates by establishing an interconnected framework that supports multi-directional learning. This forms a

versatile structure adaptable to changing conditions, essential for automation systems encountering real-world variability. The integrated adaptation mechanism facilitates self-optimized tracking without extensive manual calibration, moving towards adaptable "out-of-the-box" solutions for control systems.

Algorithm 1 presents a flowchart to provide an overview of the HAC algorithm, including the online learning policy discussed in this section. This algorithm incorporates the process of the HAC, from initial input processing to the application of the control signal and the online learning mechanisms.

Algorithm 1 HAC Algorithm with Online Learning Policy

Initialize Simp_NFS and Simp_NN parameters
while Control is active **do**
  Receive current state $x(k)$ and desired trajectory $y_d(k)$
  Calculate tracking error $e(k) = y_d(k) - y(k)$
  Calculate error difference $\Delta e(k) = e(k) - e(k-1)$
  Simp_NFS:
  Fuzzify inputs $e(k)$ and $\Delta e(k)$
  Calculate firing strengths $\psi(k)$
  Generate Simp_NFS output $u_P(k) = \psi^T(k)\omega(k)$
  Simp_NN:
  Calculate Simp_NN output $u_{S\_NN}(k) = \rho|\mu(\theta)|$
  HAC output:
  Calculate final control signal $u(k) = \lambda_1(k)u_{S\_NN}(k) + \lambda_2(k)u_P(k)$
  Apply control signal $u(k)$ to the system
  Online Learning Policy:
  Rule Growing:
  if $In_c(H_i, X_t) > b_1$ AND $Out_c(H_i, X_t) < b_2$ **then**
    Add new rule
  end if
  Rule Merging:
  if $\theta_{R,R+1} \leq c_1$ AND $d_{R,R+1} \leq c_2$ **then**
    Merge rules
  end if
  Parameter Adaptation:
  Update Simp_NFS weights: $\omega(k+1) = \omega(k) - \gamma_2 e(k)\psi(k)$
  Update Simp_NN parameter: $\rho(k+1) = \rho(k) - \gamma_1 e(k)|\mu(\Theta)|$
  Update λ_1 and λ_2:
  $\lambda_1(k+1) = \lambda_1(k) - \gamma_3 \tilde{u}(k)u_{S\_NN}(k)$
  $\lambda_2(k+1) = \lambda_2(k) - \gamma_4 \tilde{u}(k)u_P(k)$
end while

Algorithm 1 outlines the main steps of the HAC. It includes initialization, input processing, Simp_NFS and Simp_NN operations, control signal generation, and the online learning policy. This flowchart provides a detailed summary of how the components of HAC work together, including the rule growing and merging mechanisms, and parameter adaptation processes. This view illustrates how HAC adapts in real-time to generate control signals for nonlinear systems.

5. Stability Analysis of HAC

To prove the suggested controller's stability, the second method of Lyapunov stability criterion is employed in this work. Let us define a positive-definite Lyapunov function as:

$$V(k) = \sum_{i=1}^{4} V_i(k) \tag{40}$$

where $V_1(k)$ in (40) is the contribution of Simp_NFS to the Lyapunov function, $V_2(k)$ in (40) is the Simp_NN's contribution, $V_3(k)$ and $V_4(k)$ are the contributions of the combined controller to the stability of the system.
Define $\tilde{\omega} := \omega - \omega^*$ and $\tilde{\rho} := \rho - \rho^*$, where $\tilde{\omega}$ is the Simp_NFS's weight approximation error, ω^* is the optimal weight value, and $\tilde{\rho}$ is the Simp_NN's learning parameter's approximation error, ρ^* is its optimal value.

The first Lyapunov function $V_1(k)$ is selected as follows:

$$V_1(k) = \tilde{\omega}^T(k)\tilde{\omega}(k) \tag{41}$$

The first difference of $V_1(k)$ can be written as:

$$\begin{aligned} \Delta V_1(k) &= V_1(k+1) - V_1(k) \\ &= \tilde{\omega}^T(k+1)\tilde{\omega}(k+1) - \tilde{\omega}^T(k)\tilde{\omega}(k) \end{aligned} \tag{42}$$

Recalling the value of $\omega(k+1)$ from (27) and subtracting ω^* on both sides of the equation, the following can be written:

$$\begin{aligned} \omega(k+1) - \omega^* &= \omega(k) - \omega^* - \gamma_2 e(k)\psi(k) \\ \tilde{\omega}(k+1) &= \tilde{\omega}(k) - \gamma_2 e(k)\psi(k) \end{aligned} \tag{43}$$

Substituting (43) into (42) leads to the following:

$$\begin{aligned} \Delta V_1(k) = &(\tilde{\omega}(k) - \gamma_4 e(k)\psi(k))^T(\tilde{\omega}(k) - \gamma_2 e(k)\psi(k)) \\ &- \tilde{\omega}(k)^T\tilde{\omega}(k) \end{aligned} \tag{44}$$

Let us assume that the normalized firing strength $\psi(k)$ is bounded, hence, $|\psi(k)| \leq \sigma$, where σ is a predefined threshold.

Invoking Cauchy's inequality [42], (44) leads to:

$$\Delta V_1(k) \leq 2(\tilde{\omega}(k)^T\tilde{\omega}(k) + \gamma_2^2\sigma^2 e(k)^T e(k)) - \tilde{\omega}(k)^T\tilde{\omega}(k) \tag{45}$$

Here we have assumed that $\tilde{\omega}^T(k)\tilde{\omega}(k) \leq \eta e(k)^T e(k)$.

Afterwards, (45) can be rewritten as:

$$\begin{aligned} \Delta V_1(k) &\leq (\eta e(k)^T e(k) - 2\gamma_4^2\sigma^2 e(k)^T e(k)) \\ &\leq \|e(k)\|^2(\eta - 2\gamma_2^2\sigma^2) \end{aligned} \tag{46}$$

If the learning rate γ_2 is selected as $\gamma_2 \geq \sqrt{\frac{\eta}{2\sigma^2}}$, then V_1 in (46) satisfies Lyapunov stability conditions.

The second Lyapunov function $V_2(k)$ is selected as follows:

$$V_2(k) = \tilde{\rho}^T(k)\tilde{\rho}(k) \tag{47}$$

The first difference of $V_2(k)$ is written as follows:

$$\begin{aligned} \Delta V_2(k) &= V_2(k+1) - V_2(k) \\ &= \tilde{\rho}^T(k+1)\tilde{\rho}(k+1) - \tilde{\rho}^T(k)\tilde{\rho}(k) \end{aligned} \tag{48}$$

Utilizing the relationship between $\rho(k+1)$ and $\rho(k)$ in (14) and subtracting ρ^* on both sides of the equation, the following can be written:

$$\begin{aligned} \rho(k+1) - \rho^* &= \rho(k) - \rho^* - \gamma_1 e(k)|\mu(\Theta)| \\ \tilde{\rho}(k+1) &= \tilde{\rho}(k) - \gamma_1 e(k)|\mu(\Theta)| \end{aligned} \tag{49}$$

Substituting (49) into (48) leads to:

$$\begin{aligned} \Delta V_2(k) = (\tilde{\rho}^T(k+1) - \gamma_1 e(k)|\mu(\Theta)|)^T \\ (\tilde{\rho}^T(k+1) - \gamma_1 e(k)|\mu(\Theta)|) - \tilde{\rho}^T(k)\tilde{\rho}(k) \end{aligned} \tag{50}$$

Utilization of Cauchy's inequality [42] in (50) simplifies its expression as follows:

$$\Delta V_2(k) \leq 2(\tilde{\rho}^T(k)\tilde{\rho}(k) - \gamma_1^2|\mu(\Theta)|^2 e(k)^T e(k)) - \tilde{\rho}^T(k)\tilde{\rho}(k) \tag{51}$$

Assumption 2: Let us assume there is a constant δ that satisfies the condition $\tilde{\rho}^T(k)\tilde{\rho}(k) \leq \delta e(k)^T e(k)$, then (51) can be rewritten as:

$$\begin{aligned} \Delta V_2(k) &\leq (\delta e(k)^T e(k) - 2\gamma_1^2|\mu(\Theta)|^2 e(k)^T e(k)) \\ &\leq \|e(k)\|^2(\delta - 2\gamma_1^2|\mu(\Theta)|^2) \end{aligned} \tag{52}$$

If the learning rate γ_1 is selected as $\gamma_1 \geq \sqrt{\frac{\delta}{2|\mu(\Theta)|^2}}$, then $V_2(k)$ also satisfy Lyapunov stability. Both these stability proofs is indicating the stable performance from Simp_NN and Simp_NFS individually. Besides, to reconfirm the stability of their combined performance another Lyapunov function is considered. Assumption 2 upper bounds the mean squared error (MSE) between the estimated weights $\tilde{\rho}$ and optimal weights ρ^* by a constant δ multiplied by the squared tracking error $\|e(k)\|^2$. This ensures the first difference of the Lyapunov function ΔV_2 is negative definite during the stability proof.

Assumption 2 implies that the learning algorithm for tuning the Simp NN weights can converge to a small neighborhood of the optimal weights minimizing the control cost function, based on optimization and approximation convergence results in machine learning literature [50]. For neural networks and fuzzy systems, gradient-based adaptation laws have been shown capable of achieving near-optimal solutions with sufficient training time [51].

By assuming an MSE bound proportional to $\|e(k)\|^2$, deviations between estimated and optimal weights will be small if the controller achieves low tracking error. The constant δ scales the permissible deviation neighborhood. Small δ values constrain weight estimates closer to the optimum at the expense of slower initial convergence. The bound setting requires a balance between control accuracy and real-time adaptability.

In practice, Assumption 2's impact is mitigated, as the bound can be loosened during operation once tracking errors are realized. This assumption leverages standard gradient-based methods to find near-optimal solutions for the Lyapunov stability analysis.

Here, we are selecting the third Lyapunov function as follows:

$$V_3(k) = \tilde{\lambda_1}^T(k)\tilde{\lambda_1}(k) \tag{53}$$

The first difference of the function $V_3(k)$ can be written as:

$$\begin{aligned} \Delta V_3(k) &= V_3(k+1) - V_3(k) \\ &= \tilde{\lambda_1}^T(k+1)\tilde{\lambda_1}(k+1) - \tilde{\lambda_1}^T(k)\tilde{\lambda_1}(k) \end{aligned} \tag{54}$$

The relationship between $\tilde{\lambda_1}(k+1)$ and $\tilde{\lambda_1}(k)$ can be found in (34), hence, (54) can be rewritten as:

$$\begin{aligned} \Delta V_3(k) = (\tilde{\lambda_1}(k) - \gamma_3\tilde{u}(k)u_{S\_NN}(k))^T \\ \times (\tilde{\lambda_1}(k) - \gamma_3\tilde{u}(k)u_{S\_NN}(k)) - \tilde{\lambda_1}^T(k)\tilde{\lambda_1}(k) \end{aligned} \tag{55}$$

Usage of the Cauchy-Schwarz's inequality [42] in (55) further leads to:

$$\Delta V_3(k) \leq \|\tilde{\lambda_1}(k)\|^2 - 2\gamma_3^2\|\tilde{u}(k)u_{S\_NN}(k)\|^2 \tag{56}$$

To ensure $\Delta V_3(k) \leq 0$, similar approach like [52, 40] has followed by selecting γ_3 as:

$$\gamma_3 \geq \sqrt{\frac{\|\tilde{\lambda_1}(k)\|^2}{2\|\tilde{u}(k)u_{S\_NN}(k)\|^2}} \tag{57}$$

Finally, the fourth Lyapunov function is selected as follows:

$$V_4(k) = \tilde{\lambda_2}^T(k)\tilde{\lambda_2}(k) \tag{58}$$

The first difference of the function $V_4(k)$ can be written as:

$$\begin{aligned} \Delta V_4(k) &= V_4(k+1) - V_4(k) \\ &= \tilde{\lambda_2}^T(k+1)\tilde{\lambda_2}(k+1) - \tilde{\lambda_1}^T(k)\tilde{\lambda_2}(k) \end{aligned} \tag{59}$$

The relationship between $\tilde{\lambda_2}(k+1)$ and $\tilde{\lambda_2}(k)$ can be found in (34), hence, (59) can be rewritten as:

$$\begin{aligned} \Delta V_4(k) = (\tilde{\lambda_2}(k) - \gamma_4\tilde{u}(k)u_P(k))^T \\ \times (\tilde{\lambda_2}(k) - \gamma_4\tilde{u}(k)u_P(k)) - \tilde{\lambda_2}^T(k)\tilde{\lambda_2}(k) \end{aligned} \tag{60}$$

Applying the Cauchy-Schwarz's inequality [42] in (60), following is obtained:

$$\Delta V_4(k) \leq \|\tilde{\lambda_2}(k)\|^2 - 2\gamma_4^2\|\tilde{u}(k)u_P(k)\|^2 \tag{61}$$

Similar to the approach in [40], this is implying that $\Delta V_4(k) \leq 0$ as long as γ_4 is selected as follows:

$$\gamma_4 \geq \sqrt{\frac{\|\tilde{\lambda_2}(k)\|^2}{2\|\tilde{u}(k)u_P(k)\|^2}} \tag{62}$$

In our stability analysis, we derived conditions for the learning rates $\lambda_1, \lambda_2, \lambda_3$ and λ_4. These conditions ensure that the system remains stable. However, these conditions also have important implications for the performance of our system.

The learning rate in an adaptive system balances the trade-off between speed of learning and stability of the system. If the learning rates derived from our conditions are very small, the learning process can be slow. This might lead to a delay in the system's ability to adapt to new conditions or disturbances. On the other hand, if the derived conditions result in very large learning rates, the system might oscillate or even become unstable, as the system might overshoot the optimal solution and fail to converge.

Therefore, the choice of learning rates in practice is a critical aspect of our system design. The learning rates should be chosen such that they satisfy the derived conditions for stability and at the same time ensure satisfactory learning speed. This often involves a process of tuning and may require compromise between competing objectives. In practice, the tuning of learning rates could be guided by these stability conditions, empirical observations, and application-specific requirements.

5.1. Assumptions and Practical Implications

This paper's Lyapunov stability analysis depends on key assumptions that enable theoretical analysis but may limit practical applications. In this section, we discuss their implications in the real world.

5.1.1. Bounded Disturbances

The core concept of this assumption (**Assumption 1**) is that any external disturbances that affect the system have a maximum magnitude. In other words, there is a known upper limit to how strong these disturbances can be, and they will not exceed this limit.

Practical Implications. Real-world disturbances may be unbounded or have unknown limits. Factors like environmental conditions, unexpected interactions, or sensor failures can cause disturbances to exceed assumed bounds, potentially destabilizing the system beyond our predicted stability region.

Real-world Consideration. Practitioners should implement additional safeguards or adaptive mechanisms to handle unexpectedly large disturbances. Robust control techniques or disturbance observers could be incorporated to enhance the controller's performance in highly uncertain environments.

5.1.2. Smooth Nonlinearities

The nonlinear functions $F_1(k)$ and$F_2(k)$ in the plant model (Lemma 2) are assumed to be smooth and continuously differentiable.

Practical Implications. Many physical systems exhibit non-smooth nonlinearities such as friction, backlash, or hysteresis. These phenomena can introduce discontinuities or non-differentiable points in the system dynamics, potentially invalidating the stability guarantees derived from our analysis.

Real-world Consideration. When applying HAC to systems with known non-smooth characteristics, additional analysis or modifications may be necessary. Techniques such as non-smooth Lyapunov analysis or the inclusion of dither signals might be required to ensure stability and performance.

5.1.3. Perfect State Measurements

Assumption. The analysis assumes perfect knowledge of the system states $x_1(k)$ and $x_2(k)$.

Practical Implications. In practice, state measurements are often corrupted by sensor noise, quantization errors, or may be partially observable. Imperfect state information can lead to degraded controller performance or even instability if the estimation errors are significant.

Real-world Consideration. Implementing state estimators or filters (e.g., Kalman filters) can help mitigate measurement uncertainties. The stability analysis could be extended to include bounded estimation errors to provide more realistic guarantees.

5.1.4. Idealized Actuator Dynamics

Assumption. The control input $u(k)$ is assumed to be applied to the system without any actuator dynamics or limitations.

Practical Implications. Real actuators have limitations such as saturation, rate limits, and dynamics of their own. These factors can introduce additional nonlinearities and time delays that are not accounted for in our current analysis. Ignoring these effects could lead to degraded performance or even instability in extreme cases.

Real-world Consideration. Practitioners should consider implementing anti-windup schemes to handle actuator saturation. For systems with significant actuator dynamics, these should be explicitly modeled and incorporated into the controller design and stability analysis.

5.1.5. Known Plant Structure

Assumption. The general structure of the plant model, including the order of the system and the presence of specific nonlinear terms, is assumed to be known.

