Development of advanced autonomous learning algorithms for nonlinear system identification and control

Md Meftahul Ferdaus

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy

School of Engineering and Information Technology
University of New South Wales, Australia

UNSW SYDNEY | Australia's Global University

Thesis/Dissertation Sheet

Surname/Family Name	:	**Ferdaus**
Given Name/s	:	**Md Meftahul**
Abbreviation for degree as give in the University calendar	:	**Ph.D.**
Faculty	:	**University of New South Wales Canberra**
School	:	**School of Engineering and Information Technology**
Thesis Title	:	**Development of advanced autonomous learning algorithms for modeling and controlling nonlinear dynamical systems**

Abstract 350 words maximum: (PLEASE TYPE)

Identification or modeling of nonlinear dynamical systems, data stream analysis, etc. are handled by algorithmic development of autonomous learning machines like evolving fuzzy and neuro-fuzzy systems (ENFSs) characterized by the single-pass learning mode and the open-structure property. Such features enable their effective handling of fast and rapidly changing natures of data streams. The underlying bottleneck of ENFSs lies in its design principle, which involves a high number of free parameters (rule premise and rule consequent) to be adapted in the training process. From this literature gap, a novel ENFS, namely Parsimonious Learning Machine (PALM) is proposed in this thesis. To reduce the number of network parameters significantly, PALM features utilization of a new type of fuzzy rule based on the concept of hyperplane clustering where it has no rule premise parameters. It is capable of automatically generating, merging, and tuning the hyperplane-based fuzzy rule in a single-pass manner. The efficacy of PALM has been evaluated through numerical study with data streams and to model nonlinear aerial vehicle system. The proposed models showcase significant improvements in terms of computational complexity and the number of required parameters against several renowned ENFSs while attaining comparable and often better predictive accuracy.

The ENFSs have also been utilized to develop three autonomous intelligent controllers (AICons) in this thesis. All these controllers start operating from scratch and no offline training is required. To cope with the dynamic behavior of the plant, these controllers can add, merge or prune the rules on demand. Among these controllers, in the G-controller, integration of generalized adaptive resonance theory provides a compact structure, which lowers its computational cost. Another AICon namely PAC is rooted with PALM's architecture. The threshold-dependency of PALM is replaced with the concept of bias-variance trade-off in PAC. In the last AICon called RedPAC, the network parameters have further reduced to one parameter per rule. All the controllers' efficacy is evaluated by observing various trajectory tracking performance of unmanned aerial vehicles. The tracking accuracy is comparable or better than the benchmark controllers where the proposed AICons incur significantly fewer parameters to attain similar or better performance.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

INCLUSION OF PUBLICATIONS STATEMENT

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in their thesis in lieu of a Chapter if:

- The student contributed greater than 50% of the content in the publication and is the "primary author", ie. the student was responsible primarily for the planning, execution and preparation of the work for publication
- The student has approval to include the publication in their thesis in lieu of a Chapter from their supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis

Please indicate whether this thesis contains published material or not.

☐ *This thesis contains no publications, either published or submitted for publication (if this box is checked, you may delete all the material on page 2)*

☒ *Some of the work described in this thesis has been published and it has been documented in the relevant Chapters with acknowledgement (if this box is checked, you may delete all the material on page 2)*

☐ *This thesis has publications (either published or submitted for publication) incorporated into it in lieu of a chapter and the details are presented below*

CANDIDATE'S DECLARATION

I declare that:

- I have complied with the Thesis Examination Procedure
- where I have used a publication in lieu of a Chapter, the listed publication(s) below meet(s) the requirements to be included in the thesis.

Name	Signature	Date (dd/mm/yy)

ORIGINALITY STATEMENT

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.’

Signed ……..

Date ……..

COPYRIGHT STATEMENT

Signed …….....

Date …….....

AUTHENTICITY STATEMENT

‘I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.’

Signed …….....

Date …….....

Abstract

Identification of nonlinear dynamical systems, data stream analysis, etc. is usually handled by autonomous learning algorithms like evolving fuzzy and evolving neuro-fuzzy systems (ENFSs). They are characterized by the single-pass learning mode and open structure-property. Such features enable their effective handling of fast and rapidly changing natures of data streams. The underlying bottleneck of ENFSs lies in its design principle, which involves a high number of free parameters (rule premise and rule consequent) to be adapted in the training process. This figure can even double in the case of the type-2 fuzzy system. From this literature gap, a novel ENFS, namely Parsimonious Learning Machine (PALM) is proposed in this thesis.

To reduce the number of network parameters significantly, PALM features utilization of a new type of fuzzy rule based on the concept of hyperplane clustering, where it has no rule premise parameters. PALM is proposed in both type-1 and type-2 fuzzy systems where all of them characterize a fully dynamic rule-based system. Thus, it is capable of automatically generating, merging, and tuning the hyperplane-based fuzzy rule in a single-pass manner. Moreover, an extension of PALM, namely recurrent PALM (rPALM), is proposed and adopts the concept of teacher-forcing mechanism in the deep learning literature. The efficacy of both PALM and rPALM have been evaluated through numerical study with data streams and to identify nonlinear unmanned aerial vehicle system. The proposed models showcase significant improvements in terms of computational complexity and the number of required parameters against several renowned ENFSs while attaining comparable and often better predictive accuracy.

The ENFSs have also been utilized to develop three autonomous intelligent controllers (AICons) in this thesis. They are namely Generic (G) controller, Parsimonious controller (PAC), and Reduced Parsimonious Controller (RedPAC). All these controllers start operating from scratch with an empty set of fuzzy rules,

and no offline training is required. To cope with the dynamic behavior of the plant, these controllers can add, merge or prune the rules on demand. Among three AICons, the G-controller is built by utilizing an advanced incremental learning machine, namely Generic Evolving Neuro-Fuzzy Inference System. The integration of generalized adaptive resonance theory provides a compact structure of the G-controller. Consequently, the faster evolution of structure is witnessed, which lowers its computational cost. Another AICon namely, PAC is rooted with PALM's architecture. Since PALM has a dependency on user-defined thresholds to adapt the structure, these thresholds are replaced with the concept of bias-variance trade-off in PAC. In RedPAC, the network parameters have further reduced in contrast with PALM-based PAC, where the number of consequent parameters has reduced to one parameter per rule.

These AICons work with very minor expert domain knowledge and developed by incorporating the sliding mode control technique. In G-controller and RedPAC, the control law and adaptation laws for the consequent parameters are derived from the SMC algorithm to establish a stable closed-loop system, where the stability of these controllers are guaranteed by using the Lyapunov function and the uniform asymptotic convergence of tracking error to zero is witnessed through the implication of an auxiliary robustifying control term. While using PAC, the boundedness and convergence of the closed-loop control system's tracking error and the controller's consequent parameters are confirmed by utilizing the LaSalle-Yoshizawa theorem. Their efficacy is evaluated by observing various trajectory tracking performance of unmanned aerial vehicles. The accuracy of these controllers is comparable or better than the benchmark controllers where the proposed controllers incur significantly fewer parameters to attain similar or better tracking performance.

Acknowledgments

This thesis would not have been possible without the encouragement and support of a number of individuals. This is the right place to show them my sincere appreciation. First and foremost, I would like to express my gratitude to my supervisor, Dr. Sreenatha G. Anavatti, whose passionate supervision, support, and motivation helped me perform to the best of my abilities. His untiring guidance helped me to get engaged in this attractive research domain and to collaborate with top-ranked laboratory around the globe. His charming personality and friendly behavior assisted me in conducting my research smoothly and successfully.

I am really grateful to my secondary supervisor Dr. Matthew A. Garratt for his notable support during experimentation at the unmanned aerial vehicle laboratory of the University of New South Wales (UNSW), Canberra. I would like to thank my another secondary supervisor Dr. Mahardhika Pratama, for supporting me with vital research ideas and computational support at the computational intelligence laboratory of the Nanyang Technological University (NTU) Singapore.

It is beyond the realm of imagination to utter the role of my benevolent colleagues and friends in UNSW Canberra. Thanks to Ahmad Jobran Al-Mahasneh, Ayad Al-Mahturi, Dr. Fendy Santoso, Dr. Sobers Francis, Sumana Biswas, Deepak Rajamohan, Binod Aryal, Praveen Kumar Muthusamy, Phi Vu, Tanmoy Das, Tanmoy Dam, Wasim Reza, Mohammad Mamun, Md Imran Kabir, Asaduzzaman Khan, Atiqul Islam, Md Mohiuddin Khan, Md Rasel Mahmud, Dr. Md Shohel Ahmed, Ahsanul Habib, Himel, Dr. Forhad Zaman, Dr. Ripon Chakrabortty, Shivang Pathak, Dr. Mehedi Hasan, Dr. Smita Tasneem, and a lot to mention for the cherished times we spent together at UNSW, Canberra. I am grateful to Dr. Habibullah, Abdulla Al Suman, Md Sohrab mahmud, Md Asikuzzaman, and Rashed for their valuable time to support me with the initial

setup in Australia. I have also spent remarkable time with Dr. Qing Cai, Choiru Zain, Andri Ashfahani, Hady, Hao Li, Dr. Zhengkun Wang at NTU Singapore.

I sincerely acknowledge the Ph.D. scholarship from UNSW, namely *Tuition Fee Scholarship (TFS) plus a Research Stipend*. I am thankful to the Research Office, especially to Elvira Berra, Craig I. Edwards, Kristin Newnham, for their continuous support and advice throughout my candidature.

Above all, I would like to thank my parents for encouraging me and praying for my success all the time. No words are adequate to thank my dear wife Mst Nafisa Tamanna Shanta for scarifying her time with me and tolerating me since I was not able to spend enough time with her and my only three years old son Ehsan Maahir. I am especially thankful to her for taking proper care of my son in my absence. Enormous supports and inspiration from my family help me to accomplish this endeavor.

List of Publications

Journal Papers

[1] **Md Meftahul Ferdaus**, Mahardhika Pratama, Sreenatha Anavatti, Matthew A Garratt, and Yongping Pan. Generic evolving self-organizing neuro-fuzzy control of bio-inspired unmanned aerial vehicles. *IEEE Transactions on Fuzzy Systems*, 2019 (DOI: 10.1109/TFUZZ.2019.2917808).

[2] **Md Meftahul Ferdaus**, Mahardhika Pratama, Sreenatha Anavatti, and Matthew A Garratt. PALM: An incremental construction of hyperplanes for data stream regression. *IEEE Transactions on Fuzzy Systems*, 2019 (DOI: 10.1109/TFUZZ.2019.2893565).

[3] **Md Meftahul Ferdaus**, Mahardhika Pratama, Sreenatha G Anavatti, and Matthew A Garratt. Online identification of a rotary wing unmanned aerial vehicle from data streams. *Applied Soft Computing*, 76:313–325, 2019.

[4] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Mahardhika Pratama, and Matthew A Garratt. Towards the use of fuzzy logic systems in rotary wing unmanned aerial vehicle: a review. *Artificial Intelligence Review*, pages 1–34, 2018.

[5] **Md Meftahul Ferdaus**, Mahardhika Pratama, Sreenatha G Anavatti, Matthew A Garratt, and Edwin Lughofer. PAC: A novel self-adaptive neuro-fuzzy controller for micro aerial vehicles. *Information Sciences*, 2019.

[6] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Matthew A Garratt, and Mahardhika Pratama. Development of c-means clustering based adaptive fuzzy controller for a flapping wing micro air vehicle. *Journal of Artificial Intelligence and Soft Computing Research*, 9(2):99–109, 2019.

Conference Papers

[1] **Md Meftahul Ferdaus**, Mohamad Abdul Hady, Mahardhika Pratama, Harikumar Kandath, and Sreenatha G. Anavatti. RedPAC: A Simple Evolving Neuro-Fuzzy-based Intelligent Control Framework for Autonomous Aerial Vehicles. In *2019 IEEE International Conference on Fuzzy Systems*. IEEE, 2019.

[2] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Jobran Al-Mahasneh Ahmad, Mahardhika Pratama, and Matthew A Garratt. Development of hyperplane-based adaptive t-s fuzzy controller for micro aerial robots. In *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 50–56, July 2019.

[3] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Mahardhika Pratama, and Matthew A Garratt. A novel self-organizing neuro-fuzzy based intelligent control system for a ar. drone quadcopter. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2026–2032. IEEE, 2018.

[4] **Md Meftahul Ferdaus**, Mahardhika Pratama, Sreenatha G Anavatti, and Matthew A Garratt. A generic self-evolving neuro-fuzzy controller based high-performance hexacopter altitude control system. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2784–2791, Oct 2018.

[5] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Matthew A Garratt, and Mahardhika Pratama. Evolving fuzzy inference system based online identification and control of a quadcopter unmanned aerial vehicle. In *2017 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, pages 223–228. IEEE, 2017.

[6] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Matthew A Garratt, and Mahardhika Pratama. Fuzzy clustering based modelling and adaptive controlling of a flapping wing micro air vehicle. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2017.

[7] **Md Meftahul Ferdaus**, Mahardhika Pratama, Sreenatha G Anavatti, and Matthew A Garratt. Evolving neuro-fuzzy system based online identification of a bio-inspired flapping wing micro aerial vehicle. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[8] **Md Meftahul Ferdaus**, Sreenatha G Anavatti, Matthew A Garratt, and Mahardhika Pratama. Fuzzy clustering based nonlinear system identification and controller development of pixhawk based quadcopter. In *2017 Ninth*

International Conference on Advanced Computational Intelligence (ICACI), pages 223–230. IEEE, 2017.

[9] Jobran Al-Mahasneh Ahmad, Sreenatha G Anavatti, **Md Meftahul Ferdaus**, and Matthew A Garratt. Adaptive Neural Altitude Control and Attitude Stabilization of a Hexacopter with Uncertain Dynamics. In *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology*, 2019.

[10] Choiru Zain, Mahardhika Pratama, Edwin Lughofer, **Md Meftahul Ferdaus**, Qing Cai, and Mukesh Prasad. Big Data Analytics based on PANFIS MapReduce. *Procedia Computer Science*, 144:140–152, 2018.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AICon	Autonomous Intelligent Controller
ARC	Auxiliary Robustifying Control
BI-FWMAV	Bio-Inspired Flapping Wing Micro Air Vehicle
CoA	Centroid of Area
CoG	Centre of Gravity
CoS	Centre of Sum
DCBC	Data Cloud-Based Clustering
DCM	Direction Cosine Matrix
DS	Datum Significance
EA	Evolutionary Algorithm
EFS	Evolving Fuzzy System
EIS	Evolving Intelligent System
EKF	Extended Kalman Filter
eMG	evolving Multi-variable Gaussian
ENFS	Evolving Neuro-Fuzzy System
ERS	Extended Rule Significance
eTS	evolving Takagi Sugeno
FBL	Feedback Linearization
FCM	Fuzzy C-Means
FCRM	Fuzzy C-Regression Model
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
FLS	Fuzzy Logic System

FNN	Fuzzy Neural Network
FOU	Footprint of Uncertainty
FPT	First Principle Technique
FS	Fuzzy System
FWGRLS	Fuzzily Weighted Generalized Recursive Least Square
GA	Genetic Algorithm
GART	Generalized Adaptive Resonance Theory
GENEFIS	Generic Evolving Neuro-Fuzzy Inference System
GMM	Gaussian Mixture Model
GRNN	Generalized Regression Neural Network
GT2RS	Generalized Type-2 Rule Significance
HEBC	Hyper-Ellipsoid-Based Clustering
HPBC	Hyper-Plane-Based Clustering
HSBC	Hyper-Sphere-Based Clustering
KM	karnik Mendel
LQ	Linear Quadratic
MAV	Micro Aerial Vehicle
MCI	Maximal Information Compression Index
McLM	Metacognitive Learning Machine
McSIT2RFNN	Metacognitive Scaffolding Interval Type 2 Recurrent Fuzzy Neural Network
MF	Membership Function
MIMO	Multi-Input-Multi-Output
MISO	Multi-Input-Single-Output
MoM	Mean of Maximum
MSE	Mean Squared Error
NARMAX	Nonlinear Autoregressive Moving Average Model with Exogenous
NDEI	Non-Dimensional Error Index
NFS	Neuro-Fuzzy System

NN	Neural Network
NS	Network Significance
PAC	Parsimonious Controller
PALM	Parsimonious Learning Machine
PID	Proportional Integral Derivative
RBFNN	Radial Basis Function Neural Network
RedPAC	Reduced Parsimonious Controller
RLS	Recursive Least Square
RMSE	Root Mean Squared Error
ROS	Robot Operating System
RT	Rise Time
SANFS	Self-Adaptive Neuro-Fuzzy System
SISO	Single-Input-Single-Output
SMBC	Sequential Markov Blanket Criterion
SMC	Sliding Mode Control
SSC	Self-Constructive Clustering
ST	settling time
T2GC	Type-2 Geometric Criteria
T2RMI	Type-2 Relative Mutual Information
TS	Takagi-Sugeno
TSK	Takasi-Sugeno-Kang
UAV	Unmanned Aerial Vehicle
ZEDM	Zero-order Density Maximization

Chapter 1

Introduction

1.1 Background and Motivation

Many real-world systems are inherently complex and highly nonlinear. identification of complicated nonlinear dynamical systems is challenging using linear systems theory and first principle techniques. Being capable of learning complex nonlinear relationships, Neural Networks (NNs), Fuzzy Logic Systems (FLSs), and their combination namely Neuro-Fuzzy Systems (NFSs), etc. are used to identify complex nonlinear systems [1]. Conventional NNs, NFSs, or FLSs have a predefined fixed structure. Though they can learn a system's nonlinearity by adapting their learning parameters, with a static structure, they face constraints in dealing with rapid changes in the system's dynamics. Another complex problem, like the identification of nonlinear dynamical systems, is handling the data streams.

In recent times, due to the progression in both hardware and software technologies, countless applications produce a massive amount of data in an automated way. These data are generated sequentially at a rapid rate under complex

environments. Besides, they are massive and possibly unbounded. Such online data are known as data streams. In the field of data stream mining, a complete dataset is not at hand to embark the training of a learning algorithm since the data arrive in a sequential manner. However, the classical batched-learning algorithms require the whole data to train them. It makes them infeasible to deal with data streams. Besides, the batched-learning algorithms need to be retrained by using the up-to-date training data whenever a new knowledge is observed. This retaining process is computationally expensive, which makes them impractical to employ in real-time scenarios. Due to the retraining phase, batched-learning algorithms also suffer from *catastrophic interference*, i.e., they may absolutely and rapidly forget previously learned knowledge when learning new knowledge. Some other challenges to handle data streams for the learning algorithms can be expressed as follows: 1) unbounded size of the data streams; 2) huge amount of data; 3) unknown distribution of incoming data, which may alter at different rates like slowly, rapidly, abruptly, gradually, locally, globally, cyclically, etc. over time. Such variations in the data distribution of data streams over time are known as *concept drift*; 4) data are discarded after being processed to maintain an economical and bounded memory demand.

To develop a learning algorithm to cope with the challenges in data stream mining and identification of nonlinear dynamical systems, it should expose the following desired characteristics: 1) ability to work in single-pass mode; 2) dealing with the concept drifts in streaming data; 3) handling sudden changes in complex systems' dynamics; 4) should not be computationally complex and maintain a low memory demand to deploy it in real-time under resource-constrained environment. Such features are exposed by incremental learning algorithms in the domain of online machine learning. In the realm of FLS, such learning aptitude

is demonstrated by evolving fuzzy or neuro-fuzzy systems [2]. The self-adaptive nature of the evolving neuro-fuzzy systems also inspires researchers to develop model-free Autonomous Intelligent Controllers (AICons).

FLS and NN-based intelligent controllers have been used successfully in many autonomous and nonlinear dynamical systems. There are numerous ways to develop these intelligent controllers [3]. When the mathematical model of the autonomous system to be controlled is known, it can be utilized to train those controllers in offline mode with input-output data [4]. It yields a static-structured intelligent controller with a fixed number of rules or neurons. When the system dynamics is known, another approach is to use an expert's knowledge to build the rules of the controller. In many complex autonomous systems, their mathematical model is unknown. To overcome the intelligent controllers' dependency on the mathematical model or the expert's knowledge, intelligent controllers are advanced as intelligent adaptive controllers by combining with conventional controllers [5]. Being adaptive, these fixed-structured controllers adapt their parameters to attain desired control signals. In these adaptive intelligent controllers, selecting the required number of rules or neurons beforehand is difficult. With a small number of rules or neurons, achieving the desired control accuracy may become impractical. On the other hand, a higher number of neurons or rules create a complex control structure to implement in real-time. To mitigate the limitation with predefined structure, these intelligent controllers structure need to be evolved by appending or pruning rules or neurons. Such property will make them a fully autonomous intelligent controller. Their ability to change their structure online is an expected feature in some challenging control applications such as control of Unmanned Aerial Vehicles (UAVs) in the presence of various environmental uncertainties like gust, motor degradation, etc. These desiring

features are the motivation to develop some novel evolving and self-adaptive neuro-fuzzy systems in this thesis.

1.2 Scope of Research

In this thesis, limitations of the existing evolving neuro-fuzzy systems in identifying nonlinear dynamical systems and analyzing data streams are highlighted. Evolving neuro-fuzzy systems have a high number of network parameters in both antecedent and consequent part. These parameters need to be tuned online whether they are used for data stream regression or to identify complex nonlinear systems like unmanned helicopter [6–8], or similar unmanned aircraft [9–11]. The number of parameters becomes double in the case of type-2 fuzzy systems. A higher number of parameters increases the computational complexity and requires high memory demand.

In this thesis, we also have sorted out the limitations of the existing control techniques of UAVs and the feasibility of applying evolving neuro-fuzzy systems as control tools. Conventionally, UAVs are regulated by First Principle Technique (FPT)-based controllers. Among variety of UAVs, linear control methods like Proportional Integral Derivative (PID), Linear Quadratic (LQ) methods are employed commonly in UAVs. Though the employment of these simple controllers is easy, their performance degrades sharply in dealing with environmental uncertainties due to UAV's inherent non-linearity and coupled dynamics. To overcome such limitations, FPT-based nonlinear control techniques perform better than their linear counterparts. However, a common shortcoming of all the FPT-based controllers, whether linear or nonlinear, is their dependency on the precise dynamics of the plant to be controlled. An alternative to the FPT-based

control methods is model-free control techniques. The traditional model-free control techniques like FLS or NFS are developed using experts' knowledge or require off-line training before employing in control applications. Besides, they have a fixed structure with a fixed number of rules. To overcome these limitations and to improve their control efficiency, researchers have tried to make them adaptive, where FPT-based nonlinear control methods like Sliding Mode Control (SMC), backstepping, H-∞ control method, etc. are utilized to tune the parameters of the FLS, NFS-based model-free controllers. Nevertheless, these controllers have a fixed structure, which impedes them to handle sharp changes in plant dynamics.

To mitigate the above-mentioned limitations, recently, researchers have proposed flexible structure-based autonomous controllers using evolving fuzzy or neuro-fuzzy systems. These evolving controllers have been validated for complex situation like aircraft with failures and undergoing nonlinear maneuvers [12–14]. Though these model-free autonomous controllers perform better than adaptive variants, a challenge in their real-time deployment is the requirement of tuning a high number of network parameters online with limited memory resources, especially while used in UAVs. From these research gaps, in this thesis, autonomous controllers have been developed with fewer network parameters to control UAVs.

1.3 Contributions of the Thesis

The contributions of this thesis can be summarized as follows:

1. The first contribution of the thesis is in developing a novel autonomous learning algorithm using a new evolving neuro-fuzzy structure, which is named as Parsimonious Learning Machine (PALM). In PALM, a new type

of fuzzy membership function based on the concept of hyperplane clustering is utilized, which significantly reduces the number of network parameters because it has no rule premise parameters. PALM is proposed in both type-1 and type-2 fuzzy systems. PALM characterizes a fully dynamic rule-based system, i.e., they can automatically generate, merge, and tune the hyperplane-based fuzzy rules in a single pass manner. The efficacy of PALMs have been evaluated by considering the computational complexity and the number of required parameters against several renowned evolving neuro-fuzzy systems. PALMs have been implemented for online identification of unmanned helicopter and quadcopter from real-world experimental data streams.

2. Usually, true outputs of an incremental learning algorithm are not known in the deployment mode. To circumvent PALM's dependency on the true output in the deployment phase, the so-called *Teacher Forcing* mechanism is employed in PALM. By following a similar approach like teacher forcing technique, the output of PALM is connected with the input layer of PALM at the next step. It forms a recurrent structure of the PALM (rPALM), which is the second contribution of this thesis.

3. The third contribution is in the area of AICon, where an evolving neuro-fuzzy system, namely PALM is used to develop an AICon called Parsimonious Controller (PAC). Because of using PALM, it features fewer network parameters. To remove the reliance of PAC on user-defined thresholds to adapt the structure, the bias-variance concept-based simplified method namely network significance is proposed in PAC. PAC adapts the consequent parameters with SMC theory in the single-pass fashion. The bound-

edness and convergence of the closed-loop control system's tracking error and the controller's consequent parameters are confirmed by utilizing the LaSalle-Yoshizawa theorem. Lastly, the controller's efficacy is evaluated by observing various trajectory tracking performances from a Bio-Inspired Flapping Wing Micro Aerial Vehicle (BI- FWMAV) and a rotary-wing UAV called hexacopter. This controller has also validated for aerial vehicles' nonlinear maneuvers.

4. The fourth contribution of this thesis is also in the area of AICon. PAC has been simplified as a new AICon, namely Reduced Parsimonious Controller (RedPAC). In contrast with PAC, the number of consequent parameters has further reduced to one parameter per rule in RedPAC. The SMC technique is utilized to adapt consequent parameters of RedPAC, where the SMC-based auxiliary robustifying control term has guaranteed the uniform asymptotic convergence of tracking error to zero. The proposed controller's performance has been evaluated by implementing it to control a quadcopter UAV simulator namely Dronekit.

5. Development of a multivariate Gaussian function based AICon namely, Generic controller (G-controller) is the fifth contribution in this thesis. G-controller is rooted with an incremental learning machine namely Generic Evolving Neuro-Fuzzy Inference System (GENEFIS) [1]. Control law and adaptation laws for the consequent parameters are derived from the SMC algorithm to establish a stable closed-loop system, where the stability of the G-controller is guaranteed by using the Lyapunov function.

6. Besides controlling the UAVs, online modeling of the quadcopter from experimental data streams using an autonomous neuro-fuzzy system is the

sixth contribution in this thesis. For obtaining a better predictive accuracy, Metacognitive Scaffolding Interval Type 2 Recurrent Fuzzy Neural Network (McSIT2RFNN) is utilized to model the UAV. The metacognitive concept enables the what-to-learn, how-to-learn, and when-to-learn scheme, and the scaffolding theory realizes a plug-and-play property which strengthens the online working principle of McSIT2RFNN.

1.4 Organization of the Thesis

This thesis is partitioned into seven chapters. After this introduction, Chapter 2 presents the basis of fuzzy systems, and groundwork for the research, Chapter 3-6 describes the main technical contributions and numerical experimentation results. Finally, the thesis ends with concluding remarks and future research directions in Chapter 7. Particular issues that are addressed in each chapter are outlined as follows:

1. In Chapter 2, a brief introduction to fuzzy logic system is provided. It describes the architecture of both type-1 and type-2 fuzzy system by including some popular fuzzy models, namely Takasi-Sugeno-Kang (TSK) and Mamdani. Different ways of combining fuzzy systems with neural networks and state of the art in evolving fuzzy systems are also described here. Literature regarding the challenges of using evolving fuzzy or neuro-fuzzy-based autonomous learning algorithms in data stream analysis, in modeling and controlling nonlinear dynamical systems, especially UAV systems with possible solutions, are discussed. The details in each of these contributions are presented one by one in further chapters.

2. In Chapter 3, evolving neuro-fuzzy system (ENFS) based novel autonomous

learning algorithm, namely PALM, is proposed. It utilizes a new type of fuzzy membership function based on the concept of hyperplane clustering. In PALM, the number of network parameters has reduced significantly because it has no rule premise parameters. PALM is proposed in both type-1 and type-2 fuzzy structures where all of them characterize a fully dynamic rule-based system. It is capable of automatically generating, merging, and tuning the hyperplane-based fuzzy rule in a single-pass manner. Moreover, an extension of PALM, namely recurrent PALM (rPALM), is proposed by adopting the concept of teacher-forcing mechanism in the deep learning literature.

3. In Chapter 4, evolving neuro-fuzzy-based two AICons namely, Parsimonious Controller (PAC) and Reduced Parsimonious Controller (RedPAC), are proposed. Both of them feature fewer network parameters than conventional approaches due to the absence of rule premise parameters. In contrast with PAC, the number of consequent parameters has further reduced to one parameter per rule in RedPAC. Though both PAC and RedPAC are built upon PALM, their rule growing and pruning modules are derived from the concept of bias and variance. It removes the controllers' reliance on user-defined thresholds, thereby increasing their autonomy for real-time deployment.

4. In Chapter 5, another AICon, namely Generic-controller (G-controller), is proposed. It is developed by incorporating the SMC theory with an advanced incremental learning machine, namely Generic Evolving Neuro-Fuzzy Inference System (GENEFIS). The controller starts operating from scratch with an empty set of fuzzy rule, and therefore, no offline training is required.

To cope with changing dynamic characteristics of the plant, the controller can add or prune the rules on demand. Control law and adaptation laws for the consequent parameters are derived from the SMC algorithm to establish a stable closed-loop system, where the stability of the G-controller is guaranteed by using the Lyapunov function. Due to the integration of Generalized Adaptive Resonance Theory+(GART+), multivariate Gaussian function, and SMC learning theory-based adaptation laws, the self-evolving mechanism of the G-controller is fast with a lower computational cost.

5. In Chapter 6, a quadrotor UAV has been modeled online using real-time experimental flight data streams based on an autonomous learning algorithm, namely Metacognitive Scaffolding Interval Type 2 Recurrent Fuzzy Neural Network (McSIT2RFNN). The metacognitive concept enables the what-to-learn, how-to-learn, and when-to-learn scheme, and the scaffolding theory realizes a plug-and-play property which strengthens the online working principle of the proposed evolving intelligent system.

6. In Chapter 7, a summary of the findings of all the technical contributions is presented. Besides, future issues and directions which could be pursued with the aim of making the autonomous learning algorithms more efficient for handling data streams, modeling and controlling real-world nonlinear dynamical systems.

Chapter 2

Literature Review

The survey furnished in this chapter has been published in the following articles:

- **Ferdaus, M. M.**, Anavatti, S. G., Pratama, M., Garratt, M. A. (2018). Towards the use of fuzzy logic systems in rotary wing unmanned aerial vehicle: a review. *Artificial Intelligence Review,* 1-34.
- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A. (2019). PALM: An Incremental Construction of Hyperplanes for Data Stream Regression. *IEEE Transactions on Fuzzy Systems* (DOI: 10.1109/TFUZZ.2019.2893565).
- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A., Pan, Y. (2019). Generic evolving self-organizing neuro-fuzzy control of bio-inspired unmanned aerial vehicles. *IEEE Transactions on Fuzzy Systems* DOI: 10.1109/TFUZZ.2019.2917808.
- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A. "Online Identification of a Rotary Wing Unmanned Aerial Vehicle from Data Streams." *Applied Soft Computing* 76 (2019): 313-325.

Abstract

The necessities of multi-valued logic in real-world applications with the inclusion of uncertainties traverse to the concept of fuzzy logic. Fuzzy logic system-based modeling has been proved to be useful for complex nonlinear systems. To get a broad overview of the fuzzy system, this chapter describes the basics of the fuzzy system. This chapter also covers the state of the art of various fuzzy and neuro-fuzzy systems from the static to evolving one, applications of evolving systems with existing limitations, and possible opportunities. Fuzzy logic systems have also been successfully used in variety of control applications. Fuzzy-based control approaches for Micro Aerial Vehicles (MAVs) are reviewed in this chapter too. There exists an increasing demand for a flexible and computationally efficient controller for MAVs due to a high degree of environmental perturbations. However, MAVs are mostly regulated by FPT-based controllers. Limitations of these FPT-based controllers, and possible solutions by implementing fuzzy logic, neuro-fuzzy-based model-free adaptive and evolving controllers are discussed in this chapter.

2.1 Introduction

With the current technological advancements, the autonomy of numerous systems has raised significantly. Consequently, they are becoming complicated and highly nonlinear dynamical systems. To model such a system by mathematical equations, or linear system theories is difficult. Inspired by the ability of fuzzy logic systems, Neural networks, or Neuro-fuzzy systems to learn complex nonlinear relationships, they have been utilized to model or identify nonlinear dynamical systems. For instance, the identification of nonlinear helicopter dynamics from flight

data using Nonlinear Auto Regressive eXogenous input (NARX) model, neural network with internal memory known as Memory Neuron Networks (MNN), and Recurrent MultiLayer Perceptron (RMLP) networks was accomplished in [8]. The identification of a quadcopter unmanned aerial vehicle from experimental data streams has been observed using a neuro-fuzzy system in [15] and using fuzzy system in [16]. To improve the performance of the fuzzy systems, neuro-fuzzy systems, or neural networks in identifying nonlinear dynamical systems, researchers have attempted to make their structure autonomous and developed fuzzy system, neuro-fuzzy system, or neural network-based autonomous learning algorithms. On the other hand, to deal with significant uncertainties in the autonomous systems and the environment, and to compensate the systems' failure with no human or external interactions, the idea of fuzzy logic system-based Autonomous Intelligent Controllers (AICons) is introduced in [17, 18].

AICons should have the ability to self-govern the desired control functions to handle the system's parameter variation, gross fundamental and environmental changes without any external interventions. To achieve autonomy in a designed control method, [17] have proposed an amalgamation of the mathematical model-based conventional controller and model-free decision-making symbolic method-based intelligent controller. Autonomous controllers should have the ability to deliver a high degree of endurance to failures. To confirm the reliability of an autonomous system, autonomous controllers must detect and isolate faults by designing new control laws if necessary. They must be able to plan for the sequence of control laws to complete a complicated task. They must have the ability to interact with other systems and with the operator as well.

If we consider implementing the AICons in advanced applications like space vehicular system, AICons can help to increase the vehicle's autonomy by replacing

the tasks currently accomplished by pilots, crews, or ground stations. Some conventional AICons, namely Fuzzy Logic Controller (FLC), knowledge-based controller, and Neural Network (NN)-based controller, have been implemented already in a variety of control applications. For instance, in damping the vibration of flexible robot-arms [19], complex process control problems [20], the FLC has been implemented successfully. In managing and coordinating various activities of autonomous systems like robots, knowledge-based controllers have been implemented [21]. NN-based controllers are employed successfully to control highly nonlinear dynamical systems [22]. Though the conventional AICons have certain successful implementations, in some cases, they have exaggerated the claims. For example, some researchers have stated that conventional control techniques can not handle nonlinear dynamical systems and system uncertainties. However, these so-called conventional controllers namely Proportional Integral Derivative (PID), state-space controller, frequency domain techniques, adaptive and robust control methods, Kalman filters, Lyapunov methods, etc. have been implemented in numerous complicated nonlinear plants [23]. From the successful implementation of conventional controllers in a variety of autonomous systems, it is clearly indicating their adaptability in automated systems. Nevertheless, achievement of highly autonomous feature can be easier in AICons. Before discussing the recent advancements in fuzzy logic systems and FLCs, the basics of the fuzzy logic system has been presented in the next section.

2.2 Fundamentals of Fuzzy Logic System

In the 19th century, uncertainties were not considered in control system application [24]. However, at the end of the 19th century, researchers realized the exis-

tence of a certain amount of uncertainty in every physical system [25]. Modeling of systems is inadequate without considering the uncertainty. In the early stage, the quantification of uncertainties was accomplished by probability theory [26]. The probability theory based technique was challenged by [27], where he proposed vagueness, which has the capability of explaining some definite uncertainties. The proposed technique in [27] can provide an accurate mathematical model using membership curve. On the other hand, among the logicians, the principle of bivalence and compositionality were popular assumptions [28]. In these assumptions, each phenomenon was considered as either true or false and presented by numerical denotation of 1 and 0. However, many physical phenomenon are not possible to explain by two-valued logic. From these shortcomings, the idea of multivalued logic was introduced. Due to the research of the Polish school of logic led by Lukasiewicz J [29], the multi-valued logic gained researchers' attention and huge development occurred in between 1930 and 1940 [30, 31]. The necessities of multivalued logic in real-world applications with the inclusion of uncertainties traverse to the concept of fuzzy logic [32]. Though Zadeh has invented the fuzzy logic system in 1965, it attracted researchers after its application in the automation of the steam engine by [33] and [34]. Afterward, the use of fuzzy logic has increased very rapidly in various automation systems.

The fuzzy logic system replaces the classical bi-valued (either 1 or 0) variable to a linguistic variable and can be expressed as "big negative", "close to zero", "big positive", etc. [35]. It provides fuzzy systems the capability of imitating the human expression. The flow of work in a fuzzy system are divided into three steps namely, fuzzification, the Fuzzy Inference System (FIS) and the defuzzification respectively. In the fuzzification process, the classical or crisp data are transformed into fuzzy data using membership function, and

the degree of confidence is obtained. The fuzzy domain confidence degree is amalgamated with one of the input features to yield a rule firing strength. The fuzzy rules have an IF-THEN structure, where the first part associated with IF is known as antecedent and remaining part with THEN is called as consequent. Finally, the rule firing strength is combined with the defuzzification interface to produce the final output. In the defuzzification step, various techniques like Centre of Gravity (CoG), Centroid of Area (CoA), maximum membership method, weighted average, Centre of Sums (CoS), Mean of Maximum (MoM), etc., are utilized to convert the fuzzy element to classical or crisp data [34, 36]. The selection of proper defuzzification technique carries significance to generate the desired output. Nevertheless, there is no specific rule for picking the proper defuzzification technique [35]; it depends on empirical knowledge and types of application. To get a clearer overview of the fuzzy system, the block diagram of a FIS is presented in Figure 2.1.

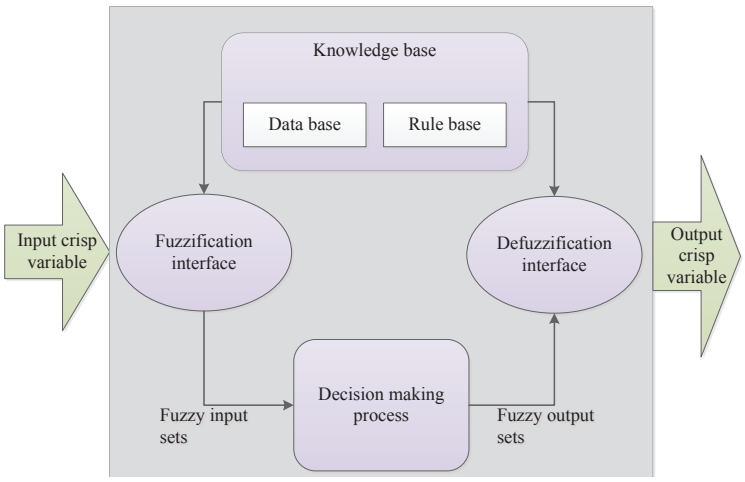


Figure 2.1: Fuzzy inference system (type 1)

Usually, two different fuzzy models are widely used in various industrial automation applications. These models are namely Mamdani [34], and Takasi-Sugeno-Kang (TSK) [37]. Mamdani fuzzy model offers a completely human-like linguistic

rule. Therefore the decision-making process delivered by Mamdani model is more interpretable than other popular models. Based on the type of membership functions, fuzzy systems can be further classified into type-1 and type-2 systems, which are described in the following subsection 2.2.1 and 2.2.2:

2.2.1 Type-1 fuzzy system

In a type-1 (T1) fuzzy system, for a certain value of the input (X), a precise value of membership grade μ_x is obtained. Such a phenomenon is pictured in Figure 2.2, where for a particular value of 2.7 a fixed membership grade of 0.5 is obtained in case of a Gaussian membership function.

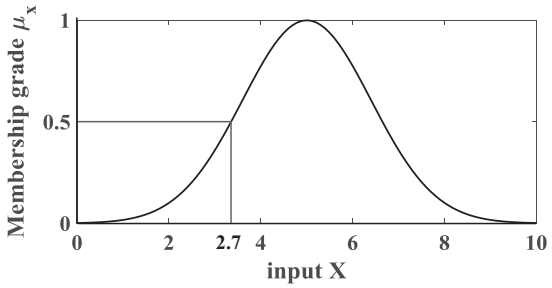


Figure 2.2: Gaussian membership function (type 1 fuzzy system)

A typical rule of the T1 Mamdani fuzzy model is described as follows:

$$\begin{aligned} R_i: \text{ If } x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \text{ and ... and } x_j \text{ is } A_j^i \\ \text{Then } y_1 \text{ is } G_1^i \text{ and } y_2 \text{ is } G_2^i \text{ and ... and } y_m \text{ is } G_m^i \end{aligned} \tag{2.1}$$

where A_j^i denotes the fuzzy set of the *i-th* rule and *j-th* input attribute, whereas G_m^i labels the *i-th* rule and *m-th* output variable. Conversely, $X \in \Re^{1\times j}$ and $Y \in \Re^{1\times m}$ characterize the input and output vectors of interest correspondingly.

By utilizing the COA defuzzification method, the output of the Mamdani

fuzzy model can be illustrated mathematically as follows:

$$y = \frac{\sum_{i=1}^{u} \mu(w_i) \times w_i}{\sum_{i=1}^{u} \mu(w_i)} \tag{2.2}$$

where u is the number of the quantization levels of the output, whereas w_i can be elicited by taking a maximum operation of the output fuzzy set.

Unlike the Mamdani fuzzy rule-based system, in the TSK fuzzy model, the consequent of the rules are not linguistic terms. Rather they are identified as any non-linear or linear continuous functions of the inputs. Therefore in TSK fuzzy model, a complex nonlinear system can be represented by continuous linear equations. A typical rule of the TSK fuzzy model can be expressed as follows:

$$R_i: \text{ If } x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \text{ and ... and } x_j \text{ is } A_j^i \; Then \; y^i = x_e \Omega_i \tag{2.3}$$

where $x_e = [1, x_1, x_2, ..., x_j] \in \Re^{1\times(j+1)}$ is an extended input vector to include the intercept of the consequent hyper-planes with the number of input dimensions j, Ω_i is a weight vector. This weight vector is possible to form as Multi-Input-Single-Output (MISO) or Multi-Input-Multi-Output (MIMO) structure. The MISO structure can be presented as follows:

$$m\Omega_i = [w_{i0}, w_{i1}, ..., w_{ic}]^T \in \Re^{(c+1)\times 1} \tag{2.4}$$

And the MIMO structure can be presented as

$$\Omega_i = \begin{bmatrix} w_{i0}^1, & w_{i0}^2, \ldots & w_{i0}^o, & \ldots & w_{i0}^m \\ w_{i1}^1, & w_{i1}^2, \ldots & w_{i1}^o, & \ldots & w_{i1}^m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ w_{ic}^1, & w_{ic}^2, \ldots & w_{ic}^o, & \ldots & w_{ic}^m \end{bmatrix} \tag{2.5}$$

where m is the number of target outputs. The output of the TSK fuzzy model can be expressed mathematically as follows:

$$y = \frac{\sum_{i=1}^{u} y^i \times w_i}{\sum_{i=1}^{u} w_i} \tag{2.6}$$

where w_i denotes the firing strength of ith rule. The next subsection describes the type-2 fuzzy system.

2.2.2 Type-2 fuzzy system

A new type of fuzzy set called type-2 fuzzy set (T2 FS) was proposed by Zadeh [38]. The precise membership values of T1 FS is replaced in T2 FS with a fuzzy set between 0 and 1. The MFs of T2 FS contains a footprint of uncertainty (FOU) as presented in Figure 2.3 (a). This FOU of the MFs helps the T2 FS to handle the uncertainties.

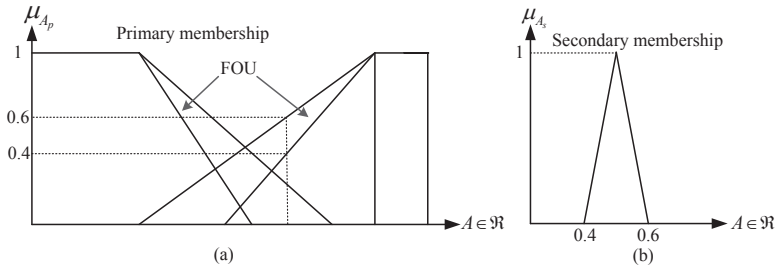


Figure 2.3: Footprint of uncertainty in type 2 fuzzy system

From Figure 2.3(a), it is evident that the membership grade (μ_{A_p}) for a certain value of A is not a crisp value. Instead, it is a function with a value ranging from 0.4 to 0.6 in the domain of primary membership. The function can be presented as a triangular secondary MF, where the weights vary from 0.4 to 0.6, as displayed in Figure 2.3(b). Stronger weighting indicates the middle value and strength of the weight decreases as it moves away from the middle value. Equal weighting is also possible, and then the T2 FS is known as Interval T2 FS (IT2 FS).

Each FOU of a typical T2 FS namely $\widetilde{A}$ has an upper MF (UMF) and a lower MF (LMF), which are presented in Figure 2.4(a). In other words, the UMF and LMF are two T1 MFs that bound the FOU. The UMF is related to the upper bound of FOU and can be presented by $\overline{\mu}_A(w)$ as follows [39]:

$$\overline{\mu}_A(w) = \sup\{g|g \in [0,1], \mu_{\widetilde{A}} > 0\} \quad \forall w \in \Re \tag{2.7}$$

Similarly, the LMF is associated with the lower bound of FOU and expressed by $\underline{\mu}_A(w)$ as follows:

$$\underline{\mu}_A(w) = \inf\{g|g \in [0,1], \mu_{\widetilde{A}} > 0\} \quad \forall w \in \Re \tag{2.8}$$

In Equation (2.7) and Equation (2.8), g is the secondary variable, and has a domain $g = [0,1]$ at each $w \in \Re$. The LMF of a FOU for the IT2 FS is presented in Figure 2.4(b), where the dashed lines within the FOU domain are T1 FSs. This phenomenon indicates the capability of IT2 FS to aggregate several T1 FSs.

The FIS of type-2 fuzzy system is presented in Figure 2.5. T2 FIS has fuzzification interface, knowledge base, and defuzzification interface like the T1 FIS. In addition, unlike the T1 variant, the T2 FIS includes an extra type reducer.

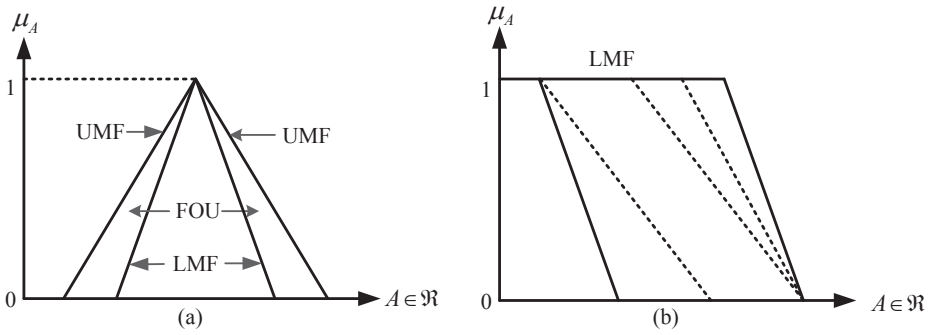


Figure 2.4: (a) LMF and UMF of FOU in T2 FS, (b) Three embedded T1 FSs in the LMF of T2 FSs FOU

Till now, the majority of type reducers are developed based on Karnik-Mendel (KM) algorithms, as explained in [40]. This type reducer projects the T2 fuzzy output sets into T1 fuzzy output sets. Like the T1 FS, T2 FS also has two commonly used architectures or models, and they are T2 Mamdani model and T2 TSK model.

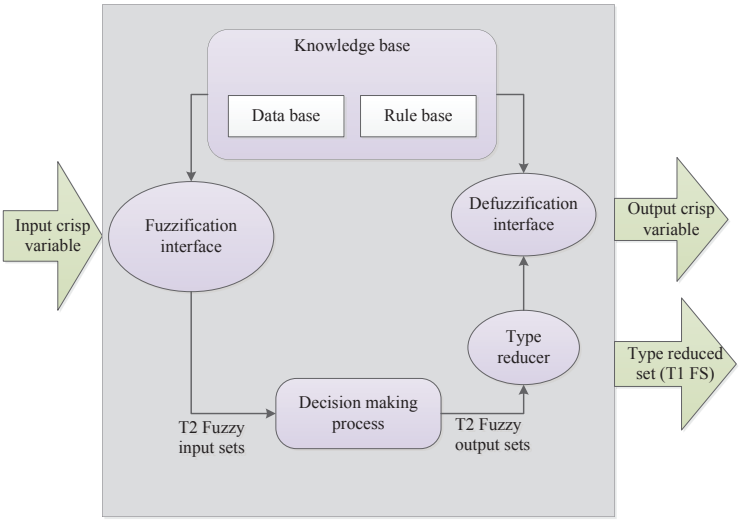


Figure 2.5: Fuzzy inference system (type 2)

2.2.3 Various conventional neuro-fuzzy systems

Fuzzy Logic Systems (FLSs), and Neural Networks (NNs) are the two primary artificial intelligent methodologies in the field of computational intelligence. The fuzzy logic system has the capability to imitate human-like behavior by utilizing linguistic rules, and the NN can learn and store information like the human brain. In FLS, the construction of Membership Functions (MFs) and linguistic rules is challenging since there is no automated way to construct the MFs and rules. Usually, the development of these MFs or rules is accomplished by expert knowledge or trial and error process. Even the experts face difficulties in constructing the MFs or rules for FLSs of large and complicated applications since those FLSs consist of a higher number of inputs, outputs, and linguistic parameters, which raises the number of possible linguistic rules exponentially. Consequently, experts fail to describe a full set of rules with corresponding MFs for obtaining satisfactory system performance. Without the expert knowledge, the problem can be solved by utilizing Evolutionary Algorithms (EAs), since the EAs do not need any information about the MFs or rule base a prior. However, the performance of EA is slow due to a large population, which affects many researchers not to use EA based FLS. NN-based fuzzy system can be a solution to the problem since NNs can construct and train the MFs and rule base and optimize the linguistic parameters. The presentation of data is vital to NNs. The NNs have high sensitivity to the input range, that is, if the variation between the maximum and minimum value is high, it affects the NNs. Researches have been conducted to combine the NNs and FLS in a way, to sum their individual advantages and reduce the limitations of both systems. Usually, there are three different ways [35] to combine the NNs and FLS, and they are namely: 1) cooperative, 2) concurrent and 3) hybrid neuro-fuzzy systems. In cooperative neuro-fuzzy systems, the learning techniques

of NNs are utilized to construct the fuzzy MFs or rule-base from the training data, and then the FLS works individually or vice versa. In the concurrent neuro-fuzzy system, the NNs and FLSs operate in parallel and help each other to determine the necessary parameters. In the hybrid neuro-fuzzy system, the architectural presentation FLS is like NN, which enables the NNs to apply their learning algorithm to FLS. However, these conventional fuzzy models are not able to evolve their structure. State of the art of the Evolving Intelligent System (EIS) based fuzzy systems, i.e. Evolving Fuzzy Systems (EFSs) are described in the following subsections.

2.3 State of the Art in Evolving Fuzzy Systems

2.3.1 Popular evolving fuzzy systems

The idea of an EIS was first implemented by [41], in their proposed Self-Organizing Neural Fuzzy Inference Network (SONFIN). SONFIN has the ability to evolve both the structure and parameter and performs excellently in single-pass learning mode. However, SONFIN doesn't have the ability to remove ineffective fuzzy rules, which results in a generation of higher numbers of rules. A Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) was developed by [42], where they have utilized an Evolving Clustering Method (ECM) for evolving the fuzzy rules. However, in both SONFIN and DENFIS, the inputs are partitioned using a distance-based approach, and thus, lack of robustness is observed against outliers. In the Self-Organizing Fuzzy Neural Network (SOFNN) [43], Ellipsoidal Basis Function (EBF) neurons have the ability of self-organizing, even if they have no prior information about the number of neurons. The development of evolving Takagi-Sugeno (eTS) fuzzy by [44] had a huge impact on the rapid growth of EIS.

The eTS algorithm is developed based upon online subtractive clustering [45], with the adoption of recursive density estimation concept. However, in the case of sequential arrival of data samples, the eTS are not able to simplify the rule base by eliminating those rules which may become useless with the upcoming data samples. A simplified version of eTS called simp_eTS was proposed by [46]; In simp_eTS, they have replaced the concept of data potential with data scatter, which induces faster computation than the eTS technique. Sequential Adaptive Fuzzy Inference System (SAFIS) was developed by [47], where they amend the rule growing and pruning modules called GAP-RBF and GGAP-RBF [48, 49] for the fuzzy system. However, the limitation of the SOFNN, eTS, simp_eTS, and SAFIS is the utilization of univariate Gaussian fuzzy rule, which does not express the scale-invariant characteristics. Furthermore, an approach called FLEXFIS is developed by [50], which exhibits an incremental version of vector quantization. However, the limitations of the FLEXFIS is the missing of simplification technique for rule base, which results in a complex structure. To solve the problem, an extended version of FLEXFIS, called FLEXFIS++ is proposed by [51]. In FLEXFIS++, they have overcome their previous limitations by accomplishing the automatic merge of redundant rules. An improved version of eTS called eTS+ was proposed by [52]. In eTS+, Angelov had simplified few methods of online rule base and reduced online dimensionality in compare to the original eTS. The simp_eTS+ technique, which is a modified version of eTS+ was developed by [53]. In simp_eTS+, Angelov employed the density increment technique for adding rules. The simp_eTS+ performs faster than the eTS in terms of the computational efficiency. However, both eTS+ and simp_eTS+ utilize the axis parallel ellipsoidal data distribution. Thus they are not effective to deal with data distributions which are not axis-parallel.

2.3.2 Multi-variable Gaussian-based evolving fuzzy system

The EFSs discussed in the previous subsection have utilized univariate Gaussian function, which are not effective to deal with non-axis-parallel data distributions. As a solution researchers have tried to look for multivariate Gaussian function based EFS. An evolving Multi-variable Gaussian (eMG)-based fuzzy classifier was developed by [54]. The eMG is an extension of evolving Participatory Learning (ePL), where the multi-variable Gaussian generalizes the basic TS fuzzy rules. However, they did not integrate the method of online dimensionality reduction in their proposed eMG based system. Another significant contribution to the field of EIS is AngelovYager (AnYa), developed by [55]. In their proposed technique, they have altered the scalar and parametrized antecedent part of the conventional Mamdani and TS fuzzy set with non-parametric data clouds. In the defuzzification process, they have replaced the Centre of Gravity (COG) method with the fuzzily weighted sum (average) technique. These approaches make the fuzzy system simple and faster with the online evolving feature. However, their proposed technique is based on the idea of relevance, which suffers from the discontinuity problem. An advanced EIS called Parsimonious Network-based Fuzzy Inference System PANFIS is developed by [56]. The PANFIS is an improved version of SAFIS, where they have modified the statistical contribution theory by the multivariate Gaussian function. After this, the PANFIS is expanded in [1] by incorporating a new online feature selection scheme. In the expanded FLEXFIS [57], they have employed multivariate Gaussian function with an off-diagonal covariance matrix. Further, they have employed geometrical rule merging principle to attain high compactness and a smooth feature weighting

concept to gently mitigate the curse of dimensionality effect (i.e., small weighted features have a minor influence in all the calculated distance). An implementation of the statistical contribution measure of the Generic Evolving Neuro-Fuzzy Inference System (GENEFIS) [1] is observed in their work for approximating feature contribution. In AnYA, the statistical contribution only considers feature saliency without taking into account mutual information across input attributes. In all the above-mentioned algorithms, the feedforward network structure is utilized, which is unable to deal with the temporal system dynamic properly. Further, the feedforward network structure is over dependant on the time-delayed input attributes.

2.3.3 Recurrent network structure-based EFS

To mitigate the limitation with the feedforward network structure, the idea of EIS has been addressed in the recurrent network structures too. This idea of evolving recurrent network has been introduced by [58] in their proposed RSONFIN, where they have inserted a global feedback loop in the previous SONFIN. However, like the SONFIN, the RSONFIN is also utilizing a distance-based clustering approach, and thus RSONFIN is not robust against the outliers. Another global recurrent network topology based EIS called TSK-type recurrent fuzzy network (TRFN) is proposed by [59]. In TRFN, a combination of the gradient descent and GA methods is observed in the adaptation process of EIS. However, the utilization of GA hinders scalability of an online scheme, since GA has computational prohibitive behavior. Besides, the global recurrent network structure performs less effectively in an online learning environment as it ignores the local learning scenario, and consequently assumes the EIS as a loosely coupled fuzzy model. Due to the global recurrent network structure, the input dimension also expands,

which may add to cause the *curse of dimensionality*. To solve the problem, local recurrent network architecture based EIS is proposed by [60], where the local recurrent connection is observed in the rule base. The concept of interactive recurrent network structure was extended by [61] in their proposed IRSFNN, where they have observed that the functional-link-based IRSFNN performs better than TSK type IRSFNN. However, with respect to their global counterpart, the interactive structure may hamper the quality of the local learning scheme since recurrent components are interconnected across different rules. All the above-mentioned algorithms have utilized the Type-1 Fuzzy System (T1 FS), which contain crisp and specific memberships. The T1 FS is not robust enough to handle the uncertainty issue, especially in time of dealing with the imprecise, inexact and inaccurate real-world data streams, where it is essential to identify the parameters accurately. Note that in the real world systems, disagreement may occur in the expert's knowledge, and noises are observed in measurements. Due to these reasons, uncertainties are present in real-world data streams.

2.3.4 KM type reduction technique-based type-2 EFS

To deal with the uncertainty issue, the idea of the Type-2 Fuzzy System (T2 FS) was developed by [38], where the membership value of the T2 FS is itself a fuzzy set between 0 and 1. Thus the MFs are not precise, and rather they are fuzzy in T2 FS. Well-established T1 FS mathematics cannot process the T2 FS [62]. Further, the T2 FS suffers from prohibitive computational complexity, which occurs due to the type reduction mechanism from type-2 to type-1. To reduce the complexity of the T2 FS, an interval type-2 fuzzy system (IT2 FS) was developed by [63], where they have assumed the weight of secondary membership function of the T2 FS to be unity. This IT2 FS can be expressed as an interval-valued fuzzy

system by using a single interval primary membership [64]. The integration of IT2 FS with EIS was introduced by [65], where they have improved their previously developed SONFIN with the type-2 fuzzy set. Afterward, they have extended their idea in local recurrent architecture [60], and to the interactive recurrent architecture [66] as well. However, due to the distance-based clustering technique, the method developed by [67] is not robust against outliers. Interval type-2 based EIS with the hybrid learning scheme is developed by [68], where they have proposed three dissimilar configurations of the interval type-2 network structure based on gradient descent method. However, their proposed network is static, and consequently not able to adapt to the changing learning environments. The IT2 FS was incorporated into the Mamdani type fuzzy system by [69]. [66] have proposed a new parameter learning scheme to address the interpretability problem of the interval type-2 based EIS. However, the feature selection process, which is required as a part of the preprocessing step, is absent in both of the previously mentioned works. All the interval type-2 based EIS, mentioned in this paragraph, use the Karnik-Mendel (KM) type reduction technique intensively and therefore suffer from the scalability issue.

2.3.5 Other type reduction technique-based type-2 EFS

An evolving type-2 fuzzy classifier called GT2FC was developed by [70], which works in a purely sequential learning scheme. The GT2FC is a zero-order classifier, where the rule consequent is set in the class label. Therefore, they can work without using the KM type reduction method. But, the accuracy of the zero-order classifier is normally lower than a first order or higher-order classifier, because it does not predict the decision surface of the classification problem. It is worth noting though if there is no (monotonic) order of class indices, the accuracy

of a higher order classifier is compromised. In [71], the KM method was replaced with q coefficient in the fixed structure of an IT2FNN. The idea of incorporating the q coefficient in the interval type-2 EIS was also adopted by [72, 73]. It is essential to mention that all the IT2FSs discussed here are interval value fuzzy systems since single interval primary membership is employed in all the cases. In the last decade, huge research on EISs has been conducted. However, most of the underlying design is cognitive since they mainly concern about the *how to learn* issue, and do not consider the other two vital issue of *what to learn* and *when to learn*. This infers that all the data streams are in order without giving careful consideration to their impact on the training progress.

2.3.6 Metacognitive Learning Machine (McLM)-based EFS

The idea of the McLM was introduced in EIS by [74] in their proposed architecture called Self-Adaptive Resource Allocation Network (SRAN). In SRAN, they have claimed that the adaptive and evolving nature of EIS can be improved by interpreting the meta-memory model of [75] into the machine learning context. In their proposed idea, they have incorporated the issues of when-to-learn and what-to-learn, which was not included in the conventional EIS's training process. The work has been expanded by a variety of cognitive components like fully complex network architecture [76], type-1 fuzzy systems [77], and interval type-2 fuzzy systems [78]. Nevertheless, a limitation of the McLM is the absence of important learning modules in the main learning engine; therefore it depends upon the pre-and/or post-training steps. This pitfall highly distracted them from the underlying spirit of the online real-time learner, which can be predicted as a plug-and-play learning algorithm. The integration of the sample selection scheme only actualize the fully supervised learning scenario and consequently

charges costly annotation efforts by the operator.