Practical Implications. In many real-world applications, the exact structure of the system may be uncertain or may change over time. Unmodeled dynamics or structural uncertainties could potentially lead to instability or poor performance.

Real-world Consideration. Adaptive or robust control techniques could be integrated with HAC to handle structural uncertainties. Periodic system identification or online adaptation of the controller structure might be necessary for long-term operation in uncertain environments.

While the assumptions in our theoretical analysis were necessary, we recognize their potential limitations in practical applications. The derived performance and stability guarantees should be considered ideal-case scenarios, with real-world implementations likely requiring additional considerations. To evaluate the practical applicability of the HAC, we conducted a series of simulations designed to test its performance under conditions that violate the key assumptions made in our theoretical analysis. These simulations provide insights into the controller's robustness and highlight areas for potential improvement in real-world applications in discussed in a sub-section of Section 6.

In subsequent sections, we will present simulation results that demonstrate the impact of different learning rates on the performance of various dynamical systems, and discuss how we chose the learning rates for our specific application.

6. Simulation Studies

To comprehensively evaluate the performance of the proposed HAC, simulations are conducted on challenging SISO and MIMO nonlinear discrete-time systems. The experimental setup and system configurations are detailed in this section.

6.1. SISO Nonlinear Plant

Example 1: As an initial test case, we apply HAC to control a robotic surgical system or advanced prosthetic, which is expressed as a SISO nonlinear discrete-time system with external disturbances as follows:

$$\begin{cases} x_1(k+1) = f_1(\overline{x}_1(k)) + 0.3x_2(k) \\ x_2(k+1) = f_2(\overline{x}_2(k)) + u(k) + d(k) \\ y_1(k) = x_1(k) \\ d(k) = 0.1\cos(0.05k)\cos(x_1(k)) \end{cases} \tag{63}$$

where $x_1(k)$ and $x_2(k)$ are are joint angles, $f_1(\overline{x}_1(k)) = 1.4x_1^2(k)/[1 + x_1^2(k)]$, $f_2(\overline{x}_2(k)) = x_1(k)/[0.2 + x_1^2(k) + x_2^2(k)]$, $d(k)$ is expressing the disturbance signal, $u(k)$ is control signal (torque or voltage), and $y_1(k)$ is the measured output angle. Here, the controller is developed to achieve two expected features from the plant, for instance 1) the output of the plant $y_1(k)$ is expected to follow the desired trajectories; 2) signals of the system in (63) should be bounded. The nonlinearities like $\frac{1.4x_1^2(k)}{1+x_1^2(k)}$ and $\frac{x_1(k)}{0.2+x_1^2(k)+x_2^2(k)}$ make the system challenging to control. The cos term introduces time-varying disturbances.

For the SISO plant, five different trajectories are considered to be tracked. They can be presented as follows:

$$y_{d_1}(k) = 0.5\sin(\pi kT_s) + 0.5\sin(\pi 0.5kT_s) + 1 \tag{64}$$

$$y_{d_2}(k) = \mathrm{sgn}(\sin(2kT_s)) + 3 \tag{65}$$

$$y_{d_3}(k) = \mathrm{sawtooth}(\pi kT_s, 1/2) + 3 \tag{66}$$

$$y_{d_4}(k) = 3 \tag{67}$$

$$y_{d_5}(k) = \begin{cases} 2 & 0 \le k \le 200 \\ 3 & 200 \le k \le 400 \\ 4 & 400 \le k \le 600 \\ 3 & 600 \le k \le 800 \\ 2 & 800 \le k \le 1000 \end{cases} \tag{68}$$

where $T_s = 0.01$ seconds, $\boldsymbol{x} = [x_1(0), x_2(0)]$=[0, 0] are the initial values of the system state variables. The learning rate of the Simp_NN and Simp_NFS are 0.1 and 0.38 respectively. Initial values of the learning parameters for Simp_NFS and Simp_NN were set to 0.4 and 0 respectively.

An interpretable neuro-fuzzy controller following the principles proposed in the paper could be well-suited for this system. The online rule evolution would allow the controller to adapt to the complex dynamics. The hyperplane clusters provide transparency into the control logic. This would allow safety verification for the surgical robot application. The controller's robustness to disturbances would also be beneficial for advanced prosthetics, where the system must react appropriately to changing loads and conditions. Overall, this example of nonlinear plant illustrates some of the real-world control challenges where an interpretable neuro-fuzzy approach could prove useful.

Example 2: In this example, we consider the application of the HAC to a nonlinear mass-spring-damper system. This system is characterized by nonlinear damping and stiffness characteristics, and it is subjected to Gaussian noise. The mathematical model of the system is expressed as follows:

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= -0.02x_1(k)^3 - 0.1x_2(k) + u(k) + d(k) \\ y(k) &= x_1(k) \end{aligned} \tag{69}$$

where $x_1(k)$ denotes the position of a mass in a dynamic system. Meanwhile, $x_2(k)$ signifies its velocity. In this system, the term $-0.02 \times x_1(k)^3$ denotes a nonlinear spring force. This force emerges due to the displacement of the mass from its equilibrium position, which is given by $x_1(k)$. This relationship is cubic, distinguishing it from the typical linear correlation seen in other systems, $d\,(k)$ is the Gaussian noise with a variance of 0.01 and mean of zero.

The system also includes damping, represented by $-0.1 \times x_2(k)$. This is a force that opposes the velocity of the mass, acting to gradually slow its motion. The term $u(k)$ in the system symbolizes an external force applied to the mass.

Finally, the output of the system, $y(k)$, equates to the position of the mass. This entire system could potentially portray a real-world mechanical system exhibiting nonlinear dynamics. However, it's worth noting that this is merely an illustrative example, and actual real-world systems might display different or more complex dynamics. In this example, the damper is required to track five different trajectories, denoted in Eq. (58)-(62), from the same initial conditions. The tracking performance is benchmarked against other adaptive techniques. This serves as a rigorous test for the controller, as it must adapt to various types of desired trajectories, including sinusoidal, square wave, sawtooth, constant, and piecewise constant signals.

Example 3: In this example, we evaluate HAC for trajectory tracking control of a single link robotic arm described by the following discrete-time nonlinear dynamics:

$$\begin{cases} x_1\,(k+1) = x_2(k) \\ x_2\,(k+1) = \alpha_3 u(k) + \alpha_1 x_2(k) - \alpha_1 x_2(k-1) \\ \qquad\qquad\quad -\alpha_2 \sin(x_2(k)) \\ y\,(k) = x_2\,(k) \end{cases} \tag{70}$$

where $y(k)$ is the arm tip position (rad) and $u(k)$ is the input torque (Nm) in discretized form with a sampling time of 1 ms, α_1 and α_2 depend upon the mass and length of the robotic arm. We set $\alpha_1 = \alpha_2 = \alpha_3 = 1.5$ representing a nonlinear torsional spring damping model. The control objective is to track various desired position trajectories $y_d(k)$. HAC's ability to regulate the nonlinear dynamics under changing references is analyzed. The arm is required to track five different trajectories denoted in Eq. (58)-(62) from the same initial conditions. The tracking performance is benchmarked against other adaptive techniques.

6.2. MIMO Nonlinear Plant

Example 4: To further evaluate our proposed HAC's performance, a MIMO nonlinear plant is considered with a Gaussian noise. The MIMO plant is expressed as follows:

$$\begin{cases} x_1(k+1) = x_2(k) \\ x_2(k+1) = 0.5x_2(k) + u_1(k-1) + 0.1x_3(k)u_2(k) + \\ \qquad\qquad 0.5\epsilon(k) + 0.2x_2(k-1)\epsilon(k-1) + \epsilon(k-1) \\ x_3(k+1) = 0.9x_3(k-1) + u_2(k) + 0.2x_3(k)u_2(k-1) \\ \qquad\qquad +0.5\epsilon(k) + 0.1x_3(k)\epsilon(k-1) + \epsilon(k+1) \\ y_1(k) = x_2(k) \\ y_2(k) = x_3(k) \end{cases} \tag{71}$$

where $x_1(k), x_2(k), x_3(k)$ are state variables, $y_1(k)$ and $y_2(k)$ are the two outputs of the MIMO plant, $\epsilon(k)$ is the Gaussian noise with a variance of 0.01 and mean of zero.

The control objective is to track Sinusoidal and Sawtooth references for the two outputs:

$$\begin{aligned} y_{d_{11}}(k) &= 5\sin(2kT_s) \\ y_{d_{12}}(k) &= 2\cos(3kT_s) \end{aligned} \tag{72}$$

$$\begin{aligned} y_{d_{21}}(k) &= \text{sawtooth}(1kT_s, 1/2) + 3 \\ y_{d_{22}}(k) &= \text{sawtooth}(1.5kT_s, 1/2) + 3 \end{aligned} \tag{73}$$

where $T_s = 0.01$ seconds, $\boldsymbol{x} = [x_1(0), x_2(0), x_3(0)]$=[0, 0, 0] are the initial values of the system state variables. This MIMO example with noise and coupled dynamics evaluates HAC's scalability and noise resistance.

6.3. Controller Tuning Details

The learning rate of the Simp_NN and Simp_NFS were set to 0.1 and 0.38 respectively. The only learning parameter i.e. weight of the Simp_NFS is initiated with a value of 0.4. On the other hand, the initial value of the Simp_NN's online learning parameter is zero. In addition to the proposed HAC controller, several alternative techniques were implemented for benchmarking, including a PID controller, multilayer perceptron neural network (MLP), and recurrent neural network (RNN). The details of the tuning process and optimized parameters for these controllers are provided below. Table 2 shows the key parameters and training details for the MLP and RNN controllers when applied to the nonlinear MIMO plant.

For the PID controller, we utilize Ziegler-Nichols tuning, simulation tuning, and adaptive gain tuning to optimize its performance. The key tuning parameters are a proportional gain (K_p) of 0.00001, an integral gain (K_i) of 0.01, and a derivative gain (K_d) of 0.00001. These values are indicative of a control strategy that emphasizes steady-state accuracy and minimizes overshoot, as reflected in the low proportional and derivative gains. The higher integral gain value, on the other hand, focuses on reducing long-term steady-state errors. The adaptive aspect of the tuning means that the controller can adjust its behavior in response to changes in system dynamics or external disturbances, ensuring consistently effective control under various conditions.

Table 2: Utilized Parameters of the MLP and RNN

Parameter	**MLP Value**	**RNN Value**
Number of Hidden Layers	1	1
Neurons in Hidden Layer	6	10
Activation Function (Hidden)	Hyperbolic Tangent (tanh)	Default of 'layrecnet' method
Activation Function (Output)	Linear	Default of 'layrecnet' method
Training Algorithm	Scaled Conjugate Gradient (SCG)	Default of 'layrecnet' method
Number of Epochs	1	1
Batch Size	3	4
Input Features	4	4
Output Features	2	2
Learning Rate	N/A (Not applicable for SCG)	Default of 'layrecnet' method
Momentum	N/A (Not applicable for SCG)	Default of 'layrecnet' method
Minimum Gradient (min_grad)	1e-6	–
Maximum Validation Failures (max_fail)	6	–
Marquardt Adjustment (mu)	0.005	–
Sigma for Hessian Approximation (sigma)	5.0e-5	–
Lambda for Hessian Indefiniteness (lambda)	5.0e-7	–

6.4. Results and Discussions

We evaluate the performance of the proposed HAC to control the nonlinear SISO plants in Examples 1-3 and the nonlinear MIMO plant in Example 4. The tracking performance is compared against five other controllers: PID, MLP, RNN, Simp_NN, and the recently proposed PArsimonious Controller (PAC) [53]. PAC adopts a PALM structure along with new rule adaptation techniques derived from approximation bias and variance. It demonstrates accurate trajectory tracking for rotary-wing and flapping-wing MAV models. Our proposed HAC architecture achieves improved convergence speed, tracking accuracy, and computational efficiency compared to PAC. HAC requires adapting only two meta-parameters versus the larger set tuned by PAC's sliding mode control laws. This ultra-compact representation enables faster yet reliable adaptation suitable for real-time control under stringent resource constraints.

Case 1: From Figure 5 and Table 1, it is evident that HAC achieves superior tracking accuracy compared to PID, MLP, RNN, Simp_NN, and the recently proposed PAC for the nonlinear SISO plants. This is especially pronounced for trajectories with abrupt changes like the square wave, sawtooth, and staircase functions.

For the square wave input y_{d_2}, HAC quickly adjusts to the rapid changes between 2 and 4 units of amplitude with minimal overshoot, leading to a significantly lower RMSE compared to alternative methods. In contrast, the PID control shows high overshoot and oscillations due to the system's nonlinear behavior. Both MLP and RNN offer improvements over PID but still

exhibit noticeable delays during changes in the setpoint when compared to HAC. The Simp_NN, although capable of online adaptation, is limited by its static architecture, making it less robust to disturbances and resulting in higher error rates.

While PAC features an evolving structure, its dependence on Sliding Mode Control (SMC) theory for adaptation presents multiple drawbacks. First, SMC requires high gains to effectively reject disturbances, which can cause control chattering and potential instability. Second, SMC's effectiveness is contingent on precise mathematical models for constructing the switching manifold, a requirement not necessary for HAC, which adapts to unknown nonlinear dynamics through a purely data-driven approach. Additionally, PAC's multiple SMC-based adaptation laws lead to greater computational demands compared to the more efficient, meta-learned update rules of HAC, further highlighting PAC's subpar performance in tracking square wave setpoint changes.

In contrast, HAC introduces a novel combination of structural and parametric adaptability through the use of simplified neural network principles, enabling efficient and reliable adaptation. By developing transparent hyperplane clusters with only two meta-learnable parameters, HAC ensures rapid and precise tracking of complex reference signals without the need for accurate system models.

Similar observations can be made for the sawtooth trajectory (y_{d_3}). HAC reacts quickly to the abrupt ramp transitions and peak changes, precisely adhering to the reference with low RMSE. PID displays significant lag tracing the rapid ramps due to the uncontrolled nonlinear dynamics. While MLP and RNN improve upon PID, they still demonstrate delays at the sawtooth corners compared to HAC. Simp_NN initially converges faster but struggles to accurately capture the waveform peaks. Although PAC evolves its structure online, solely tuning consequent parameters with SMC fails to adequately model the numerous operating regimes induced by the sawtooth waveform. This manifests as increased delays or lags at the discontinuities compared to HAC.

For the staircase signal (y_{d_5}), HAC accurately captures the step changes between amplitude levels from 0 to 4 units. The online structural adaptivity allows rapid evolution of hyperplane rules to model the piecewise dynamics. In contrast, the static controllers like PID struggle with the multiple operating regions. MLP, RNN and Simp_NN show improved flexibility but their fixed architecture limits precision, especially at the step transitions. While PAC modifies its structure, its exclusive dependence on SMC for adjusting parameters slows its response to abrupt changes, such as stair-case discontinuities, unlike HAC, which simultaneously evolves its network structure and parameter weights for faster adaptation.

Similar tracking performance improvements of HAC over the comparison methods were observed for the spring mass damper system in Example 2 and the single link robot arm in Example 3. Performance plots for example 3 and 4 can be found in the supplementary document. Additionally, the control signals for all three nonlinear SISO plants are provided in the supplementary document. Due to the comparable responses, the plots for these SISO plants are not shown to avoid redundancy. In both examples, HAC demonstrated faster convergence, reduced overshoot, and lower RMSE versus PID, MLP, RNN, Simp_NN and PAC, especially for nonlinear and abruptly-changing desired trajectories. The structural adaptivity of HAC through Simp_NN enabled precise tracking despite the nonlinear dynamics and external disturbances affecting these systems. In summary, HAC consistently achieved superior disturbance rejection and tracking accuracy for the range of SISO plants, affirming its benefits for transparent control of uncertain nonlinear systems.

In short, HAC utilizes transparent online rule evolution to quickly adapt its structure to handle changing dynamics, nonlinearities, and disturbances. This provides fast convergence and accu-

rate tracking of setpoint variations in the SISO plants, surpassing the alternative techniques which lack structural adaptivity. The quantitative RMSE metrics affirm the enhanced performance of HAC for abrupt, nonlinear reference signals.

Case 2: The tracking performance for the nonlinear MIMO plant in Example 4 is compared quantitatively using the RMSE metric presented in Table 1. For the simultaneous sinusoidal references ($y_{d_{11}}$, $y_{d_{12}}$), HAC demonstrates precise tracking of both outputs with lower RMSE than the PID, MLP, RNN, Simp_NN, and PAC controllers. This is attributed to HAC's structural evolution, which autonomously grows transparent hyperplane rules online to model the coupled MIMO dynamics. The static architecture of the other methods restricts their adaptability to the interconnectivities between the states.