2.3.7 Scaffolding McLM (McSLM)-based EFS

The McSLM aims to tackle this issue by incorporating the Scaffolding theory. It is a prominent tutoring theory in psychology, by which a learner can solve a complex learning task to develop the how-to-learn component of the metacognitive learning scenario. The idea of McSLM was pioneered by [79], and later, this work was modified by [80]. Three issues were unsolved in their work, namely the uncertainty, temporal system dynamics, and the unknown system order. The uncertainty issue arises because of the existing McSLM's type-1 FNN architecture, which is well-known to suffer from the issue of uncertainty. McSLMs are also built on the traditional feed-forward network topology and are not robust against temporal system dynamics. The feed-forward network architecture entails prior knowledge of the lagged input features to form the input-output relationship of the regression model. In addition, the vast majority of McLMs and McSLMs in the current literature are designed for the classification problem and to the best of our knowledge, only that of [81] handles the regression cases. The work, done by [81] still leaves open questions, because it shares similar characteristics with the work of [78]. To solve these three existing problems of McSLM based EIS, a novel Metacognitive Scaffolding Based Interval type 2 Fuzzy Recurrent Neural Network (McSIT2FRNN) is proposed by [82]. Classification of all algorithms discussed in this section is summarized in Table 2.1.

Table 2.1: Summary of state-of-the art of various EFS

References	Working Principle	Structure	Hidden node
[41]	Evolving	Feedforward	type-1 spherical rule
[44]	Evolving	Feedforward	type-1 spherical rule
[46]	Evolving	Feedforward	type-1 spherical rule
[47]	Evolving	Feedforward	type-1 spherical rule
[52]	Evolving	Feedforward	type-1 axis-parallel rule
[53]	Evolving	Feedforward	type-1 axis-parallel rule
[54]	Evolving	Feedforward	type-1 non-axis-parallel rule
[55]	Evolving	Feedforward	type-1 cloud-based rule
[56]	Evolving	Feedforward	type-1 non axis-parallel rule
[1]	Evolving	Feedforward	type-1 non axis-parallel rule
[57]	Evolving	Feedforward	type-1 non axis-parallel rule
[58]	Evolving	Global Recurrent	type-1 spherical rule
[59]	Evolutionary	Global Recurrent	type-1 spherical rule
[60]	Evolving	Local Recurrent	type-1 spherical rule
[67]	Evolving	Interactive	type-1 spherical rule
[65]	Evolving	Feedforward	type-1 spherical rule
[66]	Evolving	Feedforward	type-1 spherical rule
[69]	Evolving	Feedforward	type-1 spherical rule
[70]	Evolving	Feedforward	type-2 non axis-parallel rule
[72]	Evolving	Feedforward	type-2 compensatory axis-parallel rule
[73]	Evolving	Feedforward	type-2 axis-parallel rule
[74]	Metacognitive	Feedforward	type-1 spherical rule
[76]	Metacognitive	Feedforward	type-1 spherical rule
[77]	Metacognitive	Feedforward	type-1 spherical rule
[78]	Metacognitive	Feedforward	type-2 spherical rule
[79]	Metacognitive scaffolding	Feedforward	type-1 spherical rule
[80]	Metacognitive scaffolding	Feedforward	type-2 spherical rule

2.3.8 Challenges with the existing EFS-based autonomous learning algorithms

Until now, existing EFS-based autonomous learning algorithms are usually constructed via hypersphere based or hyperellipsoid based clustering techniques (HSBC or HEBC) to automatically partition the input space into a number of fuzzy rules and rely on the assumption of normal distribution due to the

use of Gaussian membership functions [1, 44, 46, 56, 83–87]. As a result, they are always associated with rule premise parameters, the mean and width of the Gaussian function, which need to be continuously adjusted in data stream analysis. This issue complicates their implementation in a complex and deep structure. As a matter of fact, existing neuro-fuzzy systems can be seen as a single hidden layer feedforward network. Other than the HSBC or HEBC, the data cloud-based clustering (DCBC) concept is utilized in [55, 88] to construct the EFS. Unlike the HSBC and HEBC, the data clouds do not have any specific shape. Therefore, required parameters in DCBC are less than HSBC and HEBC. However, in DCBC, parameters like mean, accumulated distance of a specific point to all other points need to be calculated. In other words, it does not offer significant reduction on the computational complexity and memory demand of EFS. Hyperplane-Based Clustering (HPBC) provides a promising avenue to overcome this drawback because it bridges the rule premise and the rule consequent by means of the hyperplane construction. Although the concept of HPBC already exists since the last two decades [89–91], all of them are characterized by a static structure and are not compatible for data stream analysis due to their offline characteristics [92]. Besides, the majority of these algorithms still use the Gaussian or bell-shaped Gaussian function [93] to create the rule premise and are not free of the rule premise parameters. This problem is solved in [94], where they have proposed a new function to accommodate the hyperplanes directly in the rule premise. Nevertheless, their model also exhibits a fixed structure and operates in the batch learning node. Based on this research gap, a novel EFS, namely parsimonious learning machine (PALM), is proposed and described in chapter 3. In the next section, an overview of a nonlinear dynamical system namely UAV and challenges in its control autonomy, are discussed.

2.4 Unmanned Aerial vehicles and the Control Autonomy

Advancements in portable electronic technology over the past few years encourage researchers to work on Unmanned aerial vehicles (UAVs). They are aircraft with no aviator on-board. In relation to the wing types, UAVs are usually classified into three subdivisions, and they are namely: 1) fixed wing, 2) rotary wing, and 3) flapping wing UAVs, where the rotary wing UAVs (RUAVs) can be categorized as a helicopter, quadcopter, hexacopter, octocopter, etc. Comparison among fixed wing, rotary wing, and flapping wing UAVs in various aspects are exposed in Table 2.2.

Table 2.2: Comparison among various UAVs (H: high, M: medium, L: low)

Characteristics	**Rotary wing**	**Fixed wing**	**Flapping wing**	**References**
Maneuverability	H	L	M	[95, 96]
Expenditure	M	L	H	[97, 98]
Complexity in assembly	M	L	H	[99, 100]
Civilian usage	H	L	H	[101–104]
Military Application	M	H	M	[105, 106]
Power Consumption	L	H	M	[107, 108]
Flight safety	M	M	L	[109, 110]
Range	M	H	L	[111, 112]

Attempts to humanize the UAVs, such as with the capacity of autonomous flying in a confined space is challenging. A positive step to such achievement is the low cost of miniature electronic components like sensors, actuators, microprocessors, batteries, etc. Another point is their immense applicability in both civilian and military sectors. Usage of UAVs in the military sector has intensified in the last decades as explored in [113]. Participation of UAVs in civilian application has a lot of socioeconomic benefits. For example, images provided by satellites

are costly and depend upon weather condition. Since the UAVs can fly very close to the ground, it can take images with better resolution in severe weather. Sometimes, human cannot capture images due to the lack of accessible roads. Surveillance in such locations with wide-area and spot imagery can easily be achieved by UAVs. In agriculture industries, they can contribute ingeniously to monitor crops health, which helps farmers to make an effective decision. In energy industries UAVs can be used predominantly to inspect large solar power plants, high voltage power cables without interrupting the power supply, and to observe critical places of hydroelectric dams, where human inclusion is unsafe. In the infrastructure sector, UAVs are feasible to capture post-disaster imagery of critical infrastructure, inspect pipelines, and roads. In the transportation sector, UAVs are possible to utilize in delivering goods to remote areas. They might be a great help in a rescue mission by accessing to dangers faster and safely. To transmute all these potentialities of UAVs into reality, a major concern is to pursue the preferable control autonomy.

In a recent survey on the technological growth of small autonomous robots in [114], they split the control autonomy of UAVs into three tiers, namely 1) sensory-motor autonomy, 2) reactive autonomy, and 3) cognitive autonomy. The sensory-motor autonomy indicates the translation of human instructions into desirable control signals, such as maintaining the track of the desired trajectory with appropriate altitude, roll, pitch, and yaw angle or velocity commands. Reactive autonomy indicates the capability of UAVs to preserve user-defined positions or trajectories in the presence of environmental disturbances like wind gust, motor/actuator failure; the ability of obstacle-avoidance, retaining safe distance from the ground; proficiency to takeoff or land in moving objects like moving ship. Cognitive autonomy is an advanced version of the reactive autonomy, which

enables UAVs to learn from the environment in resolving inconsistent facts, to plan for future requirement like recharging battery. It will capacitate UAVs in simultaneous mapping and localization along with their obstacle-avoidance and path-planning aptitude. Realization of cognitive autonomy in UAVs is still far-reaching in terms of control certitude that allows them to follow commands precisely from many sensors.

The first principle techniques are one of the commonly used methods to control UAVs. The necessity of a precise mathematical model is a limitation of these methods since uncertainties are extremely difficult to model. It forwarded researchers to look for model-free knowledge-based techniques. Among various knowledge-based techniques, FLSs and NNs are prominently utilized in various UAVs. Before going through various FLSs and NFSs, their advancements and application in UAVs, the use of different first principle techniques in controlling UAVs are discussed in the next subsection of this chapter.

2.4.1 First principle techniques in controlling UAVs

Proportional Integral Derivative (PID) is one of the most commonly used first principle-based control technique for UAVs [115]. Linear Quadratic (LQ) methods are also employed successfully in different UAVs [116] due to their simple and linear structure. However, linear controllers performance degrade to deal with environmental disruption due to UAV's inherent non-linearity and coupled dynamics. In rejecting disturbance of nonlinear UAV dynamics, nonlinear control techniques namely back-stepping [117–119], Sliding Mode Control (SMC) [118, 120–123], Feedback Linearization (FBL) [121, 124], H∞ robust control [125, 126] etc. perform better than their linear counterpart.

In the backstepping control technique, the controller is divided into several

stages, which facilitates fast convergence and reduce computational time as well. [127] utilized the back-stepping control method for an UAV, where they have witnessed proper tracking of position and yaw. However, they have not validated their controller through experimentation. [128] implemented the FBL method to achieve a stable trajectory tracking performance from the quadcopter UAV, where they have witnessed a satisfactory position and velocity tracking. Nonetheless, FBL technique requires a transformation to convert the nonlinear system into an equivalent linear system by changing the variables. This linearization may cause imprecision and degrade their performance. In addition, they have not addressed disturbances like aerodynamic drag forces, parametric uncertainties in their controller. [121] associated the FBL with an adaptive SMC. They have proved their controller's robustness theoretically by deriving an adaptation rule. However, during experimentation, it was not robust against uncertainties and noise from UAVs sensors. To be summarized, the majority of these conventional control systems are based on dynamic modeling of the system, where the precise mathematical model is a necessity to regulate their performance. Besides, their control parameters are fixed or bounded. A smart solution to these problems is the utilization of model-free knowledge-based or data-driven techniques.

2.4.2 Model-free intelligent control approaches for UAVs

Being a model-free approach [129], NNs [130, 131], FLSs and NFSs [132, 133] are used in control application [3, 134–136]. To develop a FLS, NN, or a NFS-based intelligent controller with a better tracking accuracy is challenging. When the system dynamics and various characteristics of a plant to be controlled are known, then they are utilized to train the controller and to construct a fixed-structured controller with a certain number of rules, membership functions, neurons, and

layers. Due to the fixed structure and absence of parameter adaptation, these controllers cannot cope with changing plant dynamics. However, in many control applications, the plant dynamics and other system information may not be readily available. Therefore, the fixed-structured FLS, NN, or NFS-based controller become unreliable or unobtainable. Furthermore, the characteristics of real-world plants are non-stationary. A fixed-size controller may fail to regulate such plants. To improve the performance of the model-free static controllers, researchers have tried to combine them with conventional nonlinear control techniques such as backstepping [117], sliding mode techniques [137], FBL, H∞ [138], etc. In [139, 140], nonlinear adaptive neural controllers are trained offline to ensure the stability of different types of aircraft. To cope with the variations in aircraft dynamics, their controller can adapt the weights online. A neural controller was augmented with a conventional controller in [141] to enhance the fault tolerant capabilities of a high-performance fighter aircraft during the landing phase when subjected to severe winds and failures. Though these controllers can adapt to the disturbances by tuning the parameters, they cannot evolve their structure. Therefore, selecting the required number of rules to achieve the desired control accuracy is difficult [142]. The problem can be circumvented by using evolving FLS, NN, NFS-based controllers [143].

2.4.3 Autonomous intelligent control approaches for UAVs

Research on Autonomous Intelligent Controllers (AICons) with evolving structure has started at the beginning of the twenty-first century. In 2003, a self-organizing NF-based AICon was proposed in [144], where they applied system error, ϵ-completeness, and error reduction ratio in their rule evolution scheme. Their controller needed to store all preceding input-output data, which forced to com-

pute a large matrix in each step and yields a high computation cost. It makes them impractical to implement in systems like UAVs, where a fast response is expected from the controller to emulate the desired commands. Another AICon was developed in [145] by utilizing the evolving Takagi Sugeno (eTS) model [44]. Though their controller was evolving in nature, it suffers from several imperfections. First, in their structural evolution mechanism, fuzzy rules were added or replaced only, pruning of rules was missing. Besides, their AICon needed to memorize the previous data obtained from both the plant and controller. Such obligation leads to a computationally costly control mechanism and made them unrealistic in the swift reaction-based control application. A hybrid AICon was developed in [146] by mixing an evolving and static TS fuzzy system. Despite their design simplicity, they required to know some parameters of the plant to be controlled. Such parameters may not be available during control operation. An AICon was also actualized using model predictive control technique [147]. However, their dependency on the plant's dynamic model restricted their application in complex nonlinear systems, where dynamic models may not be known. An Extended Minimal Resource Allocation Network (EMRAN)-based autonomous fault-tolerant neural control scheme was developed in [148]. The EMRAN was utilized to grow or prune the hidden neurons in their autonomous controller. The weights of their neural controller were adapted using an Extended Kalman Filter (EKF). The thresholds associated with the growing and pruning criteria and EKF were optimized using Genetic Algorithm (GA).

A back-stepping-based autonomous radial basis function (RBF) neural controller was developed in [149] reconfigurable flight control of aircraft in the presence of large changes in the aerodynamic characteristics and failures. Lyapunov theory was used to prove the stability in the ultimate bounded sense for their

controller. An evolving controller using Interval Type-2 (IT2) NF structure was proposed in [150] to control a quadcopter UAV. Nonetheless, their IT2NF system was functioning as an uncertainty and perturbation observer, and a PD controller was used to control the attitude and position of the quadcopter. In [151], an evolving NF controller was proposed for a simulated quadcopter UAV plant. Though their controller successfully generated and pruned fuzzy rules on the fly with a satisfied tracking accuracy, they only had considered a simplified dynamic model of the quadcopter to be controlled.

2.4.3.1 Adaptation of consequent parameters in AICon

In AICons, the adaptation of consequent parameters plays a crucial role to attain the desired accuracy. To adapt the AICon's rule consequent parameters, gradient-based methods are typically used [152]. However, the gradient-based controllers perform well only when they were used to control plants with a slow variation in dynamics. Furthermore, gradient-based methods like dynamic back-propagation techniques comprise of partial derivatives. Such algorithms can not guarantee fast convergence speed, particularly in complex non-convex search space. In addition, there are chances to be trapped in a local minimum [153]. Alternately, evolutionary algorithms were attempted in [154] to tune parameters. Nevertheless, the stability of their proposed controller was not confirmed, and the fast response was not ensured. Such constraints can be handled simply by imposing SMC theory to adapt the consequent parameters as witnessed in [5].

2.4.4 Challenges in real-time implementation of the AICons for UAVs

Though the evolving fuzzy and neuro-fuzzy-based AICons have advantages over their existing static and adaptive variants, a challenge in their real-time implementation is their complex structure. In AICons, a large number of parameters are required to be tuned online, which is very difficult to achieve by using the limited memory resource of the existing UAV systems. To minimize the parameters, AICons with minimum parameters are developed in Chapter 4. Another limitation of the AICons is their predefined thresholds to shape their structure. To overcome such dependency, the bias-variance concept-based network significance method is proposed to develop AICon in Chapter 4.

2.5 Summary

In this chapter, the fundamental concept about the fuzzy system, state of the art of various fuzzy and neuro-fuzzy systems from the static to evolving ones are discussed. It includes the challenges in data stream analysis, in modeling nonlinear dynamical systems, identification, and control of UAVs. The limitations of conventional FPT-based controllers for UAVs make various fuzzy and neuro-fuzzy system an appropriate candidate to control various UAVs. However, the majority of them are a batch learning-based system, which causes an unsatisfactory performance in handling uncertainties. The problem is partially solved by tuning the parameters of the fuzzy system. Still, they fail to cope with sudden changes in UAVs flight behavior due to their static structure. In such cases, evolving fuzzy and neuro-fuzzy system-based autonomous learning algorithms are the appropriate candidates since they can change their structure

to cope with sharp changes. Evolving fuzzy and neuro-fuzzy-based autonomous learning algorithms' application in modeling and controlling nonlinear dynamical systems like UAVs with existing limitations, and possible opportunities are also reviewed in this chapter. From the literature survey, some research gaps have been identified with autonomous learning algorithms in dealing with nonlinear dynamical systems. The details in each of these contributions are presented one by one in further chapters.

Chapter 3

PALM: An Incremental Construction of Hyperplanes for Data Stream Regression

The work furnished in this chapter has been published in the following article:

- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A. (2019). PALM: An Incremental Construction of Hyperplanes for Data Stream Regression. *IEEE Transactions on Fuzzy Systems* (DOI: 10.1109/TFUZZ.2019.2893565).

Abstract

In this chapter, a Self-Adaptive Neuro-Fuzzy Systems (SANFS) based advanced autonomous learning algorithm, namely PArsimonious Learning Machine (PALM), is proposed. PALM features utilization of a new type of fuzzy rule based on the concept of hyperplane clustering, which significantly reduces the number of

network parameters because it has no rule premise parameters. PALM is proposed in both type-1 and type-2 fuzzy systems where all of them characterize a fully dynamic rule-based system. Therefore, it is capable of automatically generating, merging and tuning the hyperplane-based fuzzy rules in the single-pass manner. Moreover, an extension of PALM, namely recurrent PALM (rPALM), is proposed that adopts the concept of teacher-forcing mechanism in the deep learning literature. The efficacy of PALM has been evaluated through numerical study with six real-world and synthetic data streams from public database and our project of autonomous vehicles. The proposed model showcases significant improvements in terms of computational complexity and number of required parameters against several renowned SANFSs while attaining comparable and often better predictive accuracy.

3.1 Introduction

Advances in both hardware and software technologies have triggered the generation of a large quantity of data in an automated way. Such applications can be exemplified by space, autonomous systems, aircraft, meteorological analysis, stock market analysis, sensors networks, users of the internet, etc., where the generated data are not only massive and unbounded but also produced at a rapid rate under complex environments. Such online data are known as data stream [155, 156]. A data stream can be expressed more formally [157] as $S = \left\{x^1, x^2, ..., x^N\right\}$ i.e. $S = \{x^i\}_{i=1}^{N}$, where x^i is an enormous sequence of data objects and possibly unbounded ($N \to \infty$). Each of the data objects can be defined by an n-dimensional feature vector as $x^i = [x_j^i]_{j=1}^{n}$, which may belong to a continuous, categorical, or mixed feature space. In the field of data stream mining, developing a learning algorithm

as a universal approximator is challenging due to the following factors: 1) the whole data to train the learning algorithm is not readily available since the data arrive sequentially; 2) the size of a data stream is not bounded; 3) dealing with a huge amount of data; 4) distribution of the incoming unseen data may slide over time slowly, rapidly, abruptly, gradually, locally, globally, cyclically or otherwise. Such variations in the data distribution of data streams over time are known as *concept drift* [158, 159]; 5) data are discarded after being processed to suppress memory consumption into a practical level.

To cope with above-stated challenges in data streams, the learning machine should be equipped with the following features: 1) capability of working in single-pass mode; 2) handling various concept drifts in data streams; 3) has low memory burden and computational complexity to enable real-time deployment under resource-constrained environment. In the realm of fuzzy system, such learning aptitude is demonstrated by SANFS [41]. Until now, existing SANFSs are usually constructed via hypersphere-based or hyperellipsoid-based clustering techniques (HSBC or HEBC) to automatically partition the input space into a number of fuzzy rules and rely on the assumption of normal distribution due to the use of Gaussian membership functions [1, 44, 46, 56, 83–87]. As a result, they are always associated with rule premise parameters, the mean and width of the Gaussian function, which need to be continuously adjusted. This issue complicates their implementation in a complex and deep structure. As a matter of fact, existing neuro-fuzzy systems can be seen as a single hidden layer feedforward network. Other than the HSBC or HEBC, the data cloud-based clustering (DCBC) concept is utilized in [55, 88] to construct the SANFS. Unlike the HSBC and HEBC, the data clouds do not have any specific shape. Therefore, the required parameters in DCBC are less than HSBC and HEBC. However, in

DCBC, parameters like mean, and accumulated distance of a specific point to all other points need to be calculated. In other words, it does not offer significant reduction on the computational complexity and memory demand of SANFS. Hyperplane-Based Clustering (HPBC) provides a promising avenue to overcome this drawback because it bridges the rule premise and the rule consequent by means of the hyperplane construction.

Although the concept of HPBC already exists since the last two decades [89–91], all of them are characterized by a static structure and are not compatible for data stream analytic due to their offline characteristics. Besides, the majority of these algorithms still use the Gaussian or bell-shaped Gaussian function [93] to create the rule premise and are not free of the rule premise parameters. This problem is solved in [94], where they have proposed a new function to accommodate the hyperplanes directly in the rule premise. Nevertheless, their model also exhibits a fixed structure and operates in the batch learning node. Based on this research gap, a novel SANFS, namely parsimonious learning machine (PALM), is proposed in this chapter.

3.2 Contribution

The proposed algorithm PALM in this chapter is equipped with some new features. The novelty of PALM can be summarized as follows:

1. **Hyperplane-based Fuzzy Rule:** PALM is constructed using the HPBC technique, and its fuzzy rule is fully characterized by a hyperplane which underpins both the rule consequent and the rule premise. This strategy reduces the rule base parameter to the level of $C * (P + 1)$ where C, P are respectively the number of fuzzy rules and input dimensions.

2. **Type-1 and Type-2 architecture:** PALM is proposed in both type-1 and type-2 versions derived from the concept of type-1 and type-2 fuzzy systems. Type-1 version incurs less network parameters and faster training speed than the type-2 version whereas type-2 version expands the degree of freedom of the type-1 version by applying the interval-valued concept leading to be more robust against uncertainty than the type-1 version.

3. **New mechanism of evolving structure:** PALM features a fully open network structure where its rules can be automatically generated, merged and updated on-demand in the one-pass learning fashion. The rule generation process is based on the self-constructing clustering approach [160, 161], checking the coherence of input, and output space. The rule merging scenario is driven by the similarity analysis via the distance and orientation of two hyperplanes. The online hyperplane tuning scenario is executed using the Fuzzily Weighted Generalized Recursive Least Square (FWGRLS) method.

4. **Recurrent PALM architecture:** An extension of PALM, namely recurrent PALM (rPALM), is put forward in this work. rPALM addresses the underlying bottleneck of HPBC method: dependency on the target variable due to the definition of point-to-hyperplane distance [162]. This concept is inspired by the teacher forcing mechanism in the deep learning literature where the activation degree of a node is calculated with respect to predictor's previous output. The performance of rPALMs have been numerically validated where their performance is slightly inferior to PALM but still highly competitive to most prominent SANFSs in terms of accuracy.

5. **Real-world application:** Two real-world problems from our project, namely

online identification of quadcopter and unmanned helicopter, are presented in this chapter and exemplify real-world streaming data problems. The two datasets are collected from indoor flight tests in the UAV lab of the University of New South Wales (UNSW), Canberra campus. These datasets, PALM, and rPALM codes are made publicly available in [163].

The efficacy of both type-1 and type-2 PALM has been numerically evaluated using six real-world and synthetic streaming data problems. Moreover, PALM is also compared against prominent SANFSs in the literature and demonstrates encouraging numerical results in which it generates compact and parsimonious network structure while delivering comparable and even better accuracy than other benchmark algorithms.

The remaining sections of this chapter is structured as follows: In Section 3.3, the network architecture of both type-1 and type-2 PALM are elaborated. Section 3.4 describes the online learning policy of type-1 PALM, while Section 3.5 presents online learning mechanism of type-2 PALM. In Section 3.7, the proposed PALM's efficacy has been evaluated through real-world and synthetic data streams. Finally, the chapter ends by drawing the concluding remarks in Section 3.8.

3.3 Network Architecture of PALM

In this section, the network architecture of PALM is presented in detail. The T-S fuzzy system is a commonly used technique to approximate complex nonlinear systems due to its universal approximation property. The rule base in the T-S fuzzy model of a multi-input single-output (MISO) system can be expressed in

the following IF-THEN rule format:

$$\begin{aligned} R^j: \quad & \text{If } x_1 \text{ is } B_1^j \text{ and } x_2 \text{ is } B_2^j \text{ and...and } x_n \text{ is } B_n^j \\ & \text{Then } y_j = b_{0j} + a_{1j}x_1 + ... + a_{nj}x_n \end{aligned} \tag{3.1}$$

where R^j stands for the jth rule, $j = 1, 2, 3, ..., R$, and R indicates the number of rules, $i = 1, 2, ..., n$; n denotes the dimension of input feature, x_n is the nth input feature, a and b are consequent parameters of the sub-model belonging to the jth rule, y_j is the output of the jth sub-model. The T-S fuzzy model can approximate a nonlinear system with a combination of several piecewise linear systems by partitioning the entire input space into several fuzzy regions. It expresses each input-output space with a linear equation as presented in Equation (3.1). Approximation using T-S fuzzy model leads to a nonlinear programming problem and hinders its practical use. A simple solution to the problem is the utilization of various clustering techniques to identify the rule premise parameters. Because of the generation of the linear equation in the consequent part, the HPBC can be applied to construct the T-S fuzzy system efficiently. The advantages of using HPBC in the T-S fuzzy model can be seen graphically in Figure 3.1.

Some popular algorithms with HPBC are Fuzzy C-Regression Model (FCRM) [164], Fuzzy C-Quadratic Shell (FCQS) [165], double FCM [89], Inter Type-2 Fuzzy C-Regression Model (IT2-FCRM) [94]. A crucial limitation of these algorithms is their non-incremental nature, which does not suit for data stream regression. Moreover, they still deploy Gaussian function to represent the rule premise of TS fuzzy model, which does not exploit the parameter efficiency trait of HPBC. To fill up this research gap, a new membership function [94] is proposed to accommodate the use of hyperplanes in the rule premise part of TS fuzzy

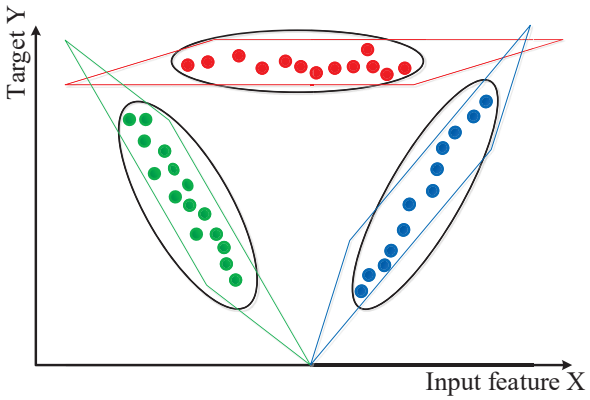


Figure 3.1: Clustering in T-S fuzzy model using hyperplanes

system. It can be expressed as:

$$\mu_B(j) = \exp\left(-\Gamma \frac{dst(j)}{\max\left(dst(j)\right)}\right) \tag{3.2}$$

where $j = 1, 2, ..., R$; R is the number of rules, Γ is an adjustment parameter which controls the fuzziness of membership grades. Based on the observation in [94], and empirical analysis with a variety of data streams in my research, the range of Γ is settled as $[1, 100]$. $dst(j)$ denotes the distance from the present sample to the jth hyperplane. In this chapter, $dst(j)$ is defined as follows [94]:

$$dst(j) = \frac{|X_t \omega_j|}{|\omega_j|} \tag{3.3}$$

where $X_t \in \Re^{1\times(n+1)}$ and $\omega_j \in \Re^{(n+1)\times 1}$ respectively stand for the input vector of the tth observation and the output weight vector of the jth rule. This membership function enables the incorporation of HPBC directly into the T-S fuzzy system with the absence of rule parameters except the first order linear function or hyperplane. Since a point to plane distance is not unique, the compatibility measure is executed using the minimum point to plane distance.

The following discusses the network structure of PALM encompassing its

type-1 and type-2 versions. PALM can be modeled as a four-layered network working in tandem, where the fuzzy rule triggers a hyperplane-shaped cluster and is induced by Equation (3.3). Since T-S fuzzy rules can be developed solely using a hyperplane, PALM is free from antecedent parameters which results in a dramatic reduction of network parameters. Furthermore, it operates in the one-pass learning fashion where it works point by point, and a data point is discarded directly once learned.

3.3.1 Structure of type-1 PALM network

In type-1 PALM network architecture, the membership function exposed in Equation (3.2) is utilized to fit the hyperplane-shaped cluster in identifying type-1 T-S fuzzy model. To understand the work flow, let us consider that a single data point x_n is fed into PALM at the nth observation. Appertaining to the concept of type-1 fuzzy system, this crisp data needs to be transformed into a fuzzy set. This fuzzification process is attained using a type-1 hyperplane-shaped membership function, which is framed through the concept of point-to-plane distance. This hyperplane-shaped type-1 membership function can be expressed as:

$$f_{T1}^{1} = \mu_B(j) = \exp\left(-\Gamma \frac{dst(j)}{\max\left(dst(j)\right)}\right) \tag{3.4}$$

where Γ is an adjustment parameter which controls the fuzziness of membership grades, $dst(j)$ in Equation (3.4) denotes the distance between the current sample and jth hyperplane as with Equation (3.3). It is defined as per definition of a point-to-plane distance [162] and is formally expressed as follows:

$$dst(j) = \left| \frac{y_d - \left(\sum_{i=1}^{n} a_{ij} x_i + b_{0j}\right)}{\sqrt{1 + \sum_{i=1}^{n} (a_{ij})^2}} \right| \tag{3.5}$$

where a_{ij} and b_{0j} are consequent parameters of the jth rule, $i = 1, 2, ..., n$; n is the number of input dimension, and y_d is the target variable. The exertion of y_d is an obstruction for PALM due to target variable's unavailability in the testing phase. This issue comes into the picture due to the definition of a point-to-hyperplane distance [162]. To eradicate such impediment, a recurrent PALM (rPALM) framework is developed here. Considering a MISO system, the IF-THEN rule of type-1 PALM can be expressed as follows:

$$R^j: \quad \text{IF } X_n \text{ is close to } f^2_{T1_j} \text{ THEN } y_j = x^j_e \omega_j \tag{3.6}$$

where x_e is the extended input vector and is expressed by inserting the intercept to the original input vector as $x_e = [1, x_1, x_2, ..., x_n]$, ω_j is the weight vector for the jth rule, y_j is the consequent part of the jth rule. Since type-1 PALM has no premise parameters, the antecedent part is simply hyperplane. It is observed from Equation (3.6) that the drawback of HPBC-based TS fuzzy system lies in the high-level fuzzy inference scheme which degrades the transparency of fuzzy rule. The intercept of the extended input vector controls the slope of hyperplane, which functions to prevent the atypical gradient problem.

The consequent part is akin to the basic T-S fuzzy model's rule consequent part ($y_j = b_{0j} + a_{1j}x_1 + ... + a_{nj}x_n$). The consequent part for the jth hyperplane is calculated by weighting the extended input variable (x_e) with its corresponding weight vector as follows:

$$f^2_{T1_j} = x^T_e \omega_j \tag{3.7}$$

Weights are used in Equation (3.7) after updating recursively by the FWGRLS method, which ensures a smooth change in the weight value. In the next step, the rule firing strength is normalized and combined with the rule consequent to

produce the end-output of type-1 PALM. The final crisp output of the PALM for a type-1 model can be expressed as follows:

$$f_{T1}^{3} = \frac{\sum_{j=1}^{R} f_{T1_j}^{1} f_{T1_j}^{2}}{\sum_{i=1}^{R} f_{T1_i}^{1}} \tag{3.8}$$

The normalization term in Equation (3.8) guarantees the partition of unity, where the sum of normalized membership degree is unity. The T-S fuzzy system is functionally-equivalent to the radial basis function (RBF) network if the rule firing strength is directly connected to the output of the consequent layer [166]. It is also depicted that the final crisp output is produced by the weighted average defuzzification scheme.

3.3.2 Network structure of the type-2 PALM

Type-2 PALM differs from the type-1 variant due to the use of interval-valued hyperplane, which generates the type-2 fuzzy rule. Akin to its type-1 version, type-2 PALM starts operating by intaking the crisp input data stream x_n to be fuzzied. Here, the fuzzification occurs with the help of type-2 interval-valued hyperplane-shaped clustering-based membership function, which can be expressed as:

$$\widetilde{f}_{out}^{1} = \exp\left(-\Gamma \frac{\widetilde{dst}(j)}{\max\left(\widetilde{dst}(j)\right)}\right) \tag{3.9}$$

where $\widetilde{f}_{out}^{1} = \left[\underline{f}_{out}^{1},\ \overline{f}_{out}^{1}\right]$ is the upper and lower hyperplane, $\widetilde{dst}(j) = \left[\overline{dst}(j),\ \underline{dst}(j)\right]$ is interval-valued distance, where $\overline{dst}(j)$ is the distance between present input samples and jth upper hyperplane, and $\underline{dst}(j)$ is that between present input samples and jth lower hyperplane. In type-2 architecture, distances among

incoming input data and upper and lower hyperplanes are calculated as follows:

$$\widetilde{dst}(j) = \left| \frac{y_d - (\sum_{i=1}^{n} \widetilde{a}_{ij} x_i + \widetilde{b}_{0j})}{\sqrt{1 + \sum_{i=1}^{n} (\widetilde{a}_{ij})^2}} \right| \tag{3.10}$$

where $\widetilde{a}_{ij} = \left[\underline{a}_{ij}; \overline{a}_{ij}\right]$ and $\widetilde{b}_{0j} = \left[\underline{b}_{0j}; \overline{b}_{0j}\right]$ are the interval-valued coefficients of the rule consequent of type-2 PALM. Like the type-1 variants, type-2 PALM has a dependency on target value (y_d). Therefore, they are also extended into type-2 recurrent structure in this chapter. The use of interval-valued coefficients results in the interval-valued firing strength, which forms the footprint of uncertainty (FoU). The FoU is the key component against uncertainty of data streams and sets the degree of tolerance against uncertainty.

In a MISO system, the IF-THEN rule of type-2 PALM can be expressed as:

$$R^j : \quad \text{IF } X_n \text{ is close to } \widetilde{f}^2_{out} \text{ THEN } y_j = x_e^j \widetilde{\omega}_j \tag{3.11}$$

where x_e is the extended input vector, $\widetilde{\omega}_j$ is the interval-valued weight vector for the jth rule, y_j is the consequent part of the jth rule, whereas the antecedent part is merely interval-valued hyperplane. The type-2 fuzzy rule is similar to that of the type-1 variant except the presence of interval-valued firing strength and interval-valued weight vector. In type-2 PALM, the consequent part is calculated by weighting the extended input variable x_e with the interval-valued output weight vectors $\widetilde{\omega_j} = \left[\underline{\omega_j},\ \overline{\omega_j}\right]$ as follows:

$$\overline{f}^2_{out_j} = x_e^j \overline{\omega}_j, \quad \underline{f}^2_{out_j} = x_e^j \underline{\omega}_j \tag{3.12}$$

The lower weight vector $\underline{\omega}_j$ for the jth lower hyperplane and upper weight vector $\overline{\omega}_j$ for the jth upper hyperplane are initialized by allocating higher value for upper

weight vector than the lower weight vector. These vectors are updated recursively by FWGRLS method, which ensures a smooth change in weight value.

Before performing the defuzzification method, the type reduction mechanism is carried out to craft the type-reduced set - the transformation from the type-2 fuzzy variable to the type-1 fuzzy variable. One of the commonly used type-reduction methods is the Karnik Mendel (KM) procedure [167]. However, in the KM method, there is an involvement of an iterative process due to the requirement of reordering the rule consequent first in ascending order before getting the cross-over points iteratively incurring an expensive computational cost. Therefore, instead of the KM method, the q design factor [168] is utilized to orchestrate the type reduction process. The final crisp output of the type-2 PALM can be expressed as follows:

$$f^3_{out} = y_{out} = \frac{1}{2}\left(y_{l_{out}} + y_{r_{out}}\right) \tag{3.13}$$

where

$$y_{l_{out}} = \frac{\sum_{j=1}^{R} q_l \underline{f}^1_{out} \underline{f}^2_{out}}{\sum_{i=1}^{R} \overline{f}^1_{out}} + \frac{\sum_{j=1}^{R} (1-q_l) \overline{f}^1_{out} \underline{f}^2_{out}}{\sum_{i=1}^{R} \underline{f}^1_{out}} \tag{3.14}$$

$$y_{r_{out}} = \frac{\sum_{j=1}^{R} q_r \underline{f}^1_{out} \overline{f}^2_{out}}{\sum_{i=1}^{R} \overline{f}^1_{out}} + \frac{\sum_{j=1}^{R} (1-q_r) \overline{f}^1_{out} \overline{f}^2_{out}}{\sum_{i=1}^{R} \underline{f}^1_{out}} \tag{3.15}$$

where $y_{l_{out}}$ and $y_{r_{out}}$ are the left and right outputs resulted from the type reduction mechanism. q_l and q_r, utilized in Equation (3.14) and (3.15), are the design factors initialized in a way to satisfy the condition $q_l < q_r$. In this q design factor [92], the q_l and q_r steer the proportion of the upper and lower rules to the final crisp outputs $y_{l_{out}}$ and $y_{r_{out}}$ of the PALM. The normalization process of the type-2 fuzzy inference scheme [71] was modified in [83] to prevent the generation of the invalid interval. The generation of this invalid interval as a result of the normalization

process of [71] was also proved in [83]. Therefore, the normalization process as adopted in [83] is applied and advanced in terms of q_l and q_r in our work. Besides, in order to improve the performance of the proposed PALM, the q_l and q_r are not left constant, rather continuously adapted using a gradient descent technique as explained in Section 3.4. Notwithstanding that the type-2 PALM is supposed to handle uncertainty better than its type-1 variant, it incurs a higher number of network parameters in the level of $2 \times R \times (n+1)$ as a result of the use of upper and lower weight vectors $\widetilde{\omega_j} = \left[\underline{\omega}_j, \overline{\omega}_j\right]$. In addition, the implementation of q-design factor imposes extra computational cost because q_l and q_r call for a tuning procedure with the gradient descent method.

3.4 Online Learning Policy in Type-1 PALM

This section describes the online learning policy of our proposed type-1 PALM. PALM is capable of starting its learning process from scratch with an empty rule base. Its fuzzy rules can be automatically generated on the fly using the self constructive clustering (SCC) method, which checks the input and output coherence. The complexity reduction mechanism is implemented using the hyperplane merging module, which vets similarity of two hyperplanes using the distance and angle concept. The hyperplane-based fuzzy rule is adjusted using the FWGRLS method in the single-pass learning fashion.

3.4.1 Mechanism of growing rules

The rule growing mechanism of type-1 PALM is adopted from the self-constructive clustering (SSC) method [160,161] to adapt the number of rules. This method has been successfully applied to automatically generate interval-valued data clouds

[88] but its use for HPBC deserves an in-depth investigation. In this technique, the rule significance is measured by calculating the input and output coherence. The coherence is measured by analyzing the correlation between the existing data samples and the target concept. Let us assume the input vector as $X_t \in \Re^n$, target vector as $T_t \in \Re^m$, hyperplane of the ith local sub-model as $\mathcal{H}_i \in \Re^{1\times(n+1)}$. Here, n is the input vector dimension, and m is the target vector dimension. Now, the input and output coherence between $X_t \in \Re^n$ and each $\mathcal{H}_i \in \Re^{1\times(n+1)}$ are calculated as follows:

$$I_c(\mathcal{H}_i,\, X_t) = \xi(\mathcal{H}_i,\, X_t) \tag{3.16}$$

$$O_c(\mathcal{H}_i,\, X_t) = \xi(X_t, T_t) - \xi(\mathcal{H}_i, T_t) \tag{3.17}$$

where $\xi(\,)$ expresses the correlation function. There are various linear and nonlinear correlation methods for measuring correlation, which can be applied. Among them, the nonlinear methods for measuring the correlation between variables are hard to employ in the online environment since they commonly use the discretization or Parzen window method. On the other hand, Pearson correlation is a widely used method for measuring correlation between two variables. However, it suffers from some limitations: it's insensitivity to the scaling and translation of variables and sensitivity to rotation [169]. To solve these problems, a method namely maximal information compression index (MCI) is proposed in [169], which has also been utilized in the SSC method to measure the correlation $\xi(\,)$ between variables as follows:

$$\begin{aligned}\xi(X_t, T_t) = \frac{1}{2}(&\mathrm{var}(X_t) + \mathrm{var}(T_t) \\ &- \sqrt{(\mathrm{var}(X_t) + \mathrm{var}(T_t))^2 - 4\mathrm{var}(X_t)(T_t)(1-\rho(X_t, T_t)^2)})\end{aligned} \tag{3.18}$$

$$\rho(X_t, T_t) = \frac{\text{cov}(X_t, T_t)}{\sqrt{\text{var}(X_t)\text{var}(T_t)}} \tag{3.19}$$

where $\text{var}(X_t)$, $\text{var}(T_t)$ express the variance of X_t and T_t respectively, $\text{cov}(X_t, T_t)$ presents the covariance between two variables X_t and T_t, $\rho(X_t, T_t)$ stands for Pearson correlation index of X_t and T_t. In a similar way, the correlation $\xi(\mathcal{H}_i,\ X_t)$ and $\xi(\mathcal{H}_i, T_t)$ can be measured using Equation (3.18) and (3.19). In addition, the MCI method measures the compressed information when a newly observed sample is ignored. Properties of the MCI method can be expressed as follows:

1. $0 \leq \xi(X_t, T_t) \leq \frac{1}{2}(\text{var}(X_t) + \text{var}(T_t))$.
2. a maximum possible correlation is $\xi(X_t, T_t) = 0$.
3. express symmetric behavior $\xi(X_t, T_t) = \xi(T_t, X_t)$.
4. invariance against the translation of the dataset.
5. express the robustness against rotation.

$I_c(\mathcal{H}_i,\ X_t)$ is projected to explore the similarity between $\mathcal{H}_i$ and X_t directly, while $O_c(\mathcal{H}_i,\ X_t)$ is meant to examine the dissimilarity between $\mathcal{H}_i$ and X_t indirectly by utilizing the target vector as a reference. In the present hypothesis, the input and output coherence need to satisfy the following conditions to add a new rule or hyperplane:

$$I_c(\mathcal{H}_i,\ X_t) > b_1 \quad \text{and} \quad O_c(\mathcal{H}_i,\ X_t) < b_2 \tag{3.20}$$

where $b_1 \in [0.01,\ 0.1]$, and $b_2 \in [0.01,\ 0.1]$ are predetermined thresholds. If the hypothesis satisfies both the conditions of Equation (3.20), a new rule is added with the highest input coherence. Besides, the accommodated data points of a rule are updated as $N_{j^*} = N_{j^*} + 1$. Also, the correlation measure functions $\xi(\ )$

are updated with Equation (3.18) and (3.19). Due to the utilization of the local learning scenario, each rule is adapted separately and therefore covariance matrix is independent to each rule $C_j(k) \in \Re^{(n+1)\times(n+1)}$, here n is the number of inputs. When a new hyperplane is added by satisfying Equation (3.20), the hyperplane parameters and the output covariance matrix of FWGRLS method are crafted as follows:

$$\pi_{R+1} = \pi_{R^*}, \quad C_{R+1} = \Omega I \tag{3.21}$$

Due to the utilization of the local learning scenario, the consequent of the newly added rules can be assigned as the closest rule, since the expected trend in the local region can be portrayed easily from the nearest rule. The value of the correction parameter Ω in Equation (3.21) is very large (10^5). Initially, the weights (ω) have a low approximation power, therefore a large value of correction factor Ω in Equation (3.21), consequently large covariance matrix C_j helps to obtain a fast convergence of the consequent parameters to the optimal solution in the least squares sense [170, 171]. This approach includes the possibility to perform incremental training from scratch without generating an initial model. The proof of such consequent parameter setting is detailed in [171]. In addition, the covariance matrix of the individual rule has no relationship with each other. Thus, when the rules are pruned in the rule merging module, the covariance matrix, and consequent parameters are deleted as it does not affect the convergence characteristics of the C matrix and consequent of remaining rules.

3.4.2 Mechanism of merging rules

In SANFS, the rule evolution mechanism usually generates redundant rules. These unnecessary rules create complicacy in the rule base, which hinders some

desirable features of fuzzy rules: transparency and tractability in their operation. Notably, in handling data streams, two overlapping clusters or rules may easily be obtained when new samples occupied the gap between the existing two clusters. Several useful methods have been employed to merge redundant rules or clusters in [1,52,57,88]. However, all these techniques are appropriate for mainly hypersphere-based or ellipsoid-based clusters.

In the realm of hyperplane clusters, there is a possibility of generating a higher number of hyperplanes in dealing with the same data-set than spherical or ellipsoidal clusters because of the nature of HPBC in which each hyperplane represents specific operating region of the approximation curve. This opens a higher chance of generating redundant rules than HSSC and HESC. Therefore, an appropriate merging technique is vital and has to achieve trade-off between diversity of fuzzy rules and generalization power of the rule base. To understand clearly, the merging of two hyperplanes due to the new incoming training data samples is illustrated in Figure 3.2.

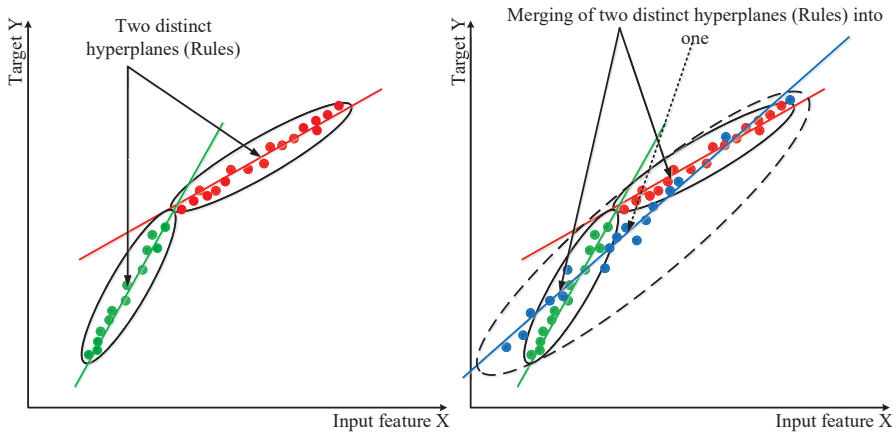


Figure 3.2: Merging of redundant hyperplanes (rules) due to new incoming training samples

In [172], to merge the hyperplanes, the similarity and dissimilarity between them

are obtained by measuring only the angle between the hyperplanes. This strategy is, however, not conclusive to decide the similarity between two hyperplanes because it solely considers the orientation of hyperplane without looking at the relationship of two hyperplanes in the target space.

In our work, to measure the similarity between the hyperplane-shaped fuzzy rules, the angle between them is estimated as follows [1, 173]:

$$\theta_{hp} = \arccos\left(\left|\frac{\omega_R^T \omega_{R+1}}{|\omega_R||\omega_{R+1}|}\right|\right) \tag{3.22}$$

where θ_{hp} is the observed angle that evolves between 0 and π radian, $\omega_R = [b_{1,R},\, b_{2,R}, ..., b_{k,R}]$, $\omega_{R+1} = [b_{1,R+1},\, b_{2,R+1}, ..., b_{k,R+1}]$.

The angle between the hyperplanes is not sufficient to decide whether the rule merging scenario should take place because it does not inform the closeness of two hyperplanes in the target space. Therefore, the spatial proximity between two hyperplanes in the hyperspace is taken into account. If we consider two hyperplanes as $l_{R1} = a_1 + xb_1$, and $l_{R2} = a_2 + xb_2$, then the minimum distance between them can be projected as follows:

$$d_{R,R+1} = \left|(a_1 - a_2).\frac{(b_1 \times b_2)}{|b_1 \times b_2|}\right| \tag{3.23}$$

For simplicity, in this chapter, the difference between weights of two consecutive rules is considered as the minimum distance between them.

The rule merging condition is formulated as follows:

$$\theta_{hp} \leq c_1 \text{ and } d_{R,R+1} \leq c_2 \tag{3.24}$$

where $c_1 \in [0.01, 0.1]$, $c_2 \in [0.001, 0.1]$ are predefined thresholds. These design

parameters are set from the empirical analysis with a variety of datasets used in this chapter. If Equation (3.24) is satisfied, fuzzy rules are merged. It is worth noting that the merging technique is only applicable in the local learning context because, in the case of global learning, the orientation and similarity of two hyperplanes have no direct correlation to their relationship.

In our merging mechanism, a dominant rule having higher support is retained, whereas a less dominant hyperplane (rule) resided by less number of samples is pruned to mitigate the structural simplification scenario of PALM. A dominant rule has a higher influence on the merged cluster because it represents the underlying data distribution. That is, the dominant rule is kept in the rule base in order for good partition of data space to be maintained and even improved. For simplicity, the weighted average strategy is adopted in merging two hyperplanes as follows:

$$\omega_{acm}^{new} = \frac{\omega_{acm}^{old} N_{acm}^{old} + \omega_{acm+1}^{old} N_{acm+1}^{old}}{N_{acm}^{old} + N_{acm+1}^{old}} \tag{3.25}$$

$$N_{acm}^{new} = N_{acm}^{old} + N_{acm+1}^{old} \tag{3.26}$$

where ω_{acm}^{old} is the output weight vector of the acmth rule, ω_{acm+1}^{old} is the output weight vector of $(acm+1)$th rule, and ω_{acm}^{new} is the output weight vector of the merged rule, N is the population of a fuzzy rule. Note that the rule acm is more influential than the rule $acm+1$, since $N_{acm} > N_{acm+1}$. The rule merging procedure is committed when no addition of rules occurs. This strategy aims to attain a stable rule evolution and prevents new rules to be merged straightaway after being introduced in the rule base. As an alternative, the Yager's participatory learning-inspired merging scenario [57] can be used to merge the two hyperplanes.

3.4.3 Adaptation of the parameters of hyperplanes

In previous work on hyperplane based T-S fuzzy system [174], recursive least square (RLS) method is employed to calculate parameters of the hyperplane. As an advancement to the RLS method, a term for decaying the consequent parameter in the cost function of the RLS method is utilized in [175] and helps to obtain a solid generalization performance - generalized recursive least square (GRLS) approach. However, their approach is formed in the context of global learning. A local learning method has some advantages over its global counterpart: interpretability and robustness over noise. The interpretability is supported by the fact that each hyperplane portrays specific operating region of approximation curve. Also, in local learning, the generation or deletion of any rule does not harm the convergence of the consequent parameters of other rules, which results in a significantly stable updating process [176].

Due to the desired features of local learning scenario, the GRLS method is extended in [1, 83]: Fuzzily Weighted Generalised Recursive Least Square (FWGRLS) method. FWGRLS can be seen also as a variation of Fuzzily Weighted Recursive Least Square (FWRLS) method [44] with the insertion of weight decay term. The FWGRLS method is formed in the proposed type-1 PALM, where the cost function can be expressed as:

$$\begin{aligned} J_{L_j}^n =&(y_t - x_e\pi_j)\Lambda_j(y_t - x_e\pi_j)+ \\ &2\beta\varphi(\pi_j) + (\pi - \pi_j)(C_j x_e)^{-1}(\pi - \pi_j) \end{aligned} \tag{3.27}$$

$$J_L^n = \sum_{j=1}^{i} J_{L_j}^n \tag{3.28}$$

where Λ_j denotes a diagonal matrix with the diagonal element of R_j, β represents

a regularization parameter, φ is a decaying factor, x_e is the extended input vector, C_j is the covariance matrix, π_j is the local subsystem of the jth hyperplane. Following the similar approach as [1], the final expression of the FWGRLS approach is formed as follows:

$$\begin{aligned}\pi_j(k) =&\pi_j(k-1) - \beta C_j(k)\nabla\varphi\pi_j(k-1)+ \\ &\Upsilon(k)(y_t(k) - x_e\pi_j(k));\ j = [1, 2, ..., R]\end{aligned} \tag{3.29}$$

where

$$C_j(k) = C_j(k-1) - \Upsilon(k)x_eC_j(k-1) \tag{3.30}$$

$$\Upsilon(k) = C_j(k-1)x_e\left(\frac{1}{\Lambda_j} + x_eC_j(k-1)x_e^T\right)^{-1} \tag{3.31}$$

with the initial conditions

$$\pi_1(1) = 0 \quad \text{and} \quad C_1(1) = \Omega I \tag{3.32}$$

where $\Upsilon(k)$ denotes the Kalman gain, R is the number of rules, $\Omega = 10^5$ is a large positive constant. The reason for using a large value is already mentioned in explaining the type-1 PALM. In this work, the regularization parameter β is assigned as an extremely small value ($\beta \approx 10^{-7}$). It can be observed that the FWGRLS method is similar to the RLS method without the term $\beta\pi_j(k)\nabla\varphi(k)$. This term steers the value of $\pi_j(k)$ even to update an insignificant amount, which helps to minimize the impact of inconsequential rules. The quadratic weight decay function chosen in PALM is written as follows:

$$\varphi(\pi_j(k-1)) = \frac{1}{2}(\pi_j(k-1))^2 \tag{3.33}$$

Its gradient can be expressed as:

$$\nabla\varphi(\pi_j(k-1)) = \pi_j(k-1) \tag{3.34}$$

By utilizing this function, the adapted-weight is shrunk to a factor proportional to its present value. It helps to intensify the generalization capability by maintaining the change of output weights into small values [177].

3.5 Online Learning Policy in Type-2 PALM

The learning policy of the type-1 PALM is extended in the context of the type-2 fuzzy system, where q design factor is utilized to carry out the type-reduction scenario. The learning mechanisms are detailed in the following subsections.

3.5.1 Mechanism of growing rules in type-2 PALM

In the realm of the type-2 fuzzy system, the SSC method has been extended to the type-2 SSC (T2SSC) in [88]. It has been adopted and extended in terms of the design factors q_l and q_r, since the original work in [88] only deals with a single design factor q. In this T2SSC method, the rule significance is measured by calculating the input and output coherence as done in the type-1 system. By assuming $\widetilde{\mathcal{H}}_i = [\overline{\mathcal{H}}_i, \underline{\mathcal{H}}_i] \in \Re^{R\times(1+n)}$ as interval-valued hyperplane of the ith local sub-model, the input and output coherence for our proposed type-2 system can be extended as follows:

$$I_{c_L}(\widetilde{\mathcal{H}}_i,\, X_t) = (1-q_l)\xi(\overline{\mathcal{H}}_i,\, X_t) + q_l\xi(\underline{\mathcal{H}}_i,\, X_t) \tag{3.35}$$

$$I_{c_R}(\widetilde{\mathcal{H}}_i,\, X_t) = (1-q_r)\xi(\overline{\mathcal{H}}_i,\, X_t) + q_r\xi(\underline{\mathcal{H}}_i,\, X_t) \tag{3.36}$$

$$I_c(\widetilde{\mathcal{H}}_i,\, X_t) = \frac{\Big(I_{c_L}(\widetilde{\mathcal{H}}_i,\, X_t) + I_{c_R}(\widetilde{\mathcal{H}}_i,\, X_t)\Big)}{2} \tag{3.37}$$

$$O_c(\widetilde{\mathcal{H}}_i,\, X_t) = \xi(X_t, T_t) - \xi(\widetilde{\mathcal{H}}_i, T_t) \tag{3.38}$$

where

$$\xi_L(\widetilde{\mathcal{H}}_i, T_t) = (1 - q_l)\xi(\overline{\mathcal{H}}_i,\, T_t) + q_l\xi(\underline{\mathcal{H}}_i,\, T_t) \tag{3.39}$$

$$\xi_R(\widetilde{\mathcal{H}}_i, T_t) = (1 - q_r)\xi(\overline{\mathcal{H}}_i,\, T_t) + q_r\xi(\underline{\mathcal{H}}_i,\, T_t) \tag{3.40}$$

$$\xi(\widetilde{\mathcal{H}}_i, T_t) = \frac{\Big(\xi_L(\widetilde{\mathcal{H}}_i, T_t) + \xi_R(\widetilde{\mathcal{H}}_i, T_t)\Big)}{2} \tag{3.41}$$

Unlike the direct calculation of input coherence $I_c(\ )$ in type-1 system, in type-2 system the $I_c(\ )$ is calculated using Equation (3.37) based on left $I_{c_L}(\ )$ and right $I_{c_R}(\ )$ input coherence. By using the MCI method in the T2SCC rule growing process, the correlation is measured using Equation (3.18) and (3.19), where (X_t, T_t) is substituted with $(\overline{\mathcal{H}}_i,\, X_t)$, $(\underline{\mathcal{H}}_i,\, X_t)$, $(\overline{\mathcal{H}}_i,\, T_t)$, $(\underline{\mathcal{H}}_i,\, T_t)$. The conditions for growing rules remain the same as expressed in Equation (3.20) and is only modified to fit the type-2 fuzzy system platform. The parameter settings for the predefined thresholds are as with the type-1 fuzzy model.

3.5.2 Mechanism of merging rules in type-2 PALM

The merging mechanism of the type-1 PALM is extended for the type-2 fuzzy model. To merge the rules, both the angle and distance between two interval-valued hyperplanes are measured as follows:

$$\widetilde{\theta}_{hp} = \arccos\left(\left|\frac{\widetilde{\omega}_R^T\widetilde{\omega}_{R+1}}{|\widetilde{\omega}_R||\widetilde{\omega}_{R+1}|}\right|\right) \tag{3.42}$$

$$\widetilde{d}_{R,R+1} = \left|(\widetilde{a}_1 - \widetilde{a}_2).\frac{(\widetilde{b}_1 \times \widetilde{b}_2)}{|\widetilde{b}_1 \times \widetilde{b}_2|}\right| \tag{3.43}$$

where $\widetilde{\theta}_{hp} = [\overline{\theta}_{hp}\ \underline{\theta}_{hp}]$, and $\widetilde{d}_{R,R+1} = [\overline{d}_{R,R+1}\ \underline{d}_{R,R+1}]$. $\widetilde{\theta}_{hp}$ and $\widetilde{d}_{R,R+1}$ also need to satisfy the condition of Equation (3.24) to merge the rules, where the same range of c_1 and c_2 are applied in the type-2 PALM. The formula of merged weight in Equation (3.25) is extended for the interval-valued merged weight as follows:

$$\widetilde{\omega}_{acm}^{new} = \frac{\widetilde{\omega}_{acm}^{old} N_{acm}^{old} + \widetilde{\omega}_{acm+1}^{old} N_{acm+1}^{old}}{N_{acm}^{old} + N_{acm+1}^{old}} \tag{3.44}$$

where $\widetilde{\omega}_{acm} = [\overline{\omega}_{acm}\ \underline{\omega}_{acm}]$. As with the type-1 PALM, the weighted average strategy is followed in the rule merging procedure of the type-2 PALM.

3.5.3 Learning of the hyperplanes' parameters in type-2 PALM

The FWGRLS method [1] is extended to adjust the upper and lower hyperplanes of the interval type-2 PALM. The final expression of the FWGRLS method is shown as follows:

$$\begin{aligned}\widetilde{\pi}_j(k) =& \widetilde{\pi}_j(k-1) - \beta\widetilde{C}_j(k)\nabla\varphi\widetilde{\pi}_j(k-1) + \\ & \widetilde{\Upsilon}(k)(y_t(k) - x_e\widetilde{\pi}_j(k));\ j = [1, 2, ..., R]\end{aligned} \tag{3.45}$$

where

$$\widetilde{C}_j(k) = \widetilde{C}_j(k-1) - \widetilde{\Upsilon}(k)x_e\widetilde{C}_j(k-1) \tag{3.46}$$

$$\widetilde{\Upsilon}(k) = \widetilde{C}_j(k-1)x_e\left(\frac{1}{\widetilde{\Lambda}_j} + x_e\widetilde{C}_j(k-1)x_e^T\right)^{-1} \tag{3.47}$$

where $\widetilde{\pi}_j = [\overline{\pi}_j \ \underline{\pi}_j]$, $\widetilde{C}_j = [\overline{C}_j \ \underline{C}_j]$, $\widetilde{\Upsilon} = [\overline{\Upsilon} \ \underline{\Upsilon}]$, and $\widetilde{\Lambda}_j = [\overline{\Lambda}_j \ \underline{\Lambda}_j]$. The quadratic weight decay function of FWGRLS method remains in the type-2 PALM to provide the weight decay effect in the rule merging scenario.

3.5.4 Adaptation of q design factors

The q design factor, as used in [83] is extended in terms of left q_l and right q_r design factors to actualize a high degree of freedom of the type-2 fuzzy model. They are initialized in such a way that the condition $q_r > q_l$ is maintained. In this adaptation process, the gradient of q_l and q_r with respect to error $E = \frac{1}{2}\left(y_d - y_{out}\right)^2$ can be expressed as follows:

$$\begin{aligned}\frac{\partial E}{\partial q_l} &= \frac{\partial E}{\partial y_{out}} \times \frac{\partial y_{out}}{\partial y_{l_{out}}} \times \frac{\partial y_{l_{out}}}{\partial q_l} \\ &= -\frac{1}{2}\left(y_d - y_{out}\right)\left(\frac{\underline{f}^1_{out}\underline{f}^2_{out}}{\sum_{i=1}^{R}\overline{f}^1_{out}} - \frac{\overline{f}^1_{out}\underline{f}^2_{out}}{\sum_{i=1}^{R}\underline{f}^1_{out}}\right)\end{aligned} \tag{3.48}$$

$$\begin{aligned}\frac{\partial E}{\partial q_r} &= \frac{\partial E}{\partial y_{out}} \times \frac{\partial y_{out}}{\partial y_{r_{out}}} \times \frac{\partial y_{r_{out}}}{\partial q_r} \\ &= -\frac{1}{2}\left(y_d - y_{out}\right)\left(\frac{\underline{f}^1_{out}\overline{f}^2_{out}}{\sum_{i=1}^{R}\overline{f}^1_{out}} - \frac{\overline{f}^1_{out}\overline{f}^2_{out}}{\sum_{i=1}^{R}\underline{f}^1_{out}}\right)\end{aligned} \tag{3.49}$$

After obtaining the gradient $\frac{\partial E}{\partial q_l}$ and $\frac{\partial E}{\partial q_r}$, the q_l and q_r are updated as follows:

$$q_l^{new} = q_l^{old} - a\frac{\partial E}{\partial q_l^{old}} \tag{3.50}$$

$$q_r^{new} = q_r^{old} - a\frac{\partial E}{\partial q_r^{old}} \tag{3.51}$$

where a is a learning rate. Based on the empirical analysis with different datasets, the value of a in this chapter is set as $a = 0.1$. Note that the learning rate is a key

of q_l and q_r convergence because it determines the step size of adjustment. An adaptive strategy as done in [80] can be implemented to shorten the convergence time without compromising the stability of the adaptation process.