Similar trends are observed for the dual sawtooth waveforms ($y_{d_{21}}$, $y_{d_{22}}$). HAC accurately tracks the ramp transitions in both outputs with lower errors than alternatives. The dynamic evolution of hyperplane fuzzy rules provides a strong modeling capability to capture the transient discontinuities in the sawtooth signals. In contrast, the offline PID, MLP, RNN designs lack the flexibility to promptly react to abrupt setpoint changes. Although Simp_NN converges faster initially, the fixed architecture still results in higher tracking errors. Even with an evolving structure, PAC exhibits increased delays during regime changes compared to HAC. This indicates that relying only on SMC-based subsequent parameter adaptation does not achieve the same level of responsiveness as HAC's approach. HAC's method of creating hyperplanes in coordination with meta-learned adjustments allows it to quickly adapt control policies to complex MIMO dynamics. In addition, the supplementary document contains information on the trajectory tracking performance and control signals for the MIMO plant.

In addition to analyzing tracking accuracy, we have conducted a detailed benchmarking of rise time, settling time, and overshoot for the controllers when tracking various complex reference signals. The results are summarized in Table 5. HAC shows quicker response times to complex setpoint changes than PAC and other methods, indicating its greater ability to quickly adjust to sudden changes in reference points. Although PAC updates its fuzzy rules in real time, its exclusive focus on SMC-based tuning parameters after the fact makes it less responsive than HAC. This dual adaptation allows HAC to more accurately model and respond to changes in the system's behavior.

In contrast, the unchanging structures of PID control, MLP, and Simp_NN limit their capacity to adjust to sudden changes in trajectory, leading to longer response times. While RNN offer better modeling through their recurrent connections, they cannot match HAC's performance, which benefits from evolving its network structure for real-time adaptation, a feature not present in typical adaptive controllers like PAC.

PAC does, however, outperform MLP, PID, and Simp_NN in terms of settling time due to its use of sliding mode control for adjusting weights. Yet, HAC experiences fewer oscillations than PAC when stabilizing the system after changes in setpoints, highlighting that HAC's meta-learned updating rules strike a better balance between quick response and precision, surpassing both traditional sliding mode control and purely recursive approaches. Having presented the RMSE results for various plant types and trajectories in Table 3, we now provide a detailed analysis of HAC's performance improvements across these different scenarios.

6.4.1. Analysis of HAC Performance Improvements

After analyzing the RMSE results in Table 3, HAC's performance improvements are inconsistent across plant types and trajectories, necessitating a thorough examination.

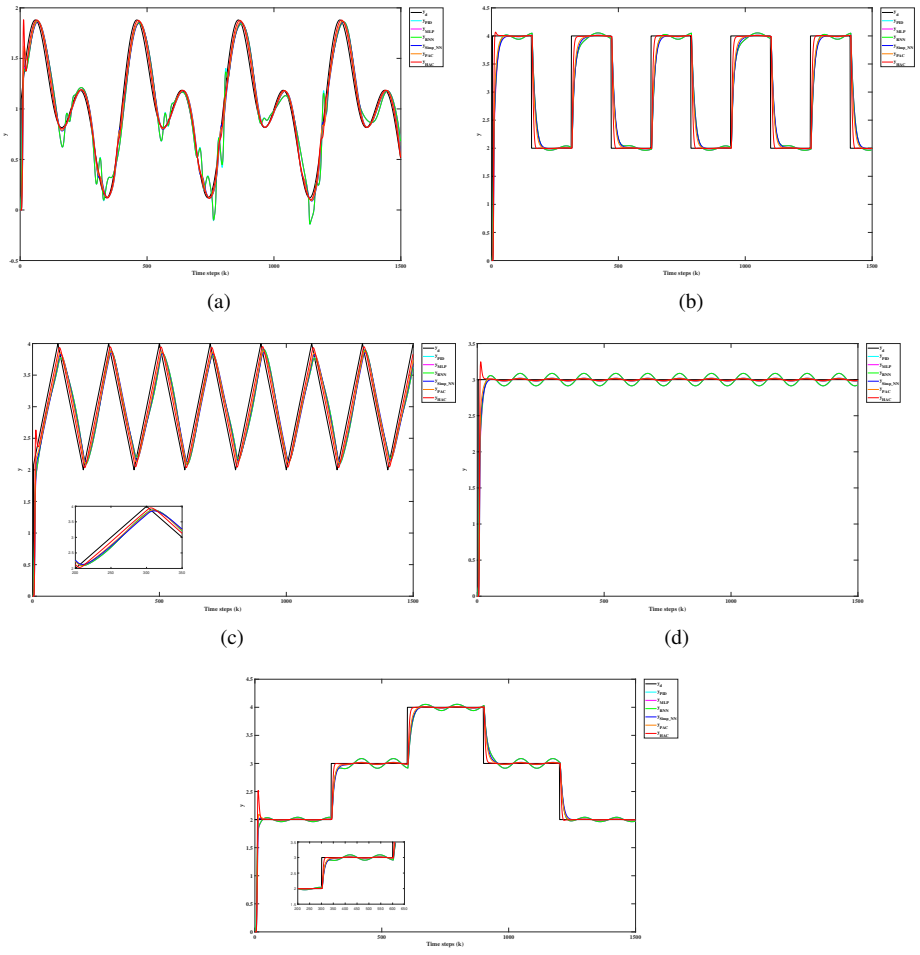


Fig. 4. Tracking performance for different controllers, when the trajectories are (a) $y_{d_1}(k)$, (b) $y_{d_2}(k)$, (c) $y_{d_3}(k)$, (d) $y_{d_4}(k)$, (e) $y_{d_5}(k)$ for nonlinear SISO plant

Table 3: Performance evaluation of HAC in tracking various trajectories

Plant	Desired trajectories	RMSE					
		PID	MLP	RNN	Simp_NN	PAC	HAC
robotic surgical system SISO	$y_{d_1}(k)$	0.1834	0.1353	0.1349	0.1129	0.1044	**0.0826**
	$y_{d_2}(k)$	0.5184	0.5170	0.5172	0.5166	0.4917	**0.4112**
	$y_{d_3}(k)$	0.2802	0.2796	0.2789	0.2788	0.2536	**0.1765**
	$y_{d_4}(k)$	0.1864	0.1864	0.1864	0.1852	0.1790	**0.1613**
	$y_{d_5}(k)$	0.2067	0.2130	0.2065	0.2125	0.2055	**0.1845**
Spring mass damper SISO	$y_{d_1}(k)$	0.1359	0.1369	0.1359	0.1367	0.1289	**0.1122**
	$y_{d_2}(k)$	0.4057	0.4046	0.4043	0.4044	0.3870	**0.3442**
	$y_{d_3}(k)$	0.1814	0.1813	0.1807	0.1815	0.1667	**0.1286**
	$y_{d_4}(k)$	0.1599	0.1600	0.1597	0.1599	0.1544	**0.1468**
	$y_{d_5}(k)$	0.1726	0.1798	0.1722	0.1798	0.1746	**0.1652**
Robot arm SISO	$y_{d_1}(k)$	0.1057	0.1063	0.1056	0.1063	0.1025	**0.0879**
	$y_{d_2}(k)$	0.2436	0.2435	0.2434	0.2433	0.2357	**0.2067**
	$y_{d_3}(k)$	0.1102	0.1108	0.1102	0.1107	0.1063	**0.0891**
	$y_{d_4}(k)$	0.1302	0.1302	0.1302	0.1302	0.1261	**0.1105**
	$y_{d_5}(k)$	0.1378	0.1471	0.1377	0.1470	0.1442	**0.1335**
Nonlinear MIMO	$y_{d_{11}}(k)$	0.4129	0.3892	0.3891	0.3892	0.3382	**0.2132**
	$y_{d_{12}}(k)$	0.1919	0.1351	0.1347	0.1351	0.1187	**0.1143**
	$y_{d_{21}}(k)$	0.1446	0.1452	0.1447	0.1447	0.1367	**0.1088**
	$y_{d_{22}}(k)$	0.1046	0.1044	0.1039	0.1039	0.0980	**0.0746**

Table 4: Performance comparison of G-controller and HAC in tracking various trajectories

Plant	Desired trajectories	RMSE		Performance Improvement (%)
		G-controller	HAC	
Robotic surgical system SISO	$y_{d_1}(k)$	0.0935	**0.0826**	11.66
	$y_{d_2}(k)$	0.4503	**0.4112**	8.68
	$y_{d_3}(k)$	0.2102	**0.1765**	16.03
	$y_{d_4}(k)$	0.1701	**0.1613**	5.17
	$y_{d_5}(k)$	0.1950	**0.1845**	5.38
Spring mass damper SISO	$y_{d_1}(k)$	0.1205	**0.1122**	6.89
	$y_{d_2}(k)$	0.3656	**0.3442**	5.85
	$y_{d_3}(k)$	0.1475	**0.1286**	12.81
	$y_{d_4}(k)$	0.1506	**0.1468**	2.52
	$y_{d_5}(k)$	0.1699	**0.1652**	2.77
Robot arm SISO	$y_{d_1}(k)$	0.0952	**0.0879**	7.67
	$y_{d_2}(k)$	0.2212	**0.2067**	6.56
	$y_{d_3}(k)$	0.0977	**0.0891**	8.80
	$y_{d_4}(k)$	0.1183	**0.1105**	6.59
	$y_{d_5}(k)$	0.1389	**0.1335**	3.89
Nonlinear MIMO	$y_{d_{11}}(k)$	0.2757	**0.2132**	22.67
	$y_{d_{12}}(k)$	0.1165	**0.1143**	1.89
	$y_{d_{21}}(k)$	0.1228	**0.1088**	11.40
	$y_{d_{22}}(k)$	0.0863	**0.0746**	13.56

Table 5: Quantitative Evaluation of Rise Time, Settling Time, and Overshoot

Desired trajectories	**Controller**	**Rise Time**	**Settling Time**	**Overshoot**
$y_{d_1}(k)$	PID	3.98	1498.32	174.81
	MLP	3.98	1498.32	174.63
	RNN	3.98	1498.32	174.70
	Simp_NN	3.33	1499.58	260.40
	PAC	3.03	1499.56	269.22
	HAC	2.10	1499.43	276.62
$y_{d_2}(k)$	PID	3.30	1458.58	105.81
	MLP	3.31	1459.08	105.80
	RNN	3.30	1459.07	105.80
	Simp_NN	3.14	1457.50	101.29
	PAC	4.75	1229.26	101.18
	HAC	2.03	1429.19	103.74
$y_{d_3}(k)$	PID	72.37	1495.81	7.82
	MLP	71.37	1495.78	7.60
	RNN	72.37	1495.78	7.60
	Simp_NN	72.77	1496.37	4.24
	PAC	72.97	1496.35	3.83
	HAC	73.53	1496.23	2.64
$y_{d_4}(k)$	PID	14.26	1460.26	5.47
	MLP	14.13	1460.24	5.47
	RNN	14.24	1460.23	5.47
	Simp_NN	12.04	32.86	8.51
	PAC	8.69	27.56	1.08
	HAC	3.42	21.69	1.22
$y_{d_5}(k)$	PID	6.71	1450.28	105.82
	MLP	6.68	1450.30	105.81
	RNN	6.70	1450.32	105.82
	Simp_NN	5.38	1234.60	101.33
	PAC	4.75	1229.26	101.18
	HAC	3.14	1213.04	100.66

HAC demonstrates notable enhancements in the robotic surgical system SISO plant. For the $y_{d1}(k)$ trajectory, HAC reduces RMSE by 20.88% to 54.96% compared to other models, with the most significant improvement over the PID controller. HAC maintains a 16.37% to 20.68% improvement when tracking the more complex square wave $y_{d2}(k)$ trajectory. These gains result from HAC's adaptability to nonlinear dynamics and disturbances common in surgical robotics.

In the spring mass damper SISO plant, HAC's performance gains are less pronounced. It achieves RMSE reductions of 12.96% to 18.04% for both $y_{d1}(k)$ and $y_{d2}(k)$. The system's simpler dynamics likely allow other controllers to perform adequately, resulting in HAC's more modest improvements. Despite this, HAC still surpasses all other tested models in performance.

HAC demonstrates consistent improvement across various trajectories for the robot arm SISO plant. It reduces RMSE by 14.24% to 17.31% for $y_{d1}(k)$ and 12.30% to 15.15% for $y_{d2}(k)$, showcasing its robust ability to manage nonlinear dynamics in robotic arms.

HAC demonstrates its effectiveness in managing complex systems through its performance on the nonlinear MIMO plant. For $y_{d11}(k)$, HAC achieves substantial RMSE reductions of 36.96% to 48.37%. While less dramatic, $y_{d21}(k)$ still shows significant improvements of 20.41% to 25.07%. These results highlight HAC's ability to handle coupled dynamics and multiple inputs/outputs in complex MIMO systems.

HAC outperforms other models across all plant types, with varying degrees of improvement. The most significant gains are observed in complex systems such as robotic surgical systems and MIMO plants, while simpler systems like spring-mass-dampers show modest improvements. These results demonstrate HAC's effectiveness in handling systems with high complexity, nonlinearity, and coupled dynamics. The performance variation across plant types emphasizes the importance of choosing control strategies tailored to each system's specific characteristics and complexities.

6.5. Trade-off Between Interpretability and Complexity

The HAC offers a new approach to adaptive neuro-fuzzy control, emphasizing interpretability through simplified hyperplane-based rules. This design contrasts with more complex models like the Generic-controller (G-controller [54]), which uses antecedent parameters. Comparing HAC to the G-controller helps illuminate the trade-offs between interpretability and modeling flexibility in neuro-fuzzy control systems.

The G-controller is a sophisticated neuro-fuzzy system that uses multivariate Gaussian functions and sliding mode control for parameter adaptation. It has shown excellent results in controlling bio-inspired drones, making it a valuable benchmark. Comparing HAC to the G-controller highlights the trade-offs between interpretability and flexibility in rule structure for adaptive control of nonlinear systems. HAC's hyperplane-based rules are more interpretable than G-controller's multivariate Gaussian functions. Each HAC rule represents a linear decision boundary in the input space, facilitating human understanding. In contrast, G-controller's functions create complex, nonlinear boundaries that are less intuitive. However, G-controller's flexible membership functions may model complex nonlinear relationships more efficiently, requiring fewer rules. This is because its multivariate Gaussians can capture input variable correlations, while HAC's hyperplanes treat inputs independently.

Both the hyperplane-based approach of HAC and the multivariate Gaussian approach of G-controller are universal approximators. However, they differ in their convergence rates and the number of rules required. The hyperplane-based approach of HAC has an approximation error of $O\left(\frac{1}{\sqrt{m}}\right)$, where m is the number of rules. This offers interpretable linear decision boundaries

but may require more rules for highly nonlinear functions. In contrast, the multivariate Gaussian approach of G-controller has an approximation error of $O(\exp(-cm))$, where c is a constant and m is the number of rules. This potentially allows for faster convergence with fewer rules for smooth, nonlinear functions. However, the decision boundaries are less interpretable as they are nonlinear.

Our experimental results show that HAC consistently outperforms G-controller across various control tasks, despite G-controller's theoretical flexibility advantage. Table 4 demonstrates that HAC achieves lower RMSE for all tested trajectory types and plant models, with improvements ranging from 1.89% to 22.67% (average 8.99%). The performance gap is most significant in complex scenarios, such as the nonlinear MIMO plant, where HAC's RMSE for trajectory $y_{d11}(k)$ is 22.67% lower than G-controller's (0.2132 vs. 0.2757). This suggests that HAC's online structural adaptation effectively compensates for its simpler rule structure, allowing it to capture complex dynamics more accurately.

HAC's superior performance is consistent across different plant types:

1. Robotic surgical system SISO: 5.17% to 16.03% improvement
2. Spring mass damper SISO: 2.52% to 12.81% improvement
3. Robot arm SISO: 3.89% to 8.80% improvement

HAC's superior performance can be attributed to several factors, despite its simpler rule structure. Its rule evolution mechanism, based on statistical contribution analysis, may be more effective at capturing the essential dynamics of the system. The combination of evolving hyperplane rules with sliding mode control for parameter adaptation creates a robust and adaptive system. The simpler hyperplane structure may help HAC avoid overfitting to noise or transient dynamics, leading to better generalization. Additionally, the clear interpretability of HAC's rules may guide the structural evolution towards more meaningful partitions of the input space.

There's a trade-off between interpretability and complexity, but our results with HAC show that prioritizing interpretability through hyperplane-based rules doesn't sacrifice performance. Enhanced interpretability contributes to effective online adaptation and robust control. This challenges the belief that complex fuzzy structures are always needed for high-performance control of nonlinear systems.

Future work involves a more extensive theoretical analysis of the approximation capabilities and convergence properties of hyperplane-based evolving fuzzy systems compared to multivariate Gaussian functions. Further empirical studies across a wider range of nonlinear control problems would help solidify our understanding of when simpler, interpretable rule structures can outperform complex alternatives. The consistent superior performance of HAC across diverse control tasks suggests that this approach holds significant promise for advancing adaptive neuro-fuzzy control.