3.6 Proposed Recurrent PALM Structure

In the PALM, hyperplane-shaped membership function is formulated exercising a distance ($dst(j)$) exposed in Equation (3.5). The ($dst(j)$) is calculated using true output value based on theory of the point to hyperplane distance [162]. Therefore, the PALM has a dependency on the true output in deployment phase. Usually, true outputs are not known in the deployment mode. To circumvent such structural shortcoming, the so-called *Teacher Forcing* mechanism [178] is employed in PALM. In teacher forcing technique, the network has connections from outputs to their hidden nodes at the next time step. Based on this concept, the output of PALM is connected with the input layer at the next step, which constructs a recurrent PALM (rPALM) architecture. The modified distance formula for the rPALM architecture can be expressed as:

$$dst(j) = \left| \frac{y_{out}^{k-1} - (\sum_{i=1}^{n} a_{ij} x_i + b_{0j})}{\sqrt{1 + \sum_{i=1}^{n} (a_{ij})^2}} \right| \tag{3.52}$$

where a_{ij} and b_{ij} are the ith coefficient of jth rule consequent, k is the current time step, and y_{out}^{k-1} is the true output variable from a prior time step.

In the case of type-2 rPALM configuration, the distance is articulated as follows:

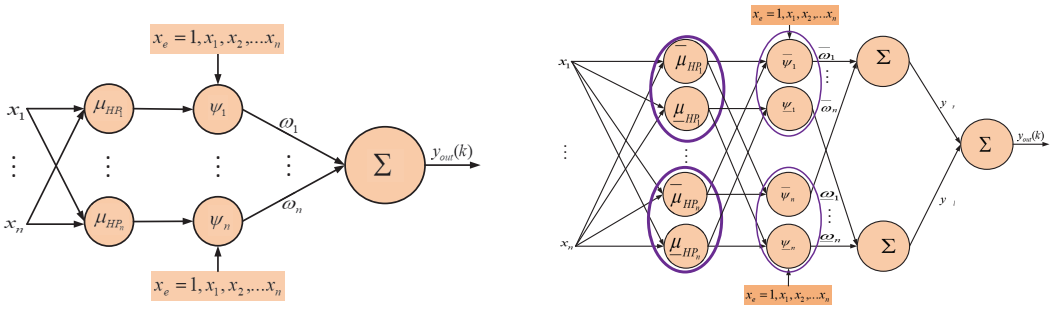
$$\widetilde{dst}(j) = \left| \frac{y_{out}^{k-1} - (\sum_{i=1}^{n} \widetilde{a}_{ij} x_i + \widetilde{b}_{0j})}{\sqrt{1 + \sum_{i=1}^{n} (\widetilde{a}_{ij})^2}} \right| \tag{3.53}$$

where $\widetilde{a}_{ij} = \left[\underline{a}_{ij}; \overline{a}_{ij}\right]$ and $\widetilde{b}_{0j} = \left[\underline{b}_{0j}; \overline{b}_{0j}\right]$ are the interval-valued coefficients of the

rule consequent of type-2 PALM, k is the current time step, and y_{out}^{k-1} is the true output variable from a prior time step.

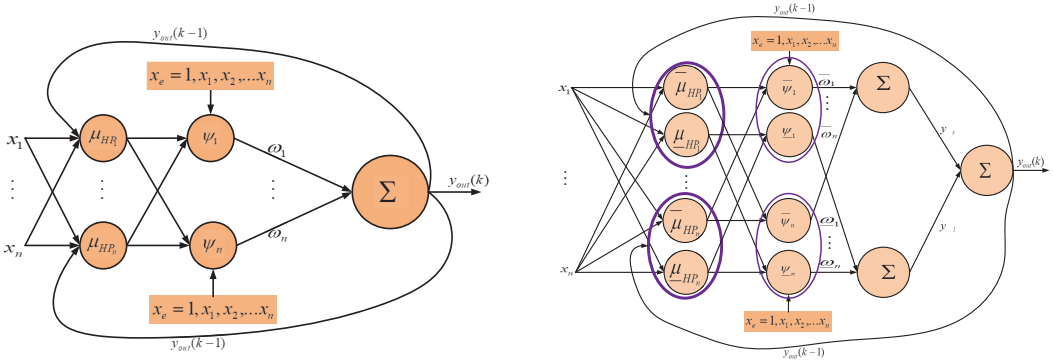
Implementation of Equation (3.52) in recurrent PALM structure solves the limitation of original PALM i.e., their reliance on true output.

To acquire a lucid conception about the network architecture of various PALMs, their high-level structures are portrayed in Figure 3.3 and Figure 3.4.



(a) (b)

Figure 3.3: Network architecture of Basic (a) Type-1 PALM, (b) Type-2 PALM



(a) (b)

Figure 3.4: Network architecture of recurrent (a) Type-1 PALM, (b) Type-2 PALM

3.7 Performance Evaluation of PALM

PALM has been evaluated through numerical studies with the use of synthetic and real-world streaming datasets. The code of PALMs and rPALMs along with these datasets have been made publicly available in [163, 179].

3.7.1 Experimental setup

3.7.1.1 Synthetic streaming datasets

Three synthetic streaming datasets are utilized in our work to evaluate the adaptive mechanism of the PALM: 1) Box-Jenkins Time Series dataset, 2) the Mackey-Glass Chaotic Time Series dataset, and 3) non-linear system identification dataset.

3.7.1.2 Box-Jenkins gas furnace time series dataset

The Box–Jenkins (BJ) gas furnace dataset is a famous benchmark problem in the literature to verify the performance of SANFSs. The objective of the BJ gas furnace problem is to model the output ($y(k)$) i.e. the CO_2 concentration from the time-delayed input ($u(k-4)$) methane flow rate and its previous output $y(k-1)$. The I/O configuration follows the standard setting in the literature as follows:

$$\widehat{y}(k) = f(u(k-4), y(k-1)) \tag{3.54}$$

This problem exposed in Equation (3.54) consists of 290 data samples where 200 samples are reserved for the training samples while the remaining 90 samples are used to test the model's generalization.

3.7.1.3 Mackey-Glass chaotic time series dataset

Mackey-Glass (MG) chaotic time series problem, having its root in [180] is a popular benchmark problem to forecast the future value of a chaotic differential delay equation by using the past values. Many researchers have used the MG dataset to evaluate their SANFSs' learning and generalization performance. This dataset is characterized by their nonlinear and chaotic behaviors where its nonlinear oscillations replicate most of the physiological processes. The MG dataset is initially proposed as a control model of the generation of white blood cells. The mathematical model is expressed as:

$$\frac{dy(k)}{dt} = \frac{by(k-\delta)}{1+y^{10}y(k-\delta)} - ay(k) \tag{3.55}$$

where $b = 0.2$, $a = 0.1$, and $\delta = 85$. The chaotic element is primarily attributed by $\delta \geq 17$. Data samples are generated through the fourth-order Range Kutta method, and our goal is to predict the system output $\widehat{y}(k+85)$ at $k = 85$ using four inputs: $y(k),\ y(k-6),\ y(k-12)$, and $y(k-18)$. This series-parallel regression model can be expressed as follows:

$$\widehat{y}(k+85) = f\left(y(k),\ y(k-6),\ y(k-12),\ y(k-18)\right) \tag{3.56}$$

For the training purpose, a total of 3000 samples between $k = 201$ and $k = 3200$ is generated with the help of the 4th-order Range-Kutta method, whereas the predictive model is tested with unseen 500 samples in the range of $k = 5001-5500$ to assess the generalization capability of the PALM.

3.7.1.4 Non-linear system identification dataset

A non-linear system identification is put forward to validate the efficacy of PALM and has frequently been used by researchers to test their SANFSs. The nonlinear dynamics of the system can be formulated by the following differential equation:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{3.57}$$

where $u(k) = \sin(2\pi k/100)$. The predicted output of the system $\widehat{y}(k+1)$ depends on the previous inputs and its own lagged outputs, which can be expressed as follows:

$$\widehat{y}(k+1) = f(y(k), y(k-1), ..., y(k-10), u(k)) \tag{3.58}$$

The first 50000 samples are employed to build our predictive model, and other 200 samples are fed to test the model's generalization ability.

3.7.1.5 Real-world streaming datasets

Three different real-world streaming datasets from two rotary wing unmanned aerial vehicle's (RUAV) experimental flight tests and a time-varying stock index forecasting data are exploited to study the performance of PALM.

3.7.1.6 Quadcopter unmanned aerial vehicle streaming data

A real-world streaming dataset is collected from a Pixhawk autopilot framework based quadcopter RUAV's experimental flight test. All experiments are performed in the indoor UAV laboratory at the University of New South Wales, Canberra campus. To record quadcopter flight data, the Robot Operating System (ROS), running under the Ubuntu 16.04 version of Linux is used. By using the ROS, a well-structured communication layer is introduced into the quadcopter

reducing the burden of having to reinvent necessary software.

During the real-time flight testing accurate vehicle position, velocity, and orientation are the required information to identify the quadcopter online. For system identification, a flight data of quadcopter's altitude containing approximately 9000 samples are recorded from VICON optical motion capture system. Among them, 60% of the samples are used for training and the remaining 40% is for testing. In this work, our model's output $y(k)$ is estimated as $\widehat{y}(k)$ from the previous point $y(k-6)$, and the system input $u(k)$, which is the required thrust to the rotors of the quadcopter. The regression model from the quadcopter data stream can be expressed as follows:

$$\widehat{y}(k) = f\left(y(k-6),\ u(k)\right) \tag{3.59}$$

3.7.1.7 Streaming data from unmanned helicopter

Another RUAV for gathering streaming dataset is a Taiwanese made Align Trex450 Pro Direct Flight Control (DFC), fly bar-less, helicopter. The high degree of non-linearity associated with the Trex450 RUAV vertical dynamics makes it challenging to build a regression model from experimental data streams. All experiments are conducted at the UAV laboratory of the UNSW Canberra campus. Flight data consists of 6000 samples collected in near hover, heave and ground effect flight conditions to simulate non-stationary environments. First 3600 samples are used for the training data, and the rest of the data are aimed to test the model. The nonlinear dependence of the unmanned helicopter is governed by the regression model as follows:

$$\widehat{y}(k+1) = f\left(y(k),\ u(k)\right) \tag{3.60}$$

where $\widehat{y}(k+1)$ is the estimated output of the helicopter system at $k = 1$.

3.7.1.8 Time-varying stock index forecasting data

Our proposed PALM has been evaluated by the time-varying dataset, namely the prediction of Standard and Poor's 500 (S&P-500 (^GSPC)) market index [181,182]. The dataset consists of sixty years of daily index values ranging from 3 January 1950 to 12 March 2009, downloaded from [183]. This problem comprises 14893 data samples. In our work, the reversed order data points of the same 60 years indexes have amalgamated with the original dataset, forming a new dataset with 29786 index values. Among them, 14893 samples are allocated to train the model and the remainder of 14893 samples are used for the validation data. The target variable is the next day S&P-500 index $y(k+1)$ predicted using previous five consecutive days indexes: $y(k),\ y(k-1),\ y(k-2),\ y(k-3)$ and $y(k-4)$. The functional relationship of the predictive model is formalized as follows:

$$\widehat{y}(k+1) = f\left(y(k),\ y(k-1),\ y(k-2),\ y(k-3)\ y(k-4)\right) \tag{3.61}$$

From Equation (3.61), it is clearly observed that 5 inputs are utilized to develop the predictive model. This dataset carries the sudden drift property which happens around 2008. This property corresponds to the economic recession in the US due to the housing crisis in 2009.

3.7.2 Results and discussion

In this work, we have developed PALM by implementing type-1 and type-2 fuzzy concept, where both of them are simulated under two parameter-optimization scenarios: 1) Type-1 PALM (L); 2) Type-1 PALM (G); 3) Type-2 PALM (L); 4)

Type-2 PALM (G). L denotes the *Local* update strategy while G stands for the *Global* learning mechanism. Basic PALM models are tested with three synthetic and three real-world streaming datasets. Furthermore, the models are compared against eight prominent variants of SANFSs, namely DFNN [166], GDFNN [184], FAOSPFNN [185], eTS [44], simp_eTS [46], GENEFIS [1], PANFIS [56], and pRVFLN [88]. Experiments with real-world and synthesis data streams are repeated with recurrent PALM. All experimental results using the rPALM are also purveyed in the supplementary document. Proposed PALMs' efficacy has been evaluated by measuring the root mean square error (RMSE), and nondimensional error index (NDEI) written as follows:

$$MSE = \frac{\sum_{k=1}^{N}(y_t - y_k)^2}{N_{T_s}},\ RMSE = \sqrt{MSE} \tag{3.62}$$

$$NDEI = \frac{RMSE}{Std(T_s)} \tag{3.63}$$

where N_{T_s} is the total number of testing samples, and $Std(T_s)$ denotes a standard deviation over all actual output values in the testing set. A comparison is produced under the same computational platform in Intel(R) Xeon(R) E5-1630 v4 CPU with a 3.70 GHz processor and 16.0 GB installed memory.

3.7.2.1 Results and discussion on synthetic streaming data- sets

Table 3.1 sums up the outcomes of the Box-Jenkins time series for all benchmark models. Among various models, our proposed type-2 PALM (G) clearly outperforms other consolidated algorithms in terms of predictive accuracy. For instance, the measured NDEI is just 0.0598 - the lowest among all models. Type-2 PALM (G) generates thirteen (13) rules to achieve this accuracy level. Although the number of generated rules is higher than that of remaining models, this

accuracy far exceeds its counterparts whose accuracy hovers around 0.29. A fair comparison is also established by utilizing very close number of rules in some benchmark strategies namely eTS, simp_eTS, PANFIS, pRVFLN, Interval-Valued Metacognitive Scaffolding Fuzzy Neural Network (RIVMcSFNN) and GENEFIS. By doing so the lowest observed NDEI among the benchmark variations is 0.29, delivered by GENEFIS. It is substantially higher than the measured NDEI of type-2 PALM (G). The advantage of HPBC is evidenced by the number of PALM's network parameters, where with thirteen rules and two inputs, PALM evolves only 39 parameters, whereas the number of network parameters of other algorithm, for instance GENEFIS is 117. PALM requires the fewest parameters than all the other variants of SANFS as well and affects positively to execution speed of PALM. On the other hand, with only one rule the NDEI of PALM is also lower than the benchmark variants as observed in type-2 PALM (L) from Table 3.1, where it requires only 3 network parameters. It is important to note that the rule merging mechanism is active in the case of only local learning scenario. Here the number of induced rules are 8 and 2, which is lower than 8 and 14 in their global learning versions. In both cases of G and L, the NDEI is very close to each other with a very similar number of rules. In short, PALM constructs a compact regression model using the Box-Jenkins time series with the least number of network parameters while producing the most reliable prediction.

The prediction of Mackey–Glass chaotic time series is challenging due to the nonlinear and chaotic behavior. Numerical results on the Mackey–Glass chaotic time series dataset is consolidated in Table 3.2, where 500 unseen samples are used to test all the models. Due to the highly nonlinear behavior, an NDEI lower than 0.2 was obtained from only GENEFIS [1] among other benchmark

Table 3.1: Modeling of the Box-Jenkins Time Series using various Self-Adaptive Neuro-Fuzzy Systems

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.7800	4.8619	1	6	200	0.0933
GDFNN	[184]	0.0617	0.3843	1	7	200	0.0964
FAOSPFNN	[185]	0.0716	0.4466	1	4	200	0.0897
eTS	[44]	0.0604	0.3763	5	30	200	0.0635
simp_eTS	[46]	0.0607	0.3782	3	18	200	1.5255
GENEFIS	[1]	0.0479	0.2988	2	18	200	0.0925
PANFIS	[56]	0.0672	0.4191	2	18	200	0.3162
pRVFLN	[88]	0.0478	0.2984	2	10	200	0.0614
RIVMcSFNN	[83]	0.0582	0.3632	2	36	200	0.3447
Type-1 PALM (L)	-	0.0484	0.3019	8	24	200	0.1972
Type-1 PALM (G)	-	0.0439	0.2739	8	24	200	0.1244
Type-2 PALM (L)	-	0.0377	**0.2355**	**2**	12	200	0.2723
Type-2 PALM (G)	-	**0.0066**	**0.0410**	14	84	200	0.3558

algorithms. However, it costs 42 rules and requires a big number (1050) of network parameters. On the contrary, with only 13 rules, 65 network parameters and faster execution, the type-2 PALM (G) attains NDEI of 0.0685, where this result is traced within 2.45 seconds due to the deployment of fewer parameters than its counterparts. The use of rule merging method in local learning mode reduces the generated rules to five (5) - type-1 PALM (L). A comparable accuracy is obtained from type-1 PALM (L) with only 5 rules and 25 network parameters. An accomplishment of such accuracy with few parameters decreases the computational complexity in predicting complex nonlinear system as witnessed from type-1 PALM (L) in Table 3.2. Due to low computational burden, the lowest execution time of 0.7771 seconds is achieved by the type-1 PALM (G).

PALM has been utilized to estimate a high-dimensional non-linear system with 50000 training samples. This study case depicts similar trend where PALM is capable of delivering comparable accuracy but with much less computational complexity and memory demand. The deployment of rule merging module lessens

Table 3.2: Modeling of the Mackey–Glass Chaotic Time Series using various Self-Adaptive Neuro-Fuzzy Systems

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	3.0531	12.0463	1	10	3000	11.1674
GDFNN	[184]	0.1520	0.6030	1	13	3000	12.1076
FAOSPFNN	[185]	0.2360	0.9314	1	6	3000	13.2213
eTS	[44]	0.0734	0.2899	48	480	3000	8.6174
simp_eTS	[46]	0.0623	0.2461	75	750	3000	20.9274
GENEFIS	[1]	0.0303	**0.1198**	42	1050	3000	4.9694
PANFIS	[56]	0.0721	0.2847	33	825	3000	4.8679
pRVFLN	[88]	0.1168	0.4615	2	18	2993	0.9236
Type-1 PALM (L)	-	0.0688	0.2718	5	25	3000	0.8316
Type-1 PALM (G)	-	0.0349	0.1380	18	90	3000	**0.7771**
Type-2 PALM (L)	-	0.0444	0.1755	11	110	3000	2.8138
Type-2 PALM (G)	-	**0.0159**	**0.0685**	13	130	3000	2.4502

the number of rules from 9 to 5 in case of type-1 PALM, and 3 from 21 in type-2 PALM. The NDEI of PALMs with such a small number of rules is also similar to other SANFS variants. To sum up, the PALM can deal with data-streaming examples with low computational burden due to the utilization of few network parameters, where it maintains a comparable or better predictive accuracy.

Table 3.3: Modeling of the non-linear system using various Self-Adaptive Neuro- Fuzzy Systems

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.0380	0.0404	2	12	50000	2149.246
GDFNN	[184]	0.0440	0.0468	2	14	50000	2355.726
FAOSPFNN	[185]	0.0027	0.0029	4	16	50000	387.7890
eTS	[44]	0.07570	0.08054	7	42	50000	108.5791
simp_eTS	[46]	0.07417	0.07892	7	42	50000	129.5552
GENEFIS	[1]	**0.00041**	**0.00043**	6	54	50000	10.9021
PANFIS	[56]	0.00264	0.00281	27	243	50000	42.4945
pRVFLN	[88]	0.06395	0.06596	2	10	49999	12.0105
Type-1 PALM (L)	-	0.08808	0.09371	5	15	50000	**9.9177**
Type-1 PALM (G)	-	0.07457	0.07804	9	27	50000	10.5712
Type-2 PALM (L)	-	0.03277	0.03487	3	18	50000	13.7455
Type-2 PALM (G)	-	0.00387	0.00412	21	126	50000	55.4865

3.7.2.2 Results and discussion on real-world data streams

Table 3.4 outlines the results of identification a quadcopter RUAV from experimental flight test data. A total 9112 samples of quadcopter's hovering test with a very high noise from motion capture technique namely VICON [186] is recorded. Building SANFS using the noisy streaming dataset is computationally expensive as seen from a high execution time of the benchmark SANFSs. Contrast with these standard SANFSs, quick execution time is seen from PALMs. It happens due to the requirement of few network parameters. Besides, PALM arrives at encouraging accuracy as well. For instance, the lowest NDEI at just 0.1538 is elicited in type-2 PALM (G). To put it plainly, due to utilizing incremental HPBC, PALM can perform better than its counterparts SANFSs driven by HSSC and HESC methods when dealing with noisy datasets.

Table 3.4: Online identification of the quadcopter utilizing various Self-Adaptive Neuro-Fuzzy Systems

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.1469	0.6925	1	6	5467	19.0962
GDFNN	[184]	0.1442	0.6800	1	7	5467	20.1737
FAOSPFNN	[185]	0.2141	1.0097	12	48	5467	25.4000
eTS	[44]	0.1361	0.6417	4	24	5467	3.0686
simp_eTS	[46]	0.1282	0.6048	4	24	5467	3.9984
GENEFIS	[1]	0.1327	0.6257	1	9	5467	1.7368
PANFIS	[56]	0.1925	0.9077	47	424	5467	6.0244
pRVFLN	[88]	0.1191	0.5223	1	5	5461	0.9485
Type-1 PALM (L)	-	0.1311	0.6182	2	6	5467	0.6605
Type-1 PALM (G)	-	0.1122	0.5290	2	6	5467	**0.5161**
Type-2 PALM (L)	-	**0.1001**	**0.4723**	3	18	5467	1.7049
Type-2 PALM (G)	-	**0.0326**	**0.1538**	4	24	5467	1.6802

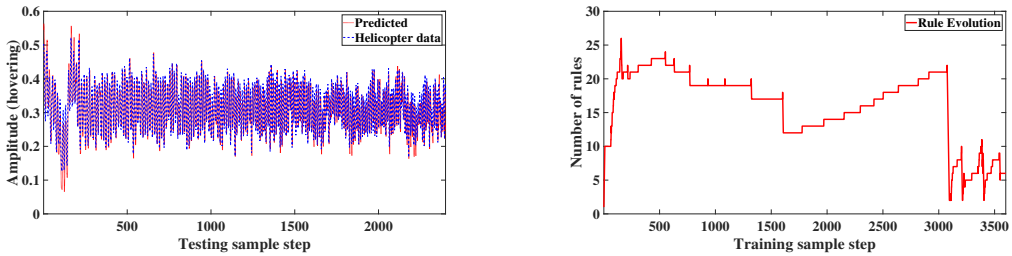
The identification of an unmanned helicopter (Trex450 Pro) from experimental flight data at hovering condition are tabulated in Table 3.5. The highest

identification accuracy with the NDEI of only 0.1380 is obtained from the proposed type-2 PALM (G) with 9 rules. As with the previous experiments, the activation of rule merging scenario reduces the fuzzy rules significantly from 11 to 6 in type-1 PALM, and from 9 to 6 in type-2 PALM. The highest accuracy is produced by type-2 PALM with only 4 rules due to most likely uncertainty handling capacity of type-2 fuzzy system.

Table 3.5: Online identification of the helicopter utilizing various Self-Adaptive Neuro-Fuzzy Systems

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.0426	0.6644	1	6	3600	8.7760
GDFNN	[184]	0.0326	0.5082	2	14	3600	11.2705
FAOSPFNN	[185]	0.0368	0.5733	2	8	3600	2.4266
eTS	[44]	0.0535	0.8352	3	18	3600	1.3822
simp_eTS	[46]	0.0534	0.8336	3	18	3600	2.3144
GENEFIS	[1]	0.0355	0.5541	2	18	3600	0.6736
PANFIS	[56]	0.0362	0.5652	9	81	3600	1.4571
pRVFLN	[88]	0.0329	0.5137	2	10	3362	1.0195
Type-1 PALM (L)	-	0.0363	0.5668	6	18	3600	0.9789
Type-1 PALM (G)	-	0.0313	0.4886	11	33	3600	0.9517
Type-2 PALM (L)	-	0.0201	0.3141	6	36	3600	2.3187
Type-2 PALM (G)	-	0.0088	**0.1380**	9	54	3600	1.9496

PALM's prediction on the helicopter's hovering dynamic and its rule evolution are depicted in Figure 3.5. These figures are produced by the type-2 PALM(L).



(a) (b)

Figure 3.5: (a) Online identification of helicopter (in hovering condition); (b) rule evolution in that identification using type-2 PALM (L)

For further clarification, the fuzzy rule extracted by type-1 PALM(L) in case of identifying helicopter can be shown as follows:

$$R^1 : \text{IF } X \text{ is close to } \Big([1, x_1, x_2]\times [0.0186, -0.0909, 0.9997]^T\Big), \text{ THEN } y_1 = 0.0186 - 0.0909x_1 + 0.9997x_2 \tag{3.64}$$

In Equation (3.64), 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 of Table V assembles an extended input vector like: $x_e^1 = [1, x_1, x_2]$. The weight vector is: $[\omega_{01}, \omega_{11}, \omega_{21}] = [0.0186, -0.0909, 0.9997]$. In case of Type-2 local learning configuration, a rule can be stated as follows:

$$\begin{aligned} R^1 : \text{IF } X \text{ is close to } \Big(&\big([1, x_1, x_2]\times [0.0787, -0.3179, 1.0281]^T\big), \big([1, x_1, x_2]\times \\ &[0.2587, -0.1767, 1.2042]^T\big)\Big) \\ \text{THEN } y_1 &= [0.0787, 0.2587] + [-0.3179, -0.1767]x_1 + \\ &[1.0281, 1.2042]x_2 \end{aligned} \tag{3.65}$$

where Equation (3.65) is expressing the first rule among 6 rules formed in that experiment in Type-2 PALM's local learning scenario. Since the PALM has no premise parameters, the antecedent part is just presenting the interval-valued hyperplanes. The consequent part is noting but $y_1 = x_e^1\widetilde{\omega}$, where $\widetilde{\omega} \in \Re^{(2(n+1))\times 1}$, n is the number of input dimension. Since 2 inputs are availed in the experiment of

Table 3.5, the extended input vector is: $x_e^1 = [1, x_1, x_2]$, and interval-valued weight vectors are: $\left[\underline{\omega_{01}},\ \overline{\omega_{01}}\right] = [0.0787, 0.2587]$; $\left[\underline{\omega_{11}},\ \overline{\omega_{11}}\right] = [-0.3179, -0.1767]$; and $\left[\underline{\omega_{21}},\ \overline{\omega_{21}}\right] = [1.0281, 1.2042]$.

The numerical results on the time-varying Stock Index Forecasting S&P-500 (ˆGSPC) problem are organized in Table 3.6. The lowest number of network parameters is obtained from PALMs, and subsequently, the fastest training speed of 2.0326 seconds is attained by type-1 PALM (L). All consolidated benchmark algorithms generate the same level of accuracy around 0.015 to 0.06.

Table 3.6: Modeling of the Time-varying Stock Index Forecasting using various Self-Adaptive Neuro-Fuzzy Systems

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.00441	0.01554	1	12	14893	347.7522
GDFNN	[184]	0.30363	1.07075	1	16	14893	344.4558
FAOSPFNN	[185]	0.20232	0.71346	1	7	14893	15.1439
eTS	[44]	0.01879	0.06629	3	36	14893	30.1606
simp_eTS	[46]	0.00602	0.02124	3	36	14893	29.4296
GENEFIS	[1]	0.00849	0.02994	3	108	14893	2.2076
PANFIS	[56]	0.00464	0.01637	8	288	14893	5.2529
pRVFLN	[88]	0.00441	0.01555	1	11	11170	2.5104
Type-1 PALM (L)	-	0.00273	0.00964	3	18	14893	**2.0326**
Type-1 PALM (G)	-	**0.00235**	**0.00832**	5	30	14893	2.2802
Type-2 PALM (L)	-	0.00442	0.01560	2	24	14893	4.0038
Type-2 PALM (G)	-	0.00421	0.01487	3	36	14893	3.9134

3.7.3 Results of using recurrent PALM

Like the original PALM, four kinds of rPALMs are developed by using local (L) and global (G) learning scenario, and type-1 and type-2 fuzzy architecture. They are namely: 1) Type-1 rPALM (L); 2) Type-1 rPALM (G); 3) Type-2 rPALM (L); 4) Type-2 rPALM (G). They are evaluated with three synthetic and three real-world streaming datasets. rPALM models are also compared

against eight prominent variants of SANFSs, namely DFNN [166], GDFNN [184], FAOSPFNN [185], eTS [44], simp_eTS [46], GENEFIS [1], PANFIS [56], and pRVFLN [88]. Tabulated information in evaluating rPALMs in each table are as follows: RMSE, NDEI, number of generated rules, number of input features, quantity of network parameters, training samples, and time of executing the experiment. Same computational platform like the PALM's experiment i.e. Intel(R) Xeon(R) E5-1630 v4 CPU with a 3.70 GHz processor and 16.0 GB installed memory is availed in rPALM's experimentation.

End results of the Box-Jenkins time series for all models are summarized in Table 3.7. In comparison with the basic PALMs, the predictive accuracy of rPALMs has dropped slightly. However, attainments of rPALMs are better/comparable with benchmark models. The lowest NDEI of nearly 0.25 is witnessed from type-2 rPALM with global learning scheme. To obtain such accuracy it generates 8 rules, slightly higher than the reference models. Enigmatically, local learning-based rPALMs generate only 3 and 2 rules with type-1 and type-2 architecture respectively, where their predictive accuracies are very close to GENEFIS and pRVFLN, and surpassing remaining benchmark models. It is noteworthy to mention that the rule merging scenario is active in local learning version of rPALMS, not in their global learning variants. It outcomes only 2 and 3 rules, which is higher (8 rules) in global learning version. In short, rPALM structures a compact regression model utilizing the Box-Jenkins time series with the least number of network parameters (when same number of rules are considered) while producing comparable prediction.

The nonlinear and chaotic behavior of Mackey–Glass chaotic time series makes it a challenging prediction problem. Numerical results with 3000 training samples

Table 3.7: Modeling of the Box-Jenkins Time Series using various Self-Adaptive Neuro-Fuzzy Systems (considering rPALM)

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.7800	4.8619	1	6	200	0.0933
GDFNN	[184]	0.0617	0.3843	1	7	200	0.0964
FAOSPFNN	[185]	0.0716	0.4466	1	4	200	0.0897
eTS	[44]	0.0604	0.3763	5	30	200	0.0635
simp_eTS	[46]	0.0607	0.3782	3	18	200	1.5255
GENEFIS	[1]	0.0479	0.2988	2	18	200	0.0925
PANFIS	[56]	0.0672	0.4191	2	18	200	0.3162
pRVFLN	[88]	0.0479	0.2984	2	10	200	0.0614
Type-1 rPALM (L)	-	0.0589	0.3672	3	9	200	0.1130
Type-1 rPALM (G)	-	0.0589	0.3672	8	24	200	0.1302
Type-2 rPALM (L)	-	0.0495	0.3088	2	12	200	0.2101
Type-2 rPALM (G)	-	0.0416	0.2594	8	48	200	0.2234

on the Mackey–Glass chaotic time series dataset are consolidated in Table 3.8, where 500 unseen samples are used to test all the models. Due to the highly nonlinear behavior, an NDEI lower than 0.2 was obtained from only GENEFIS [1]. However, it costs 42 rules and requires a big number (1050) of network parameters. In case of proposed type-1 rPALM (G) and type-2 rPALM (G), the predictive accuracy in terms of NDEI is around 0.23. Here though the G rPALMs need 54 and 49 rules for type-1 and type-2 fuzzy structure, number of required network parameters are 490 and 270, less than half of GENEFIS. On the contrary, in local learning scenario, number of induced rules in rPALM are 2 and 13 only, which requires just 15 and 130 network parameters for type-1 and type-2 variants correspondingly. Exploitation of few parameters mitigates the computational complexity, and prediction accuracy is comparable too. Due to low computational burden, the lowest execution time of 0.3472 seconds is achieved by the type-1 rPALM (L) in Table 3.8.

rPALMs are implemented to estimate a high-dimensional non-linear system

Table 3.8: Modeling of the Mackey–Glass Chaotic Time Series using various Self-Adaptive Neuro-Fuzzy Systems (considering rPALM)

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	3.0531	12.0463	1	10	3000	11.1674
GDFNN	[184]	0.1520	0.6030	1	13	3000	12.1076
FAOSPFNN	[185]	0.2360	0.9314	1	6	3000	13.2213
eTS	[44]	0.0734	0.2899	48	480	3000	8.6174
simp_eTS	[46]	0.0623	0.2461	75	750	3000	20.9274
GENEFIS	[1]	0.0303	**0.1198**	42	1050	3000	4.9694
PANFIS	[56]	0.0721	0.2847	33	825	3000	4.8679
pRVFLN	[88]	0.1168	0.4615	2	18	2993	0.9236
Type-1 rPALM (L)	-	0.1179	0.4655	2	15	3000	**0.3472**
Type-1 rPALM (G)	-	0.0631	0.2489	54	270	3000	3.8318
Type-2 rPALM (L)	-	0.1215	0.4796	13	130	3000	3.1759
Type-2 rPALM (G)	-	0.0590	0.2330	49	490	3000	6.8859

with 50000 training samples. In contrast with PALMs, prediction accuracy has been reduced slightly in rPALMS. In this study case, similar trend like PALM is depicted. rPALM can deliver comparable accuracy with less computational complexity and memory demand. Least execution time of around 6 seconds is perceived in type-1 rPALM (L), which is half of the best-performed benchmark variants GENEFIS, which takes around 10 seconds as exposed in Table 3.9. To sum up, like PALMs, rPALMS can deal with streaming examples with low computational burden due to the utilization of few network parameters, where it maintains a comparable or better predictive accuracy.

Table 3.10 outlines the results of online identification of a quadcopter RUAV from experimental flight test data with 9112 samples, which consists of noise from motion capture technique namely VICON [186]. Building SANFS using the noisy streaming dataset is computationally expensive as seen from a high execution time of the benchmark SANFSs. Contrast with these standard SANFSs, a quick execution time is seen from rPALMs. It happens due to the requirement of few

Table 3.9: Modeling of the non-linear system using various Self-Adaptive Neuro- Fuzzy Systems (considering rPALM)

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.0380	0.0404	2	12	50000	2149.246
GDFNN	[184]	0.0440	0.0468	2	14	50000	2355.726
FAOSPFNN	[185]	0.0027	0.0029	4	16	50000	387.7890
eTS	[44]	0.07570	0.08054	7	42	50000	108.5791
simp_eTS	[46]	0.07417	0.07892	7	42	50000	129.5552
GENEFIS	[1]	**0.00041**	**0.00043**	6	54	50000	10.9021
PANFIS	[56]	0.00264	0.00281	27	243	50000	42.4945
pRVFLN	[88]	0.06395	0.06596	2	10	49999	12.0105
Type-1 rPALM (L)	-	0.1299	0.1382	2	6	50000	5.6905
Type-1 rPALM (G)	-	0.1280	0.1362	7	21	50000	9.1829
Type-2 rPALM (L)	-	0.0628	0.0668	2	12	50000	10.8614
Type-2 rPALM (G)	-	0.0611	0.0650	4	24	50000	11.1630

network parameters. Besides, rPALM arrives at encouraging accuracy as well. For instance, the lowest NDEI at just 0.4995 is elicited in type-2 rPALM (G). To put it plainly, due to utilizing incremental HPBC, rPALM can perform better than its counterparts SANFSs driven by HSSC and HESC methods when dealing with noisy datasets.

Table 3.10: Online modeling of the quadcopter utilizing various Self-Adaptive Neuro-Fuzzy Systems (considering rPALM)

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.1469	0.6925	1	6	5467	19.0962
GDFNN	[184]	0.1442	0.6800	1	7	5467	20.1737
FAOSPFNN	[185]	0.2141	1.0097	12	48	5467	25.4000
eTS	[44]	0.1361	0.6417	4	24	5467	3.0686
simp_eTS	[46]	0.1282	0.6048	4	24	5467	3.9984
GENEFIS	[1]	0.1327	0.6257	1	9	5467	1.7368
PANFIS	[56]	0.1925	0.9077	47	424	5467	6.0244
pRVFLN	[88]	0.1191	0.5223	1	5	5461	0.9485
Type-1 rPALM (G)		0.1271	0.5991	3	9	5467	0.7361
Type-1 rPALM (L)	-	0.1342	0.6328	2	6	5467	0.8366
Type-2 rPALM (G)	-	0.1059	0.4995	4	24	5467	1.6376
Type-2 rPALM (L)	-	0.1089	0.5135	3	18	5467	1.8817

The identification of a unmanned helicopter (Trex450 Pro) from experimental

flight data with 6000 samples at hovering condition are tabulated in Table 3.11. The identification accuracy with the NDEI lower than 0.50 is witnessed from the proposed type-2 rPALM (G and L) with a comparable execution time. As with the previous experiments, the activation of rule merging scenario reduces the fuzzy rules significantly from 9 to 2 in type-1 rPALM, and from 11 to 4 in type-2 rPALM. In type-1 rPALMs, fast execution time less than 1 second is evidenced due to the requirement of comparatively lower network parameters, where a comparable predictive accuracy is recorded.

Table 3.11: Online identification of the helicopter utilizing various Self-Adaptive Neuro-Fuzzy Systems (considering rPALM)

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.0426	0.6644	1	6	3600	8.7760
GDFNN	[184]	0.0326	0.5082	2	14	3600	11.2705
FAOSPFNN	[185]	0.0368	0.5733	2	8	3600	2.4266
eTS	[44]	0.0535	0.8352	3	18	3600	1.3822
simp_eTS	[46]	0.0534	0.8336	3	18	3600	2.3144
GENEFIS	[1]	0.0355	0.5541	2	18	3600	0.6936
PANFIS	[56]	0.0362	0.5652	9	81	3600	1.4571
pRVFLN	[88]	0.0329	0.5137	2	10	3362	1.0195
Type-1 rPALM (G)	-	0.0534	0.8319	9	27	3600	0.9945
Type-1 rPALM (L)	-	0.0544	0.8481	2	6	3600	0.6886
Type-2 rPALM (G)	-	0.0306	0.4765	11	66	3600	2.7619
Type-2 rPALM (L)	-	0.0310	0.4828	4	24	3600	1.6364

The numerical results on the time-varying Stock Index Forecasting S&P-500 (^GSPC) problem are organized in Table 3.12. The lowest NDEI of 0.01285 is attained in type-1 rPALM (G). Though such accuracy costs 19 rules, the execution time is only 11.6796 seconds, much lower than some prominent benchmark models namely eTS with 30.1606 seconds, and simp_eTS with 29.4296 seconds. To sum up, rPALMs achieved a comparable/better accuracy with a compact structure.

Table 3.12: Modeling of the Time-varying Stock Index Forecasting using various Self-Adaptive Neuro-Fuzzy Systems (considering rPALM)

Model	Reference	RMSE using testing samples	NDEI using testing samples	Number of rules	Network Parameters	Number of training samples	Execution time (sec)
DFNN	[166]	0.00441	0.01554	1	12	14893	347.7522
GDFNN	[184]	0.30363	1.07075	1	16	14893	344.4558
FAOSPFNN	[185]	0.20232	0.71346	1	7	14893	15.1439
eTS	[44]	0.01879	0.06629	3	36	14893	30.1606
simp_eTS	[46]	0.00602	0.02124	3	36	14893	29.4296
GENEFIS	[1]	0.00849	0.02994	3	108	14893	2.2076
PANFIS	[56]	0.00464	0.01637	8	288	14893	5.2529
pRVFLN	[88]	0.00441	0.01555	1	11	11170	2.5104
Type-1 rPALM (L)	-	0.02377	0.08385	2	12	14893	15.6844
Type-1 rPALM (G)	-	0.00364	0.01285	19	114	14893	11.6796
Type-2 rPALM (G)	-	0.00430	0.01517	4	48	14893	4.4907
Type-2 rPALM (L)	-	0.00443	0.01562	2	30	14893	5.7106

An inadequacy in rPALMs is that the rules are not transparent enough to express them in human-level linguistic fuzzy rule base. However, a typical fuzzy rule for type-1 and type-2 rPALM in case of Box-Jenkins Time Series can be exemplified as follows:

$$R^1 : \text{IF } X \text{ is close to } \Big([1, x_1, x_2]\times [1.1981, -1.0649, -0.2889]^T\Big), \text{ THEN } y_1 = 1.1981 - 1.0649x_1 - 0.2889x_2 \tag{3.66}$$

In Equation (3.66), the antecedent part is manifesting the hyperplane with recurrent connection. 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 of Table I assembles an extended input vector like: $x_e^1 = [1, x_1, x_2]$. The weight vector is: $[\omega_{01}, \omega_{11}, \omega_{21}] = [1.1981, -1.0649, -0.2889]$. In case of Type-2 local learning

configuration, a rule can be stated as follows:

$$
\begin{aligned}
&R^1 : \text{IF } X \text{ is close to } \Big(\big([1, x_1, x_2]\times \\
&[0.1533, -0.6178, 0.3445]^T\big), \big([1, x_1, x_2]\times \\
&[0.5598, -0.4119, 0.5938]^T\big) \Big) \\
&\text{THEN } y_1 = [0.1533, 0.5598] + [-0.6178, -0.4119]x_1 + \\
&[0.3445, 0.5938]x_2
\end{aligned} \tag{3.67}
$$

where Equation (3.67) is expressing the first rule among 6 rules formed in that experiment in Type-2 rPALM's local learning scenario. Since the rPALM has no premise parameters, the antecedent part is just presenting recurrent connection based the interval-valued hyperplanes. The consequent part is noting but $y_1 = x_e^1\widetilde{\omega}$, where $\widetilde{\omega} \in \Re^{(2(n+1))\times 1}$, n is the number of input dimension. Since 2 inputs are availed in the experiment of Table I, the extended input vector is: $x_e^1 = [1, x_1, x_2]$, and interval-valued weight vectors are: $\left[\underline{\omega_{01}},\ \overline{\omega_{01}}\right] = [0.0787, 0.2587]$; $\left[\underline{\omega_{11}},\ \overline{\omega_{11}}\right] = [-0.3179, -0.1767]$; $\left[\underline{\omega_{21}},\ \overline{\omega_{21}}\right] = [1.028, 1.204]$.

3.7.4 Sensitivity analysis of predefined thresholds

In the rule growing purpose, two predefined thresholds (b_1 and b_2) are utilized in our work. During various experimentation, it has been observed that the higher the value of b_1, the less the number of hyperplanes are added and vice versa. Unlike the effect of b_1, in case of b_2, at higher values, more hyperplanes are added and vice versa. To further validate this feature, the sensitivity of b_1 and b_2 is evaluated using the Box–Jenkins (BJ) gas furnace dataset. The same I/O relationship as described in the subsection 3.7.1 is applied here, where the model

is trained also with the same 200 samples and the remaining 90 unseen samples are used to test the model.

In the first test, b_2 is varied in the range of $[0.052, 0.053, 0.054, 0.055]$, while the value of b_1 is kept fixed at 0.020. On the other hand, the varied range for b_1 is $[0.020, 0.022, 0.024, 0.026]$, while b_2 is maintained at 0.055. In the second test, the altering range for b_1 is $[0.031, 0.033, 0.035, 0.037]$ and for b_2 is $[0.044, 0.046, 0.048, 0.050]$. In this test, for a varying b_1, the constant value of b_2 is 0.050, where b_1 is fixed at 0.035 during the change of b_2. To evaluate the sensitivity of these thresholds, normalized RMSE (NRMSE), NDEI, running time, and number of rules are reported in Table 3.13. The NRMSE formula can be expressed as: $NRMSE = \sqrt{\frac{MSE}{Std(T_s)}}$.

Table 3.13: Sensitivity Analysis of Rule growing thresholds

Parameters	NRMSE	NDEI	Execution time	#Rules
$b_2 = 0.055$	0.023	0.059	0.355	13
$b_2 = 0.054$	0.023	0.059	0.312	13
$b_2 = 0.053$	0.023	0.059	0.326	13
$b_2 = 0.052$	0.023	0.059	0.325	13
$b_1 = 0.020$	0.023	0.059	0.324	13
$1_2 = 0.022$	0.023	0.059	0.325	13
$b_1 = 0.024$	0.023	0.059	0.320	13
$b_1 = 0.026$	0.023	0.059	0.344	13
$b_1 = 0.037$	0.046	0.115	0.260	10
$b_1 = 0.035$	0.046	0.115	0.259	11
$b_1 = 0.033$	0.046	0.115	0.269	11
$b_1 = 0.031$	0.048	0.121	0.269	11
$b_2 = 0.050$	0.047	0.118	0.265	11
$b_2 = 0.048$	0.046	0.115	0.267	11
$b_2 = 0.046$	0.047	0.116	0.266	11
$b_2 = 0.044$	0.047	0.117	0.306	11

From Table 3.13, it has been observed that in the first test for different values of b_1 andb_2, the value of NRMSE and NDEI remains stable at 0.023 and 0.059 respectively. The execution time varies in a stable range of $[0.31, 0.35]$ seconds and

the number of generated rules is 13. In the second test, the NRMSE, NDEI, and execution time are relatively constant in the range of $[0.046, 0.048]$, $[0.115, 0.121]$, $[0.26, 0.31]$ correspondingly. The value of b_1 increases and b_2 reduces compared to test 1, and few rules are generated across different experiments of this chapter.

3.8 Summary

An advanced autonomous learning algorithm, namely PALM, is proposed in this chapter for data stream regression and modeling nonlinear dynamical systems like quadcopter and helicopter. The PALM is developed with the concept of HPBC which incurs very low network parameters. The reduction of network parameters bring down the execution times because only the output weight vector calls for the tuning scenario without compromise on predictive accuracy. PALM possesses a highly adaptive rule base where its fuzzy rules can be automatically added when necessary based on the SCC theory. It implements the rule merging scenario for complexity reduction and the concept of distance and angle is introduced to coalesce similar rules. The efficiency of the PALM has been tested in six real-world and artificial data stream regression problems where PALM outperforms recently published works in terms of network parameters and running time. It also delivers state-of-the art accuracies which happen to be comparable and often better than its counterparts. In the next chapter, PALM is utilized to develop self-adaptive controllers.

Chapter 4

PALM-based Autonomous Intelligent Controllers for Micro Aerial Vehicles

The work presented in this chapter has been published/submitted in the following articles:

- **Ferdaus, M. M.,** Pratama, M., Anavatti, S. G., Garratt, M. A. (2019). PAC: A Novel Self-Adaptive Neuro-Fuzzy Controller for Micro Aerial Vehicles, *Information Sciences, (Q1) Impact Factor: 5.524*, doi.org/10.1016/j.ins.2019.10.001.
- **Ferdaus, M. M.,** Hady, M. A., Pratama, M., Anavatti, S. G., Kandath, H., (2019). RedPAC: A Simple Evolving Neuro-Fuzzy-based Intelligent Control Framework for Autonomous Aerial Vehicles. *In 2019 IEEE International Conference on Fuzzy Systems.*

Abstract

In this chapter, evolving neuro-fuzzy system-based two Autonomous Intelligent Controllers (AICons), namely Parsimonious Controller (PAC) and Reduced Parsimonious Controller (RedPAC) are proposed. Both of them feature fewer network parameters than conventional approaches due to the absence of rule premise parameters. In contrast with PAC, the number of consequent parameters has further reduced to one parameter per rule in RedPAC. Both PAC and RedPAC are built upon PALM developed in chapter 3. In contrast with PALM, in both RedPAC and PAC, new rule growing and pruning modules are derived from the concept of bias and variance. These methods have no reliance on user-defined thresholds, thereby increasing their autonomy for real-time deployment. They adapt the consequent parameters by using the sliding mode control (SMC) theory in the single-pass fashion. The boundedness and convergence of the closed-loop control system's tracking error and the controller's consequent parameters are confirmed by utilizing the LaSalle-Yoshizawa theorem. The PAC's efficacy is evaluated by observing various trajectory tracking performance from a bio-inspired Flapping Wing Micro Aerial Vehicle (BI-FWMAV) and a rotary wing UAV called hexacopter. Lastly, RedPAC's performance has been evaluated by implementing it to control a quadcopter simulator, namely Dronekit.

4.1 Introduction

In recent times, massive applicability of Micro Aerial Vehicles (MAVs) is witnessed in both civilian [114] and military sectors [113]. In MAVs, a major concern to pursue is the preferable control autonomy. To stabilize and control the MAVs, First Principle Techniques (FPTs) are used commonly. Among

numerous FPTs, simple controllers like Proportional Integral Derivative (PID), Linear Quadratic (LQ) [116], nonlinear control techniques namely Backstepping, Sliding Mode techniques, Feedback Linearization (FBL), H$-\infty$ methods are employed successfully in different MAVs. Performance of all these controllers depends upon the preciseness of the mathematical model of MAVs. In MAVs, encountering environmental disruptions like wind gust in open space, actuator degradation, etc. are usual. Integration of all these factors into the mathematical model of MAVs is laborious or inconceivable. It yields imprecise models of MAVs, and consequently, all the FPT-based controllers' performance degrades. These deficiencies of FPT-based controllers tempt research towards mathematical model-free intelligent control methods.

Among numerous intelligent control approaches, the Fuzzy Logic Controller (FLC), Neural Network (NN), and Neuro-Fuzzy (NF) controllers are employed in diverse engineering industries [187, 188]. Majority of these controllers consist of a fixed architecture with a definite number of neurons or membership functions, rules or layers. To improve the performance of model-free static controllers, researchers have developed adaptive controllers by combining conventional nonlinear control techniques such as backstepping [117], sliding mode techniques [137], feedback linearization (FBL), H∞ [138], etc. with FLS, NN, or NF structure. Such mixture provides a mode-free robust and adaptive control scheme. In these control schemes, only the network parameters are altered, where they are maintaining a fixed structure. It enforces us to specify the number of nodes, layers, or rules beforehand. It is hard to know the exact number of rules a priori to attain desired control performance. Controllers with only a few rules may fail to produce the desired performance, whereas too many of them may originate an over-complex structure of controllers to actualize in real-time. To

circumvent such shortcomings, NN or NF controller with flexible architecture can be employed. These flexible controllers are not only able to tune parameters, but also to evolve the structure by adding or deleting layers or rules in self-adaptive fashion, which make them fully AICons.

Usually, the AICons are associated with several rule premise parameters like mean and width of hyper-spherical or hyper-ellipsoidal clusters. These parameters need to be updated continuously, which rises computational complexities. To epitomize, a bottleneck of AICons is the engagement of manifold parameters, which strike adversely in furnishing a fast response. To mitigate the computational complexity of the above discussed AICons, hyper-plane-shaped clustering techniques could be a promising avenue since they are absolutely free from premise parameters.

4.2 Contributions

In this chapter, two AICons, namely PAC and RedPAC, are developed. Both of them are rooted with a HPSC technique-based evolving NF architecture namely parsimonious learning machine (PALM) architecture. However, the complex rule-evolution mechanism of PALM has been replaced with the concept of bias-variance trade-off in developing the controllers. Main features of the PAC and RedPAC can be uttered as follows:

1. ***Premise-free fuzzy rule base system:*** Usually fuzzy logic controllers' have a rule base that consists of antecedent and consequent parts, where both parts are associated with a number of parameters. Unlike the conventional FLC, the fuzzy rules in both PAC and RedPAC are portrayed by hyper-planes. These hyper-planes corroborate both the rule premise and

consequent parts. As a result, they do not have any premise parameter. Such scheme trims the rule-based parameters to the level of $R \times (N+1)$ for PAC, where N denotes the number of input dimension and R is the number of fuzzy rules. In contrast with PAC, the RedPAC's learning parameters have been further reduced. With R rules and N inputs to RedPAC, it requires to adapt only R parameters.

2. ***New rule growing and pruning mechanism based on bias and variance:*** In PALM [189], self-constructing clustering approach [160,161] was employed to generate rules, which faces computational complexities to calculate variance and covariance among different variables. Besides, similarity analysis among hyper-planes in terms of distance and orientation between hyper-planes were measured to merge rules. To eliminate such a complex calculation of growing and pruning rules, bias-variance concept based simplified method, namely network significance is proposed in PAC and RedPAC. This concept is derived from the idea of network significance [190] in estimating the network's bias and variance. A new rule is added in the underfitting situation, while the pruning process is triggered by the overfitting case. The key difference of this chapter from [190] lies in the estimation of bias and variance for the hyperplane-shaped hidden unit.

3. ***New evolving fuzzy controller:*** In general, the evolving fuzzy controller' rule evolution methods have a reliance on a number of predefined thresholds. To eliminate such dependency on user-defined thresholds, the bias-variance concept based network significance method is exercised in the developed controllers, where they do not need any user-defined problem specific thresholds. Since the proposed controller has no premise param-

eters, its only consequent parameter, i.e. weights are adapted by using SMC learning theory to confirm a stable closed-loop system. To evaluate the controllers' stable and precise tracking performance, they have been implanted into different simulated MAV's plant.

Aforementioned features of the PAC and RedPAC are desiring to achieve desired control autonomy in MAVs. Therefore, the developed PAC is applied to control BI-FWMAV and hexacopter, and RedPAC to control a quadcopter. In addition, the whole code of the PAC is written in C programming language. It is compatible with the majority of the MAVs hardware, where its implementation is made accessible publicly in [191].

Arrangement of the remaining sections of this chapter is as follows: Challenges in formulating the FW MAV, hexacopter, and quadcopter's plant model are asserted in Section 4.3. In Section 4.4, limitations of existing evolving controllers are analyzed. Section 4.5 details the network structures of the PALM based evolving controller PAC along with the explanation on rule generation and pruning mechanism. Experimental results and performance evaluation of the proposed controller are described in Section 4.6. At last, the chapter terminates with concluding remarks in Section 4.7.

4.3 Plant Dynamics of MAVs

Three different plants are engaged in this chapter to evaluate the proposed controllers' performance. Among them, the hexacopter plant is developed at the Unmanned Aerial Vehicle (UAV) laboratory of the UNSW Canberra, the BI-FWMAV plant is inspired by the work of [192, 193], and the quadcopter simulator is developed at the Computational Intelligence Lab (CIL) of the Nanyang

Technological University Singapore. In this section, we are initiating with hexacopter's nonlinear complex plant dynamics.

4.3.1 Dynamics of hexacopter plant and associated complexities

Unlike the conventional hexacopter plant with 6 degrees of freedom (DOF) rigid body dynamics, an 8 DOF over-actuated hexacopter plant with medium fidelity is considered in this chapter. Two surplus DOF are two moving masses. The masses can slide along their own rail aligned in lengthwise and sideways consecutively. To get a synopsis about the simulated Hexacopter plant, its high-level diagram is shown in Figure 4.1. The roll and pitch command to the "control mixing" block of Figure 4.1 is driven by "attitude controller". In attitude control mechanism, the inner loop is controlled by a linear PID controller and outer loop is governed by PAC. The thrust command of "control mixing" is geared by PAC based height or position controller. Moving of mass to the longitudinal direction shifts the Center of Gravity (CoG) to X-axis, which is denoted by CG_X in Figure 4.1, and CG_Y is expressing the shift of CoG to Y-axis owing to the movement of mass to the lateral direction. Both the movements CG_X and CG_Y are supervised by the PAC. In control mixing block, a simple linear mixing composition is utilized to convert the roll, pitch, yaw, and thrust commands to the required speed of motors. These signals are availed to calculate the craved thrust and torque of individual rotors based on the relative airflow faced by each of them and the commanded motor speed. Afterward, the total vertical force and yawing torque of the plant are quantified by summing up the thrust and torque of individual rotors. The product of thrust to a single rotor and moment arm yields the rolling and pitching moments acting on the hexacopter. Finally, the controlled thrust

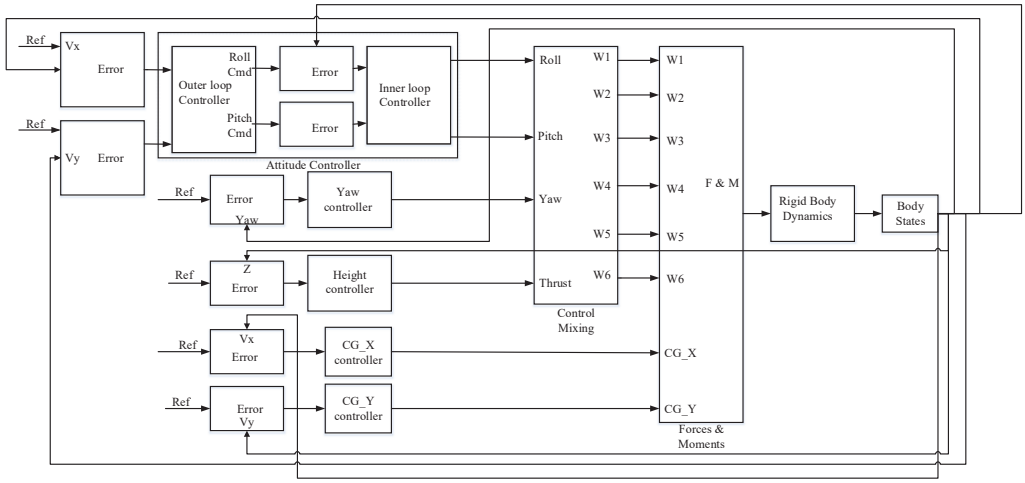


Figure 4.1: High-level presentation of the over-actuated simulated Hexacopter plant diagram

along with the yawing torque, and rolling and pitching moments are fed to the rigid body dynamics to update the body state accordingly. In upcoming paragraphs, hexacopter's nonlinear aerodynamics along with associated complexities are discoursed.

In hexacopter, body axes system is fixed to its CoG. Therefore, variations in hexacopter attitude cause rotation to the body axes. These axes play an important role in crafting the desired control signal as the sensors are fixed to the body axes. With regard to the earth surface, an additional set of three-dimensional axes (X-axis: horizontal and pointing to the north; Y-axis: horizontal and pointing to the east; and Z-axis: positive down towards the CoG), namely inertial axes are also considered in developing the nonlinear dynamic model. Now, the translational velocities along body axes are designated as u, v, and w. Similarly, body axes rotational rates are expressed as p, q, and r, right-hand axes system is considered. To define the proper orientation of the hexacopter, it is essential to specify the coordination system around which to assign rotations. Besides,

the sequence in which they employed is equally significant. In this work, a conventional aviation measure, namely Euler angles (roll: ϕ, pitch: θ; yaw: ψ) are utilized to depict the orientation of the hexacopter with respect to inertial axes. Four simplified quaternion parameters (q_0, q_1, q_2, q_3) are exerted to refrain from wraparound effect and to linearize the attitude. In addition, attitude values are stored in those parameters and converted into required Euler angles as expressed in Equation (4.1).

$$\begin{aligned} q_0 &= \cos\tfrac{\phi}{2}\cos\tfrac{\theta}{2}\cos\tfrac{\psi}{2} + \sin\tfrac{\phi}{2}\sin\tfrac{\theta}{2}\sin\tfrac{\psi}{2} \\ q_1 &= \sin\tfrac{\phi}{2}\cos\tfrac{\theta}{2}\cos\tfrac{\psi}{2} - \cos\tfrac{\phi}{2}\sin\tfrac{\theta}{2}\sin\tfrac{\psi}{2} \\ q_2 &= \cos\tfrac{\phi}{2}\sin\tfrac{\theta}{2}\cos\tfrac{\psi}{2} + \sin\tfrac{\phi}{2}\cos\tfrac{\theta}{2}\sin\tfrac{\psi}{2} \\ q_3 &= \cos\tfrac{\phi}{2}\cos\tfrac{\theta}{2}\sin\tfrac{\psi}{2} - \sin\tfrac{\phi}{2}\sin\tfrac{\theta}{2}\cos\tfrac{\psi}{2} \end{aligned} \tag{4.1}$$

4.3.1.1 Complexity in hexacopter's aerodynamics

When flying, rotors of hexacopter face relative air-stream velocity resulting from their own motion V_∞. The air-stream deflects through the rotor disc at a velocity of V_i, which alters the down-stream flow by $2V_i$. At each rotor disc, V_i can be segregated perpendicularly (V_n), and tangentially (V_t). Again, V_n is determined by summing up with the perpendicular component of V_∞, whereas, V_t is computed by adding the tangential part of V_∞. It is noteworthy that V_∞ is executed when the individual rotor is experiencing airflow during pitching, rolling, and yawing motions. To subdue the complexity of hexacopter's aerodynamics, an uniform inflow is assumed in our work. Thus, the inflow V_i remains the same at various disc radius or azimuth. After such simplification, the elemental forces can be integrated to attain a closed-form solution for thrust as expressed in Equation (4.2). Interested readers are referred to [194] for a detailed derivation of Equation

(4.2).

$$T = \frac{\rho a(\Omega R)^2 A_b}{2}\left[\frac{1}{3}\theta_0\left(1+\frac{3}{2}\mu^2\right)-\frac{1}{2}\lambda^{'}\right] \tag{4.2}$$

where θ_0 denotes blade pitch, inflow relative to the rotor disk is presented by $\lambda^{'}$, A_b is the area of the blade, R is the blade radius, a is the lift curve slope, ρ is the absolute air-density μ is expressing an advance ratio, Ω is rotational speed of rotor blade in radians per sec. $\lambda^{'}$ can be expressed as follows:

$$\lambda^{'} = \frac{V_i + V_n}{\Omega R} \text{ and } \mu = \frac{V_t}{\Omega R} \tag{4.3}$$

Based on Glauert's induced flow model [195], the mean generated velocity V_i is exposed as follows:

$$V_i = \frac{T}{2\rho A\hat{V}} \text{ where } \hat{V} = \sqrt{V_T^2 + (V_n + V_i)^2} \tag{4.4}$$

where $A = \pi R^2$. Equation (4.4) can be rearranged as follows:

$$V_i^2 = \sqrt{\left(\frac{\hat{V}}{2}\right)^2 + \left(\frac{T}{2\rho A}\right)^2} - \frac{\hat{V}^2}{2} \tag{4.5}$$

In our work, the momentum theory and blade element theory are combined through Glauert's simple inflow model, which outcomes two coupled nonlinear Equation (4.2) and (4.5). A straightforward binary search algorithm is utilized to solve them numerically and to acquire V_i by assuming a monotonic variation of thrust with V_i.