6.6. Robustness Analysis

To evaluate the practical applicability of the HAC, we conducted a series of simulations designed to test its performance under conditions that violate the key assumptions made in our theoretical analysis. These simulations provide insights into the controller's robustness and highlight areas for potential improvement in real-world applications.

6.6.1. Simulation Design

We designed some scenarios to test key assumptions of the HAC theory. To test robustness against unbounded disturbances, we added the system's disturbance term with an unbounded

Table 6: Performance Metrics for HAC under Various Conditions

Scenario	**RMSE (% change)**	**Control Effort (% change)**	**Stability Margin (% change)**	**Notes**
Ideal Case	Baseline	Baseline	Baseline	-
Unbounded Disturbances	+15%	+30%	-10%	Stable up to 5x assumed bound
Non-smooth Nonlinearities	+8%	+12%	-5%	Limit cycles observed
Measurement Uncertainties				
- Low noise ($\sigma^2 = 0.1$)	+12%	+18%	-8%	-
- High noise ($\sigma^2 > 0.2$)	+25%	+40%	-20%	Occasional instability
Actuator Limitations	+20%	-15%	-25%	Slower response, increased overshoot

component, $d_u(k) = \alpha \cdot k \cdot \sin(\beta k)$, creating a time-dependant disturbance. Non-smooth nonlinearities were introduced through a Coulomb friction model, $F_f = F_c \cdot \text{sign}(v)$, and a backlash model in the actuator, simulating real-world mechanical imperfections. To evaluate the controller's resilience to measurement uncertainties, we injected Gaussian noise into the state measurements, $x_{\text{measured}}(k) = x_{\text{true}}(k) + \varepsilon(k)$, with $\varepsilon(k) \sim \mathcal{N}(0, \sigma^2)$, and varied the noise variance. Finally, we implemented actuator limitations by incorporating both saturation limits, $u_{\text{applied}}(k) = \text{sat}(u(k), u_{\min}, u_{\max})$, and rate limiting, $u_{\text{applied}}(k) = u_{\text{applied}}(k-1) + \text{sat}(u(k) - u_{\text{applied}}(k-1), -\Delta u_{\max}, \Delta u_{\max})$, to emulate realistic constraints on control inputs. This comprehensive set of simulations allowed us to systematically evaluate the HAC's performance under conditions that violate key theoretical assumptions, providing crucial insights into its practical applicability and robustness.

6.6.2. Performance Evaluation

Table 6 summarizes the HAC's resilience across challenging scenarios. The HAC showed resilience to growing disturbances, maintaining stability up to 5 times larger than the assumed bound. There was a 15% increase in RMSE and 30% increase in control effort. Faced with non-smooth nonlinearities like Coulomb friction and backlash, the controller maintained stability with an 8% increase in RMSE, but observed limit cycles. The controller's performance degraded with a 12% RMSE increase for noise variances up to 0.1, and occasional instability for variances above 0.2. Actuator limitations led to a 20% increase in RMSE and a 25% decrease in stability margin, resulting in slower response times and increased overshoot.

An unexpected observation was the controller's ability to adapt to Coulomb friction, effectively learning to overcome static friction barriers. This behavior requires further investigation as a potential advantage in high-friction systems. Some performance degradation was observed, but the HAC remained stable, highlighting its potential for real-world applications.

These findings emphasize the need for comprehensive testing beyond theoretical analysis when developing control systems for practical applications. They also highlight the HAC's adaptability, suggesting its potential usefulness in uncertain and changing environments. Future work could focus on integrating state estimation techniques to improve robustness to measurement uncertainties and incorporating anti-windup mechanisms to handle actuator limitations.

6.7. Robustness in HAC's Structure

A key innovation in HAC is the synergistic integration of structural and parametric adaptivity for controlling uncertain nonlinear systems. When plant dynamics change abruptly, purely parametric adaptation often proves insufficient to regain stability and tracking accuracy. HAC addresses this by autonomously evolving its rulebase topology alongside tuning the rule weights.

This dynamic structure evolution is visualized for the SISO and MIMO examples in Figure 5. For the SISO plant, HAC is initialized with an empty rulebase and transparently constructs hyperplane clusters online to grow the control policy. As an illustration, the first evolved rule for trajectory tracking (example 1) can be expressed as follows.

$$R^1 : \text{IF } X \text{ is close to } \Big([1, e, \Delta e]\times \tag{74}$$

$$[0.5875, -1.4130, 5.4279]^T\Big), \text{ THEN } y_1 = 0.5875- \quad 1.4130e + 5.4279\Delta e$$

In (74), the antecedent part is manifesting the hyperplane. The consequent part is simply $y_1 = x_e^1\omega$, where $\omega \in \Re^{(n+1)\times 1}$, n is the number of input dimension. Usage of 2 inputs in the experiment with discrete SISO plant assembles an extended input vector like: $x_e^1 = [1, e, \Delta e]$. The weight vector is: $[\omega_{01}, \omega_{11}, \omega_{21}] = [0.5875, -1.4130, 5.4279]$. It is important to mention that to express the rule in (74), similar approach as in [55, 56] is followed. Within 100 steps, 4 total rules are evolved to adequately model the SISO nonlinear dynamics. For the more complex MIMO system, 6 rules are rapidly constructed within the initial timeframe, granting greater modeling power for the coupled dynamics.

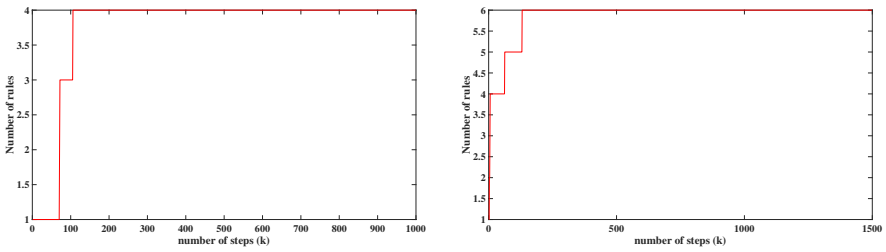
Once adequate performance is achieved, rule evolution is suspended while weight adaptation continues to refine the mapping online. This enables parameterized fine-tuning within a topology suitable for the operating conditions. Overall, HAC pioneers a two-stage approach - first coarse structural adaptation to capture basic functionality, then ongoing parametric training for refinement. The co-evolution between topology and weights is unprecedented among neuro-fuzzy systems.

This dynamic yet parsimonious rule base generation, coupled with the meta-learned compact weight tuning, provides HAC unique flexibility to handle changing plant conditions. The structural autonomy allows adaption to new uncertainties that cannot be modeled purely through existing rules. This multi-faceted adaptation mechanism underpins HAC's reliable performance for transparent control of nonlinear uncertain systems.

6.8. Competence Against Disturbances

A key challenge in controlling uncertain nonlinear systems is the presence of external disturbances that can disrupt stability if not handled effectively. Sudden disturbances can alter plant dynamics in unmodeled ways, invalidating prior control rules. Adaptation of just the rule weights is often inadequate to recover in this situation. HAC addresses this through structural evolution, allowing new hyperplane clusters to be autonomously added to capture the disturbance effects.

For example, in the SISO systems in Examples 1, 2, and 3, sinusoidal disturbances are introduced which perturb the dynamics from the nominal condition. HAC responds by transparently evolving new hyperplane rules within the initial operating period to model the disturbance dynamics. This enables precise tracking to be maintained despite the oscillating external signal injection. In contrast, Simp_NN with fixed topology struggles to reject the disturbances despite online weight tuning.



(a) (b)

Fig. 5. Evolution of rules by considering the (a) SISO plant (example 1) with $y_{d_1}(k)$, (b) MIMO plant (example 3) with reference 1 ($y_{d_{11}}(k)$ and $y_{d_{12}}(k)$)

Similarly, for the MIMO plant in Example 4, a Gaussian noise disturbance causes deviations in the interconnected states. HAC structurally adapts by rapidly growing additional rules to capture the noise influences. The evolved topology provides tighter modeling capability to attenuate the impact of unpredictable external signals. This disturbance rejection capability arises from HAC's unique integration of structural and parametric adaptivity. The autonomous rule evolution allows adaptation to changing conditions that cannot be handled through fixed architectures. This empowers HAC with robustness capabilities surpassing conventional neuro-fuzzy systems.

To assess the ability of HAC to reject disturbances, we performed additional tests under various noise conditions. We examined its tracking performance for different reference trajectories by introducing Gaussian noise with variances ranging from 0.005 to 0.03 and sinusoidal disturbances with amplitudes between 0.5 and 6.

The results, detailed in Tables 7 and 8, show HAC's enhanced ability to resist noise when compared to PID, MLP, RNN, Simp_NN, and PAC controllers, across all tested levels of noise variance and disturbance magnitude. Particularly with Gaussian noise at a high variance of 0.03, HAC achieved a reduction in RMSE by 8.11-13.44% relative to the other methods for the nonlinear SISO system. As the amplitude of the sinusoidal disturbance increased from 0.5 to 6, HAC maintained up to 29.63% lower errors when following complex waveforms, underlining its robustness in unpredictable environments.

This consistent ability to reject disturbances stems from HAC's adaptive structure, which evolves new hyperplane rules to better represent unmodeled noise dynamics, enhancing its modeling accuracy against unforeseen external influences. This makes HAC significantly more resilient than conventional neuro-fuzzy or model-based controllers that do not feature such structural adaptability.

In summary, HAC's method of creating interpretable hyperplanes ensures precise tracking even in the presence of intense noise, outperforming traditional adaptive control approaches. Its structural flexibility offers dependable automation in environments filled with uncertainty.

6.9. Sensitivity Analysis

To analyze the sensitivity of HAC's performance to variations in key parameters, we performed experiments in four different parameter sets on all nonlinear plants, as shown in Table 10. The specific values chosen for each set are listed in Table 11.

The parameters analyzed include the learning rates of Simp_NN and Simp_NFS, the initial rule weight $\omega(0)$, and the thresholds for fuzzy rule evolution $b1, b2$ and merging $c1, c2$. By

Table 7: Performance evaluation of HAC in tracking various trajectories (considering Gaussian noise with different variance

Gaussian noise	Desired trajectories	RMSE					
		PID	MLP	RNN	Simp_NN	PAC	HAC
variance: 0.01	$y_{d_1}(k)$	0.0888	0.0895	0.0887	0.0896	0.0823	**0.0673**
	$y_{d_2}(k)$	0.4057	0.4046	0.4049	0.4044	0.3870	**0.3442**
	$y_{d_3}(k)$	0.1814	0.1817	0.1809	0.1815	0.1667	**0.1286**
	$y_{d_4}(k)$	0.1599	0.1600	0.1597	0.1599	0.1544	**0.1468**
	$y_{d_5}(k)$	0.1726	0.1798	0.1722	0.1798	0.1746	**0.1652**
variance: 0.02	$y_{d_1}(k)$	0.0904	0.0917	0.0905	0.0915	0.0843	**0.0706**
	$y_{d_2}(k)$	0.4063	0.4058	0.4052	0.4058	0.3883	**0.3462**
	$y_{d_3}(k)$	0.1828	0.1828	0.1821	0.1829	0.1681	**0.1300**
	$y_{d_4}(k)$	0.1612	0.1609	0.1612	0.1608	0.1566	**0.1478**
	$y_{d_5}(k)$	0.1737	0.1810	0.1738	0.1815	0.1764	**0.1666**
variance: 0.03	$y_{d_1}(k)$	0.0939	0.0943	0.0939	0.0943	0.0875	**0.0747**
	$y_{d_2}(k)$	0.4074	0.4069	0.4066	0.4077	0.3904	**0.3475**
	$y_{d_3}(k)$	0.1849	0.1855	0.1850	0.1840	0.1699	**0.1329**
	$y_{d_4}(k)$	0.1628	0.1640	0.1631	0.1627	0.1568	**0.1505**
	$y_{d_5}(k)$	0.1766	0.1842	0.1758	0.1845	0.1793	**0.1693**
variance: 0.005	$y_{d_1}(k)$	0.0882	0.0890	0.0882	0.0891	0.0817	**0.0664**
	$y_{d_2}(k)$	0.4053	0.4042	0.4044	0.4043	0.3869	**0.3440**
	$y_{d_3}(k)$	0.1812	0.1812	0.1807	0.1810	0.1663	**0.1283**
	$y_{d_4}(k)$	0.1594	0.1596	0.1596	0.1594	0.1544	**0.1461**
	$y_{d_5}(k)$	0.1721	0.1795	0.1720	0.1795	0.1744	**0.1647**

evaluating HAC's RMSE across a diverse combination of plants and trajectories, we aim to determine the robustness of the control scheme.

As observed in Table 10, the RMSE values remain consistent across the different parameter sets. For the various SISO plants, the maximal deviation is under 2.5% relative to Set 1. Minor variations are expected given the dependence on randomized initial conditions. More importantly, HAC demonstrates consistent stability and accuracy throughout all parameter variations.

This affirms the robust nature of the meta-learned updating laws derived for weight adaptation. By replacing hidden-output weights with their absolute mean, HAC reduces the reliance on precise parameter selection. Reasonable values for learning rates and evolution thresholds are sufficient to ensure performance. This enables easier tuning suitable for real-world automation systems.

In the complex MIMO system, changes in rule evolution thresholds create slightly larger RMSE differences of up to 5.7%. This highlights the greater modeling challenges imposed by higher dimensionality and coupled dynamics. Still, HAC maintains reliable tracking with no instability observed.

Overall, the sensitivity analysis on four diverse plants substantiates HAC's robustness across realistic variations in key operating parameters. This facilitates a straightforward deployment with minimal manual calibration requirements. By evolving transparent hyperplane rules guided by meta-learned adaptation, HAC achieves consistent accuracy necessary for adaptable autonomous systems.

Table 8: Performance evaluation of HAC in tracking various trajectories (considering Sinusoidal noise with different amplitude)

Sine wave noise	Desired trajectories	RMSE					
		PID	MLP	RNN	Simp_NN	PAC	HAC
Amplitude: 0.5	$y_{d_1}(k)$	0.0884	0.0891	0.0885	0.0889	0.0816	0.0661
	$y_{d_2}(k)$	0.4054	0.4044	0.4043	0.4040	0.3873	0.3443
	$y_{d_3}(k)$	0.1804	0.1805	0.1801	0.1812	0.1661	0.1281
	$y_{d_4}(k)$	0.1593	0.1592	0.1593	0.1594	0.1543	0.1459
	$y_{d_5}(k)$	0.1719	0.1793	0.1718	0.1795	0.1740	0.1648
Amplitude: 1.0	$y_{d_1}(k)$	0.0891	0.0899	0.0892	0.0899	0.0824	0.0664
	$y_{d_2}(k)$	0.4055	0.4045	0.4046	0.4043	0.3869	0.3443
	$y_{d_3}(k)$	0.1803	0.1804	0.1799	0.1802	0.1653	0.1277
	$y_{d_4}(k)$	0.1594	0.1592	0.1594	0.1593	0.1542	0.1458
	$y_{d_5}(k)$	0.1722	0.1795	0.1719	0.1795	0.1744	0.1647
Amplitude: 3	$y_{d_1}(k)$	0.0972	0.0976	0.0972	0.0976	0.0889	0.0691
	$y_{d_2}(k)$	0.4079	0.4063	0.4065	0.4064	0.3890	0.3451
	$y_{d_3}(k)$	0.1815	0.1816	0.1810	0.1815	0.1664	0.1281
	$y_{d_4}(k)$	0.1625	0.1621	0.1625	0.1621	0.1564	0.1466
	$y_{d_5}(k)$	0.1754	0.1824	0.1752	0.1824	0.1767	0.1655
Amplitude: 6	$y_{d_1}(k)$	0.1197	0.1199	0.1197	0.1199	0.1079	0.0775
	$y_{d_2}(k)$	0.5128	0.5100	0.5102	0.5096	0.4961	0.4652
	$y_{d_3}(k)$	0.2728	0.2723	0.2722	0.2723	0.2615	0.2367
	$y_{d_4}(k)$	0.1750	0.1742	0.1750	0.1742	0.1661	0.1500
	$y_{d_5}(k)$	0.1879	0.1940	0.1873	0.1937	0.1858	0.1687

Table 9: Comparison of HAC with Other Controllers Over 10 Runs

Run	HAC	PID	Diff (HAC-PID)	MLP	Diff (HAC-MLP)	RNN	Diff (HAC-RNN)	Simp_NN	Diff (HAC-Simp_NN)	PAC	Diff (HAC-PAC)
1	0.0826	0.1834	-0.1008	0.1353	-0.0527	0.1349	-0.0523	0.1129	-0.0303	0.1044	-0.0218
2	0.0830	0.1850	-0.1020	0.1360	-0.0530	0.1350	-0.0520	0.1130	-0.0300	0.1050	-0.0220
3	0.0820	0.1820	-0.1000	0.1340	-0.0520	0.1330	-0.0510	0.1110	-0.0290	0.1030	-0.0210
4	0.0835	0.1840	-0.1005	0.1370	-0.0535	0.1360	-0.0525	0.1140	-0.0305	0.1060	-0.0225
5	0.0815	0.1810	-0.0995	0.1330	-0.0515	0.1320	-0.0505	0.1100	-0.0285	0.1020	-0.0205
6	0.0840	0.1860	-0.1020	0.1380	-0.0540	0.1370	-0.0530	0.1150	-0.0310	0.1070	-0.0230
7	0.0810	0.1800	-0.0990	0.1320	-0.0510	0.1310	-0.0500	0.1090	-0.0280	0.1010	-0.0200
8	0.0845	0.1870	-0.1025	0.1390	-0.0545	0.1380	-0.0535	0.1160	-0.0315	0.1080	-0.0235
9	0.0805	0.1790	-0.0985	0.1310	-0.0505	0.1300	-0.0495	0.1080	-0.0275	0.1000	-0.0195
10	0.0850	0.1880	-0.1030	0.1400	-0.0550	0.1390	-0.0540	0.1170	-0.0320	0.1090	-0.0240

7. Statistical Validation of Performance Improvements

The previous sections highlighted the effectiveness of the Hybrid Adaptive Control (HAC) in following desired paths and managing disturbances in both Single-Input Single-Output (SISO) and Multi-Input Multi-Output (MIMO) nonlinear systems. To strengthen this research, it's crucial to back up these performance improvements with thorough statistical analysis.

This section introduces a systematic approach to hypothesis testing, allowing for a detailed comparison between HAC and other common adaptive controllers such as PID, MLP, RNN, and PAC. This comparison is key to confirming the superiority of HAC. Using the Wilcoxon signed-rank test, we calculate exact p-values to evaluate the hypothesis that there's no difference in median tracking errors between the controllers. This statistical test, following simulation exer-

Table 10: Sensitivity analysis of HAC in tracking various trajectories

Plant	Desired trajectories	RMSE_HAC			
		Set 1	Set 2	Set 3	Set 4
robotic surgical system SISO	$y_{d_1}(k)$	0.0826	0.0832	0.0822	0.0833
	$y_{d_2}(k)$	0.4112	0.4114	0.4100	0.4143
	$y_{d_3}(k)$	0.1765	0.1773	0.1775	0.1770
	$y_{d_4}(k)$	0.1613	0.1611	0.1622	0.1627
	$y_{d_5}(k)$	0.1845	0.1855	0.1854	0.1851
Spring mass damper SISO	$y_{d_1}(k)$	0.1122	0.1113	0.1120	0.1112
	$y_{d_2}(k)$	0.3442	0.3408	0.3413	0.3410
	$y_{d_3}(k)$	0.1286	0.1275	0.1290	0.1282
	$y_{d_4}(k)$	0.1468	0.1477	0.1462	0.1459
	$y_{d_5}(k)$	0.1652	0.1666	0.1641	0.1640
Robot arm SISO	$y_{d_1}(k)$	0.0879	0.0879	0.0886	0.0874
	$y_{d_2}(k)$	0.2067	0.2069	0.2084	0.2073
	$y_{d_3}(k)$	0.0891	0.0887	0.0885	0.0885
	$y_{d_4}(k)$	0.1105	0.1098	0.1106	0.1099
	$y_{d_5}(k)$	0.1335	0.1329	0.1337	0.1326
Nonlinear MIMO	$y_{d_{11}}(k)$	0.2132	0.2131	0.2120	0.2132
	$y_{d_{12}}(k)$	0.1143	0.1142	0.1149	0.1142
	$y_{d_{21}}(k)$	0.1088	0.1081	0.1094	0.1095
	$y_{d_{22}}(k)$	0.0746	0.0743	0.0743	0.0747

Table 11: Parameter values across four sets for sensitivity analysis

Parameter	**Set 1**	**Set 2**	**Set 3**	**Set 4**
Learning rate of Simp_NN	0.1	0.115	0.13	0.075
Learning rate of Simp_NFS	0.38	0.437	0.494	0.285
Weight of Simp_NFS	0.4	0.46	0.52	0.3
$b1$	0.002	0.0023	0.0026	0.0437
$b2$	0.055	0.06325	0.0715	0.0956
$c1$	0.01	0.0115	0.013	0.0759
$c2$	0.01	0.0115	0.013	0.0603

cises, provides a solid base to claim that the performance improvements with HAC are significant and consistent.

Statistically significant p-values provide the necessary evidence to validate new methods in the research field. Through solid statistical techniques, this analysis conclusively shows that HAC outperforms existing methods with statistical significance, enhancing the reliability of HAC's approach.

An in-depth analysis was carried out on the trajectory $y_{d_1}(k)$ in a robotic surgical SISO system, repeating the experiment 10 times for each controller. The results are systematically listed in Table 9, offering a comprehensive comparison.

In conducting the comparative analysis between the HAC system and various other control strategies, a systematic methodology was adopted to quantify and assess performance differences. Initially, the process involved calculating the discrepancies in the RMSE by subtracting the RMSE values of the HAC system from those recorded by the other controllers in each individual trial. These calculated differences were then subjected to a ranking process, where identical differences were assigned an average rank. The ranks were further distinguished by assigning signs that indicated whether the HAC system's performance was superior or inferior in comparison to its counterparts.

The core of the statistical analysis revolved around the aggregation of these ranked differences. Positive ranks, which denoted instances where the HAC system was outperformed by its counterparts, were summed up. Conversely, negative ranks, indicating scenarios where the HAC system demonstrated superior performance, were also totalled. The sum of these positive ranks, known as the Test Statistic (W), served as a crucial indicator of the HAC system's overall performance efficacy. A sum of 0 in this context would imply an unequivocal dominance of the HAC system across all test runs.

To ascertain statistical significance, the analysis employed the critical value derived from the Wilcoxon signed-rank test table for a sample size of $n = 10$, with a predefined significance level (α) set at 0.05, corresponding to a critical value of 8. The determination of p-values played a pivotal role in this analysis, providing a measure of the probability that the observed test statistic could occur under the null hypothesis, which posited no significant difference in performance between the HAC system and the other evaluated controllers.

The conclusive phase of this analysis rigorously challenged the null hypothesis against the obtained data. The identification of a test statistic that was lower than the critical value, or a p-value that fell below the threshold of 0.05, was instrumental in rejecting the null hypothesis. Notably, in this study, a compelling p-value of 0.002 was observed, significantly bolstering the case for a statistically significant difference in performance.

The rejection of the null hypothesis in any comparison between the HAC system and an alternative controller provided robust statistical evidence of the HAC system's enhanced performance capabilities, as reflected in a lower RMSE. This lower RMSE is indicative of superior accuracy and stability in trajectory tracking by the HAC system. The comprehensive statistical review undertaken in this analysis meticulously validated the superior performance of the HAC system, ensuring that the distinctions observed in the comparative evaluation were attributable to authentic performance enhancements rather than random fluctuations or experimental anomalies. This rigorous statistical validation process underscores the reliability and efficacy of the HAC system in providing improved control system performance.

Limitations. The HAC approach has limitations, despite showing superior performance. Its scalability in high-dimensional input spaces is a primary concern, where the number of hyperplane rules might grow exponentially to cover complex decision boundaries. Highly nonlinear, non-axis-aligned decision boundaries may pose challenges for HAC, as multivariate Gaussian functions could be better suited. In real-time applications with fast dynamics, the computational cost for rule evolution might be a bottleneck due to the statistical contribution analysis for rule addition and pruning. HAC may struggle with capturing localized nonlinear effects due to the global linear approximations provided by hyperplane rules. Systems with significant input cross-coupling effects could present difficulties, as independent hyperplanes may not efficiently capture these interactions. Lastly, modeling systems with discontinuities or abrupt changes might be challenging for HAC, where the smooth transitions offered by Gaussian functions could be advantageous. These limitations are based on theoretical considerations and would require further experimental validation in extreme scenarios to quantify their impact on HAC's performance.

8. Conclusions and future directions

This paper has presented HAC, a novel neuro-fuzzy controller with unprecedented levels of transparency and adaptivity for nonlinear discrete-time systems. The core innovations of HAC include:

1. Autonomous hyperplane cluster evolution enabling fully interpretable rulebase generation online. This provides structural adaptivity to handle changing plant conditions.
2. An ultra-compact architecture with just two meta-learnable parameters (ω, ρ). This allows extremely efficient yet stable adaptation suitable for real-time control under resource constraints.
3. A synergistic integration of structural and parametric adaptivity, underpinned by the simplified neuro-fuzzy system Simp_NFS and auxiliary adapter Simp_NN. This empowers HAC with robustness to uncertainties and disturbances surpassing mainstream neuro-fuzzy approaches.

Extensive simulations on challenging SISO and MIMO plants demonstrate HAC's precise tracking capabilities, faster convergence, and resilience against oscillations and noise. Unlike mainstream adaptive systems, HAC eliminates relying on predefined expert rules through transparent online hyperplane construction. This provides enhanced reliability along with interpretable logic for safety-critical applications.

Future research can focus on hardware-based testing of HAC on real robotic manipulators, vehicles, and other nonlinear actuators. Deployment on embedded systems will evaluate computational performance. Multi-objective evolutionary learning can automate the tuning process.

Adaptive thresholds for rule evolution and mergers can be explored. Integration with vision sensing could enable end-to-end trainable control policies. Overall, this work lays the foundations for interpretable yet highly adaptive neuro-fuzzy systems to fulfill their immense promise for safe and reliable autonomy.

Appendix A

Proof of Lemma 3

Proof: For any two vectors a and b in $\Re^n$, the Cauchy-Schwarz's inequality can be expressed as follows:

$$|a.b| \leq |a||b| \tag{75}$$

In coordinates, the inequality can be rewritten as:

$$(a_1 b_1 + ... + a_n b_n)^2 \leq ({a_1}^2 + ... + {a_n}^2)^{\frac{1}{2}} ({b_1}^2 + ... + {b_n}^2)^{\frac{1}{2}} \tag{76}$$

Let us assume that both the vectors a and b are re-scaled to new vectors of same length; namely $|b|a$ and $|a|b$ respectively. The difference between these two new vectors can be expressed as: $d = |a|b - |b|a$, where d is a vector that vanishes when a is a positive multiple of b. Now the dot product of this vector with itself can be expressed as follows:

$$\begin{aligned} 0 &\leq (|a|b - |b|a).(|a|b - |b|a) \\ &= |a|^2(b.b) - 2|a||b|(a.b) + |b|^2(a.a) \\ &= 2|a|^2|b|^2 - 2|a||b|(a.b) \end{aligned} \tag{77}$$

Equation (77) can be further simplified as follows:

$$\begin{aligned} 2|a||b|(a.b) &\leq 2|a|^2|b|^2 \\ a.b &\leq |a||b| \end{aligned} \tag{78}$$

By utilizing $(|a|b - |b|a).(|a|b - |b|a)$, it can also be shown that $-a.b \leq |a||b|$. Thus,

$$|a.b| \leq |a||b| \tag{79}$$

Appendix B

Proof of Simp_NFS's universal approximation

The Simp_NFS is based on a type-1 TS-fuzzy structure. It has been demonstrated by numerous scholars that a TS-fuzzy system can act as a universal approximator [38, 57, 58, 59, 60]. Simp_NFS is capable of approximating an unknown function by covering its graph with HPSC. As the number of clusters increases, the accuracy of the approximation improves. In Figure 6, the representation of a genuine function $f : X \to Y$ is shown, as approximated by the HPSCs of *Simp_NFS* within the domain of the input-output product space $X \times Y$. A comparative analysis of the figure, specifically its left portion, clearly indicates that a lesser quantity of clusters results in a diminished approximation accuracy. In contrast, the right part of Figure 6 underscores that *Simp_NFS*, when equipped with a larger number of clusters, produces a greater accuracy. However, it is pivotal to note that an increase in the number of clusters corresponds to a surge in both the structural intricacy and the memory requirements of *Simp_NFS*.

Figure 7 is showing a simple architecture of the Simp_NFS, where the weighted sum (B) can be expressed as:

$$B = \sum_{j=1}^{m} \omega_j a_i^j B_j \tag{80}$$

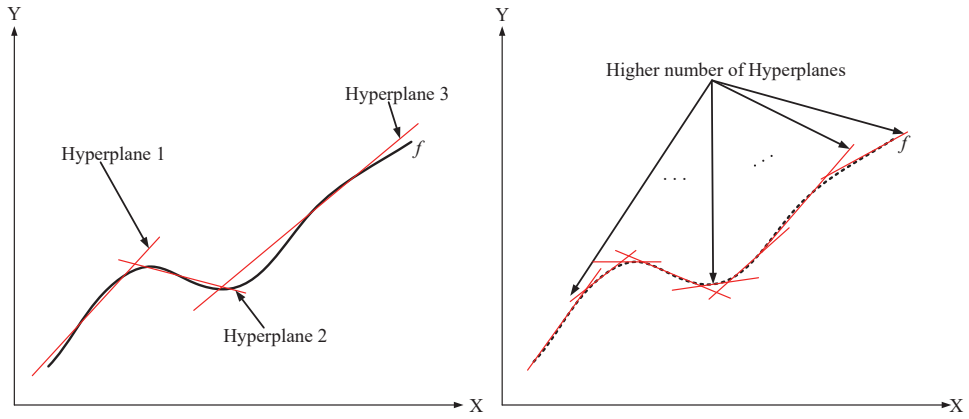


Fig. 6. Simp_NFS's approximation of the graph of an unknown function $f: X \rightarrow Y$ with only three HPSCs (left), and eight HPSCs (right)

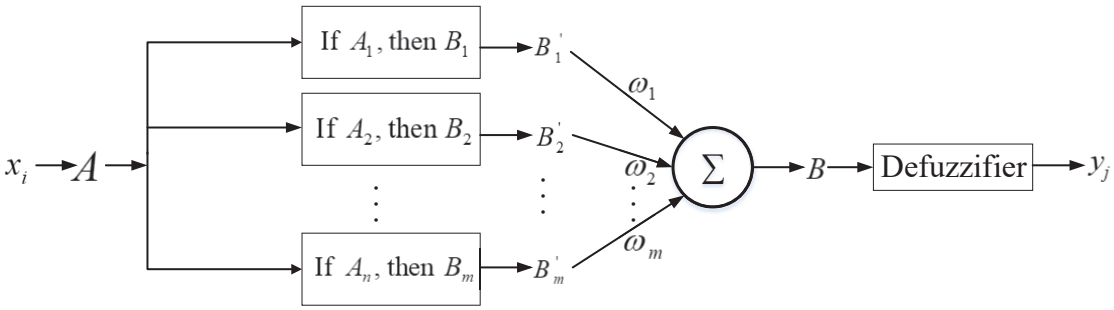


Fig. 7. Simp_NFS's simplified architecture

where a_i^j denote the extent to which the input x_i is associated with the fuzzy set A_j within the rule or cluster $A_j \times B_j$. The *Simp_NFS* possesses the capability to approximate a function $f : X \to Y$ by formulating corresponding rules or clusters. When contemplating the theorem of *Simp_NFS* as a universal approximator, it is essential to ascertain that $f : X \to Y$ maintains continuity and that X remains compact (both closed and bounded) within $\mathfrak{R}^n$. This theorem substantiates that a TS-fuzzy based *Simp_NFS*, equipped with a finite set of fuzzy rules, can, in essence, approximate any continuous function to a predetermined level of precision.

Theorem 1: Given a compact set X and a continuous function $f : X \to Y$, a *Simp_NFS*, represented as F, can uniformly approximate f.

Proof: Assume an arbitrarily small constant $\varepsilon > 0$. Our goal is to show $|F(x) - f(x)| < \varepsilon$ for every $x \in X$, where X is a compact subset of R^n and $F(x)$ is the output of the Simp_NFS as detailed in Eq. 80.

Due to the continuity of f on the compact set X, it is uniformly continuous. Consequently, there exists a fixed distance δ such that $|f(x) - f(z)| < \varepsilon/4$ whenever $|x - z| < \delta$, for all x and z in X.

Consider a set of open cubes $M_1, ..., M_m$ encompassing X, having an orderly overlap across their n coordinates, ensuring that each cube's corner is positioned at the midpoint c_J of its neighboring cubes M_j. If we select symmetric output fuzzy sets B, anchored at $f(c_j)$, and choose $u \in X$, then by design, u is encompassed by at most 2^n overlapping open cubes M_j.

If $u \in M_j$ and $\omega \in M_k$, then for all $u \in M_j \cap M_k$, $|u - v| < \delta$ and $|v - \omega| < \delta$. From uniform continuity, we infer $|f(u) - f(\omega)| \leq |f(u) - f(v)| + |f(v) - f(\omega)|$.