The drag confronted by the rotor blades through air results the yawing torque N. It is formulated by dividing the total rotor power P_{tot} by rotational speed of

rotor blade as follows:

$$N = \frac{P_{tot}}{\Omega} \tag{4.6}$$

The total power P_{tot} is an amalgamation of induced power P_{ind} and profile power P_0, expressed as $P_{tot} = P_{ind} + P_0$. Induced power P_{ind} is the power needed to yield induced velocity V_i and to overcome gravitational force, whereas P_0 is the power to beat the profile drag of the rotor blades. These powers can be articulated as follows:

$$P_{ind} = k_{ind} T V_i + T V_c \tag{4.7}$$

$$P_0 = \frac{\sigma C_{D_0}}{8}(1 + \kappa\mu^2) \tag{4.8}$$

In Equation (4.7), k_{ind} is a correlation factor, which is imported to atone the non-uniformly produced velocity and tip loss effects. On the other hand, κ in Equation (4.8) is to rectify the skewed flow and other detrimental effects in the forward flight, V_c is presenting the climbing speed achieved by the rotor thrust in a steady climb, TV_i is the rate of work accomplished on the air and portrays the kinetic energy of the rotor downwash [194]. To reduce the obscurity in our plant dynamics, impacts of vortex ring state in sharp descent are scorned.

4.3.1.2 Rigid body dynamics of hexacopter plant

In rigid body dynamics of our hexacopter plant, Newton's second law of motion is exercised to formulate correlations between the forces and moments acting on the hexacopter and translational and rotational accelerations. Here the hexacopter plant is considered as of a traditional mass distribution, where the xz plane is generally the plane of symmetry. Such consideration makes the cross product of moments of inertia in yz and xy plane zeros i.e. $I_{yz} = I_{xy} = 0$. After this

simplified implementation, the equations are exposed in Equation (4.9). For further clarifications, readers can go through [196], where equations in (4.9) are derived elaborately.

$$\begin{aligned} F_x &= m(\dot{u} + qw - rv) \\ F_y &= m(\dot{v} + ru - pw) \\ F_z &= m(\dot{w} + pv - qu) \\ L &= I_x\dot{p} - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq \\ M &= I_y\dot{q} + rp(I_x - I_z) + I_{xz}(p^2 - r^2) \\ N &= -I_{xz}\dot{p} + I_z\dot{r} + pq(I_y - I_x) + I_{xz}qr \end{aligned} \tag{4.9}$$

where

$$\begin{aligned} I_x &= \int\int\int (y^2 + z^2)dm \\ I_y &= \int\int\int (x^2 + z^2)dm \\ I_z &= \int\int\int (x^2 + y^2)dm \\ I_{xy} &= \int\int\int xydm \\ I_{xz} &= \int\int\int xzdm \\ I_{yz} &= \int\int\int yzdm \end{aligned} \tag{4.10}$$

where m is the body mass in kg; I_x, I_y, I_z are hexacopter's mass moments of inertia with regard to x, y, and z-axis respectively in kgm^2; I_{xz} is the product of inertia. In our simulated plant, the practiced values of the above parameters are as: $m = 3kg$, $I_x = 0.04kgm^2$, $I_y = 0.04kgm^2$, $I_z = 0.06kgm^2$, $I_{xz} = 0kgm^2$.

The robustness of the hexacopter plant is maintained by storing and updating the attitude as a quaternion as exposed in Equation (4.11). Equation (4.11) is explained clearly in [197]. The realization of quaternion to update attitude eliminates the usage of trigonometric functions which would be required if we

were integrating the Euler angle based differential equations.

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \tag{4.11}$$

At this stage, the position of the hexacopter's rigid body states is updated in global coordinates relative to the inertial axes. Firstly, a conversion is required from local velocities u, v and w to global velocities $\dot{X}$, $\dot{Y}$ and $\dot{Z}$. The conversion is executed in Equation (4.12), where the local velocities are multiplied by the rotation matrix B to obtain the globals.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = B \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.12}$$

where the rotation matrix B is determined directly from the quaternions using Equation (4.13).

$$B = \begin{bmatrix} q_0^2 + q_1^2\text{-}q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_2 + q_0q_3) \\ 2(q_1q_2 - q_0q_3) & q_0^2 + q_1^2\text{-}q_2^2\text{-}q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_1q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{4.13}$$

These velocities are then integrated to obtain the global position $[X, Y, Z]$. In our simulated hexacopter plant, Equation (4.9) is implemented as a C code SIMULINK S-function. The states for our hexacopter's rigid body dynamics block are position, local velocity components in the body axes system, rotation rates, and quaternion attitude. In our experiments, the states are being updated

by a simple trapezoidal integration scheme executing at 200 updates per second. Incoming signals to the block are the forces and moments acting on the hexacopter while the outcomes are accelerations, local velocities, position, body angular rates, and attitude. A user-friendly graphical mask for the dynamics block allows users to alter the mass, moments of inertia and initial states on demand. During experimentation, errors are obtained by measuring the difference from actual to reference altitude and attitude. These errors, their derivatives, and actual values are intakes for our proposed PAC, which is free from any plant parameters. Therefore, PAC can perform in a complete model-free manner. Finally, it is crucial to note that, after manifold simplification, the hexacopter plant is still highly nonlinear, complex with numerous parameters. Efficient controlling of such plant is difficult for the model-based nonlinear controller, or model-free but the parameter-dependent conventional AICon, whereas our proposed AICon, namely PAC exposes an improved control performance.

4.3.2 Dynamics of BI-FWMAV plant

Dynamics of the BI-FWMAV plant is highly nonlinear and expresses higher complexity than the hexacopter. It is mainly due to its lightweight and smaller size. The simulated BI-FWMAV plant saves the time and expenses to set-up the experimental flight test, which is inspired by the work of [192, 193]. The top-level diagram of the simulated BI-FWMAV plant is shown in Figure 4.2.

Four wings of the BI-FWMAV are operated by four actuators as exhibited in Figure 4.2. From the analysis on dragonfly flight in [198, 199], the influence of seven different flapping parameters on the wings and actuators as well, are

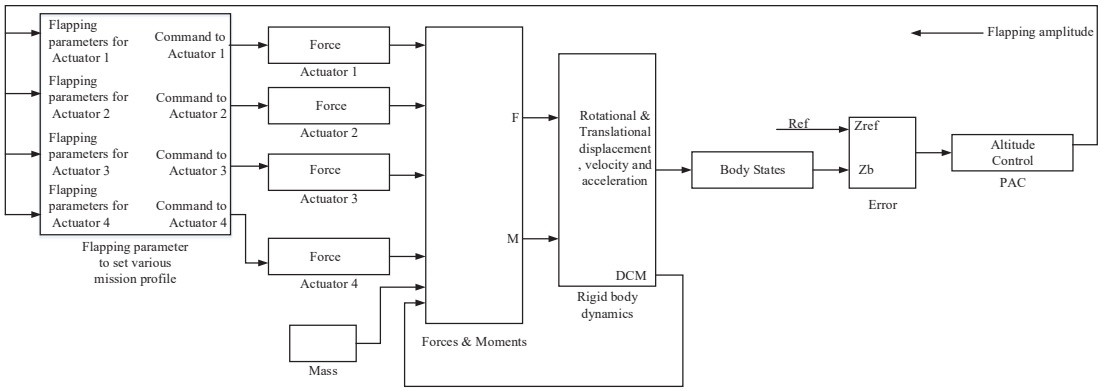


Figure 4.2: Top-level framework of the BI-FWMAV plant

considered in developing the BI-FWMAV plant. These parameters are namely stroke plane angle (Fp_{spa})(in rad) , flapping frequency (Fp_{ff}) (in Hz), flapping amplitude (Fp_{fa}) (in rad), mean angle of attack (Fp_{aoa}) (in rad), amplitude of pitching oscillation (Fp_{po}) (in rad), phase difference between the pitching and plunging motion (Fp_{pd}), and time step (Fp_{ts}) (in sec). Exploring a variety of combinations of these parameters, the BI-FWMAV can carry out take-off, rolling, pitching, and yawing, as explained in [200]. Since our proposed controller is used to regulate the altitude of the BI-FWMAV, it is essential to know the dominant parameter in determining the altitude. After a successful parametric analysis, the flapping amplitude has turned out to be the dominant one in altitude tracking. Individual forces and moments of actuators are combined to provide the demanded force and moment to the rigid body dynamics based on the relative airflow acting on each wing and the commanded actuator speed. The combined force utilized in our work can be formulated as follows:

$$F_T = F_{a_1} + F_{a_2} + F_{a_3} + F_{a_4} + (mg \times DCM) \tag{4.14}$$

where m is the mass, g is the acceleration due to gravity, $\times$ is expressing a matrix

multiplication, the matrix dimension of $mg = [0\ 0\ mg]$ is (3×1), DCM is the direction cosine matrix with a dimension of (3×3), F_{a_i} (where $i = 1, 2, 3, 4$) is the force provided by the individual actuator. Similarly, the total moment necessary for the rigid body can be demonstrated as follows:

$$M_T = M_{a_1} + M_{a_2} + M_{a_3} + M_{a_4} \tag{4.15}$$

where M_{a_i} (where $i = 1, 2, 3, 4$) is presenting the individual momentum of each wing and can be articulated as:

$$M_{a_i} = F_{a_i} \times (CG - CP_i) \tag{4.16}$$

where $i = 1, 2, 3, 4$; $CG = [0\ 0\ 0]$; and $CP_1 = [0.08\ 0.05\ 0]$; $CP_2 = [0.08\ 0.05\ 0]$; $CP_3 = [0.08\ \ -0.05\ 0]$; $CP_4 = [-0.08\ \ -0.05\ 0]$; and $'\times'$ is presenting (3×3) cross product. Finally, the accumulated force and moment are transformed into the body coordinate system, and all the required body states like three dimensional angular displacements (ϕ, θ, ψ), angular velocities $(\omega_{bx}, \omega_{by}, \omega_{bz})$ and accelerations $(\alpha_{bx} = \frac{d\omega_{bx}}{dt}, \alpha_{by} = \frac{d\omega_{by}}{dt}, \alpha_{bz} = \frac{d\omega_{bz}}{dt})$ and linear displacements (X_b, Y_b, Z_b), linear velocities (v_{bx}, v_{by}, v_{bz}) and accelerations $(a_{bx} = \frac{dv_{bx}}{dt}, a_{by} = \frac{dv_{by}}{dt}, a_{bz} = \frac{dv_{bz}}{dt})$ are acquired, and the BI-FWMAV states are updated.

4.3.2.1 Wing dynamics and aerodynamics module of the BI-FWMAV

To diminish complexity, aerodynamic force faced by a wing of the BI-FWMAV is determined by employing a three-dimensional quasi-steady model. To observe the impact of wind velocity on a small section of the wing, it was segmented into spanwise sections. Based on the local wind velocity acting on each section, their individual instantaneous forces are computed. These instantaneous forces

are then summed to attain the force faced by a wing at any instant. It is assumed that the wing is flapping in an inclined stroke plane with a certain angle. The detailed flapping profile of the wing along with their kinematics are modeled and prescribed in [192,201,202]. The flapping angle (ϕ) and the angle of attack (α_{aoa}) in the stroke plane can be manifested in a sinusoidal form as follows:

$$\phi(t) = \frac{\phi_a}{2}\cos(\pi f t) \tag{4.17}$$

$$\alpha_{aoa} = \alpha_{mn} + \alpha_p \sin(\omega dt + \Psi) \tag{4.18}$$

where ϕ_a is the flapping amplitude (in rad), f is the flapping frequency (in Hz), t is the time (in sec), α_{mn} is mean angle of attack (in rad), α_p is amplitude of pitching oscillation (in rad), dt is time step (in sec), and Ψ is the phase difference between the flapping angle and angle of attack (in rad).

Based on the experiments performed in [203, 204] to examine the effective maneuverability of MAV, the relative wind due to the movement of BI-FWMAV is considered in our work and can be presented as:

$$V_{wind} = v_b + r \times \omega_b \tag{4.19}$$

where, $v_b = [v_{bx}, v_{by}, v_{bz}]$ are linear velocities, and $\omega_b = [\omega_{bx}, \omega_{by}, \omega_{bz}]$ are angular velocities. In developing the aerodynamics module of BI-FWMAV, V_{wind} is added with the relative wind caused by the movement of the wing.

The drag, lift, and rotational coefficients captured during experimentation in [205,206] were employed in [201] to analyze the dragonfly flight simulator. The same drag, lift, and rotational coefficients are also utilized in our BI-FWMAV

plant and can be expressed as follows:

$$C_{dr} = 1.92 - 1.55\cos(2.04\alpha_{aoa} - 9.82^{\circ}) \tag{4.20}$$

$$C_{lf} = 0.225 + 1.58\cos(2.13\alpha_{aoa} - 7.2^{\circ}) \tag{4.21}$$

$$C_{rt} = \pi(0.75 - \hat{x}_0) \tag{4.22}$$

where the angles are presented in degrees, $\hat{x}_0$ represents the dimensionless distance of the rotation axis from the leading edge. The force (dF_{rot}) yields in each span of the wing due to its rotation, and lift (dF_{lift}) and drag (dF_{drag}) are formulated as follows:

$$dF_{lift} = 0.5\rho V^2 C_{lf} dA \tag{4.23}$$

$$dF_{drag} = 0.5\rho V^2 C_{dr} dA \tag{4.24}$$

$$dF_{rot} = C_{rt}\rho c^2 U\dot{\alpha}_{aoa} dS \tag{4.25}$$

where ρ is denoting the wind density, V is exposing the velocity of wind, dA is the area of a small section of the wing, U is the Euclidean norm of wind velocity, dS is the width and c is the length of an individual small part of a the wing.

From the above discussion, it is obvious that the BI-FWMAV plant associates profuse parameters with high nonlinearity, though we have omitted some complexity in revealing precise wing kinematics. Deriving a precise mathematical model of such highly nonlinear, complex, and the over-actuated plant is exceptionally laborious, where inclusion of uncertainties and uncharted disruption is more difficult or unfeasible in some cases. These perspectives necessitate a controller that performs precisely with a minimum or no knowledge about the system. Being model-free and self-evolving, our developed PAC is a suitable

candidate. More importantly, the impediment of conventional evolving controllers i.e. involvement of numerous free parameters is resolved here since our controller do not have any premise parameters and only depends on consequent parameters, namely weights of the network. With such simplistic evolving structure, PAC and RedPAC provide comparable and satisfactory tracking performance.

4.3.3 Dynamics of quadcopter MAV

This section describes a general nonlinear quadcopter model derived from its kinematics and dynamics [207]. The state variables are the Euler angles namely roll (ϕ), pitch (θ), and yaw (Υ), body axis angular velocities (p, q, r), inertial frame position (p_n, p_e, h) and the body axis velocity (u, v, w). The force balance equation is provided in Equation (4.26). The moment balance equation is exposed in Equation (4.27). The relation between Euler angles and (p, q, r) is given in Equation (4.28). The relationship between the inertial frame and body axis velocity components is presented in Equation (4.29).

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \begin{pmatrix} -g\ \sin\theta \\ g\ \cos\theta \sin\phi \\ g\ \cos\theta \cos\phi \end{pmatrix} \frac{1}{m} \begin{pmatrix} 0 \\ 0 \\ -F_z \end{pmatrix} \tag{4.26}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\Upsilon \end{pmatrix} \tag{4.27}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\Upsilon} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{4.28}$$

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} c\theta c\Upsilon & s\phi s\theta c\Upsilon - c\phi s\Upsilon & c\phi s\theta c\Upsilon + s\phi s\Upsilon \\ c\theta s\Upsilon & s\phi s\theta c\Upsilon + c\phi s\Upsilon & c\phi s\theta c\Upsilon - s\phi s\Upsilon \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{4.29}$$

In the above equations, g denotes the acceleration due to gravity, J_x, J_y and J_z are the moments of inertia along the X, Y, and Z-axis respectively. The input torques along body XYZ-axis are denoted by τ_ϕ, τ_θ and τ_Υ respectively. In Equation (4.29), c and s are indicating the cos and sin function respectively.

The six-DOF quadcopter dynamics is highly nonlinear, and there are uncertainty factors that affect the static or linear controller's performance. Therefore, it is hard to attain satisfactory performance from simple controllers like PID, LQR. Though the nonlinear controllers perform better than the linear counterparts, the nonlinear ones need information about the precise plant dynamics, which may not be known. In this case, model-free control approaches proved improved performance than modeled linear or nonlinear variants, as discussed in the previous section. However, due to their static structure, their performance deteriorates with the sudden changes of plant dynamics. Though the AICons solve the shortcomings of static model-free controllers, a high number of learning parameters limit their performance in controlling plants like a quadcopter, where fast response is expected from the controller. Thus, to yield fast response, an evolving intelligent controller with minimum learning parameters called RedPAC

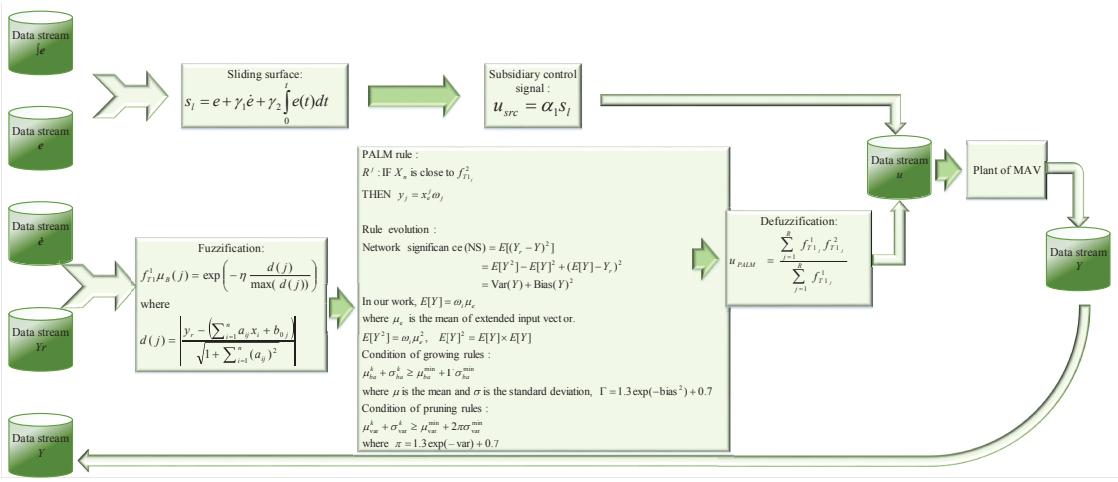


Figure 4.3: Flow of data streams in closed-loop PAC

is utilized to control the quadcopter.

4.4 Problem Statement

In a closed-loop control system, data comes in a sequential online manner to the controller, which may contain various uncertainties and disturbances. Such a randomly distributed sequence of incoming data to the controller can be exposed formally as $e = \{e^1, e^{1+t}, e^{1+2t}..., e^f\}$, denoted by "Data steam of e" in Figure 4.3, where t is the time step (in sec) of the closed-loop control system, f is the final time until which plants need to be controlled. Here, e is indicating the data stream of error, which is the difference between the reference trajectory data stream Y_r and observed corresponding out data stream from the plant Y. The remaining incoming data streams to the closed-loop system are data stream of derivative of error ($\dot{e}$), and integral of error ($\int e$) as displayed in Figure 4.3.

To tackle random sequence of incoming data, the controller should hold some desiring features such as 1) able to work in single-pass mode; 2) deal with various

uncertainties of the incoming data; 3) perform with low memory burden and computational complexity to enable real-time deployment under resource constrained environment. In the realm of NFS, such learning proficiency is manifested by evolving NFSs [41]. However, large number of free parameters associated with the evolving NFS-based AICons cause complex computation. From the perspective of controlling MAV, a swift response from the controller is much expected, where the tuning of several premise parameters in AICon is a hindrance to fulfill such expectation. To decipher the involvement of profuse premise parameters in AICon, their premise parameter-dependent hyper-spherical or hyper-ellipsoidal clustering techniques require to substitute with premise parameter-free clustering method. From this research gap, a hyper-plane based clustering method [189] is utilized in our work, which is composed of the consequent parameter rather than any premise parameters as explained in [189]. Another inadequacy of state of the art AICons is their affiliation to user-defined parameters to evolve their structure. Those parameters claim to alter with respect to the corresponding closed-loop system. Our proposed AICon is free from such parameter, and they have been superseded with the bias-variance concept. To get a clearer view, the flow of data streams in our closed-loop system is displayed in Figure 4.3. Unlike the PALM [189], in the fuzzification layer of PAC, we always have incoming target/reference data stream Y_r as shown in the fuzzification block of the Figure 4.3. To obtain an explicit impression, the self-evolving formations of our proposed PAC and RedPAC are enumerated in the next section.

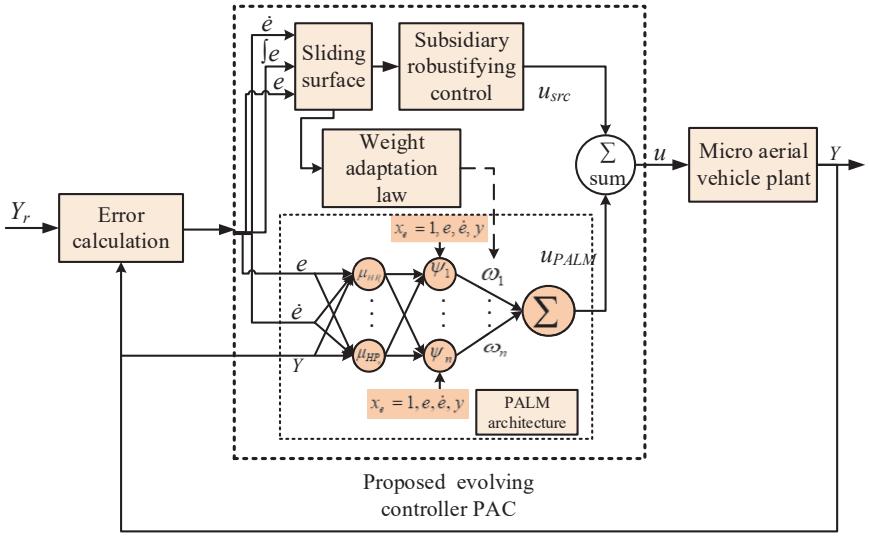


Figure 4.4: Self-adaptive PAC's closed-loop mechanism

4.5 Structure of PAC and RedPAC

Both the PAC and RedPAC are three-layered NF systems. Their evolving architecture is rooted with TS fuzzy model, where classical hyper-spherical [145], hyper-ellipsoidal [56], or data-cloud based [88] clusters are substituted with hyper-plane-based clusters. Utilization of hyper-planes have removed antecedent parameters, as explained in Chapter 3, which reduces the number of operative parameters of the developed controllers dramatically. Popular hyper-plane-based clustering techniques like fuzzy C-regression model (FCRM) [164], fuzzy C-quadratic shell (FCQS) [165], double FCM [89], inter type-2 fuzzy c-regression model (IT2-FCRM) [94] are non-incremental in nature. They can not entertain evolving hyper-planes. Additionally, they deploy hyper-spherical functions, for instance, Gaussian function to accommodate hyper-planes. To mitigate such inadequacies, a new membership function [94] is used in both PAC and RedPAC. The detailed architecture of the PAC and RedPAC are disclosed in Figure 4.5 and Figure 4.4.

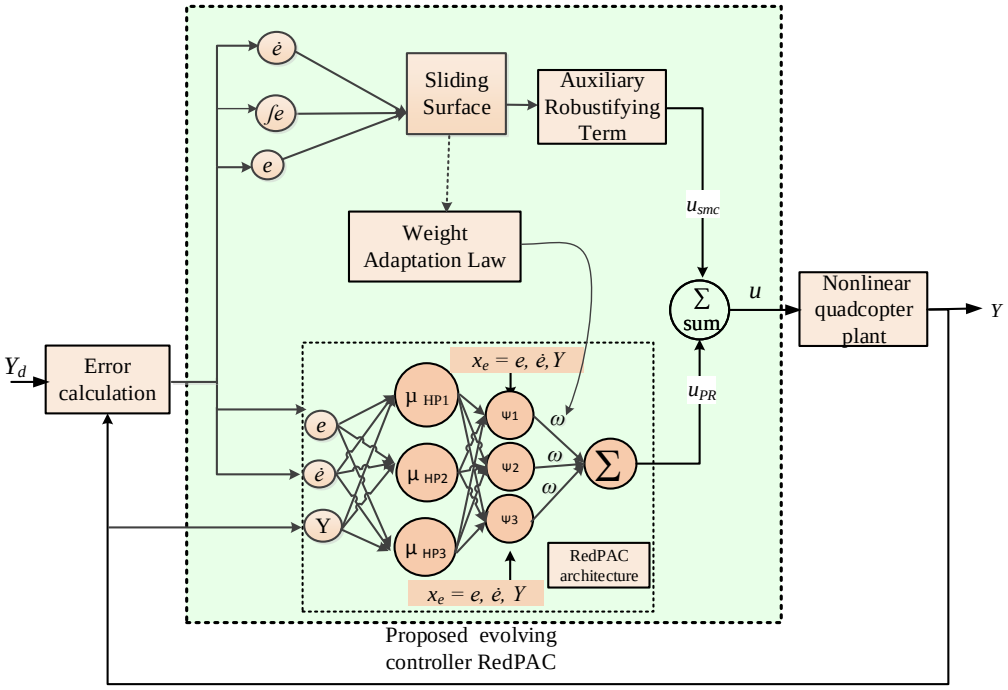


Figure 4.5: Closed-loop work flow in RedPAC

In our developed evolving closed-loop control systems, the networks are fed by three inputs, namely error (e), the derivative of error ($\dot{e}$) and actual plant's output (Y) as displayed in Figure 4.5 and Figure 4.4. Referring to the theory of fuzzy system, these crisp data $(e, \dot{e}, Y)$ need to be transformed into a fuzzy set, which is the initial step in the controllers' work flow. This fuzzification process is accomplished by adopting hyper-plane-shaped clustering (HPSC)-based membership function, which is framed through the concept of point-to-plane distance. The employment HPSC-based membership function can be expressed as follows:

$$f_{T1}^{1} = \mu_B(j) = \exp\left(-\eta \frac{d(j)}{\max\left(d(j)\right)}\right) \tag{4.30}$$

where η is a regulating parameter which adjusts the fuzziness of membership grades. Based on the observation in [94, 189], and empirical analysis with different MAV plant in our work, the range of η is fixed as $[1, 100]$ in PAC. After

numerous flight tests with the drone simulator (Dronekit SITL), the value of η is chosen as 0.5 in RedPAC. This membership function empowers the utilization of hyper-plane-based clusters directly into the PALM network without any rule parameters except the first order linear function or hyperplane. Because a point to plane distance is not unique, the compatibility measure is executed using the minimum point to plane distance. $d(j)$ in Equation (4.30) denotes the distance between the current data point and jth hyperplane as with Equation (4.32). It is determined by following the definition of a point-to-plane distance [162]. In PAC, it can be formally expressed as follows [189]:

$$d(j) = \left| \frac{Y_r - (\sum_{i=1}^{n} \omega_{ij} x_i + \omega_{0j})}{\sqrt{1 + \sum_{i=1}^{n} (\omega_{ij})^2}} \right| \tag{4.31}$$

In case of RedPAC, the expression is as follows:

$$d(j) = \left| \frac{Y_r - (\sum_{i=1}^{n} x_i)\omega_j}{\sqrt{\omega_j^2}} \right| \tag{4.32}$$

where ω_{0j} and ω_{ij} are consequent parameters of the jth rule, $i = 1, 2, ..., n$; n is the number of inputs for both the controllers, Y_r is the desired reference for the plants. Unlike the ω_{ij} of PAC in Equation (4.31), in RedPAC, we are using only ω_j as expressed in Equation (4.32). If we choose $i = n$ and $j = R$, then from Equation (4.31) and Equation (4.32), it is clear that the number of weights in PAC is $R \times (n + 1)$, which is only R in RedPAC. It is noteworthy to state that a type-1 fuzzy structure is facilitated in both PAC and RedPAC. In light of a MISO system, the IF-THEN rule can be exposed as follows:

$$R^j : \quad \text{IF } X_n \text{ is close to } f^2_{T1_j} \text{ THEN } y_j = x_e^j \omega_j \tag{4.33}$$

where 2 in $f^2_{T1_j}$ is indicating the output of the second layer of the underlying TS-fuzzy structure, x_e is the extended input vector and is expressed by inserting the intercept to the original input vector as $x_e = [1, e, \dot{e}, Y]$, e is the error i.e. the difference between the reference and actual output of the plant, $\dot{e}$ is the error derivative i.e. the difference between the present and previous state error value, Y_r is the reference for the plant to be controlled, ω_j is the weight vector for the jth rule, y_j is the consequent part of the jth rule. The antecedent part of PALM is simply hyperplane and does not consist of any premise parameters. The intercept of the extended input vector dominates the slope of hyperplane, which eliminates the atypical gradient dilemma.

In both PAC and RedAPC, an analogous consequent part alike the basic TS-fuzzy model's rule consequent part ($y_j = b_{0j} + a_{1j}x_1 + ... + a_{nj}x_n$) is employed. The consequent part for the jth hyperplane is calculated by weighting the extended input variable (x_e) with its corresponding weight vector as follows:

$$f^2_{T1_j} = x_e^T \omega_j \tag{4.34}$$

The weight vector in Equation (4.34) is updated recursively by the SMC theory-based adaptation laws, which ensures a smooth alteration in the weight value. In the next step, the rule firing strength is normalized and added with the rule consequent to produce the end-output of PALM. The final defuzzified crisp output of the PALM can be expressed as follows:

$$u_{PR} = \frac{\sum_{j=1}^{R} f^1_{T1_j} f^2_{T1_j}}{\sum_{i=1}^{R} f^1_{T1_i}} \tag{4.35}$$

The normalization term in Equation (4.35) assures the proper partition where the sum of normalized membership degree is one. Both the controllers developed in

this chapter have a simplified structure-learning mechanism, unlike the original PALM [189] as explained in the next subsection.

4.5.1 Automatic constructive mechanism of PAC and RedPAC

In PALM [189], the self-constructive clustering technique was adopted to grow the rule. The rule significance was determined by measuring input and output coherence, where the coherence was calculated by investigating the correlation between the existing data samples and the target concept. Again, the computation of correlation has a dependency on finding variance and covariance among different variables. In addition, the PALM's rule growing method was regulated by two predefined thresholds. On the other hand, PALM's rules were merged by measuring the similarity between the hyperplane-shaped fuzzy rules. The similarity among rules were measured by observing the angle and minimum distance between them. The merging strategy was also controlled by predefined thresholds. This clearly shows the high computational cost of PALM's rule-evolving mechanism and made them incompatible in fast response based control applications. To subjugate such complexity, a simplified rule evolution technique is implemented in PAC and RedPAC by using the network significance (NS) method, which is formulated from the concept of bias-variance [190]. Here, no predefined thresholds are required to control the generation or pruning of their fuzzy rules. This network significance method based rule growing and deletion modules are clarified in the subsequent paragraphs of this section.

4.5.2 NS method-based rule-growing mechanism

The strength of PAC and RedPAC can be analyzed by witnessing the tracking error, which can be written in terms of mean square error (MSE) as follows:

$$e_{MSE} = \sum_{t=1}^{T} \frac{1}{T}\big(y_r(t) - y(t)\big)^2 \tag{4.36}$$

where e_{MSE} is denoting the mean square error, $y_r(t)$ is expressing the desired trajectory and $y(t)$ is the output of plants to be controlled. Equation (4.36) experiences two obstructions in learning mechanism of an evolving controller, such as 1) it needs to memorize all data points to get a clearer view about the controller's constructive mechanism; 2) though recursive calculation of e_{MSE} excluding preceding data is possible, it does not investigate the strength of reconstruction for uncertain upcoming data. In simple words, it does not consider the generalization capacity of evolving controllers. To mitigate such hindrance, let us consider that Y is the observed plant's output for the reference input Y_r, and $E[Y]$ is the plant's expected output. According to the definition of expectation [208], if x is continuous, then the expectation of $f(x)$ can be formulated as: $E[f(x) = \int_{-\infty}^{\infty} f(x)p(x)dx]$, where $p(x)$ is the probability density function of x. Then the network significance (NS) method can be defined as follows:

$$NS = \int_{-\infty}^{\infty} \big(Y_r(t) - Y(t)\big)^2 p(t)dt \tag{4.37}$$

where $p(t)$ is the probability density function, and t is denoting the time. For simplicity, in the following equations, it is considered that $Y(t) = Y$ and $Y_r(t) = Y_r$. By following the definition of expectation of a function $E(.)$ [208], Equation

(4.37) can be rewritten as:

$$NS = E\big[(Y_r - Y)^2\big] = E\big[(Y - E[Y] + E[Y] - Y_r)^2\big] \tag{4.38}$$

Now, the implementation of the concept of bias-variance in Equation (4.38) yields the following [209]:

$$NS = E[(Y - E[Y])^2] + (E[Y] - Y_r)^2 = \mathrm{Var}(Y) + \mathrm{Bias}(Y)^2 \tag{4.39}$$

where $\mathrm{Var}(Y(t))$ is presenting the variance of $Y(t)$, and it can be expressed as follows:

$$\begin{aligned} \mathrm{Var}(Y) = E\big[(Y - E[Y])^2\big] &= \int_{-\infty}^{\infty} \big(Y - E(Y)\big)^2 p(t)dt \\ &= E[Y^2] - (E[Y])^2 \end{aligned} \tag{4.40}$$

By following the similar approach in [209], the maximum rule firing strength considered in RedPAC is one. It simplifies its output as $Y = \sum_{j=1}^{R} \sum_{i=1}^{n} x_{ij}\omega_j$. Now applying normal distribution, the expectation of Y can be expressed as $E[Y] = \int_{-\infty}^{\infty} Y p(t)dt$. In this expression, the integration of x_i over $-\infty$ to ∞ originates the mean μ_i. Thus the $E[Y]$ can be expressed as follows:

$$E[Y] = \sum_{j=1}^{R} \sum_{i=1}^{n} \mu_{ij}\omega_j \tag{4.41}$$

By utilizing Equation (4.40) and Equation (4.41) in Equation (4.39), it can further be simplified as follows to get the final expression of NS:

$$\begin{aligned} NS &= \mathrm{Var}(Y) + \mathrm{Bias}(Y)^2 = E[Y^2] - (E[Y])^2 + (Y_r - E[Y])^2 \\ &= \sum_{j=1}^{R} \sum_{i=1}^{n} \mu_{ij}^2 \omega_j - (E[Y] \times E[Y]) + (Y_r - E[Y])^2 \end{aligned} \tag{4.42}$$

Equation (4.42) is manifesting the final expression of NS. From Equation (4.39), we have observed that the NS contains both variance and bias. Therefore, a high value of NS may indicate a high variance (over-complex network with profuse fuzzy rules) or a high bias (oversimplified network) problem. Such phenomenon can not be elucidated simply by system error index. Augmentation of a new rule is inferred to subjugate the high bias dilemma. Nonetheless, such phenomenon is not convenient for high variance context since the addition of rules magnifies the network complexity. To retain a compact network structure with satisfactory tracking performance, the concept of bias-variance trade-off is inserted in our work to calculate the NS. Such regulation of the rules of our controller has no reliance on user-defined parameters [210, 211]. By confirming the fundamental objective of rule growing procedure to ease the high bias dilemma, the condition of growing rules in our work is expressed as follows:

$$\mu_{ba}^{k} + \sigma_{ba}^{k} \geq \mu_{ba}^{min} + \Gamma\sigma_{ba}^{min} \tag{4.43}$$

where μ_{ba}^{k} is denoting mean of bias and σ_{ba}^{k} is standard deviation of bias at the kth observation whilst μ_{ba}^{min} and σ_{ba}^{min} are pointing the minimum mean and standard deviation of bias up to kth time instant. In computing these variables, no preceding data are required. Their values are being updated directly based on the availability of upcoming signals to the PALM. When Equation (4.43) is satisfied, the values of μ_{ba}^{min} and σ_{ba}^{min} are to be reset. To perceive an improved tracking performance from the commencing of PAC's control operation, a rapid decay in the bias value is expected. It is retained in formulating the settings of bias in Equation (4.43) as long as the plant does not encounter any uncertainties or disturbances. The presence of any disruptions in the control system will

the elevate value of bias, which cannot be addressed directly by adapting the consequent parameter of the PAC. To elucidate such hindrance, Equation (4.43) is originated from the adaptive sigma rule, where Γ controls the degree of confidence of the sigma rule. In PAC, the Γ is expressed as $\Gamma = 1.3\exp(-\text{bias}^2) + 0.7$ and in RedPAC $\Gamma = 1.5\exp(-\text{bias}^2) + 0.5$, which revolves Γ between 1 and 2. Consequently, it obtains the level of confidence from around 68% to 96%. Such scheme enhances the flexibility in the rule-growing module to adapt to the environmental perturbations. It also eliminates the dependency of evolving controller on user-defined problem-dependent parameters. To sum up, a high bias usually signifies an oversimplified network, which is solved by adding rules. However, it is avoided in case of low bias since it may magnify the variance.

4.5.3 Mechanism of pruning rules

The high complexity of the PAC and RedPAC's network is caused by the high variance. On that ground, control of variance is essential to reduce the network complexity by pruning the fuzzy rules. Since a high variance indicates the overfitting condition, the rule pruning scheme initiates from the evaluation of variance. Like the rule growing mechanism of PAC, a statistical process control technique is embraced in rule pruning module to trace the high variance dilemma as follows:

$$\mu_{var}^{k} + \sigma_{var}^{k} \geq \mu_{var}^{min} + 2\pi\sigma_{var}^{min} \tag{4.44}$$

where μ_{var}^{k} is denoting mean and σ_{var}^{k} is the standard deviation of variance at the kth observation while μ_{var}^{min} and σ_{var}^{min} are pointing the minimum of mean and standard deviation of variance up to kth time instant. Here, the term π is adopted as $\pi = 1.3\exp(-\text{var}) + 0.7$ in PAC and $\pi = 0.2 \times (1.5\exp(-\text{variance}) + 0.5)$ in

RedPAC, π is a dynamic constant and regulating the degree of confidence in the sigma rule. The term 2 is Equation (4.44) holds the direct pruning after growing. Furthermore, μ_{var}^{min} and σ_{var}^{min} are reset when the condition in Equation (4.44) is fulfilled.

After the execution of Equation (4.44), the significance of each rule is examined via the idea of network significance, and inconsequential rules are pruned to reduce the overfitting condition. The significance of rules are tested via the concept of network significance, adapted to evaluate each rule's statistical contribution. The significance of ith rule is determined as its average activation degree for all possible incoming data samples or its expected values as expressed in Equation (4.39). Considering the normal distribution assumption, the importance of ith rule can be expressed as $HS_i = \omega_i \mu_e$. A small value of HS_i indicates that the ith rule plays a small role to recover the clean input attributes. Therefore, it can be pruned with a very insignificant loss of tracking accuracy. Since the contribution of the ith rule is calculated in terms of the expectation $E(Y)$, the least contributing rule having the lowest HS is regarded inactive. When the overfitting condition occurs, or Equation (4.44) is satisfied, the rule with the lowest HS is pruned and can be expressed as follows:

$$\text{Pruning} \longrightarrow \underset{i=1,...R}{min} HS_i \tag{4.45}$$

The condition in Equation (4.45) targets to mitigate the overfitting situation by deleting the least significant rule. It also indicates that the desired trajectory tracking performance can still be achieved with the rest $R-1$ rules. Furthermore, this rule pruning strategy enhances the generalization power of the developed controllers by reducing their variance, which helps to deal with a variety of

disturbances.

4.5.4 Adaptation of weights in PAC and RedPAC

Unlike the conventional evolving controller, the evolving neuro-fuzzy controllers developed in this chapter do not possess any premise parameters, consequently free from the computation of tuning those parameters. Only the consequent parameters need to be adjusted to realize desired control efficacy. Inspired by the smooth employment and regulations of SMC theory in various neuro-fuzzy systems [212–214], in our work SMC learning theory is functioned to adapt the controllers' weight, which ascertains stability in the closed-loop control system. Besides, it confirms adequate robustness in a system against exterior disturbances, parameter variations, and uncertainties [200]. In designing SMC, a time-varying *sliding surface* that restricts motion of a system to a plane can be exposed as follows:

$$S_{ss}(u_{PALM}, u) = u_{src}(t) = u_{PALM}(t) + u(t) \tag{4.46}$$

In RedPAC, the sliding surface for quadcopter simulator can be expressed as:

$$s_l = K_1 e + K_2 \dot{e} + K_3 \int_0^t e(t)dt \tag{4.47}$$

where, $K_1 = 20$ and $K_2 = 0.02$ and $K_3 = 0.002$ are pre-defined thresholds.

In PAC, the sliding surface for BI-FWMAV and hexacopter plant to be controlled can be expressed as:

$$s_l = e + \gamma_1 \dot{e} + \gamma_2 \int_0^t e(t)dt \tag{4.48}$$

where, $\gamma_1 = \frac{\alpha_2}{\alpha_1}$, $\gamma_2 = \frac{\alpha_3}{\alpha_1}$, e is the error i.e. the divergence from the trajectory obtained from the plant to the reference one. Here, the sliding parameters are initiated with tiny values such as $\alpha_1 = 1 \times 10^{-2},\ \alpha_2 = 1 \times 10^{-3},\ \alpha_3 \approx 0$. These parameters are further evolved by different learning rates. Proper assignment of these rates supports to secure the desired parameters with minimal time, which affirms to gain stability in the closed-loop system swiftly. Engagement of these self-organizing sliding parameters shapes a fully AICon. The definition maintained in PAC and RedPAC is as follows:

Definition : After a specific time t_k, a sliding motion will be formed on the sliding manifold $S_{ss}(u_{PR}, u) = u_{src}(t) = 0$, where the state $S_{ss}(t)\dot{S}_{ss}(t) = u_{src}(t)\dot{u}_{src}(t) < 0$ to be convinced for the entire time period with some non-trival semi-open sub-interval of time expressed as $[t,\ t_k) \subset (0,\ t_k)$.

To enforce the above-mentioned definition of sliding mode condition, weights of the proposed controllers are adapted accordingly.

In our proposed controller, the reliance of the subsidiary robustifying control term on the sliding surface can be formulated as follows:

$$u_{src}(t) = \alpha_1 s_l \tag{4.49}$$

This subsidiary robustifying control term u_{src} may endure high-frequency oscillations in contributing to the control input [212]. Such repulsive occurrence in sliding mode control theory is termed as chattering effect. To suppress this chattering effect, control systems are primarily facilitated with saturation or sigmoid functions. In this work, due to simplicity, a saturation function is used to alleviate those detrimental consequence.

The outcome from PALM (u_{PALM}) in PAC and RedPAC can be expressed as

follows:

$$u_{PALM}(t) = \psi^T(t)\omega(t) \tag{4.50}$$

The overall control signal as observed in Figure 4.4 can be declared as follows:

$$u(t) = u_{src}(t) - u_{PALM}(t) \tag{4.51}$$

The adaptation law to guarantee the boundedness of the tracking error and the consequent parameter, namely weights of PAC and RedPAC is expressed as:

$$\dot{\omega}(t) = -\gamma b \mathbf{e} P \psi \tag{4.52}$$

where $\gamma > 0$, P is a positive definite matrix as exposed in Equation (4.63), b is an unknown positive constant, $\mathbf{e} = [e \;\; \dot{e}]$. These weight adaptation laws assure a stable closed-loop control system.

4.5.4.1 Proof of boundedness of error and weights in PAC and RedPAC

Proof : Let us consider an nth order nonlinear system of the form as follows:

$$X^{(n)} = F(X, \dot{X}, ..., X^{(n-1)}) + bu, \quad Y = X \tag{4.53}$$

where $F(.)$ is an unknown continuous function, b is an unknown positive constant, $u \in \Re$ is the input and $Y \in \Re$ is the output of the system. We have considered that the state vector $\mathbf{X} = (X_1, X_2, ..., X_n)^T = (X, \dot{X}, ..., X^{(n-1)})^T \in \Re^n$ is available for measurement. Our control objective in this work is to push Y to track a given reference trajectory Y_r. To be specific, the control objectives can be summarized as follows:

i) Parameters of the closed-loop control system should be uniformly, ultimately bounded to confirm the global stability of that closed-loop system. In this work, $|\mathbf{X}(t)| \leq M_{\mathbf{X}} < \infty$, $|\omega(t)| \leq M_{\omega} < \infty$, and $|u(\mathbf{X}|\omega)| \leq M_u < \infty$ for all $t \geq 0$, where $M_{\mathbf{X}}$, M_{ω}, and M_u are pre-defined design parameters.

ii) The closed-loop tracking error $e = Y_r - Y$ should be as small as possible by satisfying the conditions in (i).

Both the controllers should be able to achieve these control objectives. To show that, let us consider $\mathbf{e} = (e, \dot{e}, ..., e^{(n-1)})^T$ and $\mathbf{k} = (k_1, ..., k_n)^T \in \Re^n$ carry such features that all roots of the polynomial $h(p) = p^n + k_1 p^{(n-1)} + ... + k_n$ will be in the open left-half plane. If we know about the function $F(.)$ and constant b, then the optimal control law for PAC and RedPAC can be expressed as follows:

$$u^* = \frac{1}{b}\big[- F(\mathbf{X} + Y_r^{(n)} + \mathbf{k}^T\mathbf{e})\big] \tag{4.54}$$

Now, applying the optimal control law of Equation (4.54) in Equation (4.53) yields the follows:

$$e^{(n)} + k_1 e^{(n-1)} + ... + k_n e = 0 \tag{4.55}$$

From Equation (4.55), it is obvious that $\lim_{t\to\infty} e(t) = 0$, which is the desired control objective from the proposed controller. Since both $F(.)$ and b are unknown, the optimal control law cannot be implemented. In such circumstance, PALM is used to approximate the optimal control law. Now utilizing Equation (4.51) in Equation (4.53), the following is obtained:

$$X^{(n)} = F(\mathbf{X}) + b\big[u_{src} - u_{PALM}\big] \tag{4.56}$$

By following the approach in [215], bu^* is added and substructed to Equation

(4.56) and the error equation governing the closed-loop system can be exposed as follows:

$$e^n = -k^T \mathbf{e} + b\big[u_{src} - u_{PALM}\big] \tag{4.57}$$

By following the approach of [215, 216], the error dynamics in our work can be presented as follows:

$$\dot{\mathbf{e}} = A\mathbf{e} + b(u_{PALM} - \epsilon) \tag{4.58}$$

where $\mathbf{e} = [e \quad \dot{e}]^T$, $A = \begin{bmatrix} 0 & 1 \\ -\alpha_1 & -\alpha_2 \end{bmatrix}$, and $b = [0 \quad 1]^T$. Let us denote an ideal weight as ω^* by defining the corresponding weight error as $\tilde{\omega} = \omega - \omega^*$. Now, the error dynamics can be rewritten as:

$$\dot{\mathbf{e}} = A\mathbf{e} + b\psi^T\tilde{\omega} + b(\psi^T\omega^* - \epsilon) \tag{4.59}$$

Assume that, in the domain of interest, the ideal weight brings the term $\psi^T(t)\omega^*(t)$ to within a Δ-neighbourhood of the error ϵ. It is bounded by

$$\Delta^* \equiv \sup_z \big|\psi^T(z)\omega^* - \epsilon(z)\big| \tag{4.60}$$

where z is the vector that contains all the variables of the inversion error. The term $\big|\psi^T(z)\omega^* - \epsilon(z)\big|$ represents a residual inversion error which is not modeled by the controllers. Therefore, Δ^* is defined as the worst-case difference between the error and its best approximation.

In this work, the candidate Lyapunov function is considered as follows:

$$V = \frac{1}{2}\mathbf{e}^T P\mathbf{e} + \frac{1}{2\gamma}\tilde{\omega}^T\tilde{\omega} \tag{4.61}$$

where $\gamma > 0$. For $\alpha_1 > 0$ and $\alpha_2 > 0$, A in Equation (4.58) is Hurwitz, P is a positive definite matrix and satisfying the following equation:

$$A^T P + PA = -Q \tag{4.62}$$

where $Q > 0$. For all $Q > 0$, the solution of Equation (4.62) is > 0. For $Q = I_2$, it is implying the following value of P:

$$P = \begin{bmatrix} \frac{\alpha_2}{\alpha_1} + \frac{1}{2\alpha_2} & \frac{1}{2\alpha_1} \\ \frac{1}{2\alpha_1} & \frac{1+\alpha_1}{2\alpha_1\alpha_2} \end{bmatrix} \tag{4.63}$$

Now, the time derivative of the Lyapunov function V with the substitution of Equation (4.58) and (4.62) can be expressed as:

$$\dot{V} = -\frac{1}{2}\mathbf{e}^T Q\mathbf{e} + b\mathbf{e}^T P(\psi^T\omega^* - \epsilon) + b\tilde{\omega}^T(t)\big(\mathbf{e}^T P + \frac{1}{\gamma}\dot{\tilde{\omega}}\big) \tag{4.64}$$

The third term of Equation (4.64) is suggesting a design of the adaptation law as follows:

$$\dot{\tilde{\omega}} = \dot{\omega} = -\gamma b\mathbf{e}^T P\psi \tag{4.65}$$

After employing the above adaptation law to Equation (4.64), it can be reduced as:

$$\dot{V} = -\frac{1}{2}\mathbf{e}^T Q\mathbf{e} + \mathbf{e}^T Pb(\psi^T\omega^* - \epsilon) \leq -\frac{1}{2}||\mathbf{e}||_2^2 + \Delta^*|\mathbf{e}^T Pb| \tag{4.66}$$

Utilizing the inequality exposed in [216], we can write:

$$\mathbf{e}^T P\mathbf{e} \leq \overline{\lambda}(P)||\mathbf{e}||_2^2 \tag{4.67}$$

Using Equation (4.67), the following is obtained from Equation (4.66):

$$\dot{V} \leq -\frac{\mathbf{e}^T P \mathbf{e}}{2\overline{\lambda}(P)} + \Delta^* \sqrt{\mathbf{e}^T P \mathbf{e}} \sqrt{\overline{\lambda}(P)} \tag{4.68}$$

Equation (4.68) is strictly negative when:

$$\sqrt{\mathbf{e}^T P \mathbf{e}} > 2\Delta^* (\overline{\lambda}(P))^{3/2} \tag{4.69}$$

According to the LaSalle-Yoshizawa theorem [217], since $\dot{V}$ is strictly negative, it is sufficient to prove that $\mathbf{e}$ and $\omega(t)$ will be remained bounded. In addition, if $\Delta^* = 0$, there will be no approximation error, then $e(t) \to 0$ as $t \to \infty$. It is guaranteeing the asymptotic stability of the system.

Unlike the conventional neuro-fuzzy systems, consequent parameters, namely weights of both controllers are also utilized in the antecedent part as exposed in Equation (4.32). Therefore, it is important to confirm the boundedness of the weights while they were used in the antecedent part. In this work, the weights are initialized with small values (less than one). Then, they are updated recursively, where their boundedness is confirmed and explained in the above paragraphs. Absolute value is considered in the distance formula since the distance cannot be negative. The calculated distance from the weights is employed in the fuzzification layer. In Equation (4.30), η is a positive constant, the highest value for the ratio between the distance and maximum distance is one and positive. Therefore, the values for the exponent operator always remains negative and bounded by η, which confirms the stability of the antecedent part of the controllers.

4.6 Numerical Experiments

In our chapter, the proposed evolving PAC is used to regulate an over-actuated hexacopter and BI-FWMAV plant, where numerous altitude and attitude trajectories are tracked for both the MAVs. To be specific, PAC is appraised in tracking altitude of six different trajectories for BI-FWMAV. On the other hand, in hexacopter, the performance of PAC is witnessed both for tracking six different altitudes, and sinusoidal attitude. The RedPAC is used to control a quadcopter with a varying altitude. All these observations are detailed in the upcoming subsections.

4.6.1 Simulation results from BI-FWMAV

Our proposed evolving PAC was inspected for numerous tracking signals and their consequent outcomes were contrasted with a Feed-Forward Neural Network (FFNN) based nonlinear adaptive controller, a Takagi-Sugeno (TS) fuzzy controller, a PID controller, and another AICon namely G-controller described in Chapter 5 of this thesis. The PAC code was written in C programming language and made openly accessible in [191]. The performance of all these controllers was observed in a BI-FWMAV plant for a duration of 100 seconds. The characteristics of six separate altitude trajectories for BI-FWMAV were as follows: 1) an unaltered height of 10 meters exposed as $Y_r(t) = 10\ m$; 2) variable heights with sharp edges, where the heights were altering from 3 m to 6 m after 20 seconds, and then from 6 to 9 m for another 20 seconds, and vice versa afterwards for another 40 seconds. The duration of hovering at a particular height was 20

seconds; 3) variable heights with smooth edges i.e. the change from one

Table 4.1: Measured features of various controllers in operating the BI-FWMAV (RT: rise time, ST: settling time, CH: constant height, VH: variable height, SS: sum of sine, RMSE: root mean square error, ms: millisecond, m: meter, MA: maximum amplitude, PSW: periodic square wave)

Desired trajectory	**Measured features**	**Control methods**				
		PID	**FFNN**	**TS-Fuzzy**	**G-control**	**PAC**
CH (MA 10 m)	RMSE	**0.6460**	0.7108	0.6693	0.6631	0.6668
	RT (ms)	50.772	55.828	44.629	**41.208**	47.207
	ST (ms)	560.98	415.90	222.06	**127.15**	147.09
	Peak (m)	12.246	11.572	10.451	10.813	**10.306**
VH with sharp change (MA 9 m)	RMSE	**0.3303**	0.4078	2.4951	0.3324	0.3561
	RT (ms)	23.931	48.943	43.949	50.892	**13.728**
	ST (ms)	8176.4	8386.3	8329.4	**8133.2**	8166.5
	Peak (m)	9.3732	9.6740	9.3010	**9.0069**	9.2265
VH with smooth change (MA 13 m)	RMSE	0.0895	0.0556	0.0368	**0.0228**	0.0523
	RT (ms)	8.8573	11.231	1.6537	4.187	**0.1314**
	ST (ms)	9884.3	**9857.7**	9871.1	9870.5	9872.1
	Peak (m)	13.006	13.009	**13.004**	13.019	13.006
SS function (MA 11 m)	RMSE	0.4730	0.5356	**0.4631**	0.4963	0.5018
	Peak (m)	11.468	11.502	11.455	**11.431**	11.462
PSW function (MA 11 m)	RMSE	2.7771	3.3185	N/A	**2.5098**	2.5115
	RT (ms)	548.93	474.36	N/A	61.563	**59.017**
	ST (ms)	9924.1	9911.7	N/A	**9603.2**	9634.1
	Peak (m)	12.794	12.573	N/A	**11.007**	11.294
Staircase function (MA 12 m)	RMSE	0.3073	0.3791	2.1796	**0.2885**	0.3072
	RT (ms)	5996.0	4060.7	**4015.7**	5998.4	6000.3
	ST (ms)	8384.2	8259.1	8094.4	8056.6	**8055.1**
	Peak (m)	12.453	12.458	12.074	**12.007**	12.198

height to another height is not so sharp as the previous trajectory; 4) sum of sines function, which was an amalgamation of sine waves with a frequency of 0.3 $radsec^{-1}$, amplitude of 4 m, bias of 6 m, and a cosine wave having a frequency of 0.5 $radsec^{-1}$, amplitude of 3 m and bias of 3 m; 5) a periodic square wave pulse, where the amplitude was varying between 1 m to 11 m, and its frequency is 0.2 $radsec^{-1}$; 6) a staircase function, where each step had a duration of 20 seconds. The individual heights of first three steps are the same with a value of 3 m, which is 2 m in the last step. In these numerical experiments, FFNN based nonlinear adaptive controller operated better than the PID and TS-fuzzy controllers. Again, the PAC manifested better tracking performance than the FFNN controller. To acquire a deeper understanding of these manifestations, some of their desired features like root mean squared error (RMSE), Rising Time (RT) in milliseconds, Settling Time (ST) in milliseconds, and the peak values of the overshoot were captured and tabulated in Table 4.1. All these simulation results were pictured in Figure 4.6 and detailed in the next paragraph.

In Figure 4.6 (a), controllers were facilitated to track a 10 m height trajectory, where both from PID and FFNN controllers, higher overshoot with peak values more than 12 m were attested. Better performance with peak overshoots of less than 7% of the height was noticed from the TS-fuzzy, and self-adaptive controllers. With regards to peak overshoot, the lowest values were exhibited by the proposed evolving controller in all six different scenarios of Figure 4.6, which is evidently signifying the superiority of PAC's evolving structure. A network with fewer parameters supports PAC to procure prompter settlement, which was witnessed from the lowest settling time of 147.09 seconds in Figure 4.6 (a). It was considerably faster than the benchmark controllers since they demand more than 200 seconds to settle. In most cases of Figure 4.6, lowest or very comparative

settling time was observed from the PAC. The tracking accuracy of the PAC in terms of RMSE and rising time was not always the lowest one. Nonetheless, their achievements were still comparable and sometimes surpassed benchmark controllers.

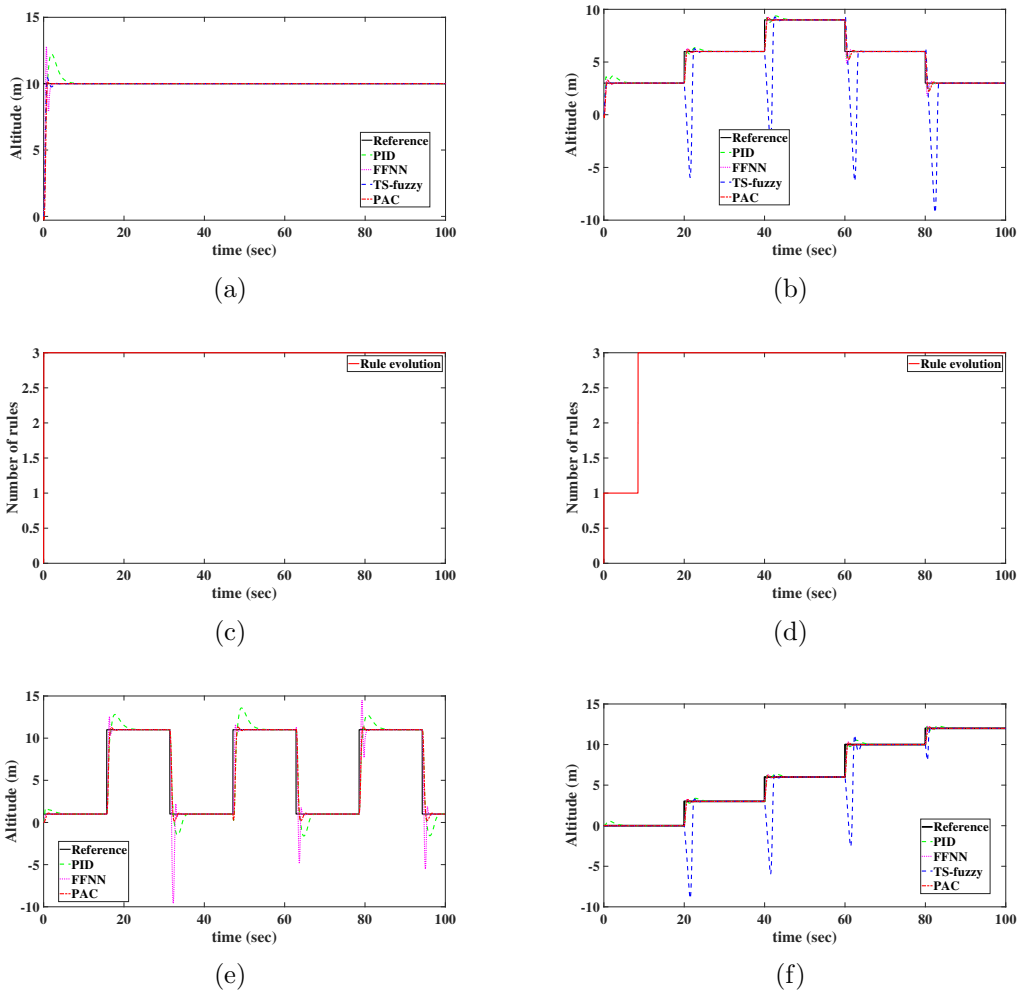


Figure 4.6: Performance observation of different controllers in tracking altitude of BI-FWMAV, when the trajectories are (a) constant hovering, (b) variable heights with sharp edges, (e) periodic square wave function, (f) staircase function, rule evolution corresponding to (c) constant hovering, (d) variable heights with sharp edges

4.6.2 Simulation results from hexacopter plant

PAC was assessed to track both the altitude and attitude (in terms of rolling and pitching) of the over-actuated hexacopter plant. Six separate trajectories of hexacopter's altitude were as follows: 1) a constant height with a value of 4 m; 2) altering heights with sharp edges, where the peak height was of 9 m; 3) altering heights with smooth edges, where the peak height was of 13 m; 4) a step function, which can be expressed as $3u(t-3)$, where $u(t)$ is a unit step function; and 5) a staircase function with a peak of 12 m; 6) sum of sines function, which was an amalgamation of a sine wave with a frequency of 0.3 $radsec^{-1}$, amplitude of 4 m, bias of 6 m, and a cosine wave having a frequency of 0.5 $radsec^{-1}$, amplitude of 3 m and bias of 3 m. In all conditions, a higher overshoot was perceived from the linear PID controller at each sharp changes as depicted in Figure 4.7. Peak of this overshoot was lesser while the linear controller was replaced with the nonlinear adaptive FFNN controller. The FFNN's performance was not consistent in all cases. Especially, in dealing with the square wave trajectory, performance deteriorates significantly as portrayed in Figure4.7 (c). This issue was managed by the evolving controller effectively owing to self-adaptive architecture. Quick settlements were also observed from PAC as recorded in Table 4.2.