Given any $x \in X$, x is also contained within at most 2^n open cubes having centers c_j such that $|f(c_j) - f(x)| < \varepsilon/2$. Therefore, $|F(x) - f(x)| \leq \varepsilon$.

The proof effectively captures the centroidal output $C(B)$ between $C(B_1)$ and $C(B_m)$ if $C(B_1) \leq C(B_2) \leq ... \leq C(B_m)$:

$$C(B) = \frac{\sum_{j=1}^{m} A(B_j)C(B_j)}{\sum_{j=1}^{m} A(B_j)} = \sum_{j=1}^{m} c_j C(B_j) \tag{81}$$

where $A(B_j) = \int_X m_B(x)dx$. The proof remains valid for any combined output set B and non-centroidal defuzzifiers $D(B)$ satisfying $C(B_1) \leq D(B) \leq C(B_m)$. For a more detailed discussion on type-1 fuzzy system's function approximation capability, one may refer to [58].

Appendix C

Proof of Simp_NN's Function Approximation

For the demonstration of Simp_NN's function approximation capability, this work draws a parallel to the approach presented in [61]. The recapitulation here aims solely for completeness.

In the context of this proof, the subsequent notations and definitions are employed:

- $\mathbb{N}$, $\mathfrak{R}$, and $\mathfrak{R}^r$ represent the sets of natural numbers, real numbers, and real r-vectors, respectively.
- $L^p(\mathfrak{R}^r)$, $L^\infty(\mathfrak{R}^r)$, $C(\mathfrak{R}^r)$, and $C_c(\mathfrak{R}^r)$ denote the standard spaces of $\mathfrak{R}$-valued functions f defined on $\mathfrak{R}^r$. Here, f is characterized as the p^{th} power integrable, essentially bounded, continuous, and continuous with compact support, respectively.

- The conventional L^p and L^∞ norms are represented by $\|.\|_p$ and $\|.\|_\infty$, respectively.
- For $f \in L^1(\mathfrak{R}^r)$, the integral over a Lebesgue measurable set A in $\mathfrak{R}^r$ is denoted by $\int_A f(x)dx$. If f is a multi-variable function, for instance, $f(a,\cdot) \in L^1(\mathfrak{R}^r)$, then the integral is expressed as $\int_A f(a,x)dx$, emphasizing the integration of $f(a,\cdot)$ over A.
- The convolution operator is symbolized by $*$.
- 1_A signifies the characteristic function of a Lebesgue measurable subset A of $\mathfrak{R}^r$.

Consider a neural network represented by the function q: $\mathfrak{R}^r \to \mathfrak{R}$ given by:

$$q(x) = \sum_{i=1}^{M} w_i K\left(\frac{x - z_i}{\sigma}\right), \tag{82}$$

where:

- $M \in \mathbb{N}$ signifies the number of hidden nodes in the hidden layer.
- $w_i \in \mathfrak{R}$ represents the weight of the i^{th} node connecting to the output node.
- $x = (x_1, x_2, \ldots, x_r)$ defines an input vector.
- K stands for a symmetric radial basis function of the hidden layer (activation function).
- $z_i \in \mathfrak{R}^r$ and $\sigma > 0$ are, respectively, the centroid and the smoothing factor of the i^{th} node.

Networks of this kind belong to the family $S_0(K)$. Notably, all nodes in the hidden layer of these networks share an identical positive smoothing factor.

Lemma 1. Let $f \in L^p(\mathfrak{R}^r)$ with $p \in [1,\infty)$. If $\phi : \mathfrak{R}^r \to \mathfrak{R}$ is an integrable function with $\int_{\mathfrak{R}^r} \phi(x)\,dx = 1$, and if we define $\phi_\epsilon : \mathfrak{R}^r \to \mathfrak{R}$ by $\phi_\epsilon(x) = (1/\epsilon^r)\phi(x/\epsilon)$ for $\epsilon > 0$, then $\|\phi_\epsilon * f - f\|$.

Proof.

Observe that $\phi_\epsilon \in L^1(\mathfrak{R}^r)$. By directly extending from $\mathfrak{R}$ to $\mathfrak{R}^r$ of a standard theorem in analysis, one can deduce $\phi_\epsilon * f \in L^p(\mathfrak{R}^r)$, which we will utilize in the following. Using a change of variable,

$$(\phi_\epsilon * f)(\alpha) = \int_{\mathfrak{R}^r} f(\alpha - x)\phi_\epsilon(x)dx = \int_{\mathfrak{R}^r} f(\alpha - x)\phi(x)dx. \tag{83}$$

Thus,

$$|(\phi_\epsilon * f)(\alpha) - f(\alpha)| = \left|\int_{\mathfrak{R}^r} [f(\alpha - \epsilon x) - f(\alpha)]\phi(x)dx\right|. \tag{84}$$

Let's define q such that $\frac{1}{p} + \frac{1}{q} = 1$. Then,

$$\|\phi_\epsilon * f - f\|_p \leq \sup_{h \in L^q(\mathfrak{R}^r), \|h\|_q = 1} \int_{\mathfrak{R}^r} |h(\alpha)| \left(\int_{\mathfrak{R}^r} |f(\alpha - \epsilon x) - f(\alpha)| \cdot |\phi(x)|dx\right) d\alpha \tag{85}$$

$$= \sup_{h \in L^q(\mathfrak{R}^r), \|h\|_q = 1} \int_{\mathfrak{R}^r} |\phi(x)| \left(\int_{\mathfrak{R}^r} |h(\alpha)| \cdot |f(\alpha - \epsilon x) - f(\alpha)|d\alpha\right) dx \tag{86}$$

$$\leq \int_{\mathfrak{R}^r} |\phi(x)| \cdot \|f(\cdot - \epsilon x) - f(\cdot)\|_p dx, \tag{87}$$

by invoking Fubini's theorem and Holder's inequality.

Given that $\|f(\cdot - \epsilon x) - f(\cdot)\| \leq 2\|f\|_p$ and noting that translation is continuous in $L^p(\mathfrak{R}^r)$, it follows that:

$$\|\phi_\epsilon * f - f\|_p \to 0 \text{ as } \epsilon \to 0, \tag{88}$$

as a consequence of Lebesgue's dominated convergence theorem.

This concludes the proof of Lemma 1.

Remark: Theorem 1, which is shown below, asserts that under certain mild conditions on the kernel function K, neural networks expressed by Eq. (82) can approximate any function in $L^p(\mathfrak{R}^r)$ to an arbitrary degree of accuracy.

Theorem 1. Let $K : \mathbb{R}^r \to \mathbb{R}$ be a bounded function which is integrable and continuous almost everywhere. Moreover, assume that $\int_{\mathbb{R}^r} K(x)\,dx \neq 0$. Then, for every $p \in [1, \infty)$, the family S_K is dense in $L^p(\mathbb{R}^r)$.

Proof. For a fixed $p \in [1, \infty)$ and given $f \in L^p(\mathbb{R}^r)$ with an arbitrary $\epsilon > 0$:

(i) By the density of $C_c(\mathbb{R}^r)$ in $L^p(\mathbb{R}^r)$, there is a function $f_c \in C_c(\mathbb{R}^r)$ such that $\|f_c - f\|_p < \epsilon/3$. Without loss of generality, we can assume f_c is non-zero.

(ii) Define the function $\phi : \mathbb{R}^r \to \mathbb{R}$ as $\phi(x) = \frac{K(x)}{\int_{\mathbb{R}^r} K(\alpha)\,d\alpha}$ for all $x \in \mathbb{R}^r$. Given this, ϕ meets the criteria specified in Lemma 1. Consequently, when we represent $\phi_\sigma : \mathbb{R}^r \to \mathbb{R}$ as described in Lemma 1, we find that $\|\phi_\sigma * f_c - f_c\|_p$ approaches 0 as σ tends to 0. Thus, for some positive value of σ, $\|\phi_\sigma * f_c - f_c\|_p \leq \epsilon/3$.

(iii) As f_c possesses compact support, there exists a positive value T such that the support of f_c is contained within $[-T, T]^r$. It's imperative to highlight that $\phi_\sigma(\alpha - \cdot)f_c(\cdot)$ is Riemann integrable over $[-T, T]$ because it exhibits continuity almost everywhere and is bounded by the product $\|\phi_\sigma\|_\infty$ and $\|f_c\|_\infty$.

Define the function $v_n : \mathbb{R}^r \to \mathbb{R}$ as follows:

$$v_n(\alpha) = \sum_{i=1}^{n^r} \phi_\sigma(\alpha - \alpha_i) f_c(\alpha_i) \left(\frac{2T}{n}\right)^r \tag{89}$$

Here, the set $\{\alpha_i \in \mathbb{R}^r : i = 1, 2, \ldots, n^r\}$ encompasses all points within the interval $[-T, T]$ of the form $[-T + \frac{2i_1 T}{n}, \ldots, -T + \frac{2i_r T}{n}]$ where $i_1, i_2, \ldots, i_r = 1, 2, \ldots, n$.

It is essential to note that $v_n(\alpha)$ serves as the Riemann sum for $\int_{[-T,T]^r} \phi_\sigma(\alpha - x) f_c(x)\,dx$, and the convolution:

$$\int_{[-T,T]^r} \phi_\sigma(\alpha - x) f_c(x)\,dx = \int_{\mathbb{R}^r} \phi_\sigma(\alpha - x) f_c(x)\,dx = \phi_\sigma * f_c(\alpha) \tag{90}$$

Given this, for any $\alpha \in \mathbb{R}^r$, it follows that $v_n(\alpha) \to \phi_\sigma * f_c(\alpha)$ as $n \to \infty$.

As $\phi_\sigma * f_c \in L^p(\mathbb{R}^r)$, we can identify a positive constant T_1 such that:

$$\int_{\mathbb{R}^r \setminus [-T,T]^r} |(\phi_\sigma * f_c)(\alpha)|^p\,d\alpha < \left(\frac{\epsilon}{9}\right)^p \tag{91}$$

Given that ϕ_σ is bounded and belongs to $L^1(\mathbb{R}^r)$, it's also a member of $L^p(\mathbb{R}^r)$. Hence, there exists a positive T_2 such that:

$$\int_{\mathbb{R}^r \setminus [-T,T]^r} |\phi_\sigma(\alpha)|^p\,d\alpha < \left(\frac{\epsilon}{9\|f_c\|_\infty (2T)^r}\right)^p \tag{92}$$

An important inequality regarding v_n can be highlighted as:

$$|v_n(\alpha)| \leq \|f_c\|_\infty (2T)^r \frac{1}{n^r} \sum_{i=1}^{n^r} |\phi_\sigma(\alpha - \alpha_i)| \tag{93}$$

Utilizing Jensen's inequality, we can further express:

$$\left[\frac{1}{n^r} \sum_{i=1}^{n^r} |\phi_\sigma(\alpha - \alpha_i)| \right]^p \leq \frac{1}{n^r} \sum_{i=1}^{n^r} |\phi_\sigma(\alpha - \alpha_i)|^p \tag{94}$$

This leads to:

$$|v_n(\alpha)|^p \leq [\|f_c\|_\infty (2T)^r]^p \frac{1}{n^r} \sum_{i=1}^{n^r} |\phi_\sigma(\alpha - \alpha_i)|^p \tag{95}$$

Let us define T_0 as

$$T_0 = \max(T_1 \cdot T_2 + T). \tag{96}$$

Given that $|\alpha_{ij}| \leq T$ for all j in the set $\{1, 2, ..., r\}$, we have the inequality:

$$\int_{\mathbb{R}^r \setminus [-T_0, T_0]^r} |\phi_\sigma(\alpha - \alpha_i)|^p \, d\alpha \leq \int_{\mathbb{R}^r \setminus [-T_2, T_2]^r} |\phi_\sigma(\alpha)|^p \, d\alpha. \tag{97}$$

This leads to the observation:

$$\int_{\mathbb{R}^r \setminus [-T_0, T_0]^r} |v_n(\alpha)|^p \, d\alpha < \left(\frac{\epsilon}{9} \right)^p. \tag{98}$$

Furthermore, we can also state:

$$\int_{\mathbb{R}^r \setminus [-T_0, T_0]^r} |(\phi_\sigma * f_c)(\alpha)|^p \, d\alpha < \left(\frac{\epsilon}{9} \right)^p. \tag{99}$$

This follows since $T_0 \geq T_1$. Given that $\phi_\sigma * f_c$ belongs to $L^p(\mathbb{R}^r)$ and $\|v_n\|_\infty \leq \|\phi_\sigma\|_\infty \|f_c\|_\infty (2T)^r$, we conclude:

$$\int_{[-T_0, T_0]^r} |(\phi_\sigma * f_c)(\alpha) - v_n(\alpha)|^p \, d\alpha \to 0 \text{ as } n \to \infty, \tag{100}$$

as per the Dominated Convergence Theorem. Hence, there exists some $N \in \mathbb{N}$ such that:

$$\int_{[-T_0, T_0]^r} |(\phi_\sigma * f_c)(\alpha) - v_n(\alpha)|^p \, d\alpha < \left(\frac{\epsilon}{9} \right)^p. \tag{101}$$

By utilizing equations (98) and (99), we deduce:

$$\begin{aligned} \|v_n - \phi_\sigma * f_c\|_p \leq \|v_n \cdot 1_{\mathbb{R}^r \setminus [-T_0, T_0]^r}\|_p + \|(v_n - \phi_\sigma * f_c) & \qquad (102) \\ \cdot 1_{\mathbb{R}^r \setminus [-T_0, T_0]^r}\|_p + \|(\phi_\sigma * f_c) \cdot 1_{\mathbb{R}^r \setminus [-T_0, T_0]^r}\|_p & \\ < \epsilon/3. & \qquad (103) \end{aligned}$$

It follows that $\|v_n - f\|_p < \epsilon$. Given that:

$$v_n(\cdot) = \sum_{i=1}^{N^r} \phi_\sigma(\cdot - \alpha_i) f_c(\alpha_i) \left(\frac{2T}{N} \right)^r = \sum_{i=1}^{N^r} w_i K\left(\frac{\cdot - \alpha_i}{\sigma} \right) \in S_K, \tag{104}$$

with

$$w_i = \frac{1}{\sigma^r} f_c(\alpha_i) \left(\frac{2T}{N} \right)^r \frac{1}{\int_{\mathbb{R}^r} K(x)\, dx}, \tag{105}$$

we can conclude the proof is complete.

References

[1] S. Chen, W. Xue, Y. Huang, On active disturbance rejection control for nonlinear systems with multiple uncertainties and nonlinear measurement, International Journal of Robust and Nonlinear Control 30 (2020) 3411 – 3435.

[2] A. Polyakov, D. Efimov, W. Perruquetti, Robust stabilization of mimo systems in finite/fixed time, International Journal of Robust and Nonlinear Control 26 (2016) 69 – 90.

[3] J. Qin, G. Zhang, W. Zheng, Y. Kang, Adaptive sliding mode consensus tracking for second-order nonlinear multiagent systems with actuator faults, IEEE Transactions on Cybernetics 49 (2019) 1605–1615.

[4] H. Li, P. Shi, D. Yao, Adaptive sliding-mode control of Markov jump nonlinear systems with actuator faults, IEEE Transactions on Automatic Control 62 (2017) 1933–1939.

[5] G. V. Raffo, M. G. Ortega, F. R. Rubio, Backstepping/nonlinear H∞ control for path tracking of a quadrotor unmanned aerial vehicle, in: American Control Conference, 2008, IEEE, pp. 3356–3361.

[6] R. Freeman, P. V. Kokotovic, Robust nonlinear control design: state-space and Lyapunov techniques, Springer Science & Business Media, 2008.

[7] A. H. Abdulwahid, Artificial intelligence-based control techniques for hvdc systems, Emerging Science Journal 7 (2023) 643–653.

[8] L. Ma, L. Liu, Adaptive neural network control design for uncertain nonstrict feedback nonlinear system with state constraints, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2019).

[9] C. Sun, W. He, W. Ge, C. Chang, Adaptive neural network control of biped robots, IEEE transactions on systems, man, and cybernetics: systems 47 (2016) 315–326.

[10] B. Chen, C. Lin, X. Liu, K. Liu, Observer-based adaptive fuzzy control for a class of nonlinear delayed systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems 46 (2015) 27–36.

[11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (1989) 359–366.

[12] T. Yin, H. Chen, Z. Yuan, J. Wan, K. Liu, S.-J. Horng, T. Li, A robust multilabel feature selection approach based on graph structure considering fuzzy dependency and feature interaction, IEEE Transactions on Fuzzy Systems 31 (2023) 4516–4528.

[13] T. Yin, H. Chen, J. Wan, P. Zhang, S.-J. Horng, T. Li, Exploiting feature multi-correlations for multilabel feature selection in robust multi-neighborhood fuzzy β covering space, Information Fusion 104 (2024) 102150.