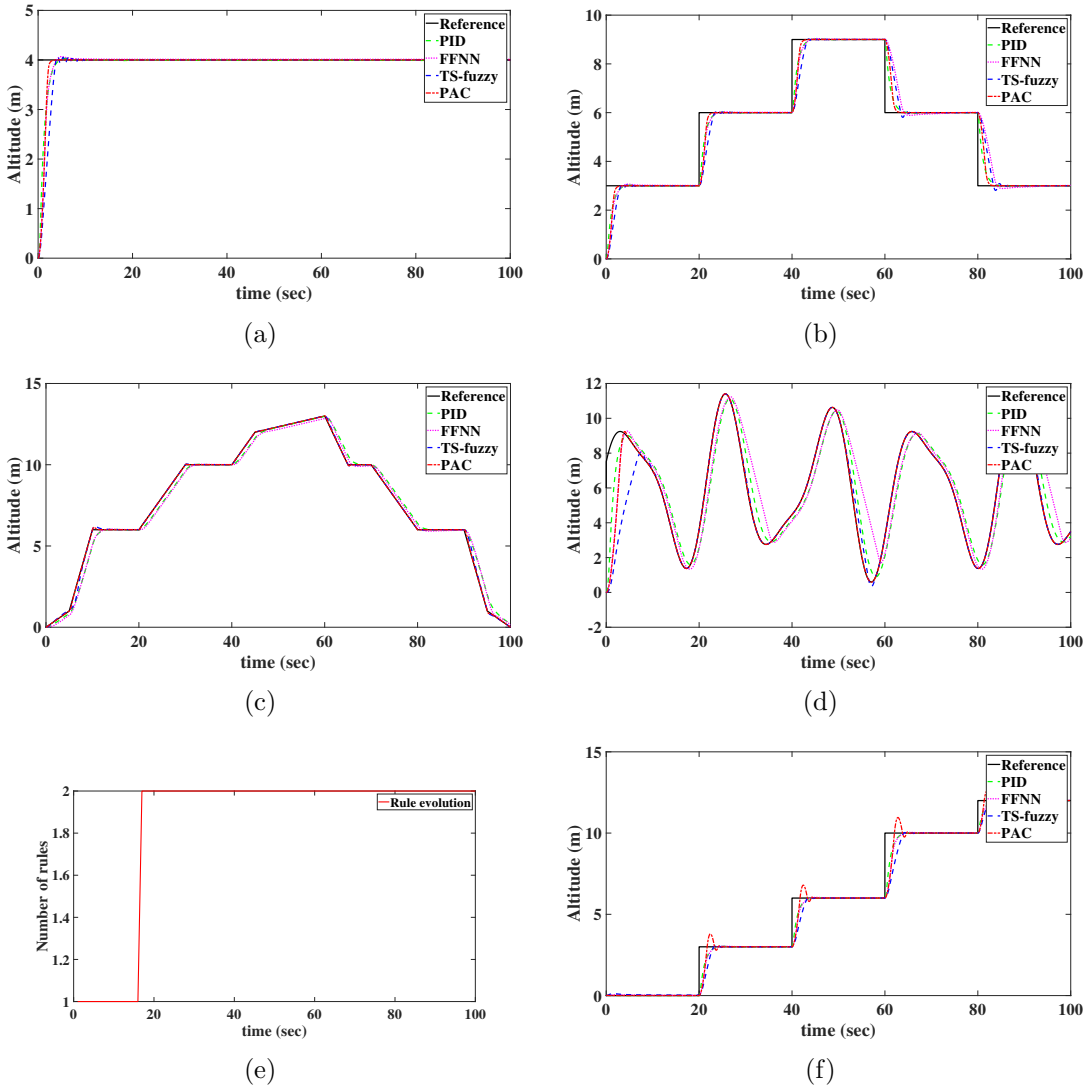


Figure 4.7: Performance observation of different controllers in tracking altitude of hexacopter, when the trajectories are (a) constant hovering, (b) variable heights with sharp edges, (d) staircase function, and (c) evolution of rules corresponding to constant hovering

Furthermore, the rolling and pitching position (in rad) was observed with a sum of sine trajectory, which was a fusion of a sine wave with a frequency of 0.3 $radsec^{-1}$, amplitude of 0.3 m, and a cosine wave possessing a frequency of 0.5 $radsec^{-1}$, amplitude of 0.5 m. The amplitude of the cosine wave was

Table 4.2: Measured features of various controllers in regulating the hexacopter (RT: rise time, ST: settling time, CH: constant height, VH: variable height, ms: millisecond, m: meter, MA: maximum amplitude, PSW: periodic square wave, rad: radian)

Desired trajectory	**Measured features**	**Control method**				
		PID	**FFNN**	**TS-fuzzy**	**G-control**	**PAC**
CH (MA 4 m)	RMSE	**0.3551**	0.4221	0.4771	0.4239	0.4204
	RT (ms)	208.97	199.15	259.03	**141.66**	144.63
	ST (ms)	372.82	364.22	368.31	274.51	**247.52**
	Peak (m)	4.0272	4.0704	4.0714	4.0909	**4.0015**
VH with sharp change (MA 9 m)	RMSE	**0.5574**	0.7588	0.7607	0.6491	0.6537
	RT (ms)	205.77	197.21	209.19	**122.12**	127.84
	ST (ms)	8368.9	8649.9	8412.3	**8249.2**	8279.0
	Peak (m)	9.0281	9.0406	9.0216	9.0022	**9.0010**
VH with smooth change (MA 13 m)	RMSE	0.3642	0.3651	0.1013	0.0273	0.0268
	RT (ms)	122.21	144.33	6.7804	5.2776	**2.4948**
	ST (ms)	9932.5	9938.4	**9869.4**	9929.0	9927.0
	Peak (m)	12.987	12.868	13.007	**12.999**	13.002
Step function (MA 3 m)	RMSE	**0.2420**	0.2795	0.3078	0.2842	0.2834
	RT (ms)	203.94	197.22	215.14	121.83	**112.14**
	ST (ms)	445.12	432.87	396.36	**300.50**	451.29
	Peak (m)	3.0289	3.0676	3.0394	**3.0040**	3.6242
Staircase function (MA 12 m)	RMSE	**0.5221**	0.6151	0.6959	0.5999	0.6237
	RT (ms)	6019.9	6030.9	5970.0	6004.2	**4227.9**
	ST (ms)	8245.3	8285.4	8212.7	**8179.5**	8391.5
	Peak (m)	12.026	12.039	12.017	**11.998**	12.587
Sum of sine function (MA	RMSE	1.2270	1.7091	1.2636	1.0956	**1.0856**
	Peak (m)	**11.129**	11.235	11.426	11.413	11.409
11 m) Pitching	RMSE	0.3513	0.0451	N/A	0.0466	**0.0109**
	RT (ms)	14.907	14.686	N/A	65.469	**10.135**
	ST (ms)	10057	10057	N/A	**9982.9**	10053
	Peak (rad)	0.5615	0.5758	N/A	**0.5398**	0.5469
Rolling	RMSE	0.1673	N/A	N/A	0.0290	**0.0259**
	RT (ms)	166.116	N/A	N/A	118.75	**91.596**
	ST (ms)	10037	N/A	N/A	**9978.9**	9979.9
	Peak (rad)	0.3907	N/A	N/A	0.3513	0.4852

substituted with 0.4 m in rolling mode, where a far precise tracking was witnessed from the PAC than PID. Interestingly, both the adaptive FFNN controller and TS-fuzzy controller failed to track the rolling trajectory, which is the reason for their absence in Figure 4.8 (a). Insertion of PAC yielded better tracking of pitching position, which is obvious from the lowest RMSE of 0.01 as recorded in Table 4.2.

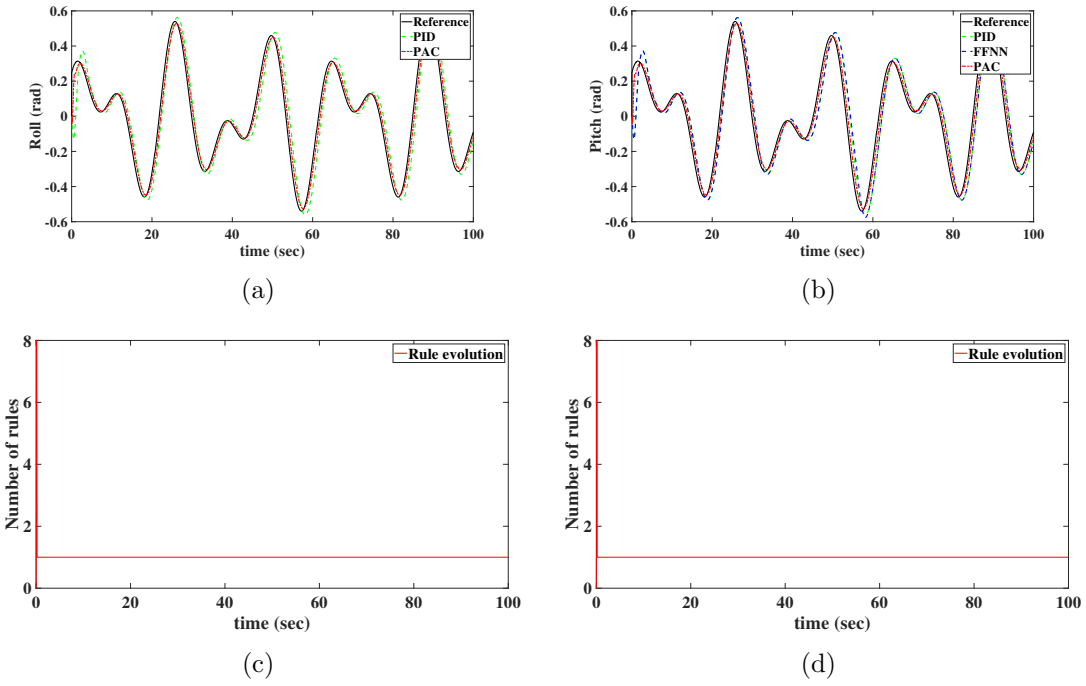


Figure 4.8: Performance observation of different controllers in tracking desired (a) rolling, (b) pitching of the hexacopter MAV, (c) evolution of rules in tracking rolling, and (d) pitching in hexacopter

To sum up, superior or comparative tracking of trajectories were witnessed in the proposed PAC. Additionally, faster responses were obtained than the benchmark controllers, testifying the benefits of having an evolving structure with minimal network parameters.

4.6.3 PAC's robustness against uncertainties and noise

In this work, a variety of disturbances were inserted in both BI-FWMAV and hexacopter's plant to verify PAC's robustness against those disturbances. For instance, in the plant dynamics of BI-FWMAV, a sudden noise with a peak of 3 m and duration of 0.1 seconds, and the discrete wind gust model Matlab block with a wind velocity 4 ms^{-1} immediately after 2 seconds was embedded in the plant. The mathematical representation of the discrete gust model is as follows:

$$V_{wind} = \begin{cases} 0 & x < 0 \\ \frac{V_m}{2}\left(1 - \cos\left(\frac{\pi x}{d_m}\right)\right) & 0 \leq x \leq d_m \\ V_m & x > d_m \end{cases} \tag{4.70}$$

where V_m is the amplitude of the gust, d_m is the length of the gust, x is the distance traveled, and V_{wind} is the resultant wind velocity in the body axis frame.

Effects of both wind gust and sudden peak noise was observed for all six different altitude trajectories of BI-FWMAV, which are depicted in Figure 4.9. From a closer view, an obvious performance degradation in dealing with disturbances was witnessed from the non-adaptive PID controller in all cases. In the FFNN controller, due to the adaptation of the network parameters, it performed better than the PID. Sometimes, the TS-fuzzy controller performs better than PID. However, its performance was not consistent for all the trajectories. Both FFNN and TS-fuzzy controllers suffered severely in tracking trajectories with sharp changes because of the absence of structure adaptation mechanism. On the other hand, quicker settlement and recovery from the adverse effect of gust were sighted from the proposed evolving controller in our numerical experiments. At the same time, the proposed PAC dominated all the benchmark controllers

in rejecting sudden peak noise since the lowest peak was viewed from PAC. Such accomplishments were possible due to the evolving structure with an adaptation of fewer parameters.

In the hexacopter dynamics, a sharp peak noise with an amplitude of 2 m and a period of 0.1 seconds was implanted to observer robustness of the controllers. Effects of disturbance was witnessed for four different altitude trajectories of hexacopter. RMSE, settling time, rise time, and peak overshoot values for all those trajectories were tabulated for all benchmark and proposed controller in Table 4.4. Such perturbation was handled effectively by PAC than its static counterparts as attested in Figure 4.10. A high peak and slow settlement was detected in PID, TS-fuzzy and FFNN controllers. On the contrary, a negligible overshoot with rapid settlements were inspected from the proposed PAC. For example, after closely observing the constant altitude trajectory in Figure 4.10 (c), recorded values of rising time were less than 6 ms from the evolving controllers, which was more than 100 ms in FFNN and PID controller. A similar phenomenon was observed in remaining trajectories, which is evidently declaring the improved robustness against uncertainties of the PAC in contrast with the benchmark static controller.

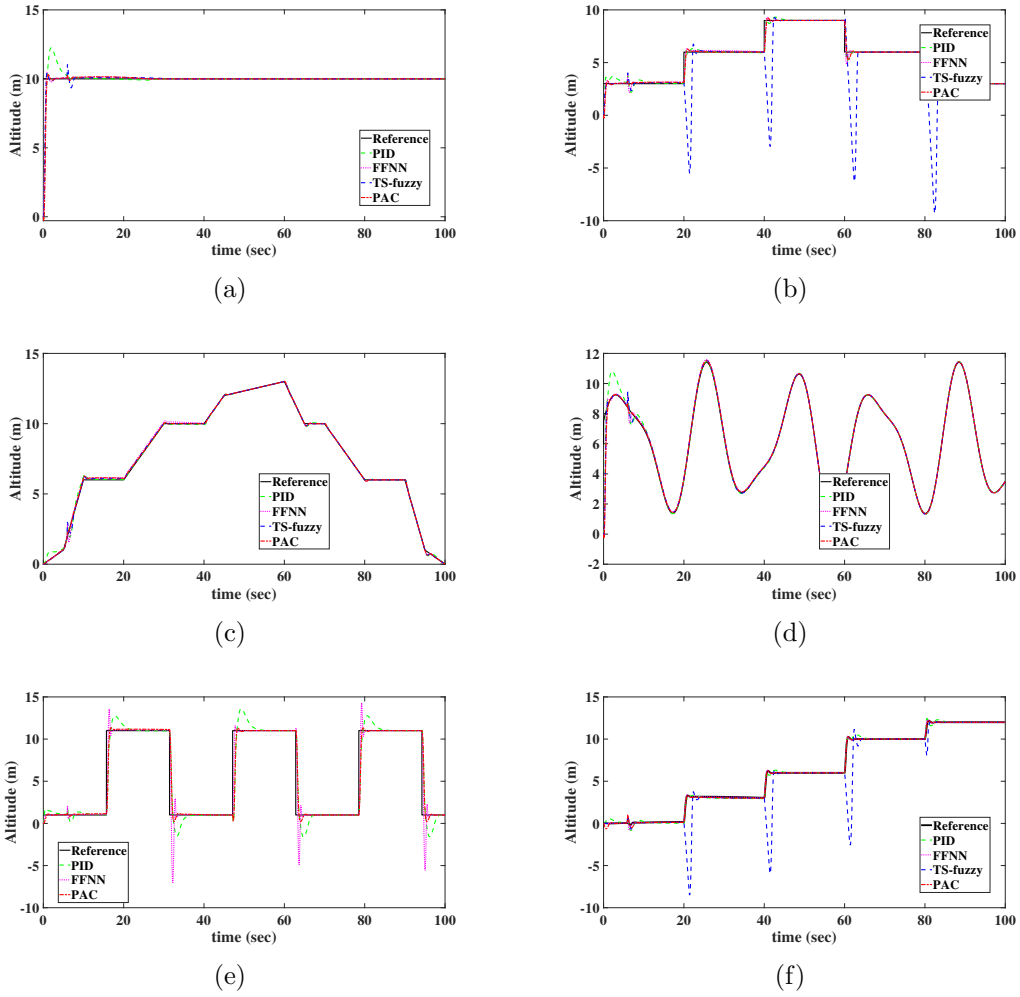


Figure 4.9: Performance observation of different controllers in tracking altitude of BI-FWMAV by considering sudden noise and wind gust uncertainty, when the trajectories are (a) constant hovering, (b) variable heights with sharp edges, (c) variables height with smooth edges, (d) sum of sine function, (e) periodic square wave function, and (f) staircase function

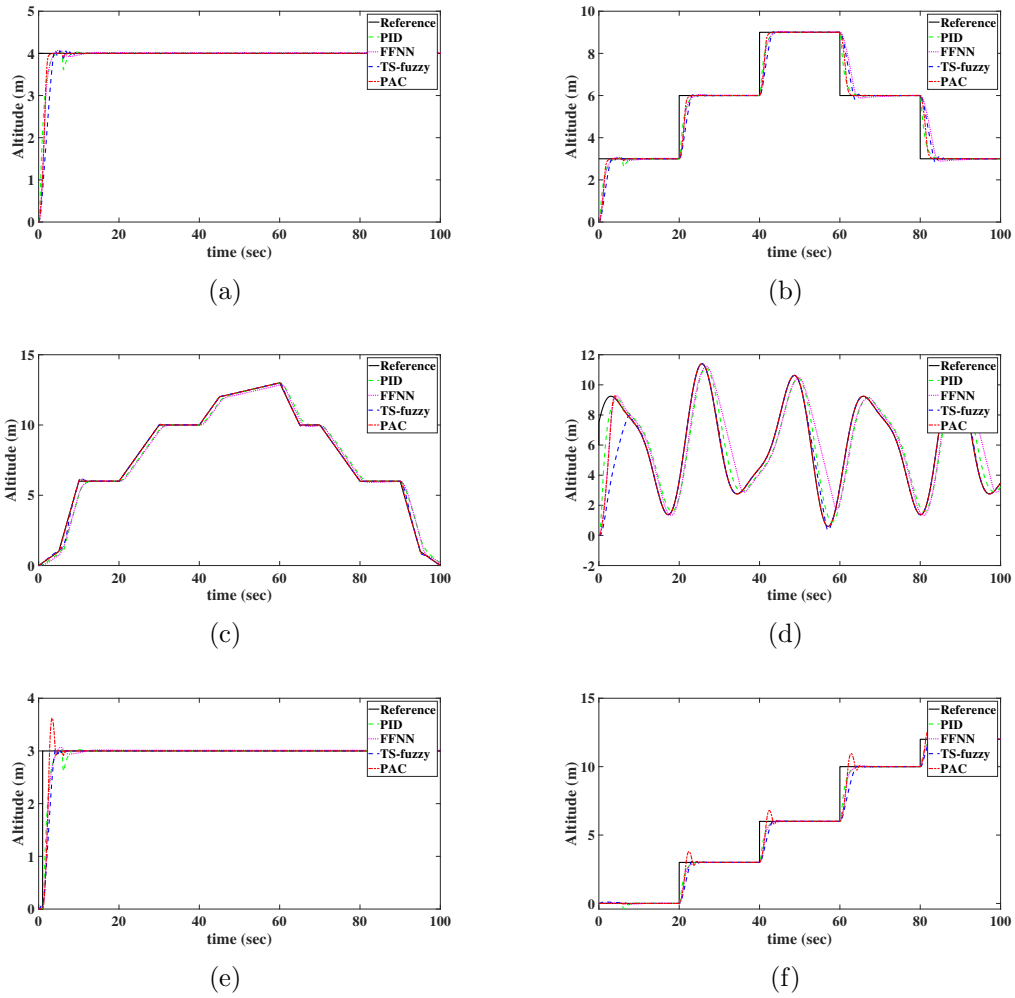


Figure 4.10: Performance observation of different controllers in tracking altitude of hexacopter considering sudden noise, when the trajectories are (a) constant hovering, (b) variable heights with sharp edges, (c) variable heights with smooth edges, (d) sum of sines function, (e) step function, and (f) staircase function

Table 4.3: Measured features of various controllers in operating the BI-FWMAV by considering a noise of sudden peak amplitude, and wind gust disturbance (RT: rise time, ST: settling time, CH: constant height, VH: variable height, SS: sum of sine, ms: millisecond, m: meter, MA: maximum amplitude, PSW: periodic square wave)

Desired trajectory	**Measured features**	**Control method**				
		PID	**FFNN**	**TS-fuzzy**	**G-control**	**PAC**
CH (MA 10 m)	RMSE	0.6536	0.7268	**0.5746**	0.6657	0.6712
	RT (ms)	50.772	55.828	44.629	**41.208**	47.207
	ST (ms)	1025.9	2707.2	743.39	**635.53**	747.43
	Peak (m)	12.247	11.573	11.035	11.022	11.035
VH with sharp change (MA 9 m)	RMSE	0.3430	0.4237	2.4603	**0.3351**	0.3611
	RT (ms)	23.931	48.943	43.949	50.892	**13.728**
	ST (ms)	8176.2	8386.3	8329.4	**8133.2**	8166.5
	Peak (m)	9.3731	9.6742	9.3010	**9.0073**	9.2270
VH with smooth change (MA 13 m)	RMSE	0.1258	0.1541	0.0823	**0.0613**	0.0899
	RT (ms)	8.8573	11.231	1.6537	4.1881	**0.1314**
	ST (ms)	9884.3	**9857.5**	9871.1	9870.5	9872.1
	Peak (m)	13.007	13.009	**13.004**	13.019	13.006
SS function (MA 11 m)	RMSE	0.4832	0.5565	**0.4685**	0.4998	0.5075
	Peak (m)	**11.468**	11.712	11.518	11.489	11.534
PSW function (MA 11 m)	RMSE	2.7739	3.2660	N/A	**2.5112**	2.5122
	RT (ms)	546.47	472.35	N/A	**57.067**	59.508
	ST (ms)	9923.9	9911.7	N/A	**9603.2**	9634.1
	Peak (m)	12.794	12.667	N/A	**12.067**	12.073
Staircase function (MA 12 m)	RMSE	0.3205	0.3960	2.1397	**0.2916**	0.3131
	RT (ms)	5996.0	4067.3	**4024.4**	5999.4	6002.2
	ST (ms)	8370.8	8156.2	8094.8	8056.5	**8055.1**
	Peak (m)	12.453	12.458	12.072	**12.007**	12.198

Table 4.4: Measured features of various controllers in regulating the hexacopter by considering noise of sudden peak amplitude (RT: rise time, ST: settling time, CH: constant height, VH: variable height, ms: millisecond, m: meter, MA: maximum amplitude)

Desired trajectory	**Measured features**	**Control method**				
		PID	**FFNN**	**TS-fuzzy**	**G-control**	**PAC**
CH (MA 4 m)	RMSE	**0.3383**	0.4067	0.4772	0.4237	0.4048
	RT (ms)	208.28	197.14	259.03	**141.66**	144.67
	ST (ms)	746.30	831.62	634.67	**274.51**	623.97
	Peak (m)	4.0293	4.0704	4.0714	4.0909	**4.0015**
VH with sharp change (MA 9 m)	RMSE	**0.5367**	0.7586	0.7479	0.6509	0.6391
	RT (ms)	203.76	195.22	209.19	**122.12**	125.84
	ST (ms)	8315.9	8583.9	8411.8	8257.2	**8232.0**
	Peak (m)	9.0281	9.0406	9.0258	9.0022	**9.0010**
VH with smooth change (MA 13 m)	RMSE	0.3788	0.3666	0.1189	0.0322	0.0281
	RT (ms)	110.21	142.33	6.7867	5.2776	**2.4948**
	ST (ms)	9879.5	9877.4	**9869.2**	9940.0	9871.0
	Peak (m)	12.987	12.868	13.005	13.008	**13.002**
Step function (MA 3 m)	RMSE	**0.2439**	0.2795	0.3078	0.2841	0.2834
	RT (ms)	203.94	197.23	215.13	121.83	**112.14**
	ST (ms)	763.38	912.24	396.34	**300.48**	631.18
	Peak (m)	3.0285	3.0676	3.0394	**3.0042**	3.6242
Staircase function (MA 12 m)	RMSE	**0.5074**	0.6000	0.6961	0.5998	0.6078
	RT (ms)	5993.9	5980.9	5973.7	6004.0	**4144.9**
	ST (ms)	8204.3	8222.4	8212.7	**8183.5**	8254.5
	Peak (m)	12.028	12.039	12.021	**11.998**	12.587
Sum of sine function (MA	RMSE	1.2123	1.7003	1.4325	1.0954	**1.0787**
	Peak (m)	**11.130**	11.235	11.400	11.413	11.409
11 m) Pitching	RMSE	0.3513	0.0451	N/A	0.0466	**0.0109**
	RT (ms)	14.907	14.686	N/A	65.469	**10.135**
	ST (ms)	10057	10057	N/A	**9982.9**	10053
	Peak (rad)	0.5615	0.5758	N/A	**0.5398**	0.5469
Rolling	RMSE	0.1673	N/A	N/A	0.0290	**0.0259**
	RT (ms)	166.116	N/A	N/A	118.75	**91.596**
	ST (ms)	10037	N/A	N/A	**9978.9**	9979.9
	Peak (rad)	0.3907	N/A	N/A	0.3513	0.4852

4.6.4 Self-adaptive mechanism of PAC

Based on the bias-variance concept explained in section 4.5, rules of the PAC have been evolved dynamically in different experiments. Before analyzing the evolution of the structure of our proposed AICon, we have tried to summarize shortfalls of the benchmark controllers used in the experiments explained in this chapter. The PID controller's realization is based upon three gain parameters namely proportional, integral and differential gain. They are typically denoted as K_p, K_i, and K_D. It requires to set values for those parameters in offline before utilizing in control operation, which may oblige repetitious efforts. Besides, those parameters can not be tuned online. Before performing the control operation, both the adaptive FFNN and TS-fuzzy controllers require offline training encouraged by the PID controller's input-output datasets. Though they adapt their network parameters during operation, they have a fixed structure with a hidden layer consists of ten fixed nodes in FFNN and five rules in TS-fuzzy controller. In contrast with the conventional evolving controllers, our proposed PAC has no premise parameters. The only parameter that needs to be adapted is weight, which is adapted here using SMC theory. Unlike the traditional evolving controllers, PAC is free from predefined problem-dependent parameters for regulating its structure.

The structure evolution in terms of added or pruned rules for some trajectories of BI-FWMAV and hexacopter is disclosed graphically in Figure 4.6 (c) and 4.6 (d), Figure 4.7 (c), and in Figure 4.8 (c) and 4.8 (d) to get a vivid insight into the evolution of rules in PAC. For further clarification, the fuzzy rule extracted

by PAC in controlling the BI-FWMAV can be expressed as follows:

$$R^1 : \text{IF } X_n \text{ is close to } \Big([1, e, \dot{e}, y_r] \times [0.0121, 0.0909, 0.4291, 0.6632]^T\Big), \quad (4.71)$$
$$\text{THEN } y = 0.0121 + 0.0909e + 0.4291\dot{e} + 0.6632y_r$$

where e is the error i.e. the difference between the reference and actual output of the plant, $\dot{e}$ is the error derivative i.e. the difference between the present and previous state error value, y_r is the reference for the plant to be controlled. Since PAC is targeted to minimize the tracking error to zero or very close to zero, it needs information about the error as an input to the closed-loop system. It is also witnessed in PAC's rule as exposed in Equation (4.71). When PAC was controlling the BI-FWMAV in tracking a constant altitude of 10 m, it generated 3 rules within 1 second at the beginning of control operation. Since the reference is unaltered and stability of the plant is achieved, PAC does not add or prune any extra rule later on as witnessed from Figure 4.6 (c). While BI-FWMAV was following a variable height trajectory with sharp changes at edges, the PAC starts operating by producing only one rule. After 8 seconds it adds two more rules and achieved system stability. After that, the changes in trajectories are handled by PAC only through tuning of weights only. It does not need any further structure evolution as observed in Figure 4.6 (d). The successful evolution of rules by confirming system stability was also achieved by PAC in controlling the hexacopter plant. For instance, PAC supports hexacopter to track a constant altitude of 4 m with two rules, as displayed in Figure 4.7 (c). While PAC was regulating the rolling of hexacopter for a sum of sine trajectory, it started operating with only one rule. Immediately after 17 seconds, it added another two rules; however one of them was pruned at 19 seconds to minimize the

overfitting phenomenon, while maintained system stability with two rules only. With these two rules, it tracked the trajectory efficiently through the SMC based weight adaptation. A similar scenario was witnessed in controlling the pitching position of the hexacopter since the same trajectory was employed. To sum up, by adapting both the structure and weights, the PAC was controlling the MAVs effectively to follow the desired trajectories very closely and by preserving the stability of the closed-loop system.

4.6.5 Performance evaluation of RedPAC in controlling quadcopter

RedPAC is tested using popular drone simulator namely Dronekit SITL (Software in the Loop) [218]. The dynamics of a SOLO quadcopter is simulated in this work. SOLO has a weight of about 1.5 kg, and it can carry a payload of 0.4 kg. Autonomous missions like search and target tracking using SOLO quadcopter is reported in the literature [219]. Inside the simulator, environmental perturbation such as measurement noise and wind gust are considered. The RedPAC controller code is executed in a MATLAB-Simulink environment and connected to the Dronekit SITL using a user datagram protocol (UDP) localhost communication. The code is made publicly accessible in [220]. In this simulation work, the RedPAC is used to control the altitude of the drone. The controller is expected to perform an altitude-hold mode operation and to track a varying altitude reference. The RedPAC generates the required thrust as denoted by u in Figure 4.5 to maintain the desired altitude. The roll, pitch, and yaw are controlled by conventional PID controllers.

In this chapter, simulation comparison between PID, SMC, PAC, and RedPAC for the quadcopter is provided to clarify that the proposed RedPAC is

Table 4.5: Testing result summary(RT: Rise Time, TS: Settling Time, MP: Maximum Peak, SSE: Steady State Error, RMSE: Root Mean Square Error)

Controller Type	Test	RT (s)	ST (±5%) (s)	MP (m)	SSE	RMSE
PID	I	5.1	27	6.2	0.05	0.87
	II	6	32	6.9	0.1	1.27
	III	6.2	20.5	6.6	0.2	1.03
	IV	5	25	6.2	0.3	0.80
	V	7.5	40	8.2	0.1	2.30
	Avg.	**5.96±1.01**	28.90 ± 7.45	5.96 ± 0.83	6.82 ± 0.83	1.25 ± 0.61
SMC	I	6.5	27	6.3	0.2	1.06
	II	7	26	6.7	0.01	1.42
	III	5.5	24	6.2	0.3	0.81
	IV	6.5	25	6.3	0.2	0.82
	V	5.5	27	6.3	0.2	0.69
	Avg.	6.20 ± 0.67	25.80 ± 1.30	6.36 ± 0.19	0.19 ± 0.09	0.96 ± 0.29
PAC	I	6	30	6	0.1	1.07
	II	7	27	5.9	0.05	1.11
	III	5	30	5.7	0.2	0.66
	IV	6	20	6	0.05	0.99
	V	6.1	20	7	0.1	1.23
	Avg.	6.02 ± 0.71	**25.4±5.08**	6.12 ± 0.51	**0.1±0.06**	1.01 ± 0.21
RedPAC	I	7	25	6	0.2	1.05
	II	6	25	6	0.05	0.84
	III	6	27	5.9	0.1	0.81
	IV	6	28	6	0.2	1.27
	V	5.7	27	5.9	0.1	0.62
	Avg.	6.14 ± 0.5	26.04 ± 1.34	**5.96±0.05**	0.13 ± 0.07	**0.96±0.25**

working better than the other linear and nonlinear ones and comparable to the predecessor controller. For a fair comparison, the parameter of PID and SMC in this simulation are set equally ($K_i = K_1 = 20$; $K_p = K_2 = 0.02$; $K_d = K_3 = 0.002$). The test was repeated five times to clarify the controllers' performance at the different unpredictable disturbances. The altitude hold performance was observed for all those controllers for 50 seconds and the desired altitude was 5 meters.

The response of quadcopter's altitude-hold initially sets to reach 2.5 meters and changes to 5 meters immediately. Below 3 seconds the altitude is reduced as the effect of disturbance to the system. All controllers were able to follow the reference signal by minimizing the steady-state error, and rejecting the disturbance influence as shown in Figure 4.11.

From the simulation results, it is observed that the PID controller had an over-

shoot with a maximum peak (MP) of around 0.6 meters. A very small overshoot of around 0.05 meters was observed from our proposed controller RedPAC. A comparatively lower MP than PID was caused by the adaptation of the evolving part, which reduced the control-input signal during the transient phase. While considering the rise time (RT) of PAC and RedPAC, they were comparable with PID and SMC. However, the settling time (ST) of PAC and RedPAC were slightly smaller than the other two controllers, which is proving the faster settling capacity of the evolving controllers than their linear or nonlinear counterparts.

The number of rules for both PAC and RedPAC was evolving from one to six and five respectively as shown in Figure 4.12. This compact number of rules is guaranteeing the proposed controller's ability to handle the complex nonlinear and uncertain quadcopter dynamics. It is also indicating that only a few rules are sufficient for the networks to meet the system's requirements. The robustness in evolving controllers structure is also validated since the rule is growing during the early transient-state and holding to a fixed value at the steady-state condition. This evolution of structure is effectively solving the limitations of the conventional controller to deal with the uncertainties from the environment.

To sum up, from the average of all five observations, the fastest RT is witnessed while the PID controller is utilized. These are recorded in Table 4.5 for an altitude-hold condition. From the PAC, the fastest settling time and minimum steady-state error are attained. Compared to the benchmark controllers, RedPAC performs better with a lower MP and lower root mean square error (RMSE). Both the PAC and RedPAC have obtained the best performance in two different performance criteria, which is indicating that they are comparable to each other with a narrow performance difference. The number of weight parameters is smaller by one-third in RedPAC than the PAC since RedPAC has three inputs

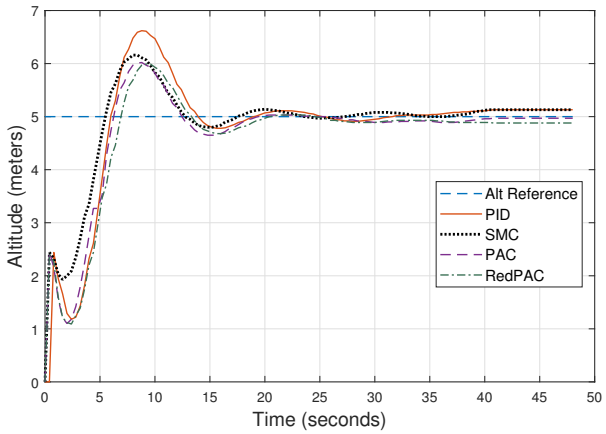


Figure 4.11: Comparison among various controllers' performance while controlling the altitude of the Dronekit simulator

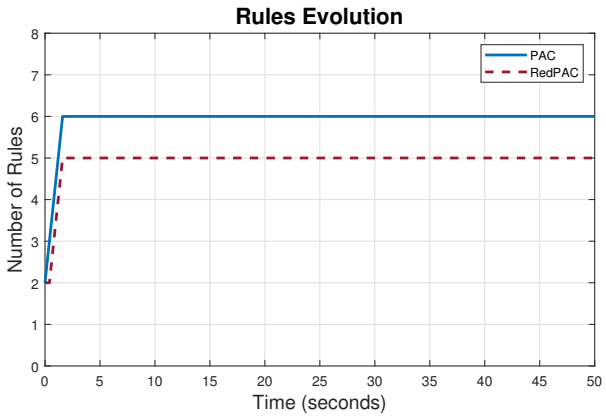


Figure 4.12: Evolution of rules in RedPAC

while maintaining a comparable or better performance.

4.7 Summary

Based on the research gap in controlling MAVs in cluttered environments, two AICons, namely PAC and RedPAC are proposed in this chapter. A bottleneck of the existing AICons is the utilization of numerous free parameters and their tuning. Such inadequacy has been mitigated in the PAC since it has no premise parameters. The only parameter used in PAC to acquire the desired tracking is the weight. Apart from that, conventional AICons adhere to user-defined

problem-based thresholds to shape their structure. In PAC and RedPAC, rather than predefined parameters, the bias-variance concept based network significance method is utilized to determine their's structure. The PAC has been verified by implementing them in BI-FWMAV and hexacopter to track diverse trajectories. Achievements are contrasted with a commonly utilized linear controller PID, an adaptive nonlinear FFNN controller, TS-fuzzy controller. Furthermore, controllers' robustness against uncertainties and disruptions is ascertained by injecting a wind gust and sudden peak to the MAVs dynamics. In controlling both plants with uncertainties, lower or comparable overshoot and settling time were observed from PAC with a simplified evolving structure, which is testifying its robustness against uncertainties and compatibility in regulating MAVs. The simulation for RedPAC is completed with a quadcopter testbed (Dronekit-SITL) which behaves very close to the real system. Better or comparable tracking performance is witnessed from RedPAC than its modeled or model-free counterparts. In the next chapter, another AICon using multivariate Gaussian function is described.

Chapter 5

Generic Evolving Neuro-Fuzzy-based Autonomous Intelligent Controller for UAVs

The work presented in this chapter has been published in the following articles:

- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A., Pan, Y. (2019). Generic evolving self-organizing neuro-fuzzy control of bio-inspired unmanned aerial vehicles. *IEEE Transactions on Fuzzy Systems* DOI: 10.1109/TFUZZ.2019.2917808.
- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A. (2018). A Generic Self-Evolving Neuro-Fuzzy Controller based High-performance Hexacopter Altitude Control System. *In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* DOI: 10.1109/SMC.2018.00475.

Abstract

In this chapter, an autonomous intelligent controller (AICon) namely Generic-controller (G-controller) is proposed. It is developed by incorporating the sliding mode control (SMC) theory with an advanced incremental learning machine, namely Generic Evolving Neuro-Fuzzy Inference System (GENEFIS). The controller starts operating from scratch with an empty set of fuzzy rule, and therefore, no offline training is required. To cope with changing dynamic characteristics of the plant, the controller can add or prune the rules on demand. Control law and adaptation laws for the consequent parameters are derived from the SMC algorithm to establish a stable closed-loop system, where the stability of the G-controller is guaranteed by using the Lyapunov function. The uniform asymptotic convergence of tracking error to zero is witnessed through the implication of an auxiliary robustifying control term. In addition, the implementation of the multivariate Gaussian function helps the controller to handle the non-axis parallel data from the plant and consequently enhances the robustness against the uncertainties and environmental perturbations. Finally, the controller's performance has been evaluated by observing the tracking performance in controlling simulated plants of bio-inspired flapping wing micro air vehicle (BIFW MAV) and hexacopter unmanned aerial vehicle (UAV) for a variety of trajectories.

5.1 Introduction

Obtaining an accurate first principle model for many UAV systems is considerably arduous due to their highly non-linear, over-actuated or under-actuated behavior. Besides, various uncertainty factors like impreciseness in the data obtained from sensors, induced noise by the sensors, outdoor environmental uncertainties like

wind gust, motor degradation, etc. are difficult or impossible to integrate into the first principle models. In such circumstances, approaches without the necessity of accurate mathematical models of the system are much appreciated. Being a model-free approach [129], the NN and FLS-based intelligent controllers have been successfully implemented in many control applications [221, 222] over the past few years. Recently, systems with an amalgamation of FLS and NN, namely Fuzzy Neural Network (FNN) based controllers, are becoming popular. FLS, NN, and FNN are employed in a variety of engineering applications [223].

To handle uncertainties in control applications, researchers have tried to develop adaptive controllers by combining FLS, NN, FNN systems SMC [224], H_∞ control, back-stepping, etc. Such amalgamation empowers the FLS, NN, FNN controllers with the feature of parameter-tuning, which provides an adaptive control structure. It assists them to mitigate the adverse effects of various uncertainties and perturbations. However, such adaptive FNN-based controllers are not able to evolve their structures by adding or pruning rules. Therefore, the number of rules needs to be determined a priori in adaptive FNN-controllers, where a selection of a few fuzzy rules may hinder to achieve the desired control performance. On the other hand, consideration of too many rules may create complexity in a hardware implementation.

The problems mentioned above can be circumvented by implementing evolving FLS, NN, FNN-based AICon [143]. They can evolve their structure by adding, or deleting rules through self-organizing techniques. All the AICons discussed in chapter 2 have utilized univariate Gaussian function, which does not expose the scale-invariant property. Besides, they are not effective in dealing with non-axis parallel data distribution. To mitigate these shortcomings, we utilized a multivariate Gaussian functions based incremental learning algorithm called

Generic Evolving Neuro-Fuzzy Inference System (GENEFIS) [1]. In this work, GENEFIS is amalgamated with SMC technique to develop an AICon, namely G-controller.

5.2 Contribution

Main features of the proposed G-controller are as follows:

1. The design of the proposed G-controller does not depend upon the plant dynamics or any other features of the plant.

2. No previous information or off-line training is required. Thus, the controller starts self-construction from scratch with only one rule at the beginning, and then it adds or deletes rules to follow the desired trajectory. Besides, the application of a fast kernel-based metric approach helps to capture the fuzzy set and rule level redundancy.

3. Integration of the Generalized Adaptive Resonance Theory+ (GART+) helps to upgrade the premise parameters with respect to input data distributions, and utilization of multivariate Gaussian function aids the controller to handle a variety of data generated from the sudden change in plants, or from uncertainties, environmental perturbations.

4. Adaptation laws for the GENEFIS based G-controller's consequent parameters are derived from the SMC learning theory, which confirms a stable closed-loop control system. Instead of predefined values, the sliding parameters in the SMC theory is also self-organizing in this controller. A robustifying auxiliary control term ensures uniform asymptotic convergence

of tracking error to zero. Finally, the stability of the G-controller is proved using the Lyapunov function.

5. Successful evaluation of the proposed G-controller through implementing it into the simulated BIFW MAV and hexacopter plant.

The above-mentioned characteristics of the proposed autonomous G-controller make them an appropriate candidate to control autonomous vehicles like BIFW MAV, quadcopter, hexacopter, octocopter etc. with better accuracy than a stand-alone FPT-based controller. Furthermore, the control algorithm is developed using C programming language to make it compatible with all types of hardware, where their implementation is made publicly available in [225].

The organization of rest of this chapter is as follows: The evolving architecture of the GENEFIS based G-controller is explained in Section 5.3. Section 5.4 represents SMC learning algorithm based adaptation of the proposed G-controller. The results are summarized, and analysed in Section 5.5. Finally, the chapter ends with the concluding remarks encompassed in Section 5.6.

5.3 Architecture of the Evolving G-Controller

The self-organizing mechanism of the G-controller is adopted from GENEFIS developed in [1]. GENEFIS is a TS FLS that features multidimensional membership functions in the input space where the contours are ellipsoid in arbitrary positions. Each estimated one-dimensional membership function represents a portion in the input space partition by assigning the Gaussian function's own center and width. Concurrently, in the GENEFIS-based G-controller, first-order polynomials are the consequent part of the fuzzy rules. In G-controller, a typical

fuzzy rule can be expressed as follows:

$$\text{IF } Z \text{ is } R_i, \text{ then } \eta_i = a_{0i} + a_{1i}\zeta_1 + a_{2i}\zeta_2 + ... + a_{ki}\zeta_k \tag{5.1}$$

where Z is an input vector of interest, R_i represents the ith rule (membership function) constructed from a concatenation of fuzzy sets and epitomizing a multi-dimensional kernel, a_{ni} $(n = 1, 2, ..., k)$ are the consequent parameters, k denotes the dimension of input feature, ζ_k is the kth input feature. The predicted output of the self-evolving model can be expressed as:

$$\begin{aligned}\hat{\eta} = \sum_{i=1}^{j} \psi_i(\zeta)\eta_i(\zeta) &= \frac{\sum\limits_{i=1}^{j} R_i\eta_i}{\sum\limits_{i=1}^{j} R_i} \\ &= \frac{\sum_{i=1}^{j} \exp(-(Z-\Theta_i)\Sigma_i^{-1}(Z-\Theta_i)^T)\eta_i}{\sum_{i=1}^{j} \exp(-(Z-\Theta_i)\Sigma_i^{-1}(Z-\Theta_i)^T)}\end{aligned} \tag{5.2}$$

In Equation (5.2), $\Theta_i \in \Re^{1\times j}$ is the centroid of the ith fuzzy rule, j is the number of fuzzy rules, $\Sigma_i \in \Re^{j\times j}$ is a non-diagonal covariance matrix whose diagonal components are expressing the spread of the multivariate Gaussian function, η_i is the consequent part of the ith rule.

5.3.1 Statistical contribution-based rule growing mechanism

The Datum Significance (DS) method developed in [48] is utilized as a rule growing mechanism in G-controller. The original DS method is geared into the multivariate Gaussian membership function and polynomial consequents, which is the crux of GENEFIS [1]. The integration of the multivariate Gaussian

membership function into the original DS method can be observed here as follows:

$$D_{sgn} = |e_{rn}| \int_Z \exp\left(-\frac{(Z - Z_n)\Sigma^{-1}(Z - Z_n)^T}{(Z - \Theta)\Sigma^{-1}(Z - \Theta)^T}\right) \frac{1}{\mathcal{H}(Z)} dz \tag{5.3}$$

where D_{sgn} denotes the significance of the nth datum, Z_n is the current input to the controller in a closed-loop control system $Z_n \in \Re$, and $\mathcal{H}(Z)$ is the range of input Z. In a closed-loop control system, error (e) indicates the difference between the desired reference and plant's output, which is usually fed as input to the controller. Error e_{rn} mentioned in Equation (5.3) can be expressed as:

$$|e_{rn}| = |tr_n - \eta_n| \tag{5.4}$$

where tr_n is the plant's output of the closed-loop control system, and η_n is control output from the G-controller at nth time instant. After applying $k-$fold numerical integration to Equation (5.3), the following is obtained:

$$D_{sgn} = |e_{rn}| \left(\frac{\det(\Sigma_{j+1})}{\mathcal{H}(Z)}\right)^k \tag{5.5}$$

When the statistical contribution of the datum is higher than the existing rules, it becomes an appropriate candidate to be a new rule. Therefore, the DS criterion can be amended mathematically as follows:

$$D_{sgn} = |e_{rn}| \frac{\det(\Sigma_{j+1})^k}{\sum_{i=1}^{j+1} \det(\Sigma_i)^k} \tag{5.6}$$

When a sample lies far away from the nearest rule, a high value of D_{sgn} is obtained from Equation (5.6) even with a small value of e_{rn}. In such a situation, generalization capability of the self-evolving neuro-fuzzy controller remains good

without the addition of any new rules. Therefore, a high value of D_{sgn} does not always indicate the necessity of a rule evolution. On the other hand, a high value of e_{rn} may be observed in an overfitting phenomenon. In such a case, the addition of a new rule may worsen the overfitting phenomenon. Thus, a separation is needed in Equation (5.6) to cover the two above-mentioned discernible situations.

To overcome an overfitting scenario, it is important to monitor the effect of a newly injected sample on e_{rn}, since structural learning is not occurring in every observation. In other words, the rule growing mechanism is probably turned on when the rate of change of e_{rn} is positive. In the proposed controller, the mean and variance of e_{rn} are measured by recursively updating e_{rn} and standard deviation [53] as follows:

$$\bar{e}_{rn} = \frac{n-1}{n}\bar{e}_{rn-1} + \frac{1}{k}\bar{e}_{rn} \tag{5.7}$$

$$\bar{\sigma}^2_{rn} = \frac{n-1}{n}\bar{\sigma}^2_{rn-1} + \frac{1}{k}(\bar{e}_{rn} - \bar{e}_{rn-1}) \tag{5.8}$$

When $\bar{e}_{rn} + \bar{\sigma}^2_{rn} - (\bar{e}_{rn-1} + \bar{\sigma}^2_{rn-1}) > 0$, the DS criterion is simplified in our chapter as follows:

$$D_{sgn} = \frac{\det(\Sigma_{j+1})^k}{\sum_{i=1}^{j+1}\det(\Sigma_i)^k} \tag{5.9}$$

The condition in expanding the rule base utilizing Equation (5.9) is $D_{sgn} \geq g$, where g is a predefined threshold. Equation (5.9) represents an encouraging generalization and summarization of the datum since a new rule can omit possible overfitting effects. Besides, this DS criterion can predict the probable contribution of the datum during its lifetime.

5.3.2 Statistical contribution-based rule pruning mechanism

The Extended Rule Significance (ERS) method was put forward by [48]. The ERS concept appraises the statistical contribution of the fuzzy rules when the number of observations or time instants are approaching infinity. The default ERS theory is not possible to integrate directly into GENEFIS due to the incompatibility of default ERS concept with the concept of neuro-fuzzy structure. ERS theory is modified to fit them with G-controller. In this chapter, the concept of the statistical contribution of the fuzzy rules can be expressed mathematically as follows:

$$\mathcal{E}(i,n) = |\delta_i|\mathcal{E}_i, \text{ where } |\delta_i| = \sum_{i=1}^{k+1} |\eta_i| \tag{5.10}$$

$$\mathcal{E}_i = \int_Z \exp(-(Z-\Theta_i^n)\Sigma_i^{-1}(Z-\Theta_i^n)^T)\frac{1}{\mathcal{H}(Z)}dz \tag{5.11}$$

From Equation (5.10), it can be anticipated that the contribution of fuzzy rules is a summary of the total contribution of input and output parts of the fuzzy rules, where $\mathcal{E}_i$ is expressing the modified version of original input contribution explained in [48, 49], and δ_i is expressing the contribution of output parameters. Usually, inverse covariance matrix Σ_i^{-1} in Equation (5.11) has a smaller size than that of Z, which necessitates an amendment in Equation (5.11) as follows:

$$\mathcal{E}_i \approx \frac{1}{\mathcal{H}(Z)}\left(2\int_0^\infty \exp\left(-\left(\frac{Z^2}{\det(\Sigma_i)}\right)\right)dz\right)^k \tag{5.12}$$

By using the k fold numerical integration, the final version of ERS theory can

be expressed as:

$$\mathcal{E}_{inf}^{i} = \sum_{i=1}^{j+1} \eta_i \frac{\det(\Sigma_i)^k}{\sum_{i=1}^{j} \det(\Sigma_i)^k} \tag{5.13}$$

When $\mathcal{E}_{inf}^{i} \leq k_e$, it is presumed that the clusters or rules cannot capture the latest incoming data to GENEFIS. It can be deduced that the hypervolume of the triggered cluster indicates the significance of the fuzzy rule. Thus, when the volume of the ith cluster is much lower than the summation of volumes of all clusters, that rule is considered as inconsequential. Such a rule is pruned to protect the rule base evolution from its adverse effect. Here, k_e exhibits a plausible trade-off between compactness and generalization of the rule base. The allocated value for δ is $\delta = [0.0001, 1]$, and $k_e = 10\%$ of δ.

5.3.3 Adaptation of the rule premise parameters

Generalized Adaptive Resonance Theory+ (GART+) [226] is used in G-controller as a technique of granulating input features and adapting premise parameters. It is observed that GART [227] and its successor namely improved GART (IGART) [228] suffer from a cluster or rule growing problem. In GART the compatibility measure is done utilizing the maximal membership degree of a new datum to all available rules. In the first round, if the selected rule expresses a higher membership degree than a predefined threshold ρ_a, then it is declared as a winning rule and the match-tracking mechanism is executed. However, if the first round winning rule fails to beat the match-tracking threshold ρ_b, it deactivates that rule and increases the value of threshold ρ_a to find a better candidate. A larger width is required in the next selected cluster to cope with the increased value of ρ_a. Otherwise, it fabricates a new cluster. Nonetheless, a cluster with larger radii may contain more than one distinguishable data clouds and thereby marginalizing the

other clusters in every training episode. In an incremental learning environment, this effect is known as *cluster delamination* [229], pictorially exhibited in Figure 5.1. To relieve from the *cluster delamination* effect in G-controller, the size of the fuzzy rule is constrained by using GART+, which allows a limited grow or shrink of clusters.

5.3.3.1 Improved selection procedure of winning rule

To determine the most compatible or winning rule, Bayes's decision theory is utilized in GART+ [226]. To stimulate a more appropriate selection of the winning rule, the Bayesian concept does not only consider the proximity of a rule or cluster to the inserted datum, but also the dominance of the cluster with respect to the other categories through the measure of cluster's prior-probability. That is, the prior probability can count the number of samples falling in the outreach of the cluster, and is expressed as follows:

$$\hat{P}_r(\psi_i) = \frac{N_i}{\sum_{i=1}^{j} N_i} \tag{5.14}$$

where N_i indicates the number of times that ith cluster or fuzzy rule wins the competition.

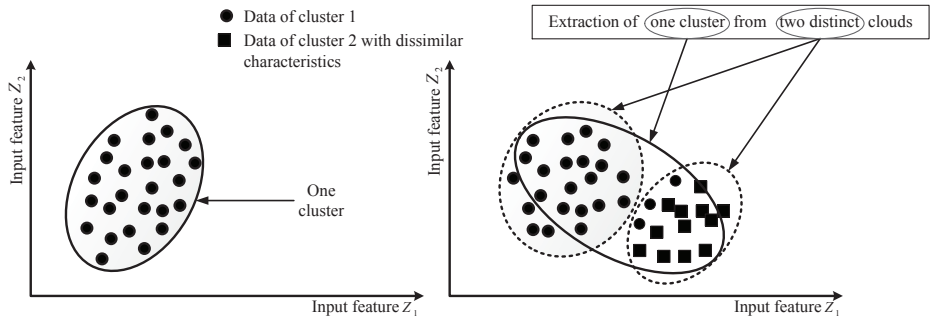


Figure 5.1: Cluster delamination effect (adapted from [1] with proper permission)

When a new input datum to the controller finds two categories with almost similar distances but with different population numbers, Bayes's decision theory assists to select the cluster with more data points and declare it as a winning cluster. In the proposed G-controller, the posterior probability of the ith cluster can be represented as follows:

$$\hat{P}_r(\psi_i|Z) = \frac{\hat{p}_r(Z|\psi_i)\hat{P}_r(\psi_i)}{\sum_{i=1}^{j}\hat{p}_r(Z|\psi_i)\hat{P}_r(\psi_i)} \tag{5.15}$$

where $\hat{p}_r(\psi_i|Z)$ and $\hat{P}_r(\psi_i)$ represents the likelihood and the prior probability correspondingly. The likelihood can also be elaborated as follows:

$$\hat{p}_r(Z|\psi_i) = \frac{1}{(2\pi V_i)^{1/2}}\exp(-(Z-\Theta_i)\Sigma_i^{-1}(Z-\Theta_i)^T) \tag{5.16}$$

where V_i determines the estimated hyper-volume of feature space covered by the ith cluster, which can be expressed as:

$$V_i = \det(\Sigma_i) \tag{5.17}$$

The Bayesian concept presented in Equation (5.15) is implemented in G-controller, which can be interpreted as follows:

1. When a new sample is adjacent to existing categories, it causes a higher likelihood expressed by Equation (5.16).

2. A cluster with a large volume is forced to divide its volume in Equation (5.16), as a consequence, it delivers a lower value of the posterior probability according to Equation (5.15). This is particularly important to avoid large span clusters and to decrease the likelihood of cluster delamination effects.

3. According to Equation (5.14), categories surrounded by more incoming data samples are more worthwhile, which inflicts a high value of posterior probability.

5.3.3.2 Vigilance test

There are two goals to perform the vigilance test [83]. The first one concerns about the capability of the winning cluster to accommodate a new datum. The second goal is to reduce the size of the rule, where a rule is not allowed to have a volume higher than the threshold V_{max}, that is calculated from $V_{max} \equiv \rho_b \sum_{i=1}^{j} V_i$. The vigilance test is a way to compress the procedure of rule-deletion, update, or evolution, where a rule needs to satisfy four different conditions as presented below:

Case I: $R_{win} \geq \rho_a,\ V_{win} \leq V_{max}$

where R_{win} is the membership degree of the winning rule to seize the latest datum. The condition in Case I is indicating the capability of the selected cluster to accommodate the newest datum and emphasizing on the limited size of a cluster. In our proposed G-controller, ρ_a is set close to 1. Contrarily, the value of ρ_b is set as [0.0001, 0.1]. Then the adaptation mechanism of focal point Θ_i, and the dispersion matrix Σ_i is generated by the equations as follows:

$$\Theta_{win}^{new} = \frac{N_{win}^{old}}{N_{win}^{old}+1}\Theta_{win}^{old} + \frac{\left(Z-\Theta_{win}^{old}\right)}{N_{win}^{old}+1} \tag{5.18}$$

$$\Sigma_{win}^{new^{-1}} = \frac{\Sigma_{win}^{old^{-1}}}{1-\alpha} + \frac{\alpha}{1-\alpha} \frac{\left(\Sigma_{win}^{old^{-1}}\left(Z-\Theta_{win}^{new}\right)\right)\left(\Sigma_{win}^{old^{-1}}\left(Z-\Theta_{win}^{new}\right)\right)^{T}}{1+\alpha\left(Z-\Theta_{win}^{new}\right)\Sigma_{win}^{old^{-1}}\left(Z-\Theta_{win}^{new}\right)^{T}} \tag{5.19}$$

$$N_{win}^{new} = N_{win}^{old} + 1 \tag{5.20}$$

where α can be expressed as follows:

$$\alpha = \frac{1}{N_{win}^{old}+1} \tag{5.21}$$

where N_{win}^{old} denotes the number of incoming samples populating the winning cluster.

Besides, a major advantage of utilizing Equation (5.19) is the prompter update of the dispersion matrix (inverse covariance matrix), since a direct adjustment of the dispersion matrix is occurring without the necessity to re-inverse the dispersion matrix [171]. Concerning the conditions in Case I, some pertinent likelihoods may emerge in the rehearsal process and they are outlined as follows:

Case II: $R_{win} < \rho_a,\ V_{win} > V_{max}$

In this circumstance, the input data to the G-controller cannot be touched by any existing rules of the controller, since the inserted input data is hardly covered by any rules. The statistical contribution of the datum needs to be calculated by DS-criterion. When both conditions are satisfied, a new rule is generated and its

parameters are assigned as follows:

$$\Theta_{j+1} = Z \tag{5.22}$$

$$\text{diag}\left(\Sigma_{j+1}\right) = \frac{\max((\Theta_i - \Theta_{i-1}), (\Theta_i - \Theta_{i+1}))}{\sqrt{\frac{1}{\ln(\epsilon)}}} \tag{5.23}$$

where the value of ϵ is 0.5. Equation (5.23) ensures a sufficient coverage of the newly added rule, which is proved in [184]. It helps GENEFIS to explore untouched regions in the feature space fitting a superfluous cluster at whatever point a relatively unexploited region or knowledge is fed, which is a mandatory element to confronting possible non-stationary and evolving qualities of the self-evolving control system. Note that proper initialization of the inverse covariance matrix plays a crucial role in the success of multivariate Gaussian fuzzy rule. Although it meets the $\epsilon-$completeness criterion, Equation (5.23) requires re-inversion phase which sometimes leads to instability when the covariance matrix is not full-rank. As an alternative, the inverse covariance matrix is initialized here as follows:

$$\Sigma_0^{-1} = k_{fs} I \tag{5.24}$$

where k_{fs} is a user-defined parameter, and I is an identity matrix.

Case III: $R_{win} \geq \rho_a,\ V_{win} > V_{max}$

This situation is indicating the capability of the existing rule base to cover the current data easily. However, the width of the chosen cluster is oversized. This datum creates a redundancy when added to the rule base. To mitigate the adverse impact, one of the solutions is to replace the selected cluster merely by

this datum. Then the fuzzy region can be expressed as follows:

$$\Theta_{win} = Z \tag{5.25}$$

$$\Sigma_{win}^{new^{-1}} = \frac{1}{k_{win}} \Sigma_{win}^{old^{-1}} \tag{5.26}$$

where k_{win} is a constant with a value of 1.1, and the width of the cluster is reduced until a desirable fuzzy region is obtained while satisfying $V_{win} \leq V_{max}$.

Case IV: $R_{win} < \rho_a,\ V_{win} \leq V_{max}$

The same action is taken as in Case I, i.e., the adjustment process is executed to stimulate the cluster to move towards the incoming data.

5.4 Adaptation of the rule consequent parameters

In our chapter, an advanced evolving neuro-fuzzy system called GENEFIS is utilized to build the evolving structure and adapt premise parameters of the proposed G-controller, where the integration of multivariate Gaussian function and GART+ method helps the controller to reduce the structural complexity and to adapt with the dynamic behavior of nonlinear UAV plants. On the other hand, being robust enough to guarantee the robustness of a system against external perturbations, parameter variations, and unknown uncertainties, the SMC theory is applied to adapt the consequent parameters of the G-controller. In SMC scheme, the motion of a system is restricted to a plane known as *sliding surface*. The SMC learning theory-based adaptation laws are developed to establish a stable closed-loop system.

By following the regulations of SMC scheme as explained in [212–214], the

zero dynamics of the learning error coordinate is defined as time-varying sliding surface as follows:

$$S_{ssr}(u_g, u) = u_{ARC}(t) = u_g(t) + u(t) \tag{5.27}$$

The sliding surface for the highly nonlinear over-actuated autonomous vehicles, namely FW MAV, and hexacopter plant to be controlled is expressed as:

$$s_H = e + \lambda_1 \dot{e} + \lambda_2 \int_0^t e(\tau) d\tau \tag{5.28}$$

where, $\lambda_1 = \frac{\alpha_2}{\alpha_1}$, $\lambda_2 = \frac{\alpha_3}{\alpha_1}$, e is the error which is the difference between the actual displacement from the plant and desired position in case of altitude control. In this work, in case the BIFW MAV plant, the sliding parameter α_1 is initialized with a small value 1×10^{-2}, whereas α_2 is initialized with 1×10^{-3}, and $\alpha_3 \approx 0$. Each of the parameters is then evolved by using learning rates. These learning rates are set in such a way so that the sliding parameters can achieve the desired value in the shortest possible time to create a stable closed-loop control system. A higher initial value of the sliding parameters is avoided, since it may cause a big overshoot at the beginning of the trajectory. It can be abstracted that, to make our proposed G-controller absolutely autonomous, these sliding parameters are self-organizing rather than predefined constant values.

Definition : After a certain time t_k a sliding motion will be developed on the sliding manifold $S_{ssr}(u_g, u) = u_{ARC}(t) = 0$, where the state $S_{ssr}(t)\dot{S}_{ssr}(t) = u_{ARC}(t)\dot{u}_{ARC}(t) < 0$ to be satisfied for the whole time period with some nontrival semi-open sub-interval of time expressed as $[t,\ t_k) \subset (0,\ t_k)$.

It is expected to produce such online adaptation of consequent parameters of

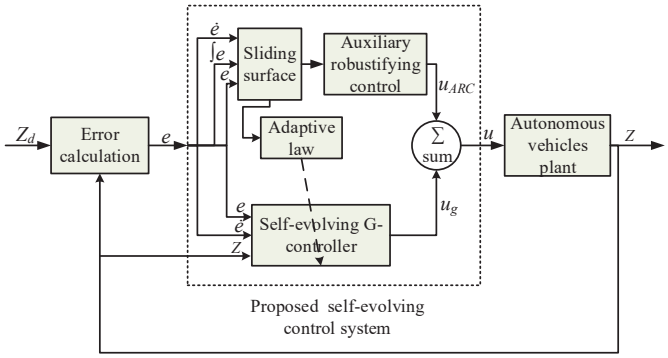


Figure 5.2: Self-evolving G-controller based closed-loop control system

the proposed G-controller that the sliding mode condition of the aforestated definition is enforced. The adaptation process of the proposed method is summarized below.

The adaptation laws for the consequent parameters of the G-controller are chosen as:

$$\dot{\omega}(t) = -\alpha_1 G(t)\psi(t)s_H(t), \quad \text{where} \quad \omega(0) = \omega_0 \in \Re^{nR\times 1} \tag{5.29}$$

where the term $G(t)$ can be updated recursively as follows:

$$\dot{G}(t) = -G(t)\psi(t)\psi^T(t)G(t), \quad \text{where} \quad G(0) = G_0 \in \Re^{nR\times nR} \tag{5.30}$$

where n is the number of inputs to the controller, and R is the number of generated rules. These adaptation laws guarantee a stable closed-loop control system, where the plants to be controlled can be of various order.

Proof : The sliding parameter-dependent robustifying auxiliary control term of the proposed controller can be expressed as follows:

$$u_{ARC}(t) = \alpha_1 s_H \tag{5.31}$$

The robustifying auxiliary control term u_{ARC} may suffer from high-frequency oscillations in the control input. It is an undesirable phenomenon in sliding mode controller and known as chattering effect. Due to simplicity, saturation, or sigmoid functions are mostly used to reduce the chattering effect. In this work, a saturation function is utilized to mitigate the adverse effect of chattering.

The G-controller's final output signal can be expressed as follows:

$$u_g(t) = \psi^T(t)\omega(t) \tag{5.32}$$

The overall control signal as observed in Figure (5.2) can be obtained as follows:

$$u(t) = u_{ARC}(t) - u_g(t) \tag{5.33}$$

The cost function can be defined as:

$$\begin{aligned} J(t) &= \int_0^t s_H^2(\tau)d\tau \\ &= \frac{1}{\alpha_1^2}\int_0^t (u(\tau) + u_g(\tau))^2 d\tau \\ &= \frac{1}{\alpha_1^2}\int_0^t (u(\tau) + \psi^T(t)\omega(\tau))^2 d\tau \end{aligned} \tag{5.34}$$

The gradient of J with respect to ω is as follows:

$$\begin{aligned}
&\nabla_\omega J(t) = 0 \\
&\Rightarrow \int \psi(\tau)u(\tau)d\tau + \omega(t)\int_0^t \psi(\tau)\psi^T(\tau)d\tau = 0 \\
&\Rightarrow \omega(t) = \left[\int_0^t \psi(\tau)\psi^T(\tau)d\tau\right]^{-1}\int_0^t \psi(\tau)u(\tau)d\tau \qquad (5.35)\\
&\Rightarrow \omega(t) = -G(t)\int_0^t \psi(\tau)u(\tau)d\tau \qquad (5.36)\\
&\Rightarrow G^{-1}(t)\omega(t) = -\int_0^t \psi(\tau)u(\tau)d\tau \qquad (5.37)
\end{aligned}$$

where,

$$G(t) = \left[\int_0^t \psi(\tau)\psi^T(\tau)d\tau\right]^{-1} \qquad (5.38)$$

$$G^{-1}(t) = \int_0^t \psi(\tau)\psi^T(\tau)d\tau \qquad (5.39)$$

The derivative of Equation (5.39) is as follows:

$$\begin{aligned}
G^{-1}(t)\dot{G}(t)G^{-1}(t) &= -\psi(t)\psi^T(t) \\
\dot{G}(t) &= -G(t)\psi(t)\psi^T(t)G(t) \qquad (5.40)
\end{aligned}$$

From Equation (5.40), it is observed that $\dot{G}(t)$ is a negative definite and G(t) is decreasing over time, therefore $G(t) \in l_\infty$. Now executing the time derivative of Equation (5.36) and utilizing Equation 6(a)(5.31), (5.32), (5.33), and (5.37) the

following is obtained:

$$\begin{aligned}\dot{\omega}(t) &= \dot{G}(t)G^{-1}(t)\omega(t) - G(t)\psi(t)u(t) \\ &= -G(t)\psi(t)\psi^T(t)\omega(t) - G(t)\psi(t)u(t) \\ &= -G(t)\psi(t)\left(\psi^T(t)\omega(t) + u(t)\right) \\ &= -\alpha_1 G(t)\psi(t)s_H(t) \end{aligned} \tag{5.41}$$

5.4.0.1 Stability Analysis

Definition : FLS is known as a general function approximator. Therefore, in this work, it is assumed that without loss of generality there exists a ω^* such that:

$$u(t) = \psi^T\omega^*(t) + \varepsilon_f^*(z) \tag{5.42}$$

where $\varepsilon_f^*(z) = [\varepsilon_{f1}^*, \varepsilon_{f1}^*, ..., \varepsilon_{f1}^*]^T \in \Re^k$ is the minimal functional approximator error. In this work, the following is defined:

$$\tilde{\omega}(t) = \omega(t) - \omega^* \tag{5.43}$$

In addition:

$$s_H(t) = \psi^T\widetilde{\omega}(t) \tag{5.44}$$

Lemma 1 :

$$\begin{aligned}\frac{d(G^{-1}(t)\tilde{\omega}(t))}{dt} &= -G^{-1}(t)\dot{G}(t)G^{-1}(t)\tilde{\omega}(t) + G^{-1}(t)\dot{\tilde{\omega}}(t) \\ &= \psi(t)\psi^T(t)\tilde{\omega}(t) - \psi(t)s_H(t) \\ &= \psi(t)s_H(t) - \psi(t)s_H(t) \\ &= 0 \end{aligned} \tag{5.45}$$

This is indicating that $G^{-1}(t)\tilde{\omega}(t)$ is not altering with respect to time, and therefore $G^{-1}(t)\tilde{\omega}(t) = G^{-1}(0)\tilde{\omega}(0)$, $\forall_t > 0$.

$$\lim_{t\to\infty} \tilde{\omega}(t) = \lim_{t\to\infty} G(t)G^{-1}(0)\tilde{\omega}(0) \tag{5.46}$$

Since $G(t)$ is decreasing and $\tilde{\omega}(t) \in l_\infty$, $\omega(t) \in l_\infty$. In this work, the following Lyapunov function is considered:

$$V(t) = \frac{1}{2}\widetilde{\omega}^T(t)G^{-1}(t)\tilde{\omega}(t) \tag{5.47}$$

The time derivative of the Lyapunov function is as follows:

$$\begin{aligned}\dot{V}(t) &= \frac{1}{2}\widetilde{\omega}^T(t)G^{-1}\dot{\tilde{\omega}}(t) + \frac{1}{2}\widetilde{\omega}^T(t)\dot{G}^{-1}\tilde{\omega}(t) \\ &= -\widetilde{\omega}^T(t)\psi(t)s_H(t) - \frac{1}{2}\widetilde{\omega}^T(t)\psi(t)\psi^T(t)\tilde{\omega}(t) \\ &= -s_H^2(t) - \frac{1}{2}s_H^2(t) \\ &= -\frac{3}{2}s_H^2(t) \leq 0 \end{aligned} \tag{5.48}$$

From Equation (5.47), and Equation (5.48), it is observed that $V(t) > 0$, and $\dot{V}(t) \leq 0$. In addition, Equation (5.48) shows that $\dot{V}(t) = 0$, if and only if $e(t) = 0$. It is indicating that the global stability of the system is guaranteed

by the Lyapunov theorem. By utilizing Barbalat's lemma [230], it can also be observed that $e(t) \to 0$ as $t \to \infty$. It is ensuring the asymptotic stability of the system. Thus, a convergence of the system's tracking error to zero is witnessed.

5.5 Results and Discussion

Modeling and control of BIFW MAVs are recent research topics in the field of autonomous unmanned aerial vehicles (UAVs). BIFW MAV exhibits some advanced characteristics like fast flight, vertical take-off, and landing, hovering and quick turn, and enhanced maneuverability when compared to similar-sized fixed and rotary wing UAVs. To observe these features from a BIFW MAV, an advanced control mechanism is necessary. Thus, our proposed self-evolving G-controller is implemented in a BIFW MAV plant, which is inspired by the work of [192, 193]. In this chapter, the G-controller is also attempted to control the hexacopter. In case of the BIFW MAV plant, tracking of various trajectories of altitude is observed to evaluate the controller's performance. In the hexacopter plant, not only the altitude but also the attitude-tracking performances are witnessed. Being an evolving controller, the G-controller can evolve both the structure and parameters. The observed structure-evolution procedure from the G-controller's performance is explained in the following subsection 5.5.1.

5.5.1 Observed evolution in G-controller's structure

The proposed G-controller has the capability of evolving the structure by adding or pruning rules like many other evolving controllers discussed in the literature review chapter. However, unlike the existing evolving controllers, GART+, multivariate Gaussian function, SMC learning theory-based adaptation laws are

combined in the G-controller. From the amalgamation of such advanced features, a fast self-evolving mechanism is recorded with a lower computational cost. In controlling the altitude and attitude of the highly nonlinear and complex autonomous vehicles discussed in Section 4.3 of chapter 4, the activation of both the rule growing and pruning mechanism are witnessed here. Due to the evolving nature of the G-controller, the fuzzy rules are evolved in different time steps for different trajectories. This rule evolution of the G-controller with respect to various desired altitude of BIFW MAV plant are compiled in table 5.3. To observe them graphically, the number of evolved rules for various trajectories of BIFW MAV and hexacopter are plotted and disclosed in Figure 5.3.

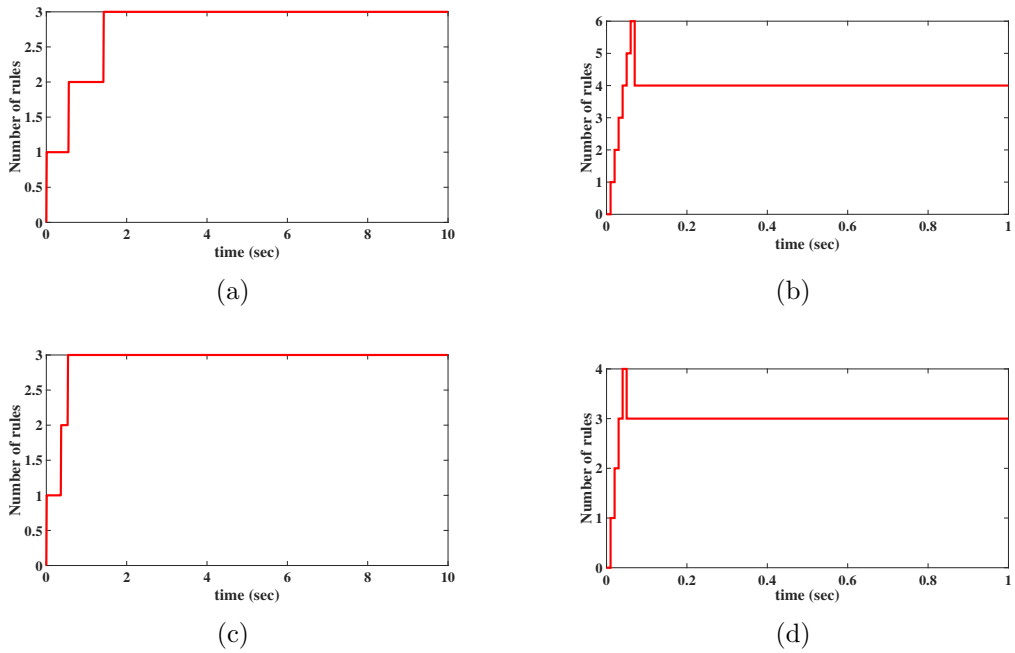


Figure 5.3: Generated rules of the self-evolving G-controller at various trajectories of BIFW MAV and hexacopter where the trajectories are (a) step function altitude for BIFW MAV, (b) customized altitude for BIFW MAV, (c) pitching for Hexacopter, (d) rolling for Hexacopter

5.5.2 Results

The G-controller's performance is observed with respect to various reference signals, and the results are compared with a TS fuzzy controller [231], and a Proportional Integral Derivative (PID) controller, Radial Basis Function Neural Network (RBFNN) and Generalized Regression Neural Network (GRNN) based controllers. Our source codes are made publicly available in [225]. In the case of the BIFW MAV, a variety of desired trajectories are utilized in the closed-loop control system to evaluate controllers performance for 100 seconds, such as 1) a constant altitude of 10 meters (m) expressed as $Z_d(t) = 10$; 2) three different step functions, where one of them is varying its amplitude from 0 m to 10 m, another is from 5 m to 10 m, and the other one is varying from -5 m to 5 m, expressed as $Z_d(t) = 10u(t-20)$, $Z_d(t) = 5u(t) + 5u(t-20)$, $Z_d(t) = -5u(t) + 10u(t-20)$ respectively; 3) three different square wave functions with a frequency of 0.1 Hz, where their amplitudes are 1 m, 4 m, and 10 m correspondingly; 4) two square wave functions with a frequency of 1 Hz. One of them has an amplitude of 1 m and the other one of 4 m; 5) a customized trajectory, where the amplitude varies from 0 to 2 m; 6) a sawtooth wave function with an amplitude of 1 m and a frequency of 1 Hz; 7) a sine wave function with an amplitude of 1 m and a frequency of 1 Hz. For all these trajectories, the performances of our proposed G-controller, TS fuzzy controller, PID controller, RBFNN controller, GRNN controller are observed and compared, where higher tracking accuracy is obtained from the G-controller. For a clearer understanding, some of these observations are presented pictorially in figures from Figure 5.4 to Figure 5.7.

The performance of various controllers for a step function $Z_d(t) = 5u(t) + 5u(t-20)$ is observed in Figure 5.4, where our proposed G-controller outperformed benchmark controllers. The performance for the square wave pulse trajectory

with an amplitude of 4 m and a frequency of 1 Hz is observed in Figure 5.5, where comparatively improved performance is witnessed from our G-controller. The TS fuzzy controller fails to follow this trajectory. Therefore, the comparisons among the PID, GRNN, RBFNN, and G-controller are exposed in Figure 5.5, where the G-controller has beaten the benchmark controllers in terms of accuracy. In case of other trajectories for BIFW MAV, superior performances are visualized by our proposed G-controller. To compare the performance among controllers, the root mean square error (RMSE), rising time, and settling time of all the controllers for various reference signals are also measured and summarized in Table 5.1, where the lowest RMSE is inspected from the G-controller. Since the G-controller starts operating from scratch with an empty fuzzy set, the rising time is comparatively longer than the PID controller. However, comparatively lower settling time is indicating the proposed controller's ability to settle to the desired trajectory sharply.

A simulated wind gust is generated using a discrete wind gust Simulink model and added to the BIFW MAV plant dynamics to check the robustness of our proposed G-controller against unknown perturbations and uncertainties. This simulated wind gust has a maximum velocity of 4 ms^{-1} and is applied to the plant after 2 seconds of starting the operation. In the presence of the wind gust, some of the trajectory tracking performances of the controllers are manifested in Figure 5.7, where tracking with a very small deviation from the trajectories are observed. However, this adverse effect has been minimized very sharply by the G-controller. The RMSEs by considering the effect of wind gust are also tabulated in Table 5.2.

Furthermore, the G-controller has been utilized to control altitude and the outer loop of attitude (roll and pitch) of the simulated over-actuated hexacopter

plant. All these results are compared with the PID, GRNN, and RBFNN controllers. In case of controlling the altitude, the controllers are employed to control the thrust of the control-mixing box of the plant. Due to the addition of the moving mass, the rolling motion is not only controlled by the velocity in Y-axis (v_y) generated by the motors, but also by the mass moving in the Y direction due to their Center of Gravity (CG)-shifting capability. Our proposed controller has been employed in both facts to control the rolling motion. Similarly, to control the pitching motion of the hexacopter, the G-controller has been used to control both the velocity in X-axis (v_x) and the mass moving in the X-direction. The altitude tracking performances of hexacopter for various trajectories have been presented in Figure 5.8, whereas the tracking of rolling and pitching are exhibited in Figure 5.9. In all cases, better tracking performances are monitored from the G-controller than that of benchmark controllers.

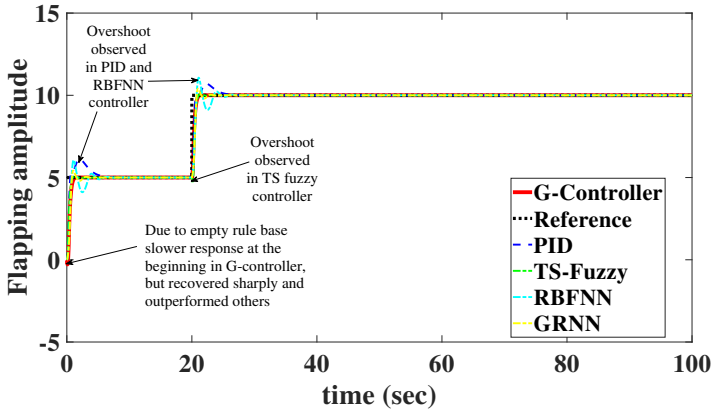


Figure 5.4: Performance observation of various controllers in altitude tracking of BIFW MAV when the trajectory is a step function $Z_d(t) = 5u(t) + 5u(t-20)$

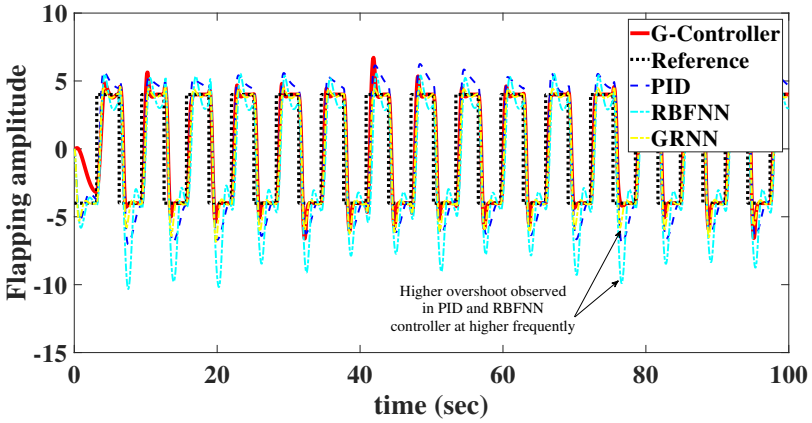


Figure 5.5: Performance observation of various controllers in altitude tracking of BIFW MAV when the trajectory is a square wave function with an amplitude of 4 m and frequency 1 Hz

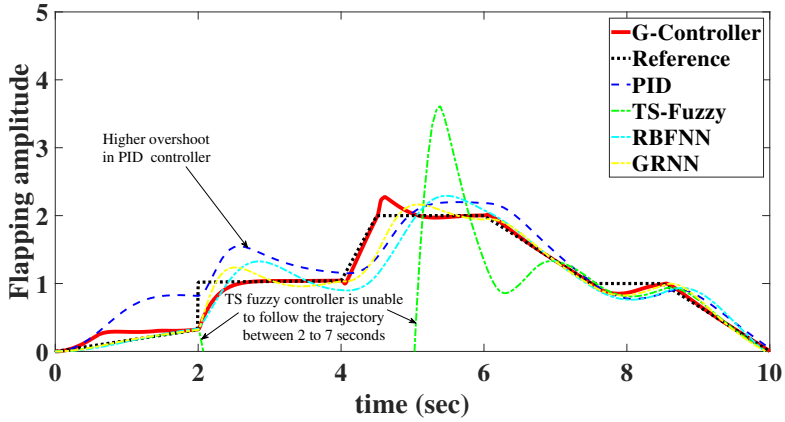


Figure 5.6: Performance observation of various controllers in BIFW MAV in case of tracking a customized trajectory

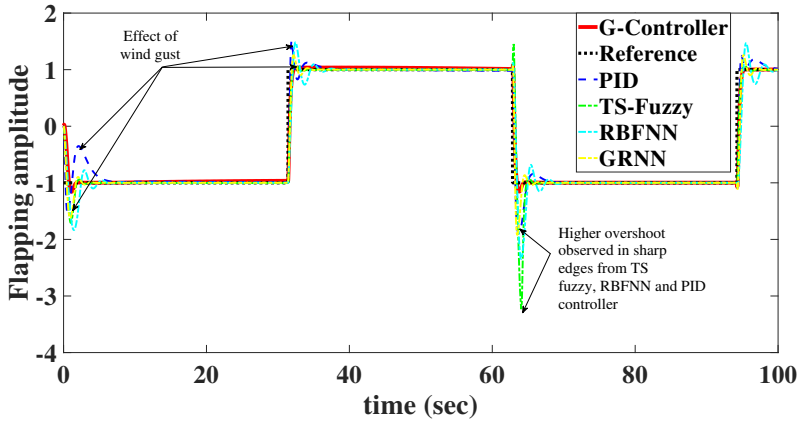


Figure 5.7: Performance observation of various controllers in altitude tracking of BIFW MAV by considering wind gust as environmental uncertainty

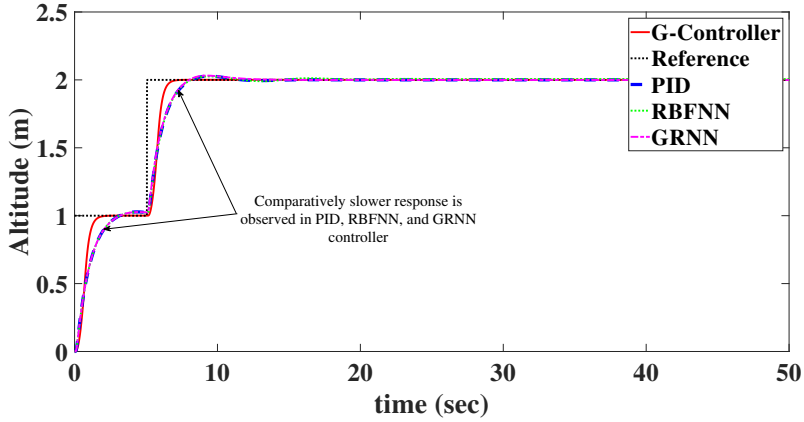
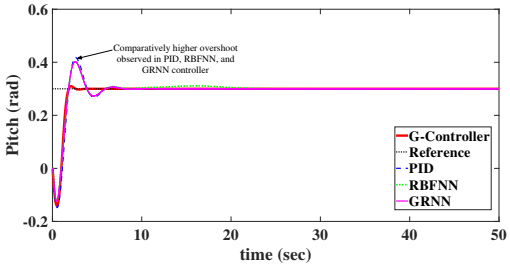
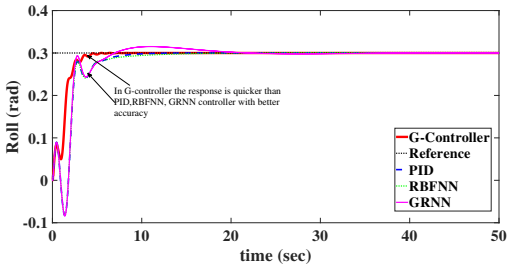


Figure 5.8: Performance observation of various controllers in tracking altitude of a hexacopter



(a)



(b)

Figure 5.9: Performance observation of various controllers in tracking desired (a) pitching and (b) rolling motion of a hexacopter

5.5.3 Discussion

Unlike the benchmark PID, TS fuzzy, RBFNN and GRNN controller, the G-controller starts the self-construction with an empty fuzzy set at the beginning of the closed-loop control system. The benchmark controllers start operating with their predefined control parameters. In case of the PID controller, control parameters (proportional gain K_p, integral gain K_i, and differential gain K_D) are obtained offline before starting the closed-loop control operation. The TS fuzzy controller consists of five rules, where univariate Gaussian membership functions are utilized in each rule. To obtain the antecedents and consequent parameters of the rules, the fuzzy controller is trained with the PID controller's input-output dataset. Similar offline training is required for the RBFNN and GRNN controllers, where both of them have utilized ten nodes in the single hidden layer. Besides, both are feed-forward neural networks, where sigmoid activation function is used. In benchmark controllers, both the parameters and the structure are fixed before the starting of the closed-loop operation. On the contrary, in G-controller not only the GENEFIS but also the parameters of the sliding surface are evolving. Those sliding parameters are initialized with a very small value, then evolved to the desired value by using different learning rates. These rates are varied with respect to the corresponding plants and desired actions. To the best of our knowledge, this approach of evolving the sliding parameters is not utilized in the existing evolving neuro-fuzzy controller. It makes the proposed G-controller a fully self-evolving controller.

5.6 Summary

The G-controller developed in this chapter is fully autonomous and evolving in nature, i.e. it can alter its structure, and system parameters online to cope with changing nonlinear dynamics of the plant to be controlled. Besides, the synthesis of SMC theory-based adaptation laws improved its robustness against various internal and external uncertainties. These desirable features make the G-controller a suitable candidate for highly nonlinear autonomous vehicles. Our proposed control algorithm is developed using C programming language considering the compatibility issues to implement directly in the hardware of a variety of autonomous vehicles like BIFW MAV, quadcopter, hexacopter, etc. The controller's performance has been evaluated by observing the tracking performance of an over-actuated BIFW MAV and an over-actuated hexacopter's plant for a variety of trajectories. The performances are compared to that of PID, TS fuzzy, RBFNN, and GRNN controller to observe the improvements of our proposed evolving G-controller. The G-controller starts building the structure from scratch with an empty fuzzy set in the closed-loop system. It causes a slow response at the very beginning of the loop, which is a common phenomenon in any self-evolving controller. However, due to the integration of GART+, multivariate Gaussian function, and SMC learning theory-based adaptation laws, the self-evolving mechanism of the G-controller is fast with a lower computational cost. In addition, wind gust has been added to the BIFW MAV plant as environmental uncertainties to evaluate the G-controller's robustness against unknown perturbations, where satisfactory tracking of the desired trajectory proves the proposed controller performance to eliminate uncertainty. The G-controller's stability is confirmed by both the Lyapunov theory and experiments. In the next

chapter, another novel evolving neuro-fuzzy system-based autonomous learning algorithm has been utilized to model an UAV online from the flight test data streams.

Table 5.1: Measured RMSE, rise and settling time of various controllers in operating the BIFW MAV

Desired Trajectory (Z_d)	**Maximum amplitude (meter)**	**Measured features**	**Without gust**				
			PID	**TS-Fuzzy**	**G-control**	**RBF-NN**	**GR-NN**
Constant height	10	**RMSE**	0.6460	0.6855	0.6631	0.7108	**0.5881**
		Rise time (ms)	50.772	61.884	**41.208**	55.828	51.493
		Settling time (ms)	560.9	261.9	**127.1**	415.9	230.8
Step function	$Z_d(t) = 5u(t) + 5u(t-20)$	**RMSE**	**0.3866**	0.4043	0.4000	0.4571	0.4401
		Rise time (ms)	2024.8	2039.0	**2023.6**	2033.3	2027.5
		Settling time (ms)	2442.5	2124.7	**2088.7**	2414.2	2129.1
	$Z_d(t) = 10u(t-20)$	**RMSE**	0.6265	0.7728	**0.5731**	0.7119	0.5896
		Rise time (sec)	54.927	60.832	**51.271**	55.785	51.544
		Settling time (ms)	2539.5	2270.0	**2104.9**	2416.0	2232.2
	$Z_d(t) = -5u(t) + 10u(t-20)$	**RMSE**	0.6695	0.8156	0.6534	0.7623	**0.6446**
		Rise time (ms)	42.710	28.145	41.823	**24.511**	26.430
		Settling time (ms)	2538.3	2267.0	**2104.7**	2407.0	2220.3
Square wave function ($f = 0.1\ Hz$)	1	**RMSE**	0.2039	0.2119	**0.1742**	0.2635	0.1942
		Rise time (ms)	**10.855**	21.463	44.097	24.417	20.409
		Settling time (ms)	9823.8	9637.9	**9515.7**	9840.1	9643.9
	4	**RMSE**	0.8908	0.8910	**0.8336**	0.9594	0.8487
		Rise time (ms)	47.903	24.645	51.995	**23.595**	25.523
		Settling time (ms)	9883.0	9664.7	**9542.7**	9826.7	9667.1
	10	**RMSE**	2.7380	312.51	**2.5405**	4.2534	2.6128
		Rise time (ms)	41.995	2394.4	**28.507**	31.603	27.292
		Settling time (ms)	9915.5	9963.0	**9611.6**	9731.5	9672.2
Square wave function($f = 1\ Hz$)	1	**RMSE**	0.6718	**0.6410**	0.5720	0.8986	0.6452
		Rise time (ms)	**12.689**	21.555	44.521	21.467	21.441
		Settling time (ms)	9952.8	9969.4	**9822.9**	9954.0	9956.5
	4	**RMSE**	3.1435	2.8409	**2.6773**	3.37674	2.7638
		Rise time (ms)	95.759	23.289	52.142	**16.512**	24.555
		Settling time (ms)	9883.8	9903.5	**9853.0**	9952.6	9874.9
Customized wave function	2	**RMSE**	0.2856	0.1771	**0.0978**	0.23799	0.1459
		Rise time (sec)	14.819	4.7401	**4.0331**	5.6073	4.4579
		Settling time (sec)	995.19	993.49	**993.46**	995.69	994.12
Sawtooth wave function($f = 1\ Hz$)	1	**RMSE**	0.5234	**0.4303**	0.4781	0.64061	0.5590
		Rise time (ms)	**174.82**	208.67	207.99	199.46	208.27
		Settling time (ms)	9986.4	9989.5	**9983.6**	9991.3	9989.3
Sine wave function ($f = 1\ Hz$)	1	**RMSE**	0.2096	0.0733	**0.0355**	0.13735	0.0725

Table 5.2: Measured RMSE, rise and settling time of various controllers in operating the BIFW MAV by considering wind gust

Desired Trajectory (Z_d)	**Maximum amplitude (meter)**	**Measured features**	**With gust**				
			PID	**TS-Fuzzy**	**G-control**	**RBF-NN**	**GR-NN**
Constant height	10	**RMSE**	0.6466	0.6867	0.6635	0.7150	**0.5897**
		Rise time (ms)	50.772	61.884	**41.208**	55.828	51.493
		Settling time (ms)	574.4	261.5	**127.1**	417.8	230.7
Step function	$Z_d(t) = 5u(t) + 5u(t-20)$	**RMSE**	**0.3868**	0.4016	0.3988	0.4520	0.4365
		Rise time (ms)	2024.7	2038.0	**2023.0**	2031.7	2025.6
		Settling time (ms)	2455.9	2127.8	**2087.1**	2406.7	2131.6
	$Z_d(t) = 10u(t-20)$	**RMSE**	0.6216	0.7638	**0.5665**	0.7001	0.5843
		Rise time (sec)	52.302	60.257	**45.895**	56.168	51.263
		Settling time (ms)	2551.4	2163.5	**2075.8**	2426.2	2138.3
	$Z_d(t) = -5u(t) + 10u(t-20)$	**RMSE**	0.6648	0.8070	0.6476	0.7511	**0.6398**
		Rise time (ms)	39.297	27.655	27.344	**24.485**	26.339
		Settling time (ms)	2550.2	2162.3	**2075.8**	2418.6	2135.3
Square wave function ($f = 0.1\ Hz$)	1	**RMSE**	0.2021	0.2140	**0.1744**	0.2703	0.1967
		Rise time (ms)	**10.669**	20.704	41.700	23.019	19.525
		Settling time (ms)	9823.5	9637.9	**9515.7**	9840.1	9643.9
	4	**RMSE**	0.8891	0.8875	**0.8310**	0.9530	0.8454
		Rise time (ms)	49.739	24.118	51.3939	**23.485**	25.277
		Settling time (ms)	9883.0	9664.7	**9542.7**	9826.7	9667.1
	10	**RMSE**	2.7339	312.52	**2.5375**	4.2429	2.6042
		Rise time (ms)	43.790	2394.4	27.961	31.507	**27.484**
		Settling time (ms)	9915.4	9963.0	**9611.6**	9731.5	9672.2
Square wave function($f = 1\ Hz$)	1	**RMSE**	0.6720	0.6329	**0.5733**	0.8968	0.6420
		Rise time (ms)	**12.682**	21.492	44.237	21.315	21.349
		Settling time (ms)	9952.1	9969.4	**9822.8**	9954.2	9956.5
	4	**RMSE**	3.1335	2.8299	**2.6766**	3.3516	2.7579
		Rise time (ms)	95.319	23.273	52.064	**16.506**	24.550
		Settling time (ms)	9883.8	9903.5	**9852.9**	9952.7	9875.3
Customized wave function	2	**RMSE**	0.2845	0.1742	**0.0969**	0.23282	0.1428
		Rise time (sec)	15.533	13.347	10.049	**1.3929**	14.723
		Settling time (sec)	994.99	993.44	**993.42**	995.24	994.14
Sawtooth wave function($f = 1\ Hz$)	1	**RMSE**	0.5239	**0.4253**	0.4775	0.63287	0.5531
		Rise time (ms)	**177.67**	211.69	209.39	203.594	211.19
		Settling time (ms)	9991.2	9989.3	**9982.8**	9986.4	9989.0
Sine wave function ($f = 1\ Hz$)	1	**RMSE**	0.1879	0.0762	**0.0394**	0.14043	0.0763

Table 5.3: Variation of rule generation at various trajectories of BIFW MAV

Desired Trajectory (Z_d)	**Maximum height (meter)**	**Rule**	**Time step of Rule Generation**
Constant height	10	1	1
		2	40
		3	45
Step function	$Z_d(t) = 5u(t) + 5u(t-20)$	1	1
		2	8
		3	26
	$Z_d(t) = 10u(t-20)$	1	1
		2	61
		3	2012
	$Z_d(t) = -5u(t) + 10u(t-20)$	1	1
		2	8
		3	13
Square wave function (frequency=0.1 Hz)	1	1	1
		2	52
		3	63
	4	1	1
		2	55
		3	129
	10	1	1
		2	95
		3	101
Square wave function (frequency=1 Hz)	1	1	1
		2	52
		3	63
	4	1	1
		2	55
		3	129
Customized wave function	1	1	1
		2	56
		3	143
Sawtooth wave function	1	1	1
		2	39
		3	48
Sine wave function	1	1	1
		2	49
		3	60

Chapter 6

Online Identification of an Unmanned Aerial Vehicle from Data Streams

The work presented in this chapter has been published in the following article:

- **Ferdaus, M. M.**, Pratama, M., Anavatti, S. G., Garratt, M. A. "Online Identification of a Rotary Wing Unmanned Aerial Vehicle from Data Streams." *Applied Soft Computing* 76 (2019): 313-325.

Abstract

This chapter proposes a novel online identification scheme, applied to a quad-copter using real-time experimental flight data streams based on an autonomous learning algorithm, namely Metacognitive Scaffolding Interval Type 2 Recurrent Fuzzy Neural Network (McSIT2RFNN). The metacognitive concept enables the what-to-learn, how-to-learn, and when-to-learn scheme, and the scaffolding theory realizes a plug-and-play property which strengthens the online working principle of

the proposed Evolving Intelligent System (EIS). Our proposed approach demonstrated significant improvements in both accuracy and complexity against some renowned existing variants of the McSLMs and EISs.

6.1 Introduction

Unmanned aerial vehicles (UAVs) are aircraft with no aviator on-board. Among various UAVs, the most commonly used one is the quadcopter. The quadcopter has six degrees of freedom (DOFs): they are three translational motions along the X, Y, and Z-axes, and three rotational motions (θ, ϕ, ψ). These 6 DOFs is regulated by four control inputs (Z, θ, ϕ, and ψ) only, which makes it an under-actuated system. Mathematical modeling of this nonlinear and under-actuated system is challenging. Until now, most quadcopter models are based on dynamic equations of the system, where the aggressive trajectories of quadcopters are difficult to integrate. In addition, various non-stationary factors like motor degradation, time-varying payload, wind gusts, and rotor damage are extremely difficult to predict and model mathematically and consequently hard to incorporate. These challenges are leading to increasing research interest in data-driven modeling techniques for system identification with real-time sensory data and limited expert knowledge.

In the data-driven techniques, system identification is a vital part. Successful system identification indicates the closeness of the input-output behavior of the identified system with the input-output behavior of the actual plant. The data-driven system identification or modeling can play an important role in quadcopter systems since their counterpart i.e. the model-based parameter identification requires several experimental tests to obtain the model parameters.

Even some parameters are difficult to obtain from the experiments and are problem-dependent. Thus, the model-based system requires a lot of effort for better accuracy. To minimize such effort, the data-driven quadcopter model can be used as a generalized model with different motors, propellers or sensor combinations. Some of the commonly used non-linear data-driven system identification techniques are: describing function method, block-structured systems, FLSs, NNs, and Nonlinear Autoregressive Moving Average Model with Exogenous inputs (NARMAX methods). Among these techniques, FLS [232–235] and NN-based [236] artificial intelligent systems are promising computational tools since they demonstrate learning capability from a set of data and approximate reasoning trait of human beings which cope with the impression and uncertainty of the decision making the process [237]. Furthermore, the fuzzy system offers a highly transparent solution which can be followed easily by the operator [238]. Identification of helicopter dynamics with the flight data using Nonlinear Auto Regressive eXogenous input (NARX) model, neural network with internal memory known as Memory Neuron Networks (MNN), and Recurrent MultiLayer Perceptron (RMLP) networks has been accomplished in [8]. Nonlinear system identification using Lyapunov-based fully tuned Radial Basis Function NN (RBFNN) and Extended Minimal resource allocating network (EMRAN) has been accomplished in [239, 240].

However, a major limitation of these conventional FLS and NN-based quadcopter modeling is the inability to evolve their structure to adapt to sudden changes. They also adopt a batched working principle which has to revisit entire dataset over multiple passes rendering them not scalable for online real-time deployment. To solve the problems that exist with conventional intelligent systems, Evolving Intelligent Systems (EISs) is a good candidate [44], since they learn

from scratch with no base knowledge and are embedded with the self-organizing property which adapts to changing system dynamics [50]. EIS fully work in a single-pass learning scenario, which is scalable for online real-time requirement under limited computational resources such as UAVs platform [53]. Nevertheless, EISs remain cognitive in nature where they still require scanning all samples regardless of their true contribution and training samples must be consumed immediately with the absence of learning capability to determine ideal periods to learn these samples [241]. The Metacognitive Learning Machine (McLM) technique enhances the adaptability of EIS by interpreting the meta-memory model of [75] where the learning process is developed in three phases, namely what-to-learn, how-to-learn and when-to-learn [242, 243]. The what-to-learn is implemented with a sample selection mechanism which determines whether to accept data samples, the how-to-learn is where the underlying training process takes places, the when-to-learn is built upon a sample reserved mechanism which allows to delay the training process of particular samples when their significance does not suffice to trigger the learning mechanism.

Recent advances in the McLM [74, 244, 245] have involved the concept of scaffolding theory as a foundation of the how-to-learn- another prominent theory in psychology to help learners to solve complex tasks. The use of scaffolding theory is claimed to generate the plug-and-play property where all learning process are self-contained in the how-to-learn without over-dependence on pre-and/or post-processing steps. It is worth noting that the scaffolding theory does not hamper the online learning property of EISs since all learning components follow strictly single-pass learning mode, which is well-suited for online real-time applications. The scaffolding theory consists of two parts: active supervision and passive supervision. The passive supervision is constructed using parameter

learning theories which demand target variables to elicit system errors for correction signals while the active supervision features three components: fading, complexity reduction, and problematizing. The complexity reduction alleviates learning complexities by applying feature selection, data normalization, etc. and the problematizing focuses on concept drifts in data distributions while the fading component is meant to reduce the network complexity by discarding inactive components using the pruning and merging scenarios.

6.2 Contribution

A novel online system identification of quadcopter based on a recently developed McSLM [83], namely McSIT2RFNN, is proposed in this chapter. McSIT2RFNN is structured as a six-layered network architecture actualizing interval type-2 Takagi Sugeno Kang (TSK) fuzzy inference scheme. This network architecture features a local recurrent connection which functions as an internal memory component to cope with the temporal system dynamics and to minimize the use of time-delayed input attributes [83]. Note that the local recurrent link does not compromise the local learning property because the spatio-temporal firing strength is generated by feeding previous states of system dynamics back to itself [83]. The rule-layer consists of interval type-2 multivariate Gaussian functions with uncertain means which characterizes scale-invariant trait and maintains inter-correlation among input variables.The rule consequent layer is constructed by the nonlinear Chebyshev polynomial up to the second-order, which expands the degree of freedom of a rule consequent [82]. The polynomial is utilized here to rectify the approximation power of the zero-or first-order TSK rule consequent.

McSIT2RFNN features unique online learning techniques where synergy be-

tween the metacognitive learning scenario and the scaffolding theory comes into picture while retaining computationally light working principle through a fully one-pass learning scenario for online real-time applications. The learning process starts from the what-to-learn process using an online active learning mechanism, which actively extracts relevant training samples for the training process while ruling out inconsequential samples for the training process. Selected training samples are then processed further in the how-to-learn designed under the scaffolding concept. The problematizing facet of the scaffolding theory is depicted by the rule growing mechanism which assesses statistical contribution of data points to be a candidate of a new rule. This scenario controls stability-and plasticity dilemma in learning from data streams since it guides to proper network complexity for a given problem and addresses changing data distributions by introducing a new rule when a change is detected. A rule recall scenario is put forward to represent the problematizing aspect which tackles the temporal or recurring drift. This learning mechanism plays a vital role during real-flight missions of the UAV because previously seen flight conditions often re-appear again in the future. The complexity reduction component is portrayed by an online feature selection scenario which puts into perspective relevance and redundancy of input features. This learning component lowers the input dimension which contributes positively to models generalization and computational complexity. The fading process relies on the rule merging scenario and the rule pruning scenario. The rule pruning scenario removes obsolete rules which are no longer relevant to current training concept by studying mutual information between fuzzy rule and the target variable. Significantly overlapping rules are coalesced into a single rule by the rule merging scenario and this mechanism is capable of cutting down network complexity and improving interpretability of rule semantics. The efficacy

of our proposed methodology was carefully investigated through simulations using real-world flight data as well as real-time flight tests. Our algorithm was benchmarked with several prominent algorithms, and it was shown that our algorithm produced the most encouraging performance in attaining a trade-off between accuracy and complexity.

The proposed methodology carries the following advantages:

- It is compatible with online real-time deployment in the real flight tests since it works fully in the single-pass learning mode. Furthermore, McSIT2RFNN does not necessarily see all sensory data streams due to its what-to-learn component further substantiating scalability of McSIT2RFNN in handling online data streams.

- It features a highly flexible foundation which self-evolves its network structure and parameters in accordance with variations of data streams; no matter how slow, rapid, gradual, and temporal a change in data streams is.

- McSIT2RFNN is created from a combination between the interval type-2 fuzzy system which is more robust to face uncertainties than its type-1 counterparts and the recurrent network architecture which is capable of coping with temporal system dynamics and lagged input variables.

- It actualizes a plug-and-play working principle where all learning modules are embedded in only a single training scenario without the requirement of pre-and/or post-training steps.

6.3 Main Features

The major features of the evolving neuro-fuzzy system proposed in this chapter are summarized as follows:

- A novel online system identification of quadcopter based on a psychologically inspired learning machine, namely McSIT2RFNN, is proposed.
- Real-time flight tests were done where real-world flight data were obtained and preprocessed. We also made these flight data publicly available for the convenience of readers.
- The efficacy of the proposed approach was validated by numerical experimentation. This includes simulations using real-world flight data and real-time flight tests [246].

The remaining parts of this chapter are organized as follows: Section 6.4 describes the learning policy of the McSIT2RFNN technique by describing both the cognitive and meta-cognitive components. In section 6.5, the details of the quadcopter flight experiment and system identification are explained. Finally, the chapter ends with concluding remarks in section 6.6.

6.4 Online Learning Policy of McSIT2RFNN

This section describes the learning policy of Meta-cognitive Scaffolding Based Interval Type 2 Recurrent Fuzzy Neural Network (McSIT2RFNN) [82]. The McSIT2RFNN has two components namely cognitive and meta-cognitive. The cognitive component corresponds to the network structure of McSIT2RFNN while the metacognitive component consists of learning scenarios to fine-tune the cognitive component.

6.4.1 Cognitive mechanism of McSIT2RFNN

In McSIT2RFNN, a six-layered recurrent network structure with a local recurrent connection is utilized for the hidden layer. The first layer is known as the input layer, which passes the input to the second layer directly as follows:

$$\left(\eta_{out}\right)_k^1 = \left(\eta_{atv}\right)^1(x_k) = x_k \tag{6.1}$$

where η_{out} represents output of a layer, and η_{atv} denotes the forward activation function of a layer.

Unlike the conventional neuro-fuzzy system, the univariate Gaussian function is replaced by an interval-valued multivariate Gaussian function with uncertain mean and then it is utilized in the second layer of the McSIT2RFNN, which is also known as the rule layer. This Gaussian function consequently generates an interval-valued firing strength as follows:

$$\widetilde{\eta}_{out}{}^2 = \left(\eta_{atv}{}^2\right)(\eta_{out}{}^1) = \exp(-(\Gamma_n^2 - \widetilde{\zeta}_i)\Sigma_i^{-1}(\Gamma_n^2 - \widetilde{\zeta}_i)) \tag{6.2}$$

where, $\widetilde{\eta}_{out}{}^2 = [\underline{\eta_{out}}{}^2, \overline{\eta_{out}}{}^2]$, $\widetilde{\zeta}_i = [\underline{\zeta}_i, \overline{\zeta}_i]$, and $\widetilde{\zeta}_i$ is the uncertain centroid of the ith rule abiding by the condition $\underline{\zeta}_i < \overline{\zeta}_i$. If we consider to model or identify a Multi-Input-Single-Output system, the If-Then rule of the McSIT2RFNN can be expressed as follows:

$$R_j : \text{If} \quad X_n \text{ is } \widetilde{\mathcal{N}}_{out_j}{}^2 \text{ Then } y_j = x_e^j \Omega_j \tag{6.3}$$

where x_e^j, Ω_j are respectively an extended input variable resulting from a non-linear mapping of the wavelet coefficient ($x_e \in \Re^{1\times(2\mu+1)}$) and a weight vector ($\Omega \in \Re^{1\times(2\mu+1)}$). The consequent part of the rule is explained in the fifth

layer. However, the rule presented in Equation (6.3) is not transparent enough to expose atomic clause of the human-like linguistic rule [247]. It operates in a totally high dimensional space, therefore cannot be represented in fuzzy set. Since the non-axis-parallel ellipsoidal rule cannot be expressed directly in an interval type-2 fuzzy environment, a transformation strategy is required [248,249]. Such a transformation technique should have the capability of formulating the fuzzy set for the non-axis parallel ellipsoidal cluster [250,251]. The transformation strategy developed in [1] is extended in this work in terms of interval type-2 system, which can be expressed as follows:

$$\sigma_i = \frac{(\underline{r}_i + \overline{r}_i)}{2\sqrt{\Sigma_{ii}}} \tag{6.4}$$

where Σ_{ii} represents the diagonal element of the covariance matrix and $\tilde{r}_i$ denotes the Mahalanobis distance, which is $\widetilde{r}_i = (X_n - \widetilde{\zeta}_i)\Sigma_i^{-1}(X_n - \widetilde{\zeta}_i) = [\overline{r}_i \quad \underline{r}_i]$, where $\overline{r}_i$ is the upper and $\underline{r}_i$ is the lower Mahalanobis distance and $\widetilde{\zeta}_i = [\underline{\zeta}_i, \overline{\zeta}_i]$. No transformation is required for the mean or centroid $(\widetilde{\zeta}_i)$ of the multivariate Gaussian function, since it can be directly applied to the fuzzy set level. After successfully presenting the interval-valued multi-variable Gaussian function into fuzzy set, the fuzzification process of the upper and lower Gaussian membership functions with uncertain means $\widetilde{\zeta}_{j,i} = [\underline{\zeta}_{j,i}, \overline{\zeta}_{j,i}]$ is exhibited as follows:

$$\widetilde{\eta}^2_{out_{j,i}} = \exp\left(-\left(\frac{\eta^2_{atv_i} - \widetilde{\zeta}_{j,i}}{\sigma_{j,i}}\right)^2\right) \quad N(\widetilde{\zeta}^j_i, \sigma^j_i, \eta^2_{atv_i}) \;\; \widetilde{\zeta}_j = [\overline{\zeta}^j_i, \; \underline{\zeta}^j_i] \tag{6.5}$$

$$\overline{\eta}^2_{out_{j,i}} = \begin{cases} N(\overline{\zeta}^j_i, \sigma_{j,i}; \eta) & \eta^2_{atv_i} < \overline{\zeta}^j_i \\ 1 & , \;\; \overline{\zeta}^j_i < x_i < \underline{\zeta}^j_i \\ N(\widetilde{\zeta}^j_i, \sigma_{j,i}; \eta^2_{atv_i}) & \eta^2_{atv_i} > \underline{\zeta}^j_i \end{cases} \tag{6.6}$$

$$\underline{\eta}^2_{out_{j,i}} = \begin{cases} N(\underline{\zeta}^j_i, \sigma^j_i; \eta) & x_i \leq \frac{(\overline{\zeta}^j_i + \underline{\zeta}^j_i)}{2} \\ N(\overline{\zeta}^j_i, \sigma^j_i; \eta^2_{atv_i}) & x_i > \frac{(\overline{\zeta}^j_i + \underline{\zeta}^j_i)}{2} \end{cases} \tag{6.7}$$

After getting the above expression of the fuzzy set, the fuzzy rule exposed in Equation (6.3) can be transformed Equation (6.2) into a more interpretable form as follows:

$$R_j : \text{If} \quad X_1 \text{ is } \widetilde{\eta}_{out_1}{}^2 \text{ and } X_2 \text{ is } \widetilde{\eta}_{out_2}{}^2 \text{ and ... and } X_{nu} \text{ is } \widetilde{\eta}_{out_{nu}}{}^2 \text{ Then } y_j = x^i_e \Omega_i \tag{6.8}$$

where j is the number of rules, nu is the number of inputs. This transformation technique has overcome the issue of transparency of the multi-variable Gaussian function. The validity of $\widetilde{\eta}^2_{out_{j,i}} = [\overline{\eta}^2_{out_{j,i}}, \underline{\eta}^2_{out_{j,i}}]$ is proven in [83]. In the third layer, the upper and lower bound of membership degrees are connected using the product $t-$norm operator in each fuzzy set and generates an interval-valued spatial rule firing strength as follows:

$$\big(\underline{\eta_{out}}\big)^3_i = \prod_{k=1}^{nu} \big(\underline{\eta_{atv}}\big)^3_{i,k} = \prod_{k=1}^{nu} \big(\underline{\eta_{out}}\big)^{2.1}_{i,k} \ , \ \big(\overline{\eta_{out}}\big)^3_i = \prod_{k=1}^{nu} \big(\overline{\eta_{atv}}\big)^3_{i,k} = \prod_{k=1}^{nu} \big(\overline{\eta_{out}}\big)^{2.1}_{i,k} \tag{6.9}$$

The fourth layer is known as temporal firing layer. In this layer of McSIT2RFNN a local recurrent connection is observed, where the spatial firing strength of previous observation is fed back to itself and generates a temporal firing strength as follows:

$$\big(\overline{\eta_{out}}\big)^4_{i,o} = \Lambda^o_i \big(\overline{\eta_{atv}}\big)^4_i + (1 - \Lambda^o_i)\big(\overline{\eta_{out}}\big)^4_i (n-1) \tag{6.10}$$

$$\big(\underline{\eta_{out}}\big)^4_{i,o} = \Lambda^o_i \big(\underline{\eta_{atv}}\big)^4_i + (1 - \Lambda^o_i)\big(\underline{\eta_{out}}\big)^4_i (n-1) \tag{6.11}$$

where $\Lambda^o_i \in [0, 1]$ denotes a recurrent weight for the ith rule of the oth class.

The fifth layer of McSIT2RFNN is the consequent layer, where the Chebyshev polynomial up to the second-order is utilized to construct the extended input feature x_e [252]. This Chebyshev polynomial is expressed in Equation (6.12).

$$\tau_{n+1}(x) = 2x_k\tau_n(x_k) - \tau_{n-1}(x_k) \tag{6.12}$$

If X is considered as a 2-D input composition like $[x_1, x_2]$, then the extended input vector can be presented as $x_e = [1, x_1, \tau_2(x_1), x_2, \tau_2(x_2)]$, where $x_e \in \Re^{1\times(2\mu+1)}$, and μ represents the input dimension. This layer functions as an enhancement layer that maps to the original input vector to high dimensional space to rectify the mapping capability of the rule consequent. The extended input variable x_e is weighted and generates an output of the consequent layer as follows:

$$\left(\eta_{out}\right)_i^5 = x_e^i\Theta_i \tag{6.13}$$

where Θ_i is a connection weight between the temporal firing layer and the output layer. In the output layer, type reduction mechanism is observed, where q design coefficient method is used instead of commonly used Karnik-Mendel (KM) technique. The final crisp output of the McSIT2RFNN can be expressed as follows:

$$y_{out} = \left(\eta_{out}\right)^6 = \frac{(1-q_{out})\left(\overline{\eta_{out}}\right)_{i,o}^4\left(\eta_{out}\right)_i^5}{\sum_{i=1}^R\left(\underline{\eta_{out}}\right)_{i,o}^4} + \frac{q_{out}\left(\underline{\eta_{out}}\right)_{i,o}^4\left(\eta_{out}\right)_i^5}{\sum_{i=1}^R\left(\overline{\eta_{out}}\right)_{i,o}^4} \tag{6.14}$$

where R represents the number of fuzzy rules and q is the design factor $q \in \Re^{1\times no}$. The q design factor based type reduction mechanism performs by altering the proportion of the upper and lower rules to the final crisp output of McSIT2RFNN, where the normalization term of the original q design factor [71] is modified to

overcome the invalid interval as shown in [82].

6.4.2 Meta-cognitive learning mechanism of McSIT2RFNN

In meta-cognitive learning policy, incoming training data streams $\big((X_n,)$, where X_n is an input variable vector$\big)$ are fed into the what-to-learn section. In this section, the probability of a sample to stay in the existing cluster is calculated as:

$$P_r(X_n \in N_i) = \frac{\frac{1}{N_i}\sum_{n=1}^{N_i} S_M(X_N, X_n)}{\sum_{i=1}^{R}\sum_{n=1}^{N_i}\frac{S_M(X_N,X_n)}{N_i}} \tag{6.15}$$

where, X_N is representing the current incoming data stream and X_n is indicating the nth support of the ith cluster; meanwhile $S_M(X_N, X_n)$ is defining the similarity measure. Since Equation (6.15) requires to revisit previously seen samples, its recursive form is formulated as follows:

$$\begin{aligned}\frac{\sum_{n=1}^{N_i} S_M(X_N, X_n)}{N_i} &= \frac{\sum_{n=1}^{N_i-1}\sum_{j=1}^{u}(X_{n,j} - X_{N,j})^2}{(N_i-1)u} \\ &= \frac{(\sum_{j=1}^{u}(N_i-1){x_{N,j}}^2 - 2\sum_{j=1}^{u} x_{N,j}K_{i,j} + v_{N_i)}}{(N_i-1)u}\end{aligned} \tag{6.16}$$

where, $K_{N_i,j} = K_{N_i-1,j} + x_{N_i-1,j},$ and $v_{N_i} = v_{N_i-1} + \sum_{j=1}^{u} {x_{N_i-1,j}}^2$.

The necessity of a data sample to be trained by the how-to-learn section is monitored by the what-to-learn section through the computation of the sample's entropy which portrays the level of uncertainty caused by the samples as follows:

$$H_{tr}(N|X_n) = -\sum_{i=1}^{R} P_r(X_n \in N_i)\log P_r(X_n \in N_i) \tag{6.17}$$

In McSIT2RFNN structure, a highly uncertain data stream is accepted as a training sample since it helps to mitigate the uncertainties in learning the target function. However, it opens the door for outliers to be fed to the how-to-learn

section. To overcome this shortcoming, the entropy or uncertainty measured by Equation (6.17) can be weighted by its average distance to the R, which are the densest local regions of the cognitive part [253]. Thus, computation of an average distance between the enquired sample and focal point can be expressed as:

$$A_d(X) = \frac{\sum_{i=1}^{R} similarity(X, C_i)}{R} \tag{6.18}$$

where *similarity* (X, C_i) is a distance function that computes the pair-wise similarity value between two examples. Finally, combining the concept of Equation (6.18) in Equation (6.17) the H_{tr} can be modified as:

$$H_{tr} = H_{tr}(N|X_n) \times A_d(X) \tag{6.19}$$

Acceptance of a data stream depends on the magnitude of H_{tr} in Equation (6.19), where H_{tr} should be greater than or equal to a threshold as follows:

$$H_{tr} \geq \delta \tag{6.20}$$

where δ denotes an uncertainty threshold, which is not constant rather it is adjusted dynamically. In this method, δ is set as $\delta_{N+1} = \delta_N(1 \pm s_s)$, where $\delta_{N+1} = \delta_N(1 + s_s)$ creates augmentation by admitting training data from the training process for minimizing the computational load and vice versa. The value of the step size s_s is set as 0.01, which refers to the thumb rule in [254]. This tuning scenario is necessary notably in non-stationary environments since a concept change directly hits the sample consumption.

After satisfying the condition of Equation (6.20), a data stream is fed to the how-to-learn phase. The how-to-learn phase of McSIT2RFNN is derived from

the scaffolding theory. It encompasses both parameter and structural learning scenarios which are done in the strictly single-pass manner.