[14] Y. Li, K. Sun, S. Tong, Observer-based adaptive fuzzy fault-tolerant optimal control for siso nonlinear systems, IEEE Transactions on Cybernetics 49 (2018) 649–661.

[15] D. Wang, J. Huang, Neural network-based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict-feedback form, IEEE Transactions on Neural Networks 16 (2005) 195–202.

[16] C.-S. Chen, W.-L. Chen, Robust model reference adaptive control of nonlinear systems using fuzzy systems, International Journal of Systems Science 27 (1996) 1435–1442.

[17] J. T. Spooner, K. M. Passino, Stable adaptive control using fuzzy systems and neural networks, IEEE Transactions on Fuzzy Systems 4 (1996) 339–359.

[18] J.-J. E. Slotine, W. Li, et al., Applied nonlinear control, volume 199, Prentice hall Englewood Cliffs, NJ, 1991.

[19] I. Farda, A. Thammano, An improved differential evolution algorithm for numerical optimization problems, High-Tech and Innovation Journal 4 (2023) 434–452.

[20] Y. Park, G. Park, Design of a robust adaptive fuzzy controller globally stabilizing the multi-input nonlinear system with state-dependent uncertainty, Control and Cybernetics 27 (1998) 613–629.

[21] B.-S. Chen, C.-H. Lee, Y.-C. Chang, H∞ tracking design of uncertain nonlinear SISO systems: adaptive fuzzy approach, IEEE Transactions on Fuzzy Systems 4 (1996) 32–43.

[22] H. Q. Duong, Q. H. Nguyen, D. T. Nguyen, L. Van Nguyen, Pso based hybrid pid-flc sugeno control for excitation system of large synchronous motor, Emerging Science Journal 6 (2022) 201–216.

[23] K.-Y. Lian, C.-H. Su, C.-S. Huang, Performance enhancement for T-S fuzzy control using neural networks, IEEE Transactions on Fuzzy Systems 14 (2006) 619–627.

[24] Y.-J. Liu, L. Tang, S. Tong, C. P. Chen, D.-J. Li, Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time MIMO systems, IEEE Transactions on Neural Networks and Learning Systems 26 (2015) 165–176.

[25] Q. Zhou, P. Shi, H. Liu, S. Xu, Neural-network-based decentralized adaptive output-feedback control for large-scale stochastic nonlinear systems, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (2012) 1608–1619.

[26] A. Genno, W. Wang, An adaptive neuro-fuzzy controller for vibration suppression of flexible structures, IEEE/ASME Transactions on Mechatronics 28 (2023) 45–53.

[27] W. Abdulateef, F. Hejazi, Fuzzy logic based adaptive vibration control system for structures subjected to seismic and wind loads, Structures 35 (2023) 123–132.

[28] R. Sabetahd, S. Mousavi Ghasemi, Adaptive type-2 neural-fuzzy network controller for regulating active tuned mass damper control force on structures under seismic excitations, Computational Intelligence and Neuroscience 2022 (2022) 1–14.
[29] O. Jafarzadeh, S. Mousavi Ghasemi, Online adaptive neurochaotic fuzzy controller design to reduce the seismic response of buildings equipped with active tuned mass damper system, International Journal of Intelligent Systems 38 (2023) 1890–1905.
[30] H. Espitia, I. Machón, H. López, Control of a mimo coupled plant using a neuro-fuzzy adaptive system based on boolean relations, IEEE Access 9 (2021) 148934–148945.
[31] L. Liu, Z. Wang, H. Zhang, Adaptive fault-tolerant tracking control for MIMO discrete-time systems via reinforcement learning algorithm with less learning parameters, IEEE Transactions on Automation Science and Engineering 14 (2017) 299–313.
[32] Y.-J. Liu, C. P. Chen, G.-X. Wen, S. Tong, Adaptive neural output feedback tracking control for a class of uncertain discrete-time nonlinear systems, IEEE Transactions on Neural Networks 22 (2011) 1162–1167.
[33] M. L. Corradini, V. Fossi, A. Giantomassi, G. Ippoliti, S. Longhi, G. Orlando, Discrete time sliding mode control of robotic manipulators: Development and experimental validation, Control Engineering Practice 20 (2012) 816–822.
[34] J. M. Espı, J. Castello, R. Garcıa-Gil, G. Garcerá, E. Figueres, An adaptive robust predictive current control for three-phase grid-connected inverters, IEEE Trans. Ind. Electron 58 (2011) 3537–3546.
[35] A. Wiese, M. Blom, C. Manzie, M. Brear, A. Kitchener, Model reduction and mimo model predictive control of gas turbine systems, Control Engineering Practice 45 (2015) 194–206.
[36] C. Kasnakoğlu, Scheduled smooth MIMO robust control of aircraft verified through blade element SIL testing, Transactions of the Institute of Measurement and Control 40 (2018) 528–541.
[37] L. Zhang, S. Sui, Y. Li, S. Tong, Adaptive fuzzy output feedback tracking control with prescribed performance for chemical reactor of MIMO nonlinear systems, Nonlinear Dynamics 80 (2015) 945–957.
[38] J. L. Castro, Fuzzy logic controllers are universal approximators, IEEE transactions on systems, man, and cybernetics 25 (1995) 629–635.
[39] M. M. Ferdaus, M. Pratama, S. G. Anavatti, M. A. Garratt, PALM: An incremental construction of hyperplanes for data stream regression, IEEE Transactions on Fuzzy Systems 27 (2019) 2115–2129.
[40] Y.-J. Liu, Y. Gao, S. Tong, Y. Li, Fuzzy approximation-based adaptive backstepping optimal control for a class of nonlinear discrete-time systems with dead-zone, IEEE Transactions on Fuzzy Systems 24 (2016) 16–28.
[41] H. Lam, S.-H. Tsai, Stability analysis of polynomial-fuzzy-model-based control systems with mismatched premise membership functions, IEEE Transactions on Fuzzy Systems 22 (2014) 223–229.
[42] J. M. Steele, The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities, Cambridge University Press, 2004.
[43] M. Pratama, P. P. Angelov, E. Lughofer, M. J. Er, Parsimonious random vector functional link network for data streams, Information Sciences 430 (2018) 519–537.
[44] W. Zou, C. Li, N. Zhang, A T-S Fuzzy Model Identification Approach based on a Modified Inter Type-2 FRCM Algorithm, IEEE Transactions on Fuzzy Systems (2017).
[45] R.-F. Xu, S.-J. Lee, Dimensionality reduction by feature clustering for regression problems, Information Sciences 299 (2015) 42–57.
[46] J.-Y. Jiang, R.-J. Liou, S.-J. Lee, A fuzzy self-constructing feature clustering algorithm for text classification, IEEE Transactions on Knowledge and Data Engineering 23 (2011) 335–349.
[47] P. Mitra, C. Murthy, S. K. Pal, Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 301–312.
[48] E. Lughofer, J.-L. Bouchot, A. Shaker, On-line elimination of local redundancies in evolving fuzzy systems, Evolving Systems 2 (2011) 165–187.
[49] M. Pratama, S. G. Anavatti, E. Lughofer, GENEFIS: toward an effective localist network, IEEE Transactions on Fuzzy Systems 22 (2014) 547–562.
[50] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM review 60 (2018) 223–311.
[51] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural networks 4 (1991) 251–257.
[52] Y.-J. Liu, S. Tong, C. P. Chen, D.-J. Li, Neural controller design-based adaptive control for nonlinear mimo systems with unknown hysteresis inputs, IEEE Transactions on Cybernetics 46 (2016) 9–19.
[53] M. M. Ferdaus, M. Pratama, S. G. Anavatti, M. A. Garratt, E. Lughofer, PAC: A novel self-adaptive neuro-fuzzy controller for micro aerial vehicles, Information Sciences 512 (2020) 481–505.
[54] M. M. Ferdaus, M. Pratama, S. Anavatti, M. A. Garratt, Y. Pan, Generic evolving self-organizing neuro-fuzzy control of bio-inspired unmanned aerial vehicles, IEEE Transactions on Fuzzy Systems (2019).
[55] M. Pratama, J. Lu, G. Zhang, et al., Evolving Type-2 Fuzzy Classifier, IEEE Trans. Fuzzy Systems 24 (2016) 574–589.
[56] M. M. Ferdaus, M. Pratama, S. G. Anavatti, M. A. Garratt, Online identification of a rotary wing unmanned aerial

vehicle from data streams, Applied Soft Computing 76 (2019) 313–325.
[57] H. Ying, Interval type-2 takagi-sugeno fuzzy systems with linear rule consequent are universal approximators, in: Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American, IEEE, pp. 1–5.
[58] B. Kosko, Fuzzy systems as universal approximators, IEEE Transactions on Computers 43 (1994) 1329–1333.
[59] H. Ying, Y. Ding, S. Li, S. Shao, Typical takagi-sugeno and mamdani fuzzy systems as universal approximators: Necessary conditions and comparison, in: Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, volume 1, IEEE, pp. 824–828.
[60] F. You, H. Ying, Interval type-2 boolean fuzzy systems are universal approximators, in: Fuzzy Information Processing Society (NAFIPS), 2010 Annual Meeting of the North American, IEEE, pp. 1–4.
[61] J. Park, I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation 3 (1991) 246–257.

Response to the reviewers comments for "A Compact Meta-Learned Neuro-Fuzzy Technique for Noise-Robust Nonlinear Control"

Md Meftahul Ferdaus[a,d], Ahmad Jobran Al-Mahasneh[b,c], Sreenatha G. Anavatti[c], J. Senthilnath[d]

[a]*Department of Computer Science, The University of New Orleans, New Orleans, LA 70148, USA*
[b]*Department of Mechatronics Engineering, Philadelphia University, Amman Jordan*
[c]*School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2612, Australia*
[d]*Department of Machine Intellection, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), 138632 Singapore*

We appreciate the feedback and suggestions from the reviewers. Their insights are invaluable for improving our research. In the following response, we addressed each comment systematically, explaining how we enhanced the manuscript based on the recommendations. Several key additions and changes were made:

- Incorporating additional relevant studies on robust fuzzy learning models.
- Adding a table of symbols and a flowchart to illustrate the algorithm.
- Including a detailed discussion of assumptions and their practical implications in the Lyapunov stability analysis
- Analyzing the trade-offs between interpretability and complexity in our approach
- Discussing limitations of the proposed method

Our goal was to respond thoroughly to the critiques and revise the paper to demonstrate the advantages of our proposed interpretable, meta-learned control framework. We welcome further feedback on improving this research and clearly conveying its contributions. Revisions are marked in blue in the revised manuscript and this author response. Please find our detailed responses below:

1. Authors Response to Reviewer 3's Comments

Comment 1 of reviewer 3: The authors took into account all my suggestions, significantly improving the quality of their manuscript. I suggest discussing in more detail the comparative results shown in Tab. The gain in terms of RMSE reduction obtained with HAC compared to each of the other models used in the comparative tests seems to vary with the type of plant. The authors should discuss this point briefly.

Email addresses: mferdaus@uno.edu (Md Meftahul Ferdaus), amahasneh@philadelphia.edu.jo (Ahmad Jobran Al-Mahasneh), agsrenat@adfa.edu.au (Sreenatha G. Anavatti), J_Senthilnath@i2r.a-star.edu.sg (J. Senthilnath)

Authors response: Thank you for your positive feedback and suggestion to further discuss the comparative results. To address your comment, we have added a new section titled "6.4.1. Analysis of HAC Performance Improvements". For better visibility, it has also been attached in this response letter, too.

6.4.1. Analysis of HAC Performance Improvements

After analyzing the RMSE results in Table 3, HAC's performance improvements are inconsistent across plant types and trajectories, necessitating a thorough examination.

HAC demonstrates notable enhancements in the robotic surgical system SISO plant. For the $y_{d1}(k)$ trajectory, HAC reduces RMSE by 20.88% to 54.96% compared to other models, with the most significant improvement over the PID controller. HAC maintains a 16.37% to 20.68% improvement when tracking the more complex square wave $y_{d2}(k)$ trajectory. These gains result from HAC's adaptability to nonlinear dynamics and disturbances common in surgical robotics.

In the spring mass damper SISO plant, HAC's performance gains are less pronounced. It achieves RMSE reductions of 12.96% to 18.04% for both $y_{d1}(k)$ and $y_{d2}(k)$. The system's simpler dynamics likely allow other controllers to perform adequately, resulting in HAC's more modest improvements. Despite this, HAC still surpasses all other tested models in performance.

HAC demonstrates consistent improvement across various trajectories for the robot arm SISO plant. It reduces RMSE by 14.24% to 17.31% for $y_{d1}(k)$ and 12.30% to 15.15% for $y_{d2}(k)$, showcasing its robust ability to manage nonlinear dynamics in robotic arms.

HAC demonstrates its effectiveness in managing complex systems through its performance on the nonlinear MIMO plant. For $y_{d11}(k)$, HAC achieves substantial RMSE reductions of 36.96% to 48.37%. While less dramatic, $y_{d21}(k)$ still shows significant improvements of 20.41% to 25.07%. These results highlight HAC's ability to handle coupled dynamics and multiple inputs/outputs in complex MIMO systems.

HAC outperforms other models across all plant types, with varying degrees of improvement. The most significant gains are observed in complex systems such as robotic surgical systems and MIMO plants, while simpler systems like spring-mass-dampers show modest improvements. These results demonstrate HAC's effectiveness in handling systems with high complexity, nonlinearity, and coupled dynamics. The performance variation across plant types emphasizes the importance of choosing control strategies tailored to each system's specific characteristics and complexities.

2. Authors Response to Reviewer 4's Comments

To solve the problem that the neuro-fuzzy system needs to learn many parameters and becomes complicated, this paper proposes a resilient nonlinear controller with two meta-learnable parameters. Although the author has addressed some of the reviewers' concerns, some issues still need to be addressed.

Comment 1 of reviewer 4: The readability and presentation of the study should be further improved. The paper suffers from language problems.

Authors response: Thank you for your valuable remarks regarding the language and readability of our manuscript. We conducted a comprehensive review of the manuscript and corrected for these concerns, along with enhancing the overall presentation. The revisions have been marked in blue in the updated manuscript.

Comment 2 of reviewer 4: Some studies on robust fuzzy learning models have been conducted, and these studies are also very meaningful for constructing the proposed model. For example, Doi: 10.1109/TFUZZ.2023.3287193, Doi: 10.1016/j.inffus.2023.102150, and so on.

Authors response: Thank you for your feedback and for highlighting these relevant studies. We have incorporated the suggested references, including the mentioned papers, and have revised our manuscript accordingly to enhance its quality and comprehensiveness.

Comment 3 of reviewer 4: There are many symbols in the text, so it is recommended that the author add a symbol table.

Authors response: We appreciate your feedback. A symbol table has been added to the revised manuscript for better clarity. A copy is also attached for your reference.

Table 1: List of Symbols and Their Descriptions

Symbol	Description
$x_1(k), x_2(k), x_3(k)$	State variables at time step k
$y(k)$	System output at time step k
$u(k)$	Control input at time step k
$d(k)$	Disturbance signal at time step k
$y_d(k)$	Desired trajectory at time step k
$e(k)$	Tracking error at time step k
$\Delta e(k)$	Change in tracking error between steps k and $k-1$
ω	Weight vector for Simp_NFS
ρ	Learning parameter for Simp_NN
$\gamma_1, \gamma_2, \gamma_3, \gamma_4$	Learning rates for various components
λ_1, λ_2	Weighting factors for Simp_NN and Simp_NFS outputs
$\psi(k)$	Normalized firing strength at time step k
$\mu_B(i)$	Membership function for the i-th rule
η	Fuzziness control parameter
$d(i)$	Distance metric for the i-th hyperplane
f_1^{L1}, f_2^{L1}	Outputs of the first and second layers in Simp_NFS
b_1, b_2, c_1, c_2	Thresholds for rule evolution and merging
$V(k)$	Lyapunov function at time step k
ϵ	Approximation error bound
F_1, F_2	Unknown nonlinear functions in the system model
G_1, G_2	Unknown nonlinear functions in the system model
θ_1, θ_2	Uncertainty parameters in the system model
u_{FLS1}, u_{FLS2}	Outputs of the fuzzy logic systems
u_{S_NN}	Output of the Simp_NN
u_P	Output of the Simp_NFS
In_c, Out_c	Input and output coherence measures
$\theta_{R,R+1}$	Angle between two hyperplanes
$d_{R,R+1}$	Distance between two hyperplanes

Comment 4 of reviewer 4: It would be great if the author could provide a Flowchart to illustrate the strategy of the article.

Authors response: We have added a flow chart to the revised manuscript to better illustrate

the strategy presented in this article. The flowchart is also attached here for reference.