6.4.2.1 Mechanism of growing rules

The feature of growing rules in the how-to-learn section is governed by the Generalized Type-2 Datum Significance (GT2DQ) method forming a modification of the neuron significance of [255,256] to the context of interval-valued multi-variable fuzzy rule. Gaussian Mixture Model (GMM) is used in this method as the input density function to cope with complex and even irregular data clouds. The extended formula of the neuron significance for the multi-variable Gaussian neuron [256] is further extended to the generalized interval-valued neuron in [83], which is utilized in the rule growing mechanism of this work. To express the significance of ith multi-variable interval-valued rule, the $L_u - norm$ of the error function is weighted by the input density function which can be presented as follows:

$$\begin{aligned}\mathcal{E}_i = \|\Omega_i\|_u(1-q)\left(\int_{\Re^{nu}} \exp(-u||x-\overline{\zeta}_i||^2_{\Sigma_i} p(x)dx)\right)^{1/u} \\ + \|\Omega_i\|_u\, q\left(\int_{\Re^{nu}} \exp(-u||x-\underline{\zeta}_i||^2_{\Sigma_i} p(x)dx)\right)^{1/u}\end{aligned} \tag{6.21}$$

where the Gaussian term under the integral can be written as follows:

$$(2\pi/u)^{nu/2}\det(\Sigma_i)^{-1/2} \times N(x;\, \widetilde{\zeta}_i\Sigma_i^{-1}/u),\ \widetilde{\zeta}_i = [\underline{\zeta}_i, \overline{\zeta}_i]$$

Therefore, it can be realized that the neuron significance depends on the input density $p(x)$. Usually, the input density $p(x)$ is considered to follow simple data distributions, as explained in [49] or uniform data distribution, as described

in [56]. Utilizing the concept of GMM, $p(x)$ is able to cope with complex data distributions and can be expressed as follows:

$$p(x) = \sum_{m=1}^{M} \alpha_m N(x;\ v_m,\ \Sigma_m) \tag{6.22}$$

where $N(x; v_m, \Sigma_m)$ denotes multi-variable Gaussian probability density function with mean vector $v_m \in \Re^{1\times nu}$ and covariance matrix $\Sigma_m \in \Re^{nu\times nu}$, α_m denotes the mixing coefficients which satisfies the condition $\sum_{m=1}^{M} \alpha_m = 1$, $\alpha_m > 0$. Now using the GMM in the input density $p(x)$, the further derivation can be expressed as follows:

$$\begin{aligned}
\mathcal{E}_i =& \|\Omega_i\|_u(1-q)((2\pi/u)\det(\Sigma_i)^{-1/2} \\
&\times \sum_{m=1}^{M} \alpha_m \int_{\Re^{nu}} N(x;\overline{\zeta}_i\Sigma_i^{-1}/u)N(x;\ v_m,\ \Sigma_m)dx)^{1/u} \\
&+ \|\Omega_i\|_u q((2\pi/u)\det(\Sigma_i)^{-1/2} \\
&\times \sum_{m=1}^{M} \alpha_m \int_{\Re^{nu}} N(x;\underline{\zeta}_i\Sigma_i^{-1}/u)N(x;\ v_m,\ \Sigma_m)dx)^{1/u}
\end{aligned} \tag{6.23}$$

The integral term of Equation (6.23) is a product of two Gaussian distributions and can be solved as $\int_{\Re^{nu}} N(x;\widetilde{\zeta}_i\Sigma_i^{-1}/u)N(x;v_m,\Sigma_m)dx = N(\widetilde{\zeta}_i - v_m;\ 0, \Sigma_i^{-1}/u + \Sigma_m)$. Accordingly, the final formula of the GT2DQ method to express the significance of the ith interval-valued multivariable rule [83] is expressed as:

$$\begin{aligned}
\mathcal{E}_i =& \|\Omega_i\|_u(1-q)\big\{(2\pi/u)^{n/2}\det(\Sigma_i)^{-1/2}\overline{N}_i\gamma^T\big\}^{1/u} \\
&+ \|\Omega_i\|_u\ q\big\{(2\pi/u)^{n/2}\det(\Sigma_i)^{-1/2}\underline{N}_i\gamma^T\big\}^{1/u}
\end{aligned} \tag{6.24}$$

In Equation (6.24) the mixing coefficient is denoted by γ and can be expressed

as:

$$\gamma = [\alpha_1, ..., \alpha_m, ..., \alpha_M] \in \Re^{1\times m} \tag{6.25}$$

In Equation (6.24), $\overline{N}_i$ and $\underline{N}_i$ are defined as $\overline{N}_i = \lfloor N(\overline{\zeta}_i - v_1; 0, \Sigma_i^{-1}/u + \Sigma_1), (\overline{\zeta}_i - v_2; 0, \Sigma_i^{-1}/u + \Sigma_2), ..., (\overline{\zeta}_i - v_m; 0, \Sigma_i^{-1}/u + \Sigma_m), ..., (\overline{\zeta}_i - v_M; 0, \Sigma_i^{-1}/u + \Sigma_M)\rfloor$, $\underline{N}_i = \lfloor N(\underline{\zeta}_i - v_1; 0, \Sigma_i^{-1}/u + \Sigma_1), (\underline{\zeta}_i - v_2; 0, \Sigma_i^{-1}/u + \Sigma_2), ..., (\underline{\zeta}_i - v_m; 0, \Sigma_i^{-1}/u + \Sigma_m), ..., (\underline{\zeta}_i - v_M; 0, \Sigma_i^{-1}/u + \Sigma_M)\rfloor$.

L_2-norm is utilized in McSIT2RFNN. Based on empirical analysis and previous research [82], the value of u is 2. Besides, some parameters of the GMM, namely the mean v_m, the covariance matrix Σ_m, the mixing coefficients α_m, and the number of mixing models M, are acquired using previously recorded data points $N_{prerecord}$ like [49, 255, 256]. In today's world of big data, having access to the $N_{prerecord}$ is easy. Furthermore, the total number of training data samples is noticeably larger than that of the pre-recorded data samples. The proposed method's sensitivity with regards to an altered number of prehistory samples is analyzed in [82], which proves that the $N_{prerecord}$ is not case sensitive.

In McSIT2RFNN, the generation of the hypothetical rule depends upon an incoming data stream and therefore, $\overline{c}_i$, $\underline{c}_i$, Σ_i^{-1} are substituted with $\overline{c}_{R+1}$, $\underline{c}_{R+1}$, Σ_{R+1}^{-1}. The formula for crafting a hypothetical rule can be expressed as follow:

$$\widetilde{C}_{R+1} = X_N \pm \Delta X,\ diag(\Sigma_{R+1}) = \frac{\max((C_i - C_{i-1}), (C_i - C_{i+1}))}{\sqrt{\frac{1}{\ln(\epsilon=0.5)}}} \tag{6.26}$$

where ϵ is a predefined constant with a set value of 0.5. The ϵ regulates the proportion of rule base plenitude. ΔX is the uncertainty factor which initializes the footprint of uncertainty. In McSIT2RFNN, the value of ΔX is fixed at 0.1

for simplicity, although one can also use an optimization technique to adjust the uncertainty factor. A hypothetical rule can be added as a new rule by utilizing Equation (6.26), only if the condition of Equation (6.27) is satisfied as follows:

$$\max_{i=1,\ldots,R}(\mathcal{E}_i) \leq (\mathcal{E}_{P+1}) \tag{6.27}$$

However, this condition itself does not suffice to be the only criteria to judge the contribution of a hypothetical rule because of the fact where limited information in respect to the spatial proximity of a data sample to existing rules is included. The distance information is required to delineate its relevance to current training concept. To overcome the limitation, another rule growing condition need to be satisfied as follows:

$$Fz \leq \rho, \text{ where } Fz = \max_{i=1,\ldots,R}\left(q\left(\underline{\eta_{out}}\right)_i^3 + (1-q)\left(\overline{\eta_{out}}\right)_i^3\right) \tag{6.28}$$

where ρ denotes a critical value of the chi-square distribution χ^2 with nu degrees of freedom and a significant α level. In [82], the ρ is expressed as $\rho = \exp(-\chi^2(\alpha))$, which is similar to the expression of [257]. In McSIT2RFNN, the value of α is set as 5%. To compute the Fz of Equation (6.28), the q design factors are applied for considering the effect of lower and upper rules. When a newly added rule satisfies the condition of Equation (6.28), the new rule is sufficiently away from the existing rules, and consequently, has a low risk of overlapping. A similar approach is observed in [49,255,256]. However, McSIT2RFNN utilizes the spatial firing strength instead of measuring point to point distance [49, 255, 256]. The second section of Equation (6.28) indicates the maximum spatial firing strength, which is also known as the winning rule. Finally, a hypothetical rule is added as a new rule by complying Equation (6.26), Equation (6.27) and Equation (6.28),

where the consequent part of the new rule is expressed as follows:

$$\Omega_{R+1} = \Omega_{win}, \quad \Psi = \overline{\omega} \tag{6.29}$$

where $\overline{\omega}$ is a large positive constant of magnitude of 10^5.

When a hypothetical rule does not satisfy the condition of neither Equation (6.27) nor Equation (6.28), then it is not added as a new rule. Nonetheless, the rule is then utilized by fine tuning its antecedent part. This tuning helps to absorb information carried by the latest data stream, while it maintains the existing network architecture as follows:

$$\widetilde{C}_{win}{}^{N} = \frac{N_{win}{}^{N-1}}{N_{win}{}^{N-1}+1}\widetilde{C}_{win}{}^{N-1} + \frac{(X_N - \widetilde{C}_{win}{}^{N-1})}{N_{win}{}^{N-1}+1} \tag{6.30}$$

$$\Sigma_{win}(N)^{-1}7 = \frac{\Sigma_{win}(N-1)^{-1}}{1-\alpha} + \frac{\alpha}{1-\alpha} \frac{(\Sigma_{win}(N-1)^{-1}(X_N - \widehat{C}_{win}{}^{N-1}))(\Sigma_{win}(N-1)^{-1}(X_N - \widehat{C}_{win}{}^{N-1}))^T}{1+\alpha(X_N - \hat{C}_{win}{}^{N-1})\Sigma_{win}(old)^{-1}(X_N - \hat{C}_{win}{}^{N-1})^T} \tag{6.31}$$

$$N_{win}{}^{N} = N_{win}{}^{N-1} + 1 \tag{6.32}$$

where $\alpha = 1/(N_{win}{}^{N-1}+1)$, $\widetilde{C}_{win} = [\underline{C}_{win}, \overline{C}_{win}]$, and $\widehat{C}_{win} = (\underline{C}_{win} + \overline{C}_{win})/2$. This adaptation technique is extracted from the idea of the sequential maximum likelihood principle with an extension for incorporating the interval valued multivariate Gaussian function. Here the mid-point of uncertain centroids are utilized to adapt certain input covariance matrix. The inverse covariance matrix is adjusted directly with no re-inversion process. This re-inversion process slows down the model update. Moreover, it may cause unstable computation in the presence of an ill-defined covariance matrix. In relationship to scaffolding theory, the rule

growing and adaptation technique described in this sub-section can be categorized as the problematizing component of active supervision due to its relationship with the drift handling approach due to the capability of updating the model with respect to the learning context. To overcome the drift, McSIT2RFNN embraces a passive approach by upgrading its structure continuously in accordance with the new incoming samples and does not depend upon a dedicated drift detection approach like [258].

6.4.2.2 Mechanism of pruning rules

The idea of the neuron significance is also used in the rule pruning scheme due to its capability of detecting a superfluous fuzzy rule which does not have a significant role during its lifespan. Generalized Type-2 Rule Significance (GT2RS) method is utilized in McSIT2RFNN, which is an enhanced version of the T2ERS method through the utilization of the interval-valued multivariate Gaussian function [80]. The GT2RS technique follows the same principle like its rule growing counterpart, where a fuzzy rule's contribution is evaluated based on its statistical significance presented in Equation (6.27). To sum up, a rule is pruned from the training process after satisfying a condition as follows:

$$\begin{aligned} &\mathcal{E}_i < mean(\mathcal{E}_i) - 2std(\mathcal{E}_i),\ mean(\mathcal{E}_i) = \frac{\sum_{n=1}^{N} \mathcal{E}_{i,n}}{N}, \\ &std(\mathcal{E}_{i,n}) = \sqrt{\frac{\sum_{n=1}^{N} (\mathcal{E}_{i,n} - mean(\mathcal{E}_i))^2}{N-1}} \end{aligned} \tag{6.33}$$

The calculation of mean and standard deviation of Equation (6.33) can be done easily in a recursive way. The condition of Equation (6.33) analyzes not only the statistical contribution of ith rule during its lifetime but also the down-trend of the statistical contribution of that rule. The GT2RS method can approximate

the rule significance rigorously by considering the overall training region, which verifies the methods effectiveness. In addition, the capability of handling complex and irregular data distributions of the GMM based input density function $p(x)$ is indicating that the future contribution of the ith fuzzy rule is also taken into account during the estimation of the rule significance. Furthermore, by utilizing Equation (6.27) in GT2RS method, the influence of the local sub-model Ω_i is considered, which is usually ignored by most of the rule pruning techniques. It is worth noting that the contribution of a fuzzy rule to the overall system output is highly affected by the output weight. Low output weight forces the output of a fuzzy rule to be negligible. The GT2RS method is representing the fading component of active supervision in scaffolding theory.

In this work, using the default threshold values of growing and pruning module, only two rules are generated and no rules are pruned to identify the quadcopter from data streams with a very insignificant RMSE. Therefore, to observe the rule pruning mechanism clearly, the rule pruning threshold has been reduced from 0.9 to 0.4 and rule growing threshold from 0.45 to 0.25 in case of modeling quadcopter with 27000 samples. After that, the number of generated and pruned rules have been witnessed graphically as shown in Figure 6.1.

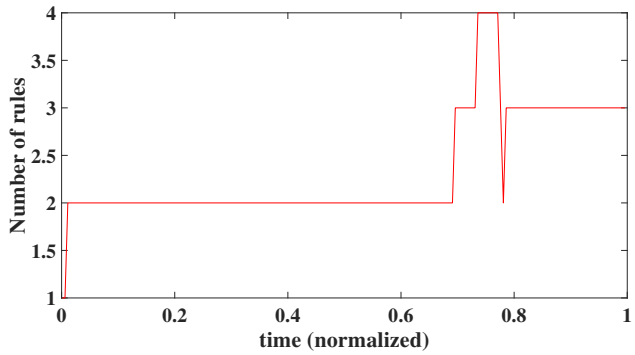


Figure 6.1: Number of added and pruned rules with a reduced threshold (in case of quadcopter model with 27000 samples)

6.4.2.3 Mechanism of forgetting and recalling rules

Type-2 Relative Mutual Information (T2RMI) method is utilized in McSIT2RFNN for detecting the obsolete rules, where the main idea is to examine the correlation between the fuzzy rules and the target concept. This T2RMI method is an improved version of the RMI method in [259] with respect to the sequential working framework of the T2RMI. Moreover, the T2RMI method is tailored to cope up with the methodology of interval type-2 fuzzy system. Unlike the RMI, in T2RMI the maximum compression index (MCI) [169] is utilized, which ameliorates the robustness of the linear correlation measure. In comparison with other linear correlation measures like Pearson coefficient, the MCI is not affected by rotation. The MCI is another improved characteristic of the T2RMI method with respect to the RMI method since the RMI method is still supported by the classic symmetrical uncertainty approach. The T2RMI also has the ability to detect outdated fuzzy rules by analyzing their relevance to the current data progression. In McSIT2RFNN, the T2RMI method is expressed as follows:

$$\xi\big(\big(\widetilde{\eta}_{out}\big)_i^3, y_{out}\big) = q_0\xi\big(\big(\underline{\eta}_{out}\big)_i^3, y_{out}\big) + (1-q_0)\xi\big(\big(\overline{\eta}_{out}\big)_i^3, y_{out}\big) \tag{6.34}$$

$$\xi\big(\big(\underline{\eta}_{out}\big)_i^3, y_{out}\big) = \frac{1}{2}\big(\mathrm{var}\big(\underline{\eta}_{out}\big)_i^3\big) + \mathrm{var}(y_{out}) - \sqrt{(\mathrm{var}\big(\underline{\eta}_{out}\big)_i^3 + \mathrm{var}(y_{out}))^2 - 4\mathrm{var}\big(\underline{\eta}_{out}\big)_i^3\ \mathrm{var}(y_{out})\big(1-\rho\big(\big(\underline{\eta}_{out}\big)_i^3, y_{out}\big)^2\big)} \tag{6.35}$$

$$\rho\big(\big(\underline{\eta}_{out}\big)_i^3, y_{out}\big) = \frac{\mathrm{cov}\big(\underline{\eta}_{out}\big)_i^3, y_{out})}{\sqrt{\mathrm{var}\big(\underline{\eta}_{out}\big)_i^3)\mathrm{var}(y_{out})}} \tag{6.36}$$

where $\mathrm{var}\big(\underline{\eta}_{out}\big)_i^3$, $\mathrm{cov}\big(\underline{\eta}_{out}\big)_i^3$, $\rho\big(\underline{\eta}_{out}\big)_i^3$ respectively represent the variance, covariance, Pearson index and output variable of the ith fuzzy rule with a lower bound. A similar technique is also applied to the upper bound of the fuzzy rule $\xi\big(\big(\overline{\eta}_{out}\big)_i^3, y_{out}\big)$. Since the spatial firing strength extracts the relevance of the fuzzy

rule in the input space, the fuzzy rule is represented by the spatial firing strength. In principle, $\xi\big(\big(\widetilde{\eta}_{out}\big)_i^3, y_{out}\big)$ implies the eigenvalue for the normal direction to the principal component of two variables $\big(\big(\widetilde{\eta}_{out}\big)_i^3, y_{out}\big)$, where maximum data compression is attained in the time of projection of information along its principal component direction. Therefore, the MCI has the ability to categorize the cost of discarding the ith rule from the training process, aiming to achieve the maximum amount of information compression. Some interesting features of the MCI is exposed in [83]. A fuzzy rule is regarded as obsolete, or the rule is forgotten after satisfying the condition as follows:

$$\begin{aligned} &\xi_{i,o} < mean(\xi_{i,o}) - 2std(\xi_{i,o}),\ mean(\xi_{i,o}) = \frac{\sum_{n=1}^{N} \xi_{i,o}^n}{N}, \\ &std(\xi_{i,o}) = \sqrt{\frac{\sum_{n=1}^{N} (\xi_{i,o}^n - mean(\xi_{i,o}))^2}{N-1}} \end{aligned} \tag{6.37}$$

The T2RMI method is also utilized to recall the discarded rules when they become relevant again to the output of the system. This function is supported by the fact that the correlation of a rule to target concept is influenced by the environments. In other words, in McSIT2RFNN a fuzzy rule is not permanently deleted and is added to a list of rules pruned by the T2RMI method $R^* = R^* + 1$, where R^* is the number of deactivated rules by the T2RMI method. Such a rule may be recalled in the future when it becomes relevant again to the systems output. This rule recall scenario makes the T2RMI method effective to deal with the cyclic drift by remembering old data distribution, which increases the relevance of the obsolete rules. The rule recall technique is activated after satisfying the condition as follows:

$$\max(\xi_{i^*}) > \max(\xi_i),\ \text{where } i^* = 1, ..., R^* \text{ and } i = 1, ..., R \tag{6.38}$$

From Equation (6.38) it is obvious that a rule is recalled when the validity of the obsolete rule is higher than any of the existing rules. Therefore, an obsolete rule brings the most compatible concept to describe the current data trend and should be reactivated as follows:

$$\widetilde{C}_{R+1} = \widetilde{C}_{i^*}, \quad {\Sigma_{R+1}}^{-1} = {\Sigma_{i^*}}^{-1}, \quad \Psi_{R+1} = \Psi_{i^*}, \Omega_{R+1} = \Omega_{i^*} \tag{6.39}$$

The rule recall scenario can be categorized as the problematizing part of the scaffolding theory.

6.4.2.4 Mechanism of merging rule

In the online identification of a quadcopter, a complete dataset may not be available. This phenomenon creates an opportunity for two rules to move together which may cause a significant overlapping as a result of the continuous adaptation of fuzzy rules [171]. Therefore, an online rule merging mechanism is required to reduce the system's complexity and to improve rule interpretability. Recently, the idea of online rule merging has been introduced in EIS by [173, 252]. However, in these approaches, an over-dependence on a problem-specific predefined threshold to determine an acceptable level of overlapping is observed, which limits the flexibility of EIS.

A novel online rule merging technique called Type-2 Geometric Criteria (T2GC) is utilized in McSIT2RFNN. T2GC is an extended version of geometric criteria of [57], which was developed for the type-1 fuzzy system. This T2GC not only observes the overlapping degree between rules but also looks at their geometric interpretation in the product space thoroughly. Two important properties of this T2GC are the overlapping degree and homogeneity. These two criteria are applied

mainly to examine the similarity of the winning rule in light of the fact that the winning rule is the only one to receive the rule premise adaptation expressed in Equation (6.34)-(6.36) and a major underlying reason of overlapping. Hence, this procedure targets to relieve the computational burden.

6.4.2.4.1 Overlapping Degree: The overlapping degree examines the similarity level of two rules to analyze their possibility of being redundant. Because of the necessity of developing a threshold-free rule merging process and the construction of McSIT2RFNN is with multi-variable Gaussian function, the Bhattacharyya distance is utilized [260]. The benefits of using the Bhattacharyya distance is that it can analyze whether two clusters are exactly disjoint, touching, or overlapping without any trouble in selecting a predefined threshold. The overlapping degree between the winning rule and other rules $i = \{1, ..., R\} \setminus \{win\}$ can be expressed as:

$$s_1(win, i) = (1 - q)\overline{s}_1(win, i) + q_o\underline{s}_1(win, i) \tag{6.40}$$

$$\overline{s}_1(win, i) = \frac{1}{8}(\overline{c}_{win} - \overline{c}_i)^T\Sigma^{-1}(\overline{c}_{win} - \overline{c}_i) + \frac{1}{2}\ln\frac{\det(\Sigma^{-1})}{\sqrt{\det(\Sigma_{win}^{-1})(\Sigma_i^{-1})}} \tag{6.41}$$

$$\underline{s}_1(win, i) = \frac{1}{8}(\underline{c}_{win} - \underline{c}_i)^T\Sigma^{-1}(\underline{c}_{win} - \underline{c}_i) + \frac{1}{2}\ln\frac{\det(\Sigma^{-1})}{\sqrt{\det(\Sigma_{win}^{-1})(\Sigma_i^{-1})}} \tag{6.42}$$

where $\Sigma^{-1} = (\Sigma_{win}^{-1} + \Sigma_i^{-1})/2$. The conditions such as $s_1(win, i) > 0, s_1(win, i) < 0,$ and $s_1(win, i) = 0$ exhibit respectively the overlapping, disjointing, and touching phenomenon of two clusters. In the McSIT2RFNN, the rule merging process is

considered mandatory when two rules are overlapping and/or touching as follows:

$$s_1(win, i) \geq 0 \tag{6.43}$$

It is important to mention that the utilization of Bhattacharyya distance is suitable for the McSIT2RFNNs rule since the multivariate Gaussian function in the Bhattacharyya distance has a one-to-one relationship with that of the McSIT2RFNN.

6.4.2.4.2 Homogeneity criterion: Homogeneity of clusters has an important role in merging two clusters, since the merging of non-homogeneous clusters may cause cluster delamination, undermining generalization and representation of local data clouds [57,261]. The cluster delamination is indicating an over-sized cluster that covers two or more distinguishable data clouds. The measure of homogeneity of clusters in McSIT2RFNN is formulated by examining the volume of the merged clusters in contrast with their individual volume as follows:

$$\underline{\nu}_{merged} + \overline{\nu}_{merged} < u(\overline{\nu}_i + \underline{\nu}_i + \overline{\nu}_{win} + \underline{\nu}_{win}) \tag{6.44}$$

Finally, after satisfying the condition of Equation (6.43), and Equation (6.44), the rules are merged. Equation (6.44) also presents a minor chance of cluster delamination since the volume of the merged cluster is less than the volume of two independent clusters, and therefore the two clusters form a joint homogeneous region. The term u is involved in Equation (6.44) to obstruct the curse of dimensionality.

After satisfying all rule merging conditions, two merging candidates are combined. Since a rule containing more supports should have higher influence to

ultimate shape and orientation of the merged cluster, the rule merging procedure is directed by the weighted average strategy [171] as follows:

$$\widetilde{C}_{merged}{}^{new} = \frac{\widetilde{C}_{win}{}^{old} N_{win}{}^{old} + \widetilde{C}_i{}^{old} N_i{}^{old}}{N_{win}{}^{old} + N_i{}^{old}},$$
$$\widetilde{C}_i = [\underline{C}_i + \overline{C}_i], \quad N_{merged}{}^{new} = N_{win}{}^{old} + N_i{}^{old} \tag{6.45}$$

$$\Sigma^{-1}_{merged}{}^{new} = \frac{\Sigma^{-1}_{win}{}^{old} N_{win}{}^{old} + \Sigma^{-1} i^{old} N_i{}^{old}}{N_{win}{}^{old} + N_i{}^{old}},$$
$$\Omega_{merged}{}^{new} = \frac{\Omega_{win}{}^{old} N_{win}{}^{old} + \Omega_i{}^{old} N_i{}^{old}}{N_{win}{}^{old} + N_i{}^{old}} \tag{6.46}$$

6.4.2.5 Mechanism of online feature selection

The feature selection mechanism plays an important role to improve the performance of EIS by reducing computational complexity and makes modeling problems easier to solve. Therefore, feature selection characterizes the complexity reduction part of scaffolding theory. Majority of these feature selection mechanisms are a part of a pre-processing step. However, very recently research in online feature selection mechanism of EIS is being conducted [171,252,261]. These techniques can minimize the significance of inconsequential features by assigning a low weight. But they still keep superfluous input attributes in the memory. Therefore the complexity issue remains unsolved. Besides, in recent EISs the online feature selection mechanism only measures the relevance between input attributes and target variables. They do not consider the redundancy among input attributes. A novel online feature selection mechanism, called Sequential Markov Blanket Criterion (SMBC) is utilized in McSIT2RFNN, which is able to mitigate all the above mentioned limitations of the existing EIS and works completely with the single-pass learning environment. It is an improved version

of MBC [262].

By analyzing the Markov blanket theory, four different types of input features are obtained with respect to their contribution; they are namely: irrelevant, weakly relevant, weakly relevant but non-redundant, and strongly relevant. The SMBC targets to eliminate irrelevant, weakly relevant input features from the training process, while keeping weak relevant but non-redundant, and strongly relevant features in the training process. In SMBC, C-Correlation and F-Correlation tests are developed and then utilized to deal with the issue of irrelevance and redundancy respectively. These two correlations are defined as follows:

Definition 1 (C-Correlation) [262]*: The relevance of the input feature is indicated by the correlation of input feature x_k and target variable t_{out}, which are measured by the C-correlation $C(x_k, t_{out})$.*

Definition 2 (F-Correlation) [262]*: The issue of redundancy is signified by the similarity degree of two different input variables $x_k, x_{k1}, k \neq k1$. The measure of similarity between two input attributes is called the F-correlation $F(x_k, x_{k1})$.*

The MCI method exposed in Equation (6.34)-(6.36) is adopted to analyze the C and F-correlation. It is accomplished by just replacing $\left(\left(\widetilde{\eta}_{out}\right)_i^3, y_{out}\right)$ in Equation (6.34)-(6.36) with (x_k, t_{out}), and (x_k, x_{k1}). The SMBC is implemented in two stages, where in the first stage the F-correlation eliminates the inconsequential features, and consequently reduce complexity. It helps the next step, the C-correlation, to run with a smaller number of input variables.

6.4.2.6 Mechanism of adapting q design factor and recurrent weight

Adaptation or fine tuning of the free network parameters of the McSIT2RFNN, namely the design coefficients and the recurrent weights are accomplished by utilizing the Zero-order Density Maximization (ZEDM) method. This ZEDM method is an improved version gradient descent technique since ZEDM utilizes error entropy as cost function unlike the mean square error (MSE) in gradient descent technique, therefore leads to a more accurate prediction. Since the accurate model of the error entropy is too complex to be derived with the first-principle technique, the cost function is formulated by utilizing the Parzen Window density estimation method and can be expressed as follows:

$$\hat{f}(0) = \frac{1}{N\phi\sqrt{2\pi}} \sum_{n=1}^{N} \exp(-\frac{{e_{n,0}}^2}{2\phi^2}) = \frac{1}{N\phi\sqrt{2\pi}} \sum_{n=1}^{N} K(\frac{-{e_{n,0}}^2}{2\phi^2}) \tag{6.47}$$

where $e_{n,0}$ represents the system error of the oth output variable, T denotes a smoothing parameter, fixed as 1 for simplicity and N is the total number of samples seen so far. It is worth noting that a recursive expression can be derived to satisfy the one-pass learning requirement. The detailed adaptation process is explained in [83].

6.4.2.7 Mechanism of adapting rule consequent

The adaptation of the rule consequent represents the passive supervision of scaffolding theory because it relies on the system error, actualizing the action-consequent mechanism. For adapting the rule consequent the Fuzzily Weighted Generalized Recursive Least Square (FWGRLS) method [1] is used in McSIT2RFNN. FWGRLS is an improved version of the Generalized Recursive Least Square (GRLS) method [175] and performs locally. This local learning scenario provides a flexible

mechanism and greater robustness, because each rule is fine-tuned separately. Thereby, entire learning procedures of a particular rule do not affect the stability and convergence of remaining rules. The local learning scenario also raises the interpretability of the TSK fuzzy rule as explained in [263]. The details of the FWGRLS method is elaborated in [1, 56, 252].

6.5 Results From Experimental Flight Data

Set-up for the quadcopter flight experiment and its online system identification results are summarized in this section.

6.5.1 Experimental set up of quadcopter flight

Our quadcopter experiments were accomplished in the indoor UAV laboratory at the University of New South Wales, Canberra campus. We use a Pixhawk autopilot framework developed by an open and independent hardware project called PX4. The Pixhawk Flight Control Unit (FCU) is manufactured and sold by 3D robotics and has three onboard sensors; namely gyroscope, accelerometer, and magnetometer. The experimental quadcopter model is displayed in Figure 6.2. To record quadcopter flight data the Robot Operating System (ROS), running under the Ubuntu 16.04 version of Linux was used. A ROS package called MAVROS was utilized in this work, where the MAVROS had enabled communication between the PX4 and a ROS enabled computer. By utilizing the ROS, a well-structured communication layer was introduced into the quadcopter that reduced the burden of having to reinvent necessary software.

During the real-time flight testing accurate vehicle position, velocity, and orientation were the required information for verification of the proposed Mc-

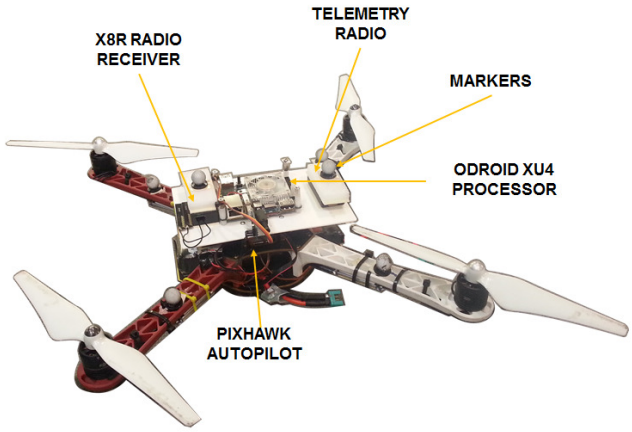


Figure 6.2: Pixhawk autopilot based experimental quadcopter model

SIT2RFNN based on-line system identification of quadcopter. In order to track the quadcopter in three-dimensional space, a VICON optical motion capture system was employed to track the UAV motion with sub-millimetre accuracy. The indoor VICON motion capture system consisted of a volume that was $10\times10\times4.3$ m^3 and formed by a netted truss framework.

The object tracking information was routed to the quadcopter via a custom SDK UDP package to the desired IP address which in our platform was an Odroid single board computer. At each time step, position, velocity, and orientation information was recorded. During testing the pilot controlled the quadcopter RUAV manually from an RC transmitter using pitch, roll and yaw and thrust commands. To record key published topics, the *rosbag* recording tool was used. The *rosbag* enables us to record and synchronize all critical experimental data via published topics that are required for online system identification. Figure 6.3 represents the way of communicating of the experimental quadcopter UAV system during all the flight tests, where the dotted lines represent wireless communication and solid lines represent the wired connection.

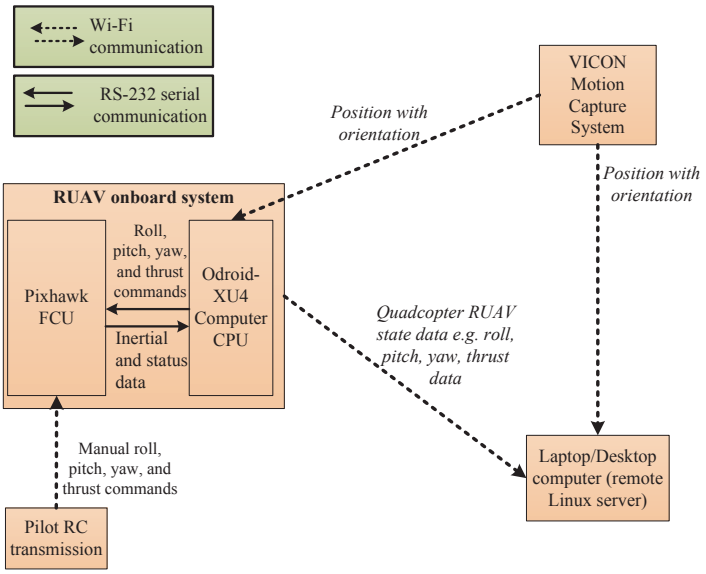


Figure 6.3: Pixhawk quadcopter RUAV's communication flow

6.5.2 Online system identification results

For system identification or moeling of the quadcopter, a variety of quadcopter flight data have been utilized. Among them, there are three different datasets of quadcopter's altitude, consisting of approximately 9,000, 27,000 and 66,000 samples. Using these datasets, the quadcopter's altitude based multi-input-single-output (MISO) online system identification model of quadcopter has been constructed by utilizing the proposed McSIT2RFNN technique [83]. The proposed technique [83] based online data-driven quadcopter model has been also structured from four inputs and output datasets (vertical altitude and the three rotational movements (θ, ϕ, ψ)). The time step of all the dataset was 0.0198 sec. For comparing and validating the accuracy of the proposed technique, the quadcoper's data-driven model has also constructed with eight different renowned EIS based neuro-fuzzy system, namely: eTS [44], simp_eTS [46], DFNN [166], GDFNN [184], FAOSPFNN [185], GENEFIS [1], Adaptive Neuro Fuzzy Inference System (ANFIS) [264], and PANFIS [56]. For the performance analysis the

RMSE, number of network parameters, number of training samples, fuzzy rules, and execution time have been considered for each algorithm. All the results are summarized in Tables from 6.1 to 6.4. Table 6.1 to Table 6.3 express the results for a MISO quadcopter model with approximately 27,000, 66,000, and 9,000 samples of quadcopter's altitude for three different flight respectively. Table 6.4 summarizes the results of the MIMO quadcopter model. It is clearly observed from the results that, among these eight different algorithms the proposed McSIT2RFNN algorithm performs the best since the lowest RMSE, fastest execution is observed. Besides, by utilizing the what-to-learn mechanism the McSIT2RFNN has reduced the number of samples required to train, which helps to reduce the execution time as observed in the Tables from 6.1 to 6.5. This sample deletion mechanism is not utilized in any other renowned variants of EIS discussed in this chapter. The altitude tracking performance of the proposed McSIT2RFNN algorithm based MISO quadcopter model with nearly 27,000 and 66,000 samples are displayed in Figure 6.5. A MIMO quadcopter model with nearly 9,000 samples for identifying thrust, roll, pitch, and yaw are displayed in Figure 6.6. From those figures, it is clearly observed that the proposed online models' output are following the desired dataset collected experimentally very closely in all the cases. The evolving and online nature of the proposed McSIT2RFNN technique helps to track the quick changes in the desired trajectory.

In this chapter, to model the quadcopter with approximately 27,000, 66,000, and 9,000 samples, two inputs ($X(t)$ and $Y(t-1)$) are utilized to generate the rules. The If-Then expression of the rule in this work is exposed in Equation (6.3). However, the rule presented in Equation (6.3) is not transparent enough to expose atomic clause of the human-like linguistic rule. It operates in a totally high dimensional space, therefore cannot be represented in fuzzy set directly.

To express such fuzzy rules with the multidimensional kernel, the phrase "close to" is conventionally used [265, 266]. As a solution, a transformation strategy is employed in this work, as expressed in Equation (6.4) to convert the high dimensional space based rules to lower dimensional human-like linguistic rules. After transformation, rules can be expressed in a conventional interval type-2 fuzzy set environment as exposed in Equation (6.5), (6.6), and (6.7). Utilizing those fuzzy set expression of Equation (6.5), (6.6), and (6.7), a more interpretable fuzzy rule is exhibited in Equation (6.8).

Now the non-axis-parallel ellipsoidal rule generated in the high dimensional space by the McSIT2RFNN in case of quadcopter model with 27000 samples can be expressed as follows:

$$
\begin{aligned}
&R_1 : \text{If} \quad X \text{ is close to} \\
&\widetilde{\eta}_{out}\left(\underline{\eta}_{out} = \left[\begin{bmatrix} -0.5934 \\ -0.2458 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right], \overline{\eta}_{out} = \left[\begin{bmatrix} 0.0066 \\ 0.3542 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right]\right) \\
&\text{Then } y_1 = -0.0043 - 0.0039X_1 - 0.0046\tau(X_1) + 0.9999X_2 - 0.0002\tau(X_2)
\end{aligned} \tag{6.48}
$$

where $\widetilde{\eta}_{out} = \left[\underline{\eta}_{out}, \overline{\eta}_{out}\right]$ is the rule antecedent of the multi-variable Gaussian function, which consists of uncertain centroids $\widetilde{\zeta} = \left[\underline{\zeta}, \overline{\zeta}\right]$, where $\underline{\zeta} = \begin{bmatrix} -0.5934 \\ -0.2458 \end{bmatrix}$, and $\overline{\zeta} = \begin{bmatrix} 0.0066 \\ 0.3542 \end{bmatrix}$, and the inverse co-variance matrix $\Sigma^{-1} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$; y_i is denoting the rule consequent of the ith rule obtained from $y_i = x_e^i \Theta_i$, where Θ_i is a connection weight between the temporal ring layer and the output layer; In this work, X is a 2-D input composition like $[x_1,\ x_2]$, then x_e is the extended input vector obtained using Chebyshev polynomial and can be expressed as

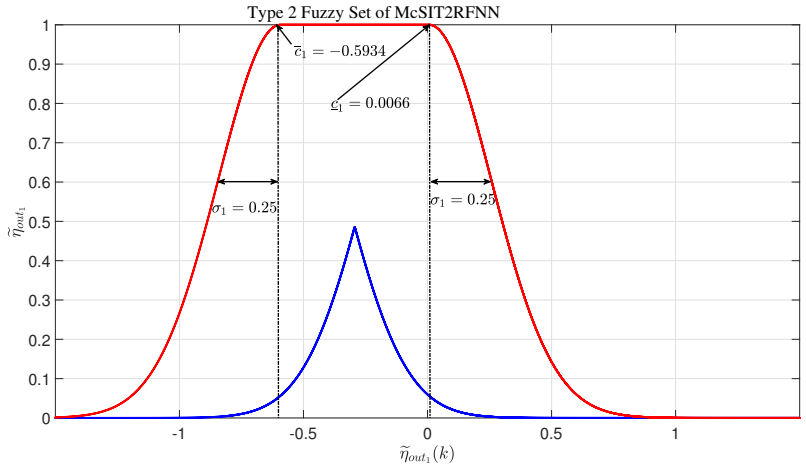


Figure 6.4: 1st Membership function (with uncertain centroid and certain width) of rule 1 of McSIT2RFNN

$x_e = [1,\ x_1,\ \tau(x_1),\ x_2,\ \tau(x_2)]$, where, $x_e \in \Re^{1\times(2\mu+1)}$, μ is expressing the input dimension.

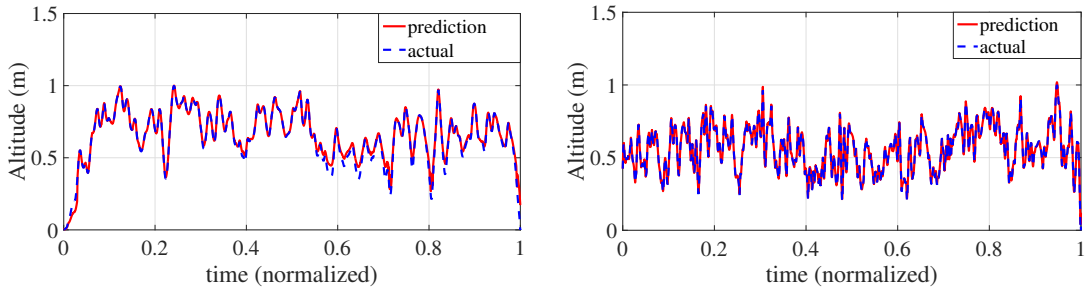
After transformation of the rule presented in Equation (6.48) to a lower-dimensional space, it can be expressed as follows:

$$\begin{aligned} &R_1 : \text{If} \quad X_1 \text{ is close to } \widetilde{\eta}_{out_1}\left(\underline{\eta}_{out_1}(-0.5934,\ 0.25), \overline{\eta}_{out_1}(0.0066,\ 0.25)\right) \\ &\text{and } X_2 \text{ is close to } \widetilde{\eta}_{out_2}\left(\underline{\eta}_{out_2}(-0.2458,\ 0.20), \overline{\eta}_{out_2}(0.3542,\ 0.20)\right), \\ &\text{Then } y_1 = -0.00432 - 0.00387X_1 - 0.00457\tau(X_1) + 0.99995X_2 - 0.00022\tau(X_2) \end{aligned} \tag{6.49}$$

where $\widetilde{\eta}_{out_1}$ and $\widetilde{\eta}_{out_2}$ stand for the interval-valued Gaussian membership function corresponding to X_1, X_2. The uncertain centroid of $\widetilde{\eta}_{out_1}$ is $\widetilde{c}_1 = [\underline{c}_1, \overline{c}_1] = [-0.5934,\ 0.0066]$, and width is $\sigma_1 = 0.25$, and for $\widetilde{\eta}_{out_2}$ those parameters are $\widetilde{c}_2 = [\underline{c}_2, \overline{c}_2] = [-0.2458,\ 0.3542]$, $\sigma_2 = 0.20$. The 1st membership function of our rule 1 is shown in Figure (6.4). The plotted membership function has uncertain centroid of $\widetilde{c}_1 = [\underline{c}_1, \overline{c}_1] = [-0.5934,\ 0.0066]$, and certain width of $\sigma_1 = 0.25$.

6.5.3 Online system identification with noisy samples

To prove the robustness of the McSIT2RFNN against uncertainties, another quadcopter flight experiment has been accomplished considering some noise from VICON optical motion capture system. The quadcopter flight dataset consists of nearly 27000 samples with a noisy 1000 samples, which is utilized to model the quadcopter. The adaptation power of the proposed algorithm against noise is clear from the lowest obtained RMSE compared to its type-1 counterparts. Furthermore, with the noisy data, it can model the quadcopter with only 546 data samples, where its type-1 variants need all the training samples i.e. 16483 (60% of the total samples). Thereby, the lowest execution time in modeling the quadcopter is also observed from the type-2 fuzzy-based proposed McSIT2RFNN algorithm. Therefore, the results are clearly indicating its improved performance and uncertainty handling capacity than the type-1 counterpart. The results are summarized in Table 6.5.



(a) with approx. 27,000 samples (b) with approx. 66,000 samples

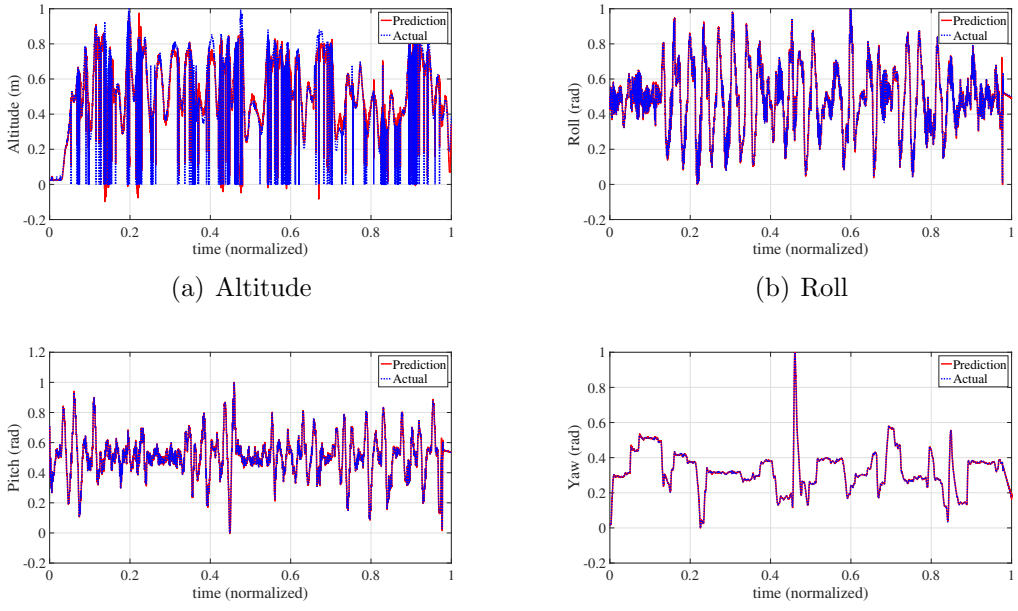
Figure 6.5: System identification of Quadcopter SISO model

Table 6.1: Online system identification result comparison of SISO quadcopter model (approx. 27,000 samples)

Algorithm	Reference	RMSE	Network Parameters	Training Samples	Number of Fuzzy Rule	Execution Time (sec)
DFNN	[166]	0.0780	10	16483	1	194.86
GDFNN	[184]	0.0067	10	16483	1	329.93
FAOSPFNN	[185]	0.0280	12	16483	2	38.10
eTS	[44]	0.0021	40	16483	4	9.39
simp_eTS	[46]	0.0020	13	16483	1	3.50
GENEFIS	[1]	0.0020	63	16483	1	3.29
PANFIS	[56]	0.0020	5	16483	1	2.92
ANFIS	[264]	0.0061	36	16483	6	33.01
McSIT2RFNN	[83]	0.0013	32	1279	2	2.27

Table 6.2: Online system identification result comparison of SISO quadcopter model (approx. 66,000 samples)

Algorithm	Reference	RMSE	Network Parameters	Training Samples	Number of Fuzzy Rule	Execution Time (sec)
DFNN	[166]	0.0810	10	39640	1	1113.69
GDFNN	[184]	0.0080	10	39640	1	1666.33
FAOSPFNN	[185]	0.0150	12	39640	2	135.59
eTS	[44]	0.0016	40	39640	2	21.03
simp_eTS	[46]	0.0015	13	39640	1	11.20
GENEFIS	[1]	0.0015	4	39640	1	7.63
PANFIS	[56]	0.0015	5	39640	1	6.93
ANFIS	[264]	0.0050	30	39640	5	34.93
McSIT2RFNN	[83]	**0.0008**	32	**2329**	2	**5.5**



(a) Altitude (b) Roll

(c) Pitch (d) Yaw

Figure 6.6: System identification of Quadcopter MIMO model with approx. 9,000 samples

Table 6.3: Online system identification result comparison of SISO quadcopter model (approx. 9,000 samples)

Algorithm	Reference	RMSE	Network Parameters	Training Samples	Number of Fuzzy Rule	Execution Time (sec)
DFNN	[166]	0.15	10	5467	1	19.77
GDFNN	[184]	0.14	10	5467	1	23.64
FAOSPFNN	[185]	0.21	12	5467	2	28.58
eTS	[44]	0.14	40	5467	4	2.10
simp_eTS	[46]	0.13	13	5467	4	1.77
GENEFIS	[1]	0.13	26	5467	1	1.10
PANFIS	[56]	0.135	5	5467	1	1.15
ANFIS	[264]	0.46	48	5467	8	36.94
McSIT2RFNN	[82]	0.13	32	769	2	**0.91**

Table 6.4: Online system identification result comparison of MIMO quadcopter model (approx. 9000 samples)

Algorithm	Reference	RMSE1	RMSE2	RMSE3	RMSE4	Network Parameters	Training Samples	Number of Fuzzy Rule	Execution Time (sec)	Input attribute
DFNN	[166]	0.26	0.18	0.14	0.15	10	5753	1	49.98	4
GDFNN	[184]	0.23	0.18	0.14	0.13	10	5753	1	85.75	4
FAOSPFNN	[185]	0.55	0.28	0.14	0.13	12	5753	1	12.11	4
eTS	[44]	0.22	0.20	0.15	0.10	292	5753	14	20.06	4
simp_eTS	[46]	0.29	0.31	0.29	0.18	104	5753	5	8.11	4
GENEFIS	[1]	0.24	0.19	0.19	0.11	4	5753	1	6.1	1
PANFIS	[56]	0.24	0.17	0.14	0.13	3	5753	1	5.4	4
McSIT2RFNN	[82]	**0.23**	**0.17**	**0.14**	**0.10**	66	**461**	2	**4.5**	4

Table 6.5: Online system identification result comparison of SISO quadcopter model (approx. 27,000 samples with 1000 noisy samples)

Algorithm	Reference	RMSE	Network Parameters	Training Samples	Number of Fuzzy Rule	Execution Time (sec)
DFNN	[166]	0.0583	10	16483	1	211.7
GDFNN	[184]	0.0141	10	16483	1	328.56
FAOSPFNN	[185]	0.0242	12	16483	2	216.47
eTS	[44]	0.0072	40	16483	4	9.39
simp_eTS	[46]	0.0212	13	16483	6	14.36
GENEFIS	[1]	0.0067	63	16483	1	3.20
PANFIS	[56]	0.0067	5	16483	1	2.79
ANFIS	[264]	0.0061	36	16483	6	32.29
McSIT2RFNN	[82]	**0.0011**	32	**546**	2	**2.13**

6.6 Summary

EIS-based autonomous learning algorithm is an appropriate candidate for modeling a complex and highly nonlinear system like quadcopter RUAV. The incorporation of McSLM with EIS makes it more appropriate. Such an advanced EIS called McSIT2RFNN is utilized to model the quadcopter with uncertainties from experimental quadcopter flight data. In McSIT2RFNN, a new local recurrent network architecture is driven by the interval-valued multivariate Gaussian function in the hidden node and the nonlinear Chebushev function in the consequent node. As with its predecessors, the McSIT2RFNN characterizes an open structure,

which can grow, prune, adjust, merge, recall its hidden node automatically and to select relevant data samples of quadcopter flight on the fly using an online active learning methodology. The McSIT2RFNN is also equipped with the online dimensionality reduction technique to cope with the curse of dimensionality. All learning mechanisms are carried out in the single-pass and local learning mode and actualize the plug-and-play learning principle, which aims to minimize the use of pre-and/or post-training steps. These features help the McSIT2RFNN method to identify the quadcopter RUAV more accurately than other variants of EIS. Thus, the accurate MISO and MIMO quadcopter modeling or better online identification results from McSIT2RFNN verifies their feasibility in modeling UAVs. In future research, an AICon for UAVs based on McSIT2RFNN will be developed.

Chapter 7

Conclusions

7.1 Research and Outcomes

In fuzzy system community, evolving neuro-fuzzy system (ENFS)- based autonomous learning algorithms are used to handle data stream regression problems and to model complex nonlinear dynamical systems due to their incremental architecture and online learning mechanism. To cope with the sudden changes in the distribution of the incoming data streams, and in the complex nonlinear systems' dynamics, these algorithms can evolve their structure autonomously. However, these algorithms face challenges in their real-time deployment with limited memory resources because of a higher number of free network parameters. To reduce the number of parameters, an ENFS-based novel autonomous learning algorithm, namely the parsimonious learning machine (PALM), is proposed in chapter 3.

ENFSs are usually constructed via hypersphere-based or hyperellipsoid-based clustering techniques (HSBC or HEBC) to automatically partition the input space into a number of fuzzy rule, where Gaussian, bell-shaped Gaussian functions

are used as membership functions. Such functions are associated with some antecedent parameters like the mean and width, which need to be adjusted continuously. To overcome the dependency on these antecedent parameters, hyperplane-based clustering (HPBC) is used in PALM to fully characterize the fuzzy rules. This strategy reduces the rule base parameter to the level of $C*(P+1)$ where C and P are the number of fuzzy rules and input dimensions, respectively. PALM features a fully open network structure where its rules can be automatically generated, merged, and updated on-demand in the one-pass learning fashion. The rule generation process is based on the self-constructing clustering approach, checking the coherence of input and output space. The rule merging scenario is driven by the similarity analysis via the distance and orientation of two hyperplanes. The online hyperplane tuning scenario is executed using the Fuzzily Weighted Generalized Recursive Least Square (FWGRLS) method. PALM is proposed in both type-1 and type-2 versions derived from the concept of type-1 and type-2 fuzzy systems. Type-1 version incurs fewer network parameters and faster training speed than the type-2 version whereas type-2 version expands the degree of freedom of the type-1 version by applying the interval-valued concept leading to be more robust against uncertainty than the type-1 version. Both type-1 and type-2 versions are simulated under two parameter optimization scenarios: 1) Type-1 PALM (L); 2) Type-1 PALM (G); 3) Type-2 PALM (L); 4) Type-2 PALM (G). L denotes the *Local* update strategy while G stands for the *Global* learning mechanism. The efficiency of the PALM has been tested in six real-world and artificial data stream regression problems where PALM outperforms recently published works in terms of network parameters and running time. It also delivers state of the art accuracies which happen to be comparable and often better than its counterparts.

Another ENFS-based advanced autonomous learning algorithm called McSIT2RFNN is utilized in chapter 6 to model the quadcopter with uncertainties from experimental quadcopter flight data. In McSIT2RFNN, a new local recurrent network architecture is driven by the interval-valued multivariate Gaussian function in the hidden node and the nonlinear Chebushev function in the consequent node. As with its predecessors, the McSIT2RFNN characterizes an open structure, which has the ability to grow, prune, adjust, merge, recall its hidden node automatically and to select relevant data samples of quadcopter flight on the fly using an online active learning methodology. The McSIT2RFNN is also equipped with the online dimensionality reduction technique to cope with the curse of dimensionality. All learning mechanisms are carried out in the single-pass and local learning mode and actualize the plug-and-play learning principle, which aims to minimize the use of pre-and/or post-training steps. These features help the McSIT2RFNN method to identify the quadcopter RUAV more accurately than other variants of ENFS. Thus, the accurate MISO and MIMO quadcopter modeling or better online identification results from McSIT2RFNN verifies their feasibility in modeling UAVs.

Besides using the evolving neuro-fuzzy systems in regression or as predictor, they have been also utilized to develop AICons for UAVs in this thesis. An advanced evolving neuro-fuzzy system, namely Generic Evolving Neuro-Fuzzy Inference System (GENEFIS) has been used to develop an AICon namely Generic-controller (G-controller) in chapter 5. It can alter its structure, and system parameters online to cope with changing dynamics of the plant to be controlled. Besides, the synthesis of SMC theory based adaptation laws improved its robustness against various internal and external uncertainties. The integration of GART+, multivariate Gaussian function, and SMC learning theory-based

adaptation laws yield a fast self-evolving mechanism of the G-controller is with a compact structure. These desirable features make the G-controller a suitable candidate for highly nonlinear autonomous vehicles. The controller's performance has been evaluated by observing the tracking performance in controlling simulated plants of UAVs, namely BIFW MAV and hexacopter for a variety of trajectories. Wind gust has been added to the BIFW MAV plant as environmental uncertainties to evaluate the G-controller's robustness against unknown perturbations, where satisfactory tracking of the desired trajectory proves the proposed AICon's performance to eliminate uncertainty. Both the Lyapunov theory and numerical experiments confirm the G-controller's stability.

Another fully AICon, namely PAC is proposed in chapter 4. A bottleneck of the existing AICons is the utilization of numerous free parameters and their online adaptation. Such inadequacy has been mitigated in our PAC since it has no premise parameters. The only parameter used in our AICon to acquire the desired tracking is the weight. For instance, if the evolving TS-fuzzy controllers were used in our experiment, with three rules, they would require 48 network parameters to be tuned, whereas PAC needs only 12 parameters with three rules. Apart from that, conventional AICons adhere to user-defined problem-based thresholds to shape their structure. In PAC, rather than predefined parameters, the bias-variance concept based network significance method is utilized to determine it's structure. The PAC has been verified by implementing them in various UAV plants namely BIFW MAV and hexacopter to track diverse trajectories. Achievements are contrasted with a commonly utilized PID controller, an adaptive nonlinear FFNN controller, and a TS-fuzzy controller. Furthermore, the controller's robustness against uncertainties and disruptions is ascertained by injecting a wind gust and sudden peak to the UAVs dynamics. In controlling

both plants with uncertainties, lower or comparable overshoot and settling time were observed from PAC with a simplified evolving structure, which is testifying its robustness against uncertainties and compatibility in regulating UAVs.

A simplified version of PAC, namely Reduced Parsimonious Controller (RedPAC) is also developed in chapter 4. In RedPAC, the number of consequent parameters has been further reduced to one parameter per rule. It eliminates the arduous problem in designing the AICon and enables its implementation to a minicomputer or embedded system. RedPAC's performance has been evaluated by implementing it to control a quadcopter simulator namely Dronekit-SITL. In addition, the trajectory tracking performance of the quadcopter is compared with its modeled or model-free counterparts namely PID, SMC, and PAC. RedPAC outperforms PID and SMC techniques. The results of tracking trajectories are also comparable to PAC. However, RedPAC needs comparatively less learning parameters to obtain similar or better tracking accuracy.

7.2 Recommendations for Future Research

Though the thesis has highlighted several limitations of the existing ENFS-based autonomous learning algorithms and proposed new algorithms with possible solutions, several open issues can be pursued to improve the developed methodology. Some promising directions for further investigation can be identified as follows:

- Our autonomous learning algorithm namely PALM exposes an open structure with autonomous appending or merging of fuzzy rules. It requires a minimum number of parameter-adaptations compared to the state of the art. However, it can be improved further in different ways. Firstly, at the present configuration, it does not have a soft feature reduction mechanism

to overlook the inconsequential input features. The significance of input features can be foreseen by measuring the mutual information among input attributes. After measuring the significance of input features, they can be evolved and pruned autonomously on the fly during the training period. Secondly, PALM is rooted with a shallow network configuration. Thus, it has lower generalization power than those of deep structures. It can be incorporated under a deep network structure.

- In this thesis, ENFS-based another autonomous learning algorithm namely McSIT2RFNN and an AICon namely G-controller is developed, where both of them have utilized multi-variate Gaussian function. It creates non-axis parallel clusters, which help them to capture the data not scattered in the main axis. However, uneven real-world data distribution are difficult to cover by a certain shape of the cluster. To overcome such limitations, the concept of clusters can be replaced by data clouds or hyperplanes, which help to create a more flexible cluster region.

- Three different AICons are developed in this thesis with their unique features. The closed-loop stability of all those controllers has been confirmed through Lyapunov theory and evaluated by numerical experimentation to regulate various UAVs. In the future, all these controllers will be executed through a hardware-based flight test of various UAVs.