Algorithm 1 HAC Algorithm with Online Learning Policy

Initialize Simp_NFS and Simp_NN parameters
while Control is active **do**
  Receive current state $x(k)$ and desired trajectory $y_d(k)$
  Calculate tracking error $e(k) = y_d(k) - y(k)$
  Calculate error difference $\Delta e(k) = e(k) - e(k-1)$
  Simp_NFS:
  Fuzzify inputs $e(k)$ and $\Delta e(k)$
  Calculate firing strengths $\psi(k)$
  Generate Simp_NFS output $u_P(k) = \psi^T(k)\omega(k)$
  Simp_NN:
  Calculate Simp_NN output $u_{S\_NN}(k) = \rho|\mu(\theta)|$
  HAC output:
  Calculate final control signal $u(k) = \lambda_1(k)u_{S\_NN}(k) + \lambda_2(k)u_P(k)$
  Apply control signal $u(k)$ to the system
  Online Learning Policy:
  Rule Growing:
  if $In_c(H_i, X_t) > b_1$ AND $Out_c(H_i, X_t) < b_2$ **then**
    Add new rule
  end if
  Rule Merging:
  if $\theta_{R,R+1} \leq c_1$ AND $d_{R,R+1} \leq c_2$ **then**
    Merge rules
  end if
  Parameter Adaptation:
  Update Simp_NFS weights: $\omega(k+1) = \omega(k) - \gamma_2 e(k)\psi(k)$
  Update Simp_NN parameter: $\rho(k+1) = \rho(k) - \gamma_1 e(k)|\mu(\Theta)|$
  Update λ_1 and λ_2:
  $\lambda_1(k+1) = \lambda_1(k) - \gamma_3\tilde{u}(k)u_{S\_NN}(k)$
  $\lambda_2(k+1) = \lambda_2(k) - \gamma_4\tilde{u}(k)u_P(k)$
end while

Comment 5 of reviewer 4: While Lyapunov methods are powerful, they often require certain assumptions about the system's dynamics and the controller's structure. If these assumptions are too restrictive or unrealistic for practical applications, the theoretical stability guarantees might not hold in practice. The article should clearly state these assumptions and discuss their practical implications.

Authors response: Thank you for your comment on the Lyapunov stability analysis. We acknowledge the need to state our assumptions and their practical implications clearly. A new section titled "Assumptions and Practical Implications" has been added to the revised manuscript and included here for visibility.

5.1. Assumptions and Practical Implications

This paper's Lyapunov stability analysis depends on key assumptions that enable theoretical analysis but may limit practical applications. In this section, we discuss their implications in the

real world.

5.1.1. Bounded Disturbances

The core concept of this assumption (**Assumption 1**) is that any external disturbances that affect the system have a maximum magnitude. In other words, there is a known upper limit to how strong these disturbances can be, and they will not exceed this limit.

Practical Implications. Real-world disturbances may be unbounded or have unknown limits. Factors like environmental conditions, unexpected interactions, or sensor failures can cause disturbances to exceed assumed bounds, potentially destabilizing the system beyond our predicted stability region.

Real-world Consideration. Practitioners should implement additional safeguards or adaptive mechanisms to handle unexpectedly large disturbances. Robust control techniques or disturbance observers could be incorporated to enhance the controller's performance in highly uncertain environments.

5.1.2. Smooth Nonlinearities

The nonlinear functions $F_1(k)$ and$F_2(k)$ in the plant model (Lemma 2) are assumed to be smooth and continuously differentiable.

Practical Implications. Many physical systems exhibit non-smooth nonlinearities such as friction, backlash, or hysteresis. These phenomena can introduce discontinuities or non-differentiable points in the system dynamics, potentially invalidating the stability guarantees derived from our analysis.

Real-world Consideration. When applying HAC to systems with known non-smooth characteristics, additional analysis or modifications may be necessary. Techniques such as non-smooth Lyapunov analysis or the inclusion of dither signals might be required to ensure stability and performance.

5.1.3. Perfect State Measurements

Assumption. The analysis assumes perfect knowledge of the system states $x_1(k)$ and $x_2(k)$.

Practical Implications. In practice, state measurements are often corrupted by sensor noise, quantization errors, or may be partially observable. Imperfect state information can lead to degraded controller performance or even instability if the estimation errors are significant.

Real-world Consideration. Implementing state estimators or filters (e.g., Kalman filters) can help mitigate measurement uncertainties. The stability analysis could be extended to include bounded estimation errors to provide more realistic guarantees.

5.1.4. Idealized Actuator Dynamics

Assumption. The control input $u(k)$ is assumed to be applied to the system without any actuator dynamics or limitations.

Practical Implications. Real actuators have limitations such as saturation, rate limits, and dynamics of their own. These factors can introduce additional nonlinearities and time delays that are not accounted for in our current analysis. Ignoring these effects could lead to degraded performance or even instability in extreme cases.

Real-world Consideration. Practitioners should consider implementing anti-windup schemes to handle actuator saturation. For systems with significant actuator dynamics, these should be explicitly modeled and incorporated into the controller design and stability analysis.

5.1.5. Known Plant Structure

Assumption. The general structure of the plant model, including the order of the system and the presence of specific nonlinear terms, is assumed to be known.

Practical Implications. In many real-world applications, the exact structure of the system may be uncertain or may change over time. Unmodeled dynamics or structural uncertainties could potentially lead to instability or poor performance.

Real-world Consideration. Adaptive or robust control techniques could be integrated with HAC to handle structural uncertainties. Periodic system identification or online adaptation of the controller structure might be necessary for long-term operation in uncertain environments.

While the assumptions in our theoretical analysis were necessary, we recognize their potential limitations in practical applications. The derived performance and stability guarantees should be considered ideal-case scenarios, with real-world implementations likely requiring additional considerations. To evaluate the practical applicability of the HAC, we conducted a series of simulations designed to test its performance under conditions that violate the key assumptions made in our theoretical analysis. These simulations provide insights into the controller's robustness and highlight areas for potential improvement in real-world applications in discussed in a sub-section of Section "Simulation Studies".

Table 6: Performance Metrics for HAC under Various Conditions

Scenario	**RMSE (% change)**	**Control Effort (% change)**	**Stability Margin (% change)**	**Notes**
Ideal Case	Baseline	Baseline	Baseline	-
Unbounded Disturbances	+15%	+30%	-10%	Stable up to 5x assumed bound
Non-smooth Nonlinearities	+8%	+12%	-5%	Limit cycles observed
Measurement Uncertainties				
- Low noise ($\sigma^2 = 0.1$)	+12%	+18%	-8%	-
- High noise ($\sigma^2 > 0.2$)	+25%	+40%	-20%	Occasional instability
Actuator Limitations	+20%	-15%	-25%	Slower response, increased overshoot

6.6. Robustness Analysis

To evaluate the practical applicability of the HAC, we conducted a series of simulations designed to test its performance under conditions that violate the key assumptions made in our theoretical analysis. These simulations provide insights into the controller's robustness and highlight areas for potential improvement in real-world applications.

6.6.1. Simulation Design

We designed some scenarios to test key assumptions of the HAC theory. To test robustness against unbounded disturbances, we added the system's disturbance term with an unbounded component, $d_u(k) = \alpha \cdot k \cdot \sin(\beta k)$, creating a time-dependant disturbance. Non-smooth nonlinearities were introduced through a Coulomb friction model, $F_f = F_c \cdot \text{sign}(v)$, and a backlash model in the actuator, simulating real-world mechanical imperfections. To evaluate the controller's resilience to measurement uncertainties, we injected Gaussian noise into the state measurements, $x_{\text{measured}}(k) = x_{\text{true}}(k) + \varepsilon(k)$, with $\varepsilon(k) \sim \mathcal{N}(0, \sigma^2)$, and varied the noise variance. Finally, we implemented actuator limitations by incorporating both saturation limits, $u_{\text{applied}}(k) = \text{sat}(u(k), u_{\min}, u_{\max})$, and rate limiting, $u_{\text{applied}}(k) = u_{\text{applied}}(k-1) + \text{sat}(u(k) - u_{\text{applied}}(k-1), -\Delta u_{\max}, \Delta u_{\max})$, to emulate realistic constraints on control inputs. This comprehensive set of simulations allowed us to systematically evaluate the HAC's performance under conditions that violate key theoretical assumptions, providing crucial insights into its practical applicability and robustness.

6.6.2. Performance Evaluation

Table 6 summarizes the HAC's resilience across challenging scenarios. The HAC showed resilience to growing disturbances, maintaining stability up to 5 times larger than the assumed bound. There was a 15% increase in RMSE and 30% increase in control effort. Faced with non-smooth nonlinearities like Coulomb friction and backlash, the controller maintained stability with an 8% increase in RMSE, but observed limit cycles. The controller's performance degraded with a 12% RMSE increase for noise variances up to 0.1, and occasional instability for variances above 0.2. Actuator limitations led to a 20% increase in RMSE and a 25% decrease in stability margin, resulting in slower response times and increased overshoot.

An unexpected observation was the controller's ability to adapt to Coulomb friction, effectively learning to overcome static friction barriers. This behavior requires further investigation as a potential advantage in high-friction systems. Some performance degradation was observed, but the HAC remained stable, highlighting its potential for real-world applications.

These findings emphasize the need for comprehensive testing beyond theoretical analysis when developing control systems for practical applications. They also highlight the HAC's adaptability, suggesting its potential usefulness in uncertain and changing environments. Future work could focus on integrating state estimation techniques to improve robustness to measurement uncertainties and incorporating anti-windup mechanisms to handle actuator limitations.

Comment 6 of reviewer 4: There is often a trade-off between interpretability and complexity. While hyperplane-based rules can be interpretable, the omission of antecedent parameters might lead to a loss of flexibility in modeling complex nonlinear relationships. The effectiveness of this trade-off needs thorough validation.

Authors response: Thank you for the important comment. To address it, we have added a new section titled "Trade-off Between Interpretability and Complexity" in the revised manuscript and attached here.

6.5. Trade-off Between Interpretability and Complexity

The HAC offers a new approach to adaptive neuro-fuzzy control, emphasizing interpretability through simplified hyperplane-based rules. This design contrasts with more complex models like the Generic-controller (G-controller [1]), which uses antecedent parameters. Comparing HAC to the G-controller helps illuminate the trade-offs between interpretability and modeling flexibility in neuro-fuzzy control systems.

The G-controller is a sophisticated neuro-fuzzy system that uses multivariate Gaussian functions and sliding mode control for parameter adaptation. It has shown excellent results in controlling bio-inspired drones, making it a valuable benchmark. Comparing HAC to the G-controller highlights the trade-offs between interpretability and flexibility in rule structure for adaptive control of nonlinear systems. HAC's hyperplane-based rules are more interpretable than G-controller's multivariate Gaussian functions. Each HAC rule represents a linear decision boundary in the input space, facilitating human understanding. In contrast, G-controller's functions create complex, nonlinear boundaries that are less intuitive. However, G-controller's flexible membership functions may model complex nonlinear relationships more efficiently, requiring fewer rules. This is because its multivariate Gaussians can capture input variable correlations, while HAC's hyperplanes treat inputs independently.

Both the hyperplane-based approach of HAC and the multivariate Gaussian approach of G-controller are universal approximators. However, they differ in their convergence rates and the number of rules required. The hyperplane-based approach of HAC has an approximation error of $O\left(\frac{1}{\sqrt{m}}\right)$, where m is the number of rules. This offers interpretable linear decision boundaries but may require more rules for highly nonlinear functions. In contrast, the multivariate Gaussian approach of G-controller has an approximation error of $O(\exp(-cm))$, where c is a constant and m is the number of rules. This potentially allows for faster convergence with fewer rules for smooth, nonlinear functions. However, the decision boundaries are less interpretable as they are nonlinear.

Table 4: Performance comparison of G-controller and HAC in tracking various trajectories

Plant	Desired trajectories	RMSE		Performance Improvement (%)
		G-controller	HAC	
Robotic surgical system SISO	$y_{d_1}(k)$	0.0935	**0.0826**	11.66
	$y_{d_2}(k)$	0.4503	**0.4112**	8.68
	$y_{d_3}(k)$	0.2102	**0.1765**	16.03
	$y_{d_4}(k)$	0.1701	**0.1613**	5.17
	$y_{d_5}(k)$	0.1950	**0.1845**	5.38
Spring mass damper SISO	$y_{d_1}(k)$	0.1205	**0.1122**	6.89
	$y_{d_2}(k)$	0.3656	**0.3442**	5.85
	$y_{d_3}(k)$	0.1475	**0.1286**	12.81
	$y_{d_4}(k)$	0.1506	**0.1468**	2.52
	$y_{d_5}(k)$	0.1699	**0.1652**	2.77
Robot arm SISO	$y_{d_1}(k)$	0.0952	**0.0879**	7.67
	$y_{d_2}(k)$	0.2212	**0.2067**	6.56
	$y_{d_3}(k)$	0.0977	**0.0891**	8.80
	$y_{d_4}(k)$	0.1183	**0.1105**	6.59
	$y_{d_5}(k)$	0.1389	**0.1335**	3.89
Nonlinear MIMO	$y_{d_{11}}(k)$	0.2757	**0.2132**	22.67
	$y_{d_{12}}(k)$	0.1165	**0.1143**	1.89
	$y_{d_{21}}(k)$	0.1228	**0.1088**	11.40
	$y_{d_{22}}(k)$	0.0863	**0.0746**	13.56

Our experimental results show that HAC consistently outperforms G-controller across various control tasks, despite G-controller's theoretical flexibility advantage. Table 4 demonstrates that HAC achieves lower RMSE for all tested trajectory types and plant models, with improvements ranging from 1.89% to 22.67% (average 8.99%). The performance gap is most significant in complex scenarios, such as the nonlinear MIMO plant, where HAC's RMSE for trajectory $y_{d11}(k)$ is 22.67% lower than G-controller's (0.2132 vs. 0.2757). This suggests that HAC's online structural adaptation effectively compensates for its simpler rule structure, allowing it to capture complex dynamics more accurately.

HAC's superior performance is consistent across different plant types:

1. Robotic surgical system SISO: 5.17% to 16.03% improvement
2. Spring mass damper SISO: 2.52% to 12.81% improvement
3. Robot arm SISO: 3.89% to 8.80% improvement

HAC's superior performance can be attributed to several factors, despite its simpler rule structure. Its rule evolution mechanism, based on statistical contribution analysis, may be more effective at capturing the essential dynamics of the system. The combination of evolving hyperplane rules with sliding mode control for parameter adaptation creates a robust and adaptive system. The simpler hyperplane structure may help HAC avoid overfitting to noise or transient dynamics, leading to better generalization. Additionally, the clear interpretability of HAC's rules may guide the structural evolution towards more meaningful partitions of the input space.

There's a trade-off between interpretability and complexity, but our results with HAC show that prioritizing interpretability through hyperplane-based rules doesn't sacrifice performance. Enhanced interpretability contributes to effective online adaptation and robust control. This challenges the belief that complex fuzzy structures are always needed for high-performance control of nonlinear systems.

Future work involves a more extensive theoretical analysis of the approximation capabilities and convergence properties of hyperplane-based evolving fuzzy systems compared to multivariate Gaussian functions. Further empirical studies across a wider range of nonlinear control problems would help solidify our understanding of when simpler, interpretable rule structures can outperform complex alternatives. The consistent superior performance of HAC across diverse control tasks suggests that this approach holds significant promise for advancing adaptive neuro-fuzzy control.

Comment 7 of reviewer 2: What is the limitation of the proposed approach?

Authors response: We appreciate the insightful question of the reviewer regarding the limitations of our proposed approach. We acknowledge that while the HAC demonstrates superior performance in various scenarios, it is not without limitations. Its scalability in high-dimensional input spaces is a primary concern, where the number of hyperplane rules might grow exponentially to cover complex decision boundaries. Highly nonlinear, non-axis-aligned decision boundaries may pose challenges for HAC, as multivariate Gaussian functions could be better suited. In real-time applications with fast dynamics, the computational cost for rule evolution might be a bottleneck due to the statistical contribution analysis for rule addition and pruning. HAC may struggle with capturing localized nonlinear effects due to the global linear approximations provided by hyperplane rules. Systems with significant input cross-coupling effects could present difficulties, as independent hyperplanes may not efficiently capture these interactions. Lastly, modeling systems with discontinuities or abrupt changes might be challenging for HAC, where the smooth transitions offered by Gaussian functions could be advantageous. These limitations

are based on theoretical considerations and would require further experimental validation in extreme scenarios to quantify their impact on HAC's performance.

References

[1] M. M. Ferdaus, M. Pratama, S. Anavatti, M. A. Garratt, Y. Pan, Generic evolving self-organizing neuro-fuzzy control of bio-inspired unmanned aerial vehicles, IEEE Transactions on Fuzzy Systems (2019).

Source Files (word or latex)

Click here to view linked References