References

[1] Pratama, M., Anavatti, S.G., Lughofer, E.: GENEFIS: toward an effective localist network. IEEE Transactions on Fuzzy Systems **22**(3) (2014) 547–562

[2] Škrjanc, I., Iglesias, J., Sanchis, A., Leite, D., Lughofer, E., Gomide, F.: Evolving Fuzzy and Neuro-Fuzzy Approaches in Clustering, Regression, Identification, and Classification: A Survey. Information Sciences (2019)

[3] Pan, Y., Yu, H.: Biomimetic hybrid feedback feedforward neural-network learning control. IEEE Transactions on Neural Networks and Learning Systems **28**(6) (2017) 1481–1487

[4] **Md Meftahul Ferdaus**, Anavatti, S.G., Garratt, M.A., Pratama, M.: Development of c-means clustering based adaptive fuzzy controller for a flapping wing micro air vehicle. Journal of Artificial Intelligence and Soft Computing Research **9**(2) (2019) 99–109

[5] Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Adaptive neuro-fuzzy control of a spherical rolling robot using sliding-mode-control-theory-based online learning algorithm. IEEE Transactions on Cybernetics **43**(1) (2013) 170–179

[6] Suresh, S., Kumar, M.V., Omkar, S., Mani, V., Sampath, P.: Neural networks based identification of helicopter dynamics using flight data. In: Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02. Volume 1., IEEE (2002) 10–14

[7] Samal, M.K., Anavatti, S., Garratt, M.: Neural network based system identification for autonomous flight of an eagle helicopter. IFAC Proceedings Volumes **41**(2) (2008) 7421–7426

[8] Kumar, M.V., Omkar, S., Ganguli, R., Sampath, P., Suresh, S.: Identification of helicopter dynamics using recurrent neural networks and flight data. Journal of the American Helicopter Society **51**(2) (2006) 164–174

[9] Bansal, S., Akametalu, A.K., Jiang, F.J., Laine, F., Tomlin, C.J.: Learning quadrotor dynamics using neural network for flight control. In: 2016 IEEE 55th Conference on Decision and Control (CDC), IEEE (2016) 4653–4660

[10] Harris, J., Arthurs, F., Henrickson, J.V., Valasek, J.: Aircraft system identification using artificial neural networks with flight test data. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE (2016) 679–688

[11] Xue, M.: UAV Trajectory Modeling Using Neural Networks. In: 17th AIAA Aviation Technology, Integration, and Operations Conference. (2017) 3072

[12] Rong, H.J., Yang, Z.X., Wong, P.K., Vong, C.M., Zhao, G.S.: Self-evolving fuzzy model-based controller with online structure and parameter learning for hypersonic vehicle. Aerospace Science and Technology **64** (2017) 1–15

[13] Doncieux, S., Meyer, J.A.: Evolving neural networks for the control of a lenticular blimp. In: Workshops on Applications of Evolutionary Computation, Springer (2003) 626–637

[14] Rong, H.J., Bai, J.M., Yang, J.: Aircraft sensor failure diagnosis using self-organizing fuzzy systems. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE (2015) 1–6

[15] Ferdaus, M.M., Pratama, M., Anavatti, S.G., Garratt, M.A.: Online identification of a rotary wing unmanned aerial vehicle from data streams. Applied Soft Computing **76** (2019) 313–325

[16] Ferdaus, M.M., Anavatti, S.G., Garratt, M.A., Pratama, M.: Evolving fuzzy inference system based online identification and control of a quadcopter unmanned aerial vehicle. In: 2017 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), IEEE (2017) 223–228

[17] Antsaklis, P.J., Passino, K.M., Wang, S.: Towards intelligent autonomous control systems: Architecture and fundamental issues. Journal of Intelligent and Robotic Systems **1**(4) (1989) 315–342

[18] Ferdaus, M.M., Anavatti, S.G., Pratama, M., Garratt, M.A.: Towards the use of fuzzy logic systems in rotary wing unmanned aerial vehicle: a review. Artificial Intelligence Review (2018) 1–34

[19] Kuo, K., Lin, J.: Fuzzy logic control for flexible link robot arm by singular perturbation approach. Applied Soft Computing **2**(1) (2002) 24–38

[20] De Silva, C.W.: Intelligent control: fuzzy logic applications. CRC press (2018)

[21] Stenmark, M., Malec, J.: Knowledge-based instruction of manipulation tasks for industrial robotics. Robotics and Computer-Integrated Manufacturing **33** (2015) 56–67

[22] Nikdel, N., Nikdel, P., Badamchizadeh, M.A., Hassanzadeh, I.: Using neural network model predictive control for controlling shape memory alloy-based manipulator. IEEE Transactions on Industrial Electronics **61**(3) (2013) 1394–1401

[23] Honegger, M., Brega, R., Schweiter, G.: Application of a nonlinear adaptive controller to a 6 dof parallel manipulator. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Volume 2., IEEE (2000) 1930–1935

[24] Sugeno, M., Tong, R.M.: Industrial applications of fuzzy control. Volume 44. North-Holland Amsterdam et al (1985)

[25] Wang, Y., Xie, L., De Souza, C.E.: Robust control of a class of uncertain nonlinear systems. Systems & Control Letters **19**(2) (1992) 139–149

[26] Bayes, T., Price, R., Canton, J.: An essay towards solving a problem in the doctrine of chances. C. Davis, Printer to the Royal Society of London (1763)

[27] Black, M.: Vagueness. an exercise in logical analysis. Philosophy of Science **4**(4) (1937) 427–455

[28] Zadeh, L.A.: Generalized theory of uncertainty: Principal concepts and ideas. In: Fundamental Uncertainty. Springer (2011) 104–150

[29] Lukasiewicz, J.: Philosophische bemerkungen zu mehrwertigen systemen des aussagenkalküls. Comtes Rendus des S eances de la Soci et e des Sciences et des Lettres de Varsovie, cl **3**(23) (1930) 51–77

[30] Gödel, K.: Zum intuitionistischen aussagenkalkül. Anz. Akad. Wiss. Wien **69** (1932) 65–66

[31] Jaśkowski, S.: Recherches sur le système de la logique intuitioniste. In: Internat. Congress Philos. Sci. Volume 6. (1936) 58–61

[32] Zadeh, L.A.: Fuzzy sets. Information and Control **8**(3) (1965) 338–353

[33] Mamdani, E.H.: Application of fuzzy algorithms for control of simple dynamic plant. In: Proceedings of the Institution of Electrical Engineers. Volume 121., IET (1974) 1585–1588

[34] Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies **7**(1) (1975) 1–13

[35] Siddique, N., Adeli, H.: Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing. John Wiley & Sons (2013)

[36] Lee, C.C.: Fuzzy logic in control systems: fuzzy logic controller. i. IEEE Transactions on Systems, Man, and Cybernetics **20**(2) (1990) 404–418

[37] Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics (1) (1985) 116–132

[38] Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences **8**(3) (1975) 199–249

[39] Aisbett, J., Rickard, J.T., Morgenthaler, D.G.: Type-2 fuzzy sets as functions on spaces. IEEE Transactions on Fuzzy Systems **18**(4) (2010) 841–844

[40] Karnik, N.N., Mendel, J.M.: Centroid of a type-2 fuzzy set. Information Sciences **132**(1) (2001) 195–220

[41] Juang, C.F., Lin, C.T.: An online self-constructing neural fuzzy inference network and its applications. IEEE Transactions on Fuzzy Systems **6**(1) (1998) 12–32

[42] Kasabov, N.K., Song, Q.: DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Transactions on Fuzzy Systems **10**(2) (2002) 144–154

[43] Leng, G., McGinnity, T.M., Prasad, G.: An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. Fuzzy Sets and Systems **150**(2) (2005) 211–243

[44] Angelov, P.P., Filev, D.P.: An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **34**(1) (2004) 484–498

[45] Yager, R.R., Filev, D.P.: Approximate clustering via the mountain method. IEEE Transactions on Systems, Man, and Cybernetics **24**(8) (1994) 1279–1284

[46] Angelov, P., Filev, D.: Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In: The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05., IEEE (2005) 1068–1073

[47] Rong, H.J., Sundararajan, N., Huang, G.B., Saratchandran, P.: Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets and Systems **157**(9) (2006) 1260–1275

[48] Huang, G.B., Saratchandran, P., Sundararajan, N.: An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **34**(6) (2004) 2284–2292

[49] Huang, G.B., Saratchandran, P., Sundararajan, N.: A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Transactions on Neural Networks **16**(1) (2005) 57–67

[50] Lughofer, E.D.: FLEXFIS: A robust incremental learning approach for evolving Takagi-Sugeno fuzzy models. IEEE Transactions on Fuzzy Systems **16**(6) (2008) 1393–1410

[51] Lughofer, E.: Flexible evolving fuzzy inference systems from data streams (FLEXFIS++). In: Learning in Non-Stationary Environments. Springer (2012) 205–245

[52] Angelov, P.: Evolving Takagi-Sugeno Fuzzy Systems from Streaming Data (eTS+). Evolving intelligent systems: methodology and applications **12** (2010) 21

[53] Angelov, P.: Fuzzily connected multimodel systems evolving autonomously from data streams. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **41**(4) (2011) 898–910

[54] Lemos, A., Caminhas, W., Gomide, F.: Adaptive fault detection and diagnosis using an evolving fuzzy classifier. Information Sciences **220** (2013) 64–85

[55] Angelov, P., Yager, R.: A new type of simplified fuzzy rule-based system. International Journal of General Systems **41**(2) (2012) 163–185

[56] Pratama, M., Anavatti, S.G., Angelov, P.P., Lughofer, E.: PANFIS: A novel incremental learning machine. IEEE Transactions on Neural Networks and Learning Systems **25**(1) (2014) 55–68

[57] Lughofer, E., Cernuda, C., Kindermann, S., Pratama, M.: Generalized smart evolving fuzzy systems. Evolving Systems **6**(4) (2015) 269–292

[58] Juang, C.F., Lin, C.T.: A recurrent self-organizing neural fuzzy inference network. IEEE Transactions on Neural Networks **10**(4) (1999) 828–845

[59] Juang, C.F.: A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. IEEE Transactions on Fuzzy Systems **10**(2) (2002) 155–170

[60] Juang, C.F., Lin, Y.Y., Tu, C.C.: A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing. Fuzzy Sets and Systems **161**(19) (2010) 2552–2568

[61] Lin, Y.Y., Chang, J.Y., Lin, C.T.: Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. IEEE Transactions on Neural Networks and Learning Systems **24**(2) (2013) 310–321

[62] Mendel, J.M., John, R.B.: Type-2 fuzzy sets made simple. IEEE Transactions on Fuzzy Systems **10**(2) (2002) 117–127

[63] Liang, Q., Mendel, J.M.: Interval type-2 fuzzy logic systems: theory and design. IEEE Transactions on Fuzzy systems **8**(5) (2000) 535–550

[64] Sola, H.B., Fernandez, J., Hagras, H., Herrera, F., Pagola, M., Barrenechea, E.: Interval type-2 fuzzy sets are generalization of interval-valued fuzzy sets: toward a wider view on their relationship. IEEE Transactions on Fuzzy Systems **23**(5) (2015) 1876–1882

[65] Juang, C.F., Tsao, Y.W.: A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. IEEE Transactions on Fuzzy Systems **16**(6) (2008) 1411–1424

[66] Juang, C.F., Chen, C.Y.: Data-driven interval type-2 neural fuzzy system with high learning accuracy and improved model interpretability. IEEE Transactions on Cybernetics **43**(6) (2013) 1781–1795

[67] Lin, Y.Y., Chang, J.Y., Pal, N.R., Lin, C.T.: A mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) with self-evolving structure and parameters. IEEE Transactions on Fuzzy Systems **21**(3) (2013) 492–509

[68] Castro, J.R., Castillo, O., Melin, P., Rodríguez-Díaz, A.: A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks. Information Sciences **179**(13) (2009) 2175–2193

[69] Tung, S.W., Quek, C., Guan, C.: eT2FIS: An evolving type-2 neural fuzzy inference system. Information Sciences **220** (2013) 124–148

[70] Bouchachia, A., Vanaret, C.: GT2FC: An online growing interval type-2 self-learning fuzzy classifier. IEEE Transactions on Fuzzy Systems **22**(4) (2014) 999–1018

[71] Abiyev, R.H., Kaynak, O.: Type 2 fuzzy neural structure for identification and control of time-varying plants. IEEE Transactions on Industrial Electronics **57**(12) (2010) 4147–4159

[72] Lin, Y.Y., Chang, J.Y., Lin, C.T.: A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (tscit2fnn) and its applications. IEEE Transactions on Industrial Electronics **61**(1) (2014) 447–459

[73] Lin, Y.Y., Liao, S.H., Chang, J.Y., Lin, C.T.: Simplified interval type-2 fuzzy neural networks. IEEE Transactions on Neural Networks and Learning systems **25**(5) (2014) 959–969

[74] Suresh, S., Dong, K., Kim, H.: A sequential learning algorithm for self-adaptive resource allocation network classifier. Neurocomputing **73**(16-18) (2010) 3012–3019

[75] Nelson, T.O.: Metamemory: A theoretical framework and new findings. Psychology of Learning and Motivation **26** (1990) 125–173

[76] Savitha, R., Suresh, S., Sundararajan, N.: Metacognitive learning in a fully complex-valued radial basis function neural network. Neural Computation **24**(5) (2012) 1297–1328

[77] Subramanian, K., Suresh, S., Sundararajan, N.: A metacognitive neuro-fuzzy inference system (McFIS) for sequential classification problems. IEEE Transactions on Fuzzy Systems **21**(6) (2013) 1080–1095

[78] Subramanian, K., Das, A.K., Sundaram, S., Ramasamy, S.: A meta-cognitive interval type-2 fuzzy inference system and its projection based learning algorithm. Evolving Systems **5**(4) (2014) 219–230

[79] Pratama, M., Anavatti, S.G., Lu, J.: Recurrent classifier based on an incremental metacognitive-based scaffolding algorithm. IEEE Transactions on Fuzzy Systems **23**(6) (2015) 2048–2066

[80] Pratama, M., Lu, J., Anavatti, S., Lughofer, E., Lim, C.P.: An incremental meta-cognitive-based scaffolding fuzzy neural network. Neurocomputing **171** (2016) 89–105

[81] Das, A.K., Subramanian, K., Sundaram, S.: An evolving interval type-2 neurofuzzy inference system and its metacognitive sequential learning algorithm. IEEE Transactions on Fuzzy Systems **23**(6) (2015) 2080–2093

[82] Pratama, M., Lughofer, E., Lu, J., Er, M.J., Anavatti, S.: Data driven modelling based on recurrent interval-valued metacognitive scaffolding fuzzy neural network. (2016)

[83] Pratama, M., Lughofer, E., Er, M.J., Anavatti, S., Lim, C.P.: Data driven modelling based on recurrent interval-valued metacognitive scaffolding fuzzy neural network. Neurocomputing **262** (2017) 4–27

[84] de Jesús Rubio, J.: Error convergence analysis of the sufin and csufin. Applied Soft Computing (2018)

[85] Pan, Y., Liu, Y., Xu, B., Yu, H.: Hybrid feedback feedforward: An efficient design of adaptive neural network control. Neural Networks **76** (2016) 122–134

[86] de Jesús Rubio, J.: Usnfis: uniform stable neuro fuzzy inference system. Neurocomputing **262** (2017) 57–66

[87] Meda-Campana, J.A.: Estimation of complex systems with parametric uncertainties using a jssf heuristically adjusted. IEEE Latin America Transactions **16**(2) (2018) 350–357

[88] Pratama, M., Angelov, P.P., Lughofer, E., Er, M.J.: Parsimonious random vector functional link network for data streams. Information Sciences **430** (2018) 519–537

[89] Kim, E., Park, M., Ji, S., Park, M.: A new approach to fuzzy modeling. IEEE Transactions on Fuzzy Systems **5**(3) (1997) 328–337

[90] Kung, C., Su, J.: Affine Takagi-Sugeno fuzzy modelling algorithm by fuzzy c-regression models clustering with a novel cluster validity criterion. IET Control Theory & Applications **1**(5) (2007) 1255–1265

[91] Li, C., Zhou, J., Xiang, X., Li, Q., An, X.: T-S fuzzy model identification based on a novel fuzzy c-regression model clustering algorithm. Engineering Applications of Artificial Intelligence **22**(4-5) (2009) 646–653

[92] Ferdaus, M.M., Pratama, M., Anavatti, S., Garratt, M.A.: PALM: An Incremental Construction of Hyperplanes for Data Stream Regression. IEEE Transactions on Fuzzy Systems (2019)

[93] Zarandi, M.F., Gamasaee, R., Turksen, I.: A type-2 fuzzy c-regression clustering algorithm for Takagi-Sugeno system identification and its application in the steel industry. Information Sciences **187** (2012) 179–203

[94] Zou, W., Li, C., Zhang, N.: A T-S Fuzzy Model Identification Approach based on a Modified Inter Type-2 FRCM Algorithm. IEEE Transactions on Fuzzy Systems (2017)

[95] Shim, D.H., Sastry, S.: An evasive maneuvering algorithm for uavs in see-and-avoid situations. In: American Control Conference, 2007. ACC'07, IEEE (2007) 3886–3891

[96] Herwitz, S., Johnson, L., Dunagan, S., Higgins, R., Sullivan, D., Zheng, J., Lobitz, B., Leung, J., Gallmeyer, B., Aoyagi, M., et al.: Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. Computers and Electronics in Agriculture **44**(1) (2004) 49–61

[97] Intelligence, S.D.: The global uav payload market 2012-2022. Strategic Defence Intelligence: White Papers (2013)

[98] Koh, L.P., Wich, S.A.: Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. Tropical Conservation Science **5**(2) (2012) 121–132

[99] Edwards, J., Baldwin, A., Bloemer, K., Callahan, K., Crawford, T., Fahringer, T.: Design and testing of soldier portable uav. In: 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition. (2012) 142

[100] Łukaszewicz, A.: Geometrical modelling of uav using parametric cax systems. In: the European Micro Air Vehicle Conference and Competition. (2009)

[101] Beard, R.W., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., Goodrich, M.: Autonomous vehicle technologies for small fixed-wing uavs. Journal of Aerospace Computing, Information, and Communication **2**(1) (2005) 92–108

[102] Folmer, E.: Exploring the use of an aerial robot to guide blind runners. ACM SIGACCESS Accessibility and Computing (112) (2015) 3–7

[103] Sarris, Z., Atlas, S.: Survey of UAV applications in civil markets. In: Proceedings of the 9th Mediterranean Conference on Control and Automation. (2001) 1–11

[104] Nonami, K.: Prospect and recent research & development for civil use autonomous unmanned aircraft as uav and mav. Journal of system Design and Dynamics **1**(2) (2007) 120–128

[105] Coffey, T., Montgomery, J.A.: The emergence of mini uavs for military applications. Defense Horizons (22) (2002) 1

[106] Samad, T., Bay, J.S., Godbole, D.: Network-centric systems for military operations in urban terrain: the role of uavs. Proceedings of the IEEE **95**(1) (2007) 92–107

[107] Aleksandrov, D., Penkov, I.: Energy consumption of mini uav helicopters with different number of rotors. In: 11th International Symposium" Topical Problems in the Field of Electrical and Power Engineering. (2012) 259–262

[108] Sotheara, S., Aso, K., Aomi, N., Shimamoto, S.: Effective data gathering and energy efficient communication protocol in wireless sensor networks employing

uav. In: Wireless Communications and Networking Conference (WCNC), 2014 IEEE, IEEE (2014) 2342–2347

[109] Weibel, R., Hansman, R.J.: Safety considerations for operation of different classes of uavs in the nas. In: AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit. (2004) 6421

[110] Clothier, R.A., Walker, R.A.: Determination and evaluation of uav safety objectives. (2006)

[111] Ruttner, F.: The life and flight activity of drones. Bee World **47**(3) (1966) 93–100

[112] Wang, I., Dobrokhodov, V., Kaminer, I., Jones, K.: On vision-based target tracking and range estimation for small uavs. In: AIAA Guidance, Navigation, and Control Conference and Exhibit. (2005) 6401

[113] Azeem, S.: Autonomous unmanned aerial vehicles. The George Washington University, Tech. Rep (2012)

[114] Floreano, D., Wood, R.J.: Science, technology and the future of small autonomous drones. Nature **521**(7553) (2015) 460

[115] Lee, K.U., Kim, H.S., Park, J.B., Choi, Y.H.: Hovering control of a quadrotor. In: 2012 12th International Conference on Control, Automation and Systems (ICCAS), IEEE (2012) 162–167

[116] Bouabdallah, S.: Design and control of quadrotors with application to autonomous flying. Technical report, Epfl (2007)

[117] Tong, S., Li, Y., Shi, P.: Observer-based adaptive fuzzy backstepping output feedback control of uncertain mimo pure-feedback nonlinear systems. IEEE Transactions on Fuzzy Systems **20**(4) (2012) 771–785

[118] Bouabdallah, S., Siegwart, R.: Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005, IEEE (2005) 2247–2252

[119] Madani, T., Benallegue, A.: Backstepping control for a quadrotor helicopter. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE (2006) 3255–3260

[120] Xu, R., Ozguner, U.: Sliding mode control of a quadrotor helicopter. In: 2006 45th IEEE Conference on Decision and Control, IEEE (2006) 4957–4962

[121] Lee, D., Kim, H.J., Sastry, S.: Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. International Journal of control, Automation and systems **7**(3) (2009) 419–428

[122] Waslander, S.L., Hoffmann, G.M., Jang, J.S., Tomlin, C.J.: Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005), IEEE (2005) 3712–3717

[123] Xiong, J.J., Zhang, G.: Discrete-time sliding mode control for a quadrotor UAV. Optik-International Journal for Light and Electron Optics **127**(8) (2016) 3718–3722

[124] Taniguchi, T., Eciolaza, L., Sugeno, M.: Quadrotor control using dynamic feedback linearization based on piecewise bilinear models. In: 2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA), IEEE (2014) 1–7

[125] Chen, M., Huzmezan, M.: A combined mbpc/2 DOF H infinity controller for a quad rotor UAV. In: AIAA Guidance, Navigation, and Control Conference and Exhibit. (2003) 5520

[126] Walker, D., Postlethwaite, I.: Advanced helicopter flight control using two-degree-of-freedom H (infinity) optimization. Journal of Guidance, Control, and Dynamics **19**(2) (1996) 461–468

[127] Fang, Z., Gao, W.: Adaptive integral backstepping control of a micro-quadrotor. In: Intelligent Control and Information Processing (ICICIP), 2011 2nd International Conference on. Volume 2., IEEE (2011) 910–915

[128] Roza, A., Maggiore, M.: Path following controller for a quadrotor helicopter. In: American Control Conference (ACC), 2012, IEEE (2012) 4655–4660

[129] Li, Y.: Approximation and robustness of fuzzy finite automata. International Journal of Approximate Reasoning **47**(2) (2008) 247–257

[130] Kim, B.S., Calise, A.J.: Nonlinear flight control using neural networks. Journal of Guidance, Control, and Dynamics **20**(1) (1997) 26–33

[131] Artale, V., Collotta, M., Pau, G., Ricciardello, A.: Hexacopter trajectory control using a neural network. In: AIP Conference Proceedings. Volume 1558., AIP (2013) 1216–1219

[132] Doitsidis, L., Valavanis, K.P., Tsourveloudis, N.C., Kontitsis, M.: A framework for fuzzy logic based uav navigation and control. In: 2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. Volume 4., IEEE (2004) 4041–4046

[133] Bacik, J., Perdukova, D., Fedor, P.: Design of fuzzy controller for hexacopter position control. In: Artificial Intelligence Perspectives and Applications. Springer (2015) 193–202

[134] Pan, Y., Sun, T., Liu, Y., Yu, H.: Composite learning from adaptive backstepping neural network control. Neural Networks **95** (2017) 134–142

[135] Pan, Y., Sun, T., Yu, H.: Peaking-free output-feedback adaptive neural control under a nonseparation principle. IEEE Transactions on Neural Networks and Learning Systems **26**(12) (2015) 3097–3108

[136] Guo, K., Pan, Y., Yu, H.: Composite Learning Robot Control with Friction Compensation: A Neural Network-Based Approach. IEEE Transactions on Industrial Electronics (2018)

[137] Chang, W.J., Qiao, H.Y., Ku, C.C.: Sliding mode fuzzy control for nonlinear stochastic systems subject to pole assignment and variance constraint. Information Sciences **432** (2018) 133–145

[138] Cheng, J., Chang, X.H., Park, J.H., Li, H., Wang, H.: Fuzzy-model-based h control for discrete-time switched systems with quantized feedback and unreliable links. Information Sciences **436** (2018) 181–196

[139] Suresh, S., Omkar, S., Mani, V., Sundararajan, N.: Nonlinear adaptive neural controller for unstable aircraft. Journal of Guidance, Control, and Dynamics **28**(6) (2005) 1103–1111

[140] Suresh, S., Omkar, S., Mani, V., Sundararajan, N.: Direct adaptive neural flight controller for f-8 fighter aircraft. Journal of Guidance, Control, and Dynamics **29**(2) (2006) 454–464

[141] Pashilkar, A., Sundararajan, N., Saratchandran, P.: A fault-tolerant neural aided controller for aircraft auto-landing. Aerospace Science and Technology **10**(1) (2006) 49–61

[142] **Md Meftahul Ferdaus**, Pratama, M., Anavatti, S.G., Garratt, M.A.: A generic self-evolving neuro-fuzzy controller based high-performance hexacopter altitude control system. In: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC). (Oct 2018) 2784–2791

[143] Leite, D., Palhares, R.M., Campos, V.C., Gomide, F.: Evolving granular fuzzy model-based control of nonlinear dynamic systems. IEEE Transactions on Fuzzy Systems **23**(4) (2015) 923–938

[144] Gao, Y., Er, M.J.: Online adaptive fuzzy neural identification and control of a class of mimo nonlinear systems. IEEE Transactions on Fuzzy Systems **11**(4) (2003) 462–477

[145] Angelov, P.: A fuzzy controller with evolving structure. Information Sciences **161**(1-2) (2004) 21–35

[146] de Barros, J.C., Dexter, A.L.: Evolving fuzzy model-based adaptive control. In: Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International, IEEE (2007) 1–5

[147] Han, H.G., Wu, X.L., Qiao, J.F.: Real-time model predictive control using a self-organizing neural network. IEEE Transactions on Neural Networks and Learning Systems **24**(9) (2013) 1425–1436

[148] Suresh, S., Narasimhan, S., Nagarajaiah, S., Sundararajan, N.: Fault-tolerant adaptive control of nonlinear base-isolated buildings using emran. Engineering Structures **32**(8) (2010) 2477–2487

[149] Pashilkar, A., Sundararajan, N., Saratchandran, P.: Adaptive back-stepping neural controller for reconfigurable flight control systems. IEEE Transactions on Control Systems Technology **14**(3) (2006) 553–561

[150] Chen, X., Li, D., Xu, Z., Bai, Y.: Robust control of quadrotor mav using self-organizing interval type-ii fuzzy neural networks (soit-iifnns) controller. International Journal of Intelligent Computing and Cybernetics **4**(3) (2011) 397–412

[151] Dong, C., Wang, N., Er, M.J.: Self-organizing adaptive robust fuzzy neural attitude tracking control of a quadrotor. In: 2016 35th Chinese Control Conference (CCC), IEEE (2016) 10724–10729

[152] Lin, C.M., Chen, T.Y.: Self-organizing cmac control for a class of mimo uncertain nonlinear systems. IEEE Transactions on Neural Networks **20**(9) (2009) 1377–1384

[153] Poston, T., Lee, C.N., Choie, Y., Kwon, Y.: Local minima and back propagation. In: International Joint Conference on Neural Networks, 1991., IJCNN-91-Seattle. Volume 2., IEEE (1991) 173–176

[154] Topalov, A.V., Kim, K.C., Kim, J.H., Lee, B.K.: Fast genetic on-line learning algorithm for neural network and its application to temperature control. In: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, IEEE (1996) 649–654

[155] Aggarwal, C.C.: Data streams: models and algorithms. Volume 31. Springer Science & Business Media (2007)

[156] Gama, J.: Knowledge discovery from data streams. CRC Press (2010)

[157] Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., De Carvalho, A.C., Gama, J.: Data stream clustering: A survey. ACM Computing Surveys (CSUR) **46**(1) (2013) 13

[158] Bose, R.J.C., Van Der Aalst, W.M., Zliobaite, I., Pechenizkiy, M.: Dealing with concept drifts in process mining. IEEE Transactions on Neural Networks and Learning Systems **25**(1) (2014) 154–171

[159] Lughofer, E., Angelov, P.: Handling drifts and shifts in on-line data streams with evolving fuzzy systems. Applied Soft Computing **11**(2) (2011) 2057–2068

[160] Xu, R.F., Lee, S.J.: Dimensionality reduction by feature clustering for regression problems. Information Sciences **299** (2015) 42–57

[161] Jiang, J.Y., Liou, R.J., Lee, S.J.: A fuzzy self-constructing feature clustering algorithm for text classification. IEEE Transactions on Knowledge and Data Engineering **23**(3) (2011) 335–349

[162] Wolfram: Point to Plane Distance, https://mathworld.wolfram.com/Point-Plane Distance.html (2018)

[163] Ferdaus, M.M., Pratama, M.: PALM code, https://www.researchgate.net /publication/332110788_PALM_code_Final (2018)

[164] Hathaway, R.J., Bezdek, J.C.: Switching regression models and fuzzy clustering. IEEE Transactions on Fuzzy Systems **1**(3) (1993) 195–204

[165] Krishnapuram, R., Frigui, H., Nasraoui, O.: Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation. i. IEEE Transactions on Fuzzy Systems **3**(1) (1995) 29–43

[166] Wu, S., Er, M.J.: Dynamic fuzzy neural networks-a novel approach to function approximation. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **30**(2) (2000) 358–364

[167] Karnik, N.N., Mendel, J.M., Liang, Q.: Type-2 fuzzy logic systems. IEEE Transactions on Fuzzy Systems **7**(6) (1999) 643–658

[168] Watkins, C.J., Dayan, P.: Q-learning. Machine Learning **8**(3-4) (1992) 279–292

[169] Mitra, P., Murthy, C., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(3) (2002) 301–312

[170] Ljung, L.: System Identification, Theory for the User. Upper Saddle River, NJ 07458, USA, Prentice-Hall. Technical report, ISBN 0-13-656695-2 (1999)

[171] Lughofer, E.: Evolving fuzzy systems-methodologies, advanced concepts and applications. Volume 53. Springer (2011)

[172] Kim, C.H., Kim, M.S.: Incremental hyperplane-based fuzzy clustering for system modeling. In: 33rd Annual Conference of the IEEE Industrial Electronics Society, 2007. IECON 2007., IEEE (2007) 614–619

[173] Lughofer, E., Bouchot, J.L., Shaker, A.: On-line elimination of local redundancies in evolving fuzzy systems. Evolving Systems **2**(3) (2011) 165–187

[174] Kim, M.S., Kim, C.H., Lee, J.J.: Evolving compact and interpretable Takagi-Sugeno fuzzy models with a new encoding scheme. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **36**(5) (2006) 1006–1023

[175] Xu, Y., Wong, K.W., Leung, C.S.: Generalized RLS approach to the training of neural networks. IEEE Transactions on Neural Networks **17**(1) (2006) 19–34

[176] Angelov, P., Lughofer, E., Zhou, X.: Evolving fuzzy classifiers using different model architectures. Fuzzy Sets and Systems **159**(23) (2008) 3160–3182

[177] MacKay, D.J.: Bayesian interpolation. Neural Computation **4**(3) (1992) 415–447

[178] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning. Volume 1. MIT press Cambridge (2016)

[179] Ferdaus, M.M., Pratama, M.: rPALM code, https://www.researchgate.net /publication/332110788_PALM_code_Final (2018)

[180] Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. Science **197**(4300) (1977) 287–289

[181] Oentaryo, R.J., Er, M.J., Linn, S., Li, X.: Online probabilistic learning for fuzzy inference system. Expert Systems with Applications **41**(11) (2014) 5082–5096

[182] Tan, J., Quek, C.: A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning. IEEE Transactions on Neural Networks **21**(6) (2010) 985–1003

[183] Yahoo, F.: S and P 500, https://finance.yahoo.com/quote/%5EGSPC? p=ĜSPC (2018)

[184] Wu, S., Er, M.J., Gao, Y.: A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. IEEE Transactions on Fuzzy Systems **9**(4) (2001) 578–594

[185] Wang, N., Er, M.J., Meng, X.: A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks. Neurocomputing **72**(16-18) (2009) 3818–3829

[186] Camera, M.C.: VICON Camera, https://www.vicon.com/ (2018)

[187] Zhang, D., Zhou, Z., Jia, X.: Networked fuzzy output feedback control for discrete-time takagi–sugeno fuzzy systems with sensor saturation and measurement noise. Information Sciences (2018)

[188] Lin, C.J.: A ga-based neural fuzzy system for temperature control. Fuzzy Sets and Systems **143**(2) (2004) 311–333

[189] Ferdaus, M.M., Pratama, M., Anavatti, S., Garratt, M.A.: PALM: An Incremental Construction of Hyperplanes for Data Stream Regression. IEEE Transactions on Fuzzy Systems (2019) 1–1

[190] Pratama, M., Ashfahani, A., Ong, Y.S., Ramasamy, S., Lughofer, E.: Autonomous deep learning: Incremental learning of denoising autoencoder for evolving data streams. arXiv preprint arXiv:1809.09081 (2018)

[191] Ferdaus, M.M., Pratama, M.: Autonomous Intelligent Controller PAC, https://www.researchgate.net/publication/326416125-PAC (2018)

[192] Wang, Z.J., Russell, D.: Effect of forewing and hindwing interactions on aerodynamic forces and power in hovering dragonfly flight. Physical Review Letters **99**(14) (2007) 148101

[193] Kok, J., Chahl, J.: A low-cost simulation platform for flapping wing mavs. In: SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring, International Society for Optics and Photonics (2015) 94290L–94290L

[194] Seddon, J.M., Newman, S.: Basic helicopter aerodynamics. Volume 40. John Wiley & Sons (2011)

[195] Glauert, H.: A general theory of the autogyro. Volume 1111. HM Stationery Office (1926)

[196] Nelson, R.C.: Flight stability and automatic control. Volume 2. WCB/McGraw Hill New York (1998)

[197] Stevens, B.L., Lewis, F.L., Johnson, E.N.: Aircraft control and simulation: dynamics, controls design, and autonomous systems. John Wiley & Sons (2015)

[198] Thomas, A.L., Taylor, G.K., Srygley, R.B., Nudds, R.L., Bomphrey, R.J.: Dragonfly flight: free-flight and tethered flow visualizations reveal a diverse array of unsteady lift-generating mechanisms, controlled primarily via angle of attack. Journal of Experimental Biology **207**(24) (2004) 4299–4323

[199] Kok, J., Chahl, J.: Systems-level analysis of resonant mechanisms for flapping-wing flyers. Journal of Aircraft **51**(6) (2014) 1833–1841

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

[201] Wang, Z.J.: The role of drag in insect hovering. Journal of Experimental Biology **207**(23) (2004) 4147–4155

[202] Wang, Z.J.: Dissecting insect flight. Annu. Rev. Fluid Mech. **37** (2005) 183–210

[203] Alexander, D.E.: Wind tunnel studies of turns by flying dragonflies. Journal of Experimental Biology **122**(1) (1986) 81–98

[204] Rüppell, G.: Kinematic analysis of symmetrical flight manoeuvres of odonata. Journal of Experimental Biology **144**(1) (1989) 13–42

[205] Dickinson, M.H., Lehmann, F.O., Sane, S.P.: Wing rotation and the aerodynamic basis of insect flight. Science **284**(5422) (1999) 1954–1960

[206] Wang, Z.J., Birch, J.M., Dickinson, M.H.: Unsteady forces and flows in low reynolds number hovering flight: two-dimensional computations vs robotic wing experiments. Journal of Experimental Biology **207**(3) (2004) 449–460

[207] Beard, R.: Quadrotor Dynamics and Control Rev0.1. https://scholarsarchive.byu.edu/facpub/1325 **1325** (2008)

[208] Imai, K.: Expectation and Functions of Random Variables (2006)

[209] Ferdaus, M.M., Pratama, M., Anavatti, S.G., Garratt, M.A., Lughofer, E.: PAC: A Novel Self-Adaptive Neuro-Fuzzy Controller for Micro Aerial Vehicles. arXiv preprint arXiv:1811.03764 (2018)

[210] Gama, J., Fernandes, R., Rocha, R.: Decision trees for mining data streams. Intelligent Data Analysis **10**(1) (2006) 23–45

[211] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys (CSUR) **46**(4) (2014) 44

[212] Kayacan, E., Cigdem, O., Kaynak, O.: Sliding mode control approach for online learning as applied to type-2 fuzzy neural networks and its experimental evaluation. IEEE Transactions on Industrial Electronics **59**(9) (2012) 3510–3520

[213] Utkin, V.I.: Sliding modes in control and optimization. Springer Science & Business Media (2013)

[214] Kayacan, E., Maslim, R.: Type-2 fuzzy logic trajectory tracking control of quadrotor vtol aircraft with elliptic membership functions. IEEE/ASME Transactions on Mechatronics **22**(1) (2017) 339–348

[215] Wang, L.X.: Stable adaptive fuzzy control of nonlinear systems. IEEE Transactions on Fuzzy Systems **1**(2) (1993) 146–155

[216] Rysdyk, R., Calise, A.: Fault tolerant flight control via adaptive neural network augmentation. In: Guidance, Navigation, and Control Conference and Exhibit. (1998) 4483

[217] Krstic, M., Kanellakopoulos, I., Kokotovic, P.V., et al.: Nonlinear and adaptive control design. Volume 222. Wiley New York (1995)

[218] Dronekit 3DR: Setting up a Simulated Vehicle (SITL) (2015)

[219] Kandath, H., Bera, T., Bardhan, R., Sundaram, S.: Autonomous Navigation and Sensorless Obstacle Avoidance for UGV with Environment Information from UAV. 2018 Second IEEE International Conference on Robotic Computing (IRC) (Jan 2018) 266–269

[220] Ferdaus, M.M., Hady, M.A., Kandath, H., Pratama, M., Anavatti, S.G.: Matlab Codes for RedPAC, https://www.researchgate.net/publication/332107396_Matlab_Codes_for _RedPAC_A_Simple_Evolving_Neuro_Fuzzy_based_Intelligent_Control _Framework_for_Autonomous_Aerial_Vehicles_FUZZ-IEEE_2019 (2018)

[221] Liu, Y.J., Tong, S.: Adaptive fuzzy identification and control for a class of nonlinear pure-feedback MIMO systems with unknown dead zones. IEEE Transactions on Fuzzy Systems **23**(5) (2015) 1387–1398

[222] Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. In: Readings in Fuzzy Sets for Intelligent Systems. Elsevier (1993) 387–403

[223] Yang, C., Li, Y.: ϵ-bisimulation relations for fuzzy automata. IEEE Transactions on Fuzzy Systems **26**(4) (2018) 2017–2029

[224] Sun, T., Pei, H., Pan, Y., Zhou, H., Zhang, C.: Neural network-based sliding mode adaptive control for robot manipulators. Neurocomputing **74**(14) (2011) 2377–2384

[225] Ferdaus, M.M., Pratama, M.: G controller code, https://www.researchgate.net/publication/325253767_gcontroller_code (2018)

[226] Oentaryo, R.J., Er, M.J., San, L., Zhai, L., Li, X.: Bayesian ART-based fuzzy inference system: A new approach to prognosis of machining processes. In: 2011 IEEE Conference on Prognostics and Health Management (PHM), IEEE (2011) 1–10

[227] Yap, K.S., Lim, C.P., Abidin, I.Z.: A Hybrid ART-GRNN Online Learning Neural Network With a ε Insensitive Loss Function. IEEE Transactions on Neural Networks **19**(9) (2008) 1641–1646

[228] Yap, K.S., Lim, C.P., Au, M.T.: Improved GART neural network model for pattern classification and rule extraction with application to power systems. IEEE Transactions on Neural Networks **22**(12) (2011) 2310–2323

[229] Lughofer, E.: A dynamic split-and-merge approach for evolving cluster models. Evolving Systems **3**(3) (2012) 135–151

[230] Slotine, J.J.E., Li, W., et al.: Applied nonlinear control. Volume 199. Prentice hall Englewood Cliffs, NJ (1991)

[231] Ferdaus, M.M., Anavatti, S.G., Garratt, M.A., Pratama, M.: Fuzzy Clustering based Modelling and Adaptive Controlling of a Flapping Wing Micro Air Vehicle. In: 2017 IEEE Symposium Series on Computational Intelligence (IEEE SSCI), IEEE (2017) 1914–1919

[232] Babaei, A., Mortazavi, M., Moradi, M.: Classical and fuzzy-genetic autopilot design for unmanned aerial vehicles. Applied Soft Computing **11**(1) (2011) 365–372

[233] Domingos, D., Camargo, G., Gomide, F.: Autonomous fuzzy control and navigation of quadcopters. IFAC-PapersOnLine **49**(5) (2016) 73–78

[234] Santoso, F., Garratt, M.A., Anavatti, S.G.: Adaptive neuro-fuzzy inference system identification for the dynamics of the AR. Drone quadcopter. In: 2016 International Conference on Sustainable Energy Engineering and Application (ICSEEA), IEEE (2016) 55–60

[235] Santoso, F., Garratt, M.A., Anavatti, S.G.: Fuzzy logic-based self-tuning autopilots for trajectory tracking of a low-cost quadcopter: A comparative study. In: 2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), IEEE (2015) 64–69

[236] Hatamleh, K.S., Al-Shabi, M., Al-Ghasem, A., Asad, A.A.: Unmanned aerial vehicles parameter estimation using artificial neural networks and iterative bi-section shooting method. Applied Soft Computing **36** (2015) 457–467

[237] Sundhararajan, M., Gao, X.Z., Vahdat Nejad, H.: Artificial intelligent techniques and its applications. Journal of Intelligent & Fuzzy Systems **34**(2) (2018) 755–760

[238] Ban, X., Gao, X., Huang, X., Yin, H.: Stability analysis of the simplest takagi-sugeno fuzzy control system using popov criterion. In: Soft Computing in Industrial Applications. Springer (2007) 63–71

[239] Sundararajan, N., Saratchandran, P., Li, Y.: Nonlinear System Identification Using Lyapunov-Based Fully Tuned RBFN. In: Fully Tuned Radial Basis Function Neural Networks for Flight Control. Springer (2002) 29–45

[240] Sundararajan, N., Saratchandran, P., Li, Y.: Real-Time Identification of Nonlinear Systems Using MRAN/EMRAN Algorithm. In: Fully Tuned Radial Basis Function Neural Networks for Flight Control. Springer (2002) 47–68

[241] Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10) (2011) 1517–1531

[242] Josyula, D.P., Vadali, H., Donahue, B.J., Hughes, F.C.: Modeling metacognition for learning in artificial systems. In: World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009, IEEE (2009) 1419–1424

[243] Flavell, J.H., Patto, M.H.S., Piaget, J.: A psicologia do desenvolvimento de Jean Piaget. (1996)

[244] Isaacson, R.M., Fujita, F.: Metacognitive knowledge monitoring and self-regulated learning: Academic success and reflections on learning. Journal of Scholarship of Teaching and Learning **6**(1) (2006) 39–55

[245] Zain, C., Pratama, M., Lughofer, E., Anavatti, S.G.: Evolving Type-2 Web News Mining. Applied Soft Computing (2017)

[246] Ferdaus, M.M., Anavatti, S.G., Garratt, M.A., Pratama, M.: Fuzzy clustering based nonlinear system identification and controller development of pixhawk based quadcopter. In: 2017 Ninth International Conference on Advanced Computational Intelligence (ICACI), IEEE (2017) 223–230

[247] Cpałka, K.: Design of Interpretable Fuzzy Systems. Volume 684. Springer (2017)

[248] Paiva, R.P., Dourado, A.: Interpretability and learning in neuro-fuzzy systems. Fuzzy Sets and Systems **147**(1) (2004) 17–38

[249] Eftekhari, M., Katebi, S., Karimi, M., Jahanmiri, A.: Eliciting transparent fuzzy model using differential evolution. Applied Soft Computing **8**(1) (2008) 466–476

[250] Gacto, M.J., Alcalá, R., Herrera, F.: Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. Information Sciences **181**(20) (2011) 4340–4360

[251] Rudziński, F.: A multi-objective genetic optimization of interpretability-oriented fuzzy rule-based classifiers. Applied Soft Computing **38** (2016) 118–133

[252] Pratama, M., Er, M.J., Anavatti, S.G., Lughofer, E., Wang, N., Arifin, I.: A novel meta-cognitive-based scaffolding classifier to sequential non-stationary classification problems. In: 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE (2014) 369–376

[253] Hajmohammadi, M.S., Ibrahim, R., Selamat, A., Fujita, H.: Combination of active learning and self-training for cross-lingual sentiment classification with density analysis of unlabelled samples. Information Sciences **317** (2015) 67–77

[254] Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Transactions on Neural Networks and Learning Systems **25**(1) (2014) 27–39

[255] Bortman, M., Aladjem, M.: A growing and pruning method for radial basis function networks. IEEE Transactions on Neural Networks **20**(6) (2009) 1039–1045

[256] Vuković, N., Miljković, Z.: A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation. Neural Networks **46** (2013) 210–226

[257] Tabata, K., Sato, M., Kudo, M.: Data compression by volume prototypes for streaming data. Pattern Recognition **43**(9) (2010) 3162–3176

[258] Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Transactions on Knowledge and Data Engineering **25**(10) (2013) 2283–2301

[259] Han, H., Qiao, J.: Nonlinear model-predictive control for industrial processes: An application to wastewater treatment process. IEEE Transactions on Industrial Electronics **61**(4) (2014) 1970–1982

[260] Bhattachayya, A.: On a measure of divergence between two statistical population defined by their population distributions. Bulletin Calcutta Mathematical Society **35**(99-109) (1943) 28

[261] Lughofer, E., Sayed-Mouchaweh, M.: Autonomous data stream clustering implementing split-and-merge concepts–towards a plug-and-play approach. Information Sciences **304** (2015) 54–79

[262] Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research **5**(Oct) (2004) 1205–1224

[263] Lughofer, E.: On-line assurance of interpretability criteria in evolving fuzzy systems–achievements, new concepts and open issues. Information Sciences **251** (2013) 22–46

[264] Jang, J.S.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics **23**(3) (1993) 665–685

[265] Pratama, M., Lu, J., Zhang, G., et al.: Evolving type-2 fuzzy classifier. IEEE Trans. Fuzzy Systems **24**(3) (2016) 574–589

[266] Pratama, M., Zhang, G., Er, M.J., Anavatti, S.: An incremental type-2 meta-cognitive extreme learning machine. IEEE Transactions on Cybernetics **47**(2) (2017) 339–353

[267] Castro, J.L.: Fuzzy logic controllers are universal approximators. IEEE Transactions on Systems, Man, and Cybernetics **25**(4) (1995) 629–635

[268] Ying, H.: 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 (2009) 1–5

[269] Kosko, B.: Fuzzy systems as universal approximators. IEEE Transactions on Computers **43**(11) (1994) 1329–1333

[270] Ying, H., Ding, Y., Li, S., Shao, S.: 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 (1998) 824–828

[271] You, F., Ying, H.: Interval type-2 boolean fuzzy systems are universal approximators. In: Fuzzy Information Processing Society (NAFIPS), 2010 Annual Meeting of the North American, IEEE (2010) 1–4

Appendix A

Pseudocodes of PALM algorithms

A.1 Pseudocodes of PALM algorithms

All the steps of our proposed algorithms are summarized in separate Pseudocodes to improve their readability. Both Pseudocodes are expressed as follows:

Algorithm A.1 Type-1 PALM algorithm

Define: Training data (X_t, T_t)=$(x_1, ..., x_n, t_1, ..., t_n)$
Predefined thresholds b_1, b_2 and c_1, c_2
Step 1: Basic architecture of type-1 PALM
procedure FUZZIFICATION
    for $i = 1$ to R **do**
        Calculate point to plane distance using (5)
        Find the maximum distance among the rules
        Calculate hyperplane based membership function using (4)
    end for
end procedure
procedure RULE BASE
    for $i = 1$ to R **do**
        Calculate consequent part for each rule using (7)
    end for
    Express the IF-THEN fuzzy rule utilizing (6)
end procedure

procedure DEFUZZIFICATION
    Calculate defuzzified crisp output using (8)
end procedure
Step 2
procedure MECHANISM OF GROWING RULES
    for $j = 1$ to n **do**
        Compute $\xi(X_j, T_o)$ using (18)
    end for
    for $i = 1$ to R **do**
        Calculate input coherence using (16)
        for $o = 1$ to k **do**
            Compute $\xi(\mathcal{H}_i, T_o)$ using (18)
        end for
        Calculate output coherence using (17)
    end for
    if (20) **then**
        Create a new rule
    else
        Accommodated data points of a rule are updated as $N_{j^*} = N_{j^*} + 1$
        Take the next sample and Go to Step 1
    end if
end procedure
procedure MECHANISM OF MERGING RULES
    for $i = 1$ to R **do**
        for $o = 1$ to k **do**
            Calculate angle between hyperplanes using (22)
            Calculate minimum distance between hyperplanes using (23)
        end for
        if (24) **then**
            Rules are merged
        end if
    end for
end procedure
procedure ADAPTATION OF OUTPUT WEIGHTS

for $i = 1$ to R **do**
    Update output weights using FWGRLS described in Section IV.C
end for
end procedure

Algorithm A.2 Type-2 PALM algorithm

Define: Training data (X_t, T_t)=$(x_1, ..., x_n, t_1, ..., t_n)$
Predefined thresholds b_1, b_2 and c_1, c_2
Step 1: Basic architecture of type-2 PALM
procedure FUZZIFICATION
    for $i = 1$ to R **do**
        Calculate point to plane distance using (10)
        Find the maximum distance among the rules
        Calculate hyperplane based membership function using (9)
    end for
end procedure
procedure RULE BASE
    for $i = 1$ to R **do**
        Calculate consequent part for each rule using (12)
    end for
    Express the IF-THEN fuzzy rule utilizing (11)
end procedure
procedure TYPE REDUCTION AND DEFUZZIFICATION
    Calculate q design factor based type reduction and defuzzified crisp output using (13), (14) and (15)
end procedure
Step 2
procedure MECHANISM OF GROWING RULES
    for $j = 1$ to n **do**
        Compute $\xi(\widetilde{\mathcal{H}}_i,\ X_j)$ using (39), (40) and (41)
    end for
    for $i = 1$ to R **do**
        Calculate input coherence using (35), (36) and (37)
        for $o = 1$ to k **do**
            Compute $\xi(\widetilde{\mathcal{H}}_i, T_o)$ using (39), (40) and (41)

end for
Calculate output coherence using (38)
end for
if (20) **then**
Create a new rule
else
Accommodated data points of a rule are updated as $N_{j^*} = N_{j^*} + 1$
Take the next sample and Go to Step 1
end if
end procedure
procedure MECHANISM OF MERGING RULES
for $i = 1$ to R **do**
for $o = 1$ to k **do**
Calculate angle between hyperplanes using (42)
Calculate minimum distance between hyperplanes using (43)
end for
if (24) **then**
Rules are merged
end if
end for
end procedure
procedure ADAPTATION OF OUTPUT INTERVAL-VALUED WEIGHTS
for $i = 1$ to R **do**
Update output weights using interval type-2 FWGRLS described in Section V.C
end for
end procedure

Appendix B

PALM as an universal approximator

B.1 Proof of PALMs as universal approximator

Our developed type-1 PALM and type-2 PALM's basic structures are followed by type-1 and type-2 TS-fuzzy architecture. It has already been proved by many researchers that both type-1 and type-2 TS-fuzzy system work as an universal approximator [267–271]. Since type-1 PALM and type-2 PALM work as universal approximator, it can clearly guarantee the predictive accuracy for the system with chaotic behaviors.

B.1.1 Type-1 PALM as universal approximator:

Type-1 PALM can approximate an unknown function by covering its graph with hyperplane-based clusters. The approximation improves as the clusters grow in number. Fig. B.1 displays how the PALM's hyerplane based clusters in the input-output product space $X \times Y$ represent the real function $f : X \to Y$. From the left part of Fig. 1 it is clearly observed that lower number of clusters have lower approximation accuracy than PALM with higher number of clusters as exposed in the right hand side of Fig. 1. On the other hand, as the clusters grows, the structural complexity and memory demand raises in PALM.

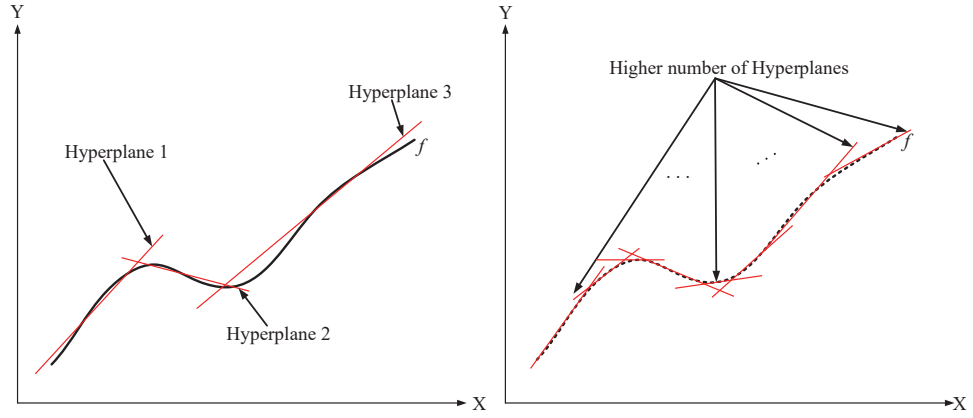


Figure B.1: Type-1 PALM's approximation of the graph of an unknown function $f : X \to Y$ with only three hyperplane based clusters (left), and eight hyperplane based clusters (right)

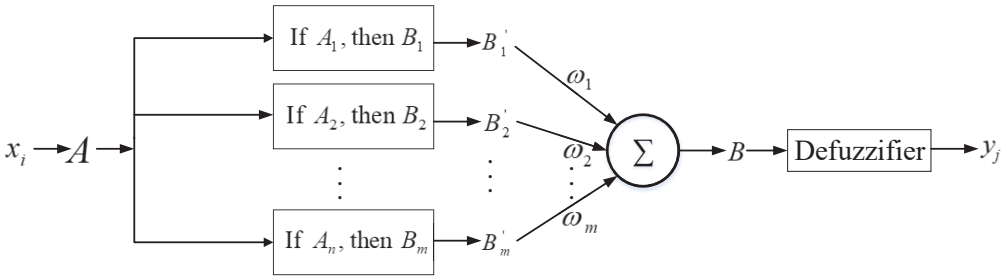


Figure B.2: Type-1 PALM's simplified architecture

From Fig. 2, the weighted sum can be expressed as:

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

where a_i^j is the degree to which input x_i belongs to fuzzy set A_j in the rule or cluster $A_j \times B_j$. Type-1 PALM can approximate a function $f : X \to Y$ by generating rules or clusters. In the theorem of PALM as universal approximator, we need to consider that $f : X \to Y$ is continuous and that X is compact (closed and bounded) in $\Re^n$. The theorem shows that in principle a TS-fuzzy based PALM with finite fuzzy rules can approximate any continuous function to any degree of accuracy.

Theorem:A type-1 PALM F uniformly approximates $f : X \to Y$ if X is compact and f is continuous.

Proof: Let us assume any small constant $\varepsilon > 0$. We have to prove that $|F(x) - f(x)| < \varepsilon$ for all $x \in X$, where X is a compact subset of R^n, $F(x)$ is the output the PALM as expressed in Eq. B.1. Continuity of f on compact X gives uniform continuity. So there is a fixed distance δ such that, for all x and z in X , $|f(x)-f(z)| < \varepsilon/4$ if $|x-z| < \delta$. We can construct a set of open cubes $M_1, ..., M_m$ that cover X and that have ordered overlap in their n coordinates so that each cube comer lies at the midpoint c_J of its neighbors M_j. If we pick symmetric output fuzzy sets B, centered on $f(c_j)$. and pick $u \in X$, then by construction u lies in at most 2^n overlapping open cubes M_j. Pick any ω in the same set of cubes. If $u \in M_j$ and $\omega \in M_k$, then for all $u \in M_j \cap M_k : |u - v| < \delta$ and $|v - \omega| < \delta$. Uniform continuity implies that $|f(u) - f(\omega)| \leq |f(u) - f(v)| + |f(v) - f(\omega)|$. So for cube centers c_j and c_k, $|f(c_j) - f(c_k)| < \varepsilon/2$. If we pick $x \in X$, then x too lies in at most 2^n open cubes with centers c_j and $|f(c_j) - f(x)| < \varepsilon/2$. Along the kth coordinate of the range space R^p the kth component of the PALM output$F(x)$ lies as in (6) on or between the kth components of the centroids of the B_j sets. So, since $|f(c_j) - f(c_k)| < \varepsilon/2$ for all $f(c_j)$, $|F(x) - f(c_j)| < \varepsilon/2$. Then $|F(x) - f(x)| \leq |F(x) - f(c_j)| + |f(c_j) - f(x)| < \varepsilon/2 + \varepsilon/2 = \varepsilon$.

The proof traps the centroidal output C(B) or y_j between the centroids $C(B_1)$and $C(B_m)$ if $C(B_1) \leq C(B_2) \leq ... \leq C(B_m)$:

$$\begin{aligned} C(B) &= \frac{\sum_{j=1}^{m} A(B_j)C(B_j)}{\sum_{j=1}^{m} A(B_j)} \\ &= \sum_{j=1}^{m} c_j C(B_j) \end{aligned} \tag{B.2}$$

for volume or area $A(B_j) = \int_X m_B(x)dx$ and for convex area coefficients $c_1, ..., c_m$. The proof works for any combined output set $B = \phi(B_1, ..., B_m)$ such that $C(B_1) \leq \phi \leq C(B_m)$. The proof also works for noncentroidal defuzzifiers $D(B)$ that obey $C(B_1) \leq D(B) \leq C(B_m)$. For further clarification, the details proof of type-1 fuzzy system's function approximation power is described in [269].

B.1.2 Type-2 PALM as universal approximator:

We will constructively prove that type-2 PALM $F_n(x)$ can uniformly approximate any multivariate polynomial $P_d(x)$ to any degree of accuracy.

Lemma $F_n(x)$ can uniformly approximate any multivariate polynomial $P_d(x)$ defined in $C^r[-1, 1]$ to any degree of accuracy, where $P_d(x) = \sum_{d_i \geq 0} \left(\beta_{d_1,...,d_r} \prod_{i=1}^{r} x_i^{d_i}\right)$ and $\sum_{i=1}^{r} d_i < d$. That is, $\forall \varepsilon > 0$, there exists a sufficiently large positive integer n^* such that $\forall n > n^*$.

$$||F_n - P_d||_{c^r[-1,1]} = \max_{x \in c^r[-1,1]} |F_n(x) - P_d(x)|| < \varepsilon \tag{B.3}$$

Proof: Let us consider a dth order polynomial as follows:

$$f(p_h) = \sum_{d_i \geq 0} \left(L_{d_1,...,d_r} . n^d \prod_{i=1}^{r} \left(\frac{p_{h,i}}{n}\right)^{d_i} \right) \tag{B.4}$$

with the same degree as $P_d(x)$ with respect to p_i, where $L_{d_1,...,d_r}$ are integers obtained from $\beta_{d_1,...,d_r}$ of $P_d(x)$ to ensure $f(p_h)$ is integer, and $L_{d_1,...,d_r} = 10^s \times \beta_{d_1,...,d_r}$; here, s is the smallest positive integer that will make all the $10^s \times \beta_{d_1,...,d_r}$ integers. It is obvious that $M(f, n) = n^d . \sum_{d_i \geq 0} |L_{d_1,...,d_r}|$ We choose $H = 10^s . \sum_{d_i \geq 0} |L_{d_1,...,d_r}|$, then

$$\frac{H.f(p_h)}{M(f, n)} = \sum_{d_i \geq 0} \left(\beta_{d_1,...,d_r} \prod \left(\frac{p_{h,i}}{n}\right)^{d_i} \right) = P_d\left(\frac{p_h}{n}\right) \tag{B.5}$$

where $\frac{p_h}{n} = \left[\frac{p_{h_1}}{n}, ..., \frac{p_{h_r}}{n}\right]$. Based on the derivation in [271] the defuzzified crisp value can be expressed as follows:

$$F_n(x) = \frac{1}{2^\tau} \sum_{h=1}^{2^\tau} \frac{\sum_{q=1}^{\tau} P_d\left(\frac{p_h + c_h}{n}\right) \lambda_{m_q}^h}{\sum_{q=1}^{\tau} \lambda_{m_q}^h} \tag{B.6}$$

For simplicity, the case of $r = 2$ is considered in this proof. We assume that $\frac{p_i}{n} \leq x_i \leq \frac{p_i}{n}$, thus

$$\left| \frac{p_{h,i} + c_{h,i}}{n} - x_i \right| \leq \frac{c_{h,i}}{n} \leq \frac{c_i^{max}}{n}, \text{ where } i = 1, 2. \tag{B.7}$$

Let, $\eta = [\eta_{h,1}, \eta_{h,2}]$, where $\eta_{h,i} = \frac{p_{h,i} + c_{h,i}}{n}$, $i = 1, 2.$ and $x = [x_1, x_2]$, $|x_i| \leq 1$ and $|\eta_{h,i}| \leq 1$. For $1 \leq d_1$and $d_2 \leq d$, let us consider that $\xi(\eta, x) = \left|\eta_{h,1}^{d_1}.\eta_{h,2}^{d_2} - x_1^{d_1}.x_2^{d_2}\right|$and note that

$$\begin{aligned}
\left|\eta_{h,1}^{d_1}.\eta_{h,2}^{d_2} - x_1^{d_1}.x_2^{d_2}\right| &\leq \left|\eta_{h,1}^{d_1} - x_1^{d_1}\right| . \left|\eta_{h,2}^{d_2}\right| + \left|x_1^{d_1}\right| . \left|\eta_{h,2}^{d_2} - x_2^{d_2}\right| \\
&\leq \left|\eta_{h,1}^{d_1} - x_1^{d_1}\right| + \left|\eta_{h,2}^{d_2} - x_2^{d_2}\right| \\
&\leq \left|\eta_{h,1}^{d_1} - x_1\right| . \left|\sum_{v=1}^{d_1} \eta_{h,1}^{d_1 - v}.x_1^{v-1}\right| + \left|\eta_{h,2}^{d_2} - x_2\right| \left|\sum_{v=1}^{d_2} \eta_{h,2}^{d_2 - v}.x_2^{v-1}\right| \\
&\leq \frac{c_1^{max}}{n} \sum_{v=1}^{d_1} 1 + \frac{c_2^{max}}{n} \sum_{v=1}^{d_2} 1 \\
&= \frac{c_1^{max}.d_1 + c_2^{max} d_2}{n}
\end{aligned} \tag{B.8}$$

Thus,

$$\begin{aligned}
&= \left| \frac{1}{2^\tau} \sum_{h=1}^{2^\tau} \frac{\sum_{q=1}^{\tau} P_d\left(\eta_{h,1}^{d_1}, \eta_{h,2}^{d_2}\right).\lambda_{m_q}^h}{\sum_{q=1}^{\tau} \lambda_{m_q}^h} - P_d(x_1, x_2) \right| \\
&\leq \frac{1}{2^\tau} \sum_{h=1}^{2^\tau} \frac{\sum_{q=1}^{\tau} \lambda_{m_q}^h \left| P_d\left(\eta_{h,1}^{d_1}, \eta_{h,2}^{d_2}\right) - P_d(x_1, x_2) \right|}{\sum_{q=1}^{\tau} \lambda_{m_q}^h} - \\
&\leq \frac{1}{2^\tau} \sum_{h=1}^{2^\tau} \frac{\sum_{q=1}^{\tau} \lambda_{m_q}^h \left(\sum_{d_1, d_2 = 0} \left| \beta_{d_1, d_2} \right| \right) \xi(\eta, x)}{\sum_{q=1}^{\tau} \lambda_{m_q}^h}
\end{aligned} \tag{B.9}$$

Combining (8)-(11), the following is achieved:

$$\|F_n - P_d\|_{C^2[-1,1]} = \max_{x_1, x_2 \in [-1,1]} |F_n(x) - P_d(x)| \leq \frac{c_1^{max} d_1 + c_2^{max} d_2}{n} \tag{B.10}$$

Thus, $\forall \varepsilon > 0$, if we choose $n^* > \sum_{i=1}^{2} \frac{c_i^{max} d_i}{\varepsilon} . \sum_{d_1, d_2 = 0}^{d} \left| \beta_{d_1, d_2} \right|$, then $\|F_n - P_d\|_{C^2[-1,1]} < \varepsilon$ for all $n > n^*$is indicating that $F_n(x)$ can approximate $P_d(x)$ uniformly.

Theorem: $F_n(x)$ can uniformly approximate any multivariate continuous function $G(x)$ in $C^r[-1,1]$ to any degree of accuracy. In other words, $\forall\varepsilon > 0$, there exists a sufficiently large positive integer n^* such that $\forall n > n^*$, (5) is true.

Proof: According to the Weierstrass approximation theorem, for any given continuous function $G(x)$ in $C^r[-1,1]$, there always exists a polynomial $P_d(x)$ capable of approximating uniformly $G(x)$ to arbitrary degree of accuracy. That is to say, $\forall\varepsilon_1 > 0$, $\|P_d - G\|_{C^r[-1,1]} < \varepsilon_1$. Furthermore, owing to lemma, $\forall\varepsilon_2 > 0$, a sufficiently large positive integer n^* can be found such that $\forall n > n^*$, $\|F_n - P_d\|_{C^r[-1,1]} < \varepsilon_2$. Thus,

$$\|F_n - G\| \leq \|F_n - P_d\| + \|P_d - G\| < \varepsilon_1 + \varepsilon_2 = \varepsilon \tag{B.11}$$

In other words, F_n can approximate uniformly G in $C^r[-1,1]$.