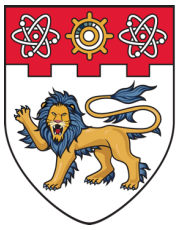


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Learning Control of Unmanned Aerial Vehicles Using Artificial Intelligence-Based Methods

Andriy Sarabakha

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the intellectual content of this thesis is the product of my original research work and has not been submitted for a higher degree to any other University or Institution.

02-01-2020
.....
Date

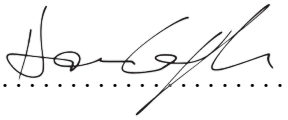

.....
Andriy Sarabakha

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

8 Jan 2020

Date



Assoc Prof Domenico Campolo

Co-Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

04/01/2020

.....

Date

.....

Assoc Prof Erdal Kayacan

Authorship Attribution Statement

This thesis contains material from ten papers published in the following peer-reviewed journals / from papers accepted at conferences in which I am listed as an author.

Chapter 2 is partially published as A. Sarabakha, C. Fu, E. Kayacan, and T. Kumbasar, "Type-2 Fuzzy Logic Controllers Made Even Simpler: From Design to Deployment for UAVs," IEEE Transactions on Industrial Electronics, vol. 65, no. 6, pp. 5069–5077, June 2018. doi:10.1109/TIE.2017.2767546. The contributions of the co-authors are as follows:

- I prepared the control algorithms and prepared the manuscript draft.
- Dr Fu assisted during the laboratory work.
- A/Prof Kayacan edited the manuscript draft.
- A/Prof Kumbasar suggested the research area.

Chapter 3 is published as C. Fu, A. Sarabakha, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi, "Input Uncertainty Sensitivity Enhanced Non-Singleton Fuzzy Logic Controllers for Long-Term Navigation of UAVs," IEEE/ASME Transactions on Mechatronics, vol. 23, no. 2, pp. 725–734, Apr. 2018. doi:10.1109/TMECH.2018.2810947, C. Fu, A. Sarabakha, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi, "Novel, Similarity-Based Non-Singleton Fuzzy Logic Control for Improved Uncertainty Handling in Quadrotor UAVs," in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, Italy, 2017, pp. 1–6. doi:10.1109/FUZZ-IEEE.2017.8015440, and C. Fu, A. Sarabakha, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi, "A Comparative Study on the Control of Quadcopter UAVs by Using Singleton and Non-Singleton Fuzzy Logic Controllers," 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Vancouver, Canada, 2016, pp. 1023–1030. doi:10.1109/FUZZ-IEEE.2016.7737800. The contributions of the co-authors are as follows:

- I coded the control algorithms and assisted during the laboratory work.
- Dr Fu prepared the manuscript' drafts.
- A/Prof Kayacan suggested the research area and revised the manuscript.
- A/Prof Wagner provided the initial ideas and revised the manuscript draft.
- Prof John revised the manuscript.
- Prof Garibaldi revised the manuscript.

Chapter 4 is partially published as E. Kayacan, A. Sarabakha, S. Coupland, R. John, M. A. Khanesar, "Type-2 Fuzzy Elliptic Membership Functions for Modeling Uncertainty," Engineering Applications of Artificial Intelligence, vol. 70, pp. 170–183, Apr. 2018. doi:10.1016/j.engappai.2018.02.004. The contributions of the co-authors are as follows:

- I performed all the laboratory work.
- A/Prof Kayacan prepared the manuscript draft.
- Dr Coupland revised the manuscript.
- Prof John revised the manuscript.
- Dr Khanesar edited the manuscript.

Chapter 5 is published as A. Sarabakha, C. Fu, and E. Kayacan, "Intuit Before Tuning: Type-1 and Type-2 Fuzzy Logic Controllers," Applied Soft Computing, vol. 81, pp. 105495, Aug. 2019. doi:10.1016/j.asoc.2019.105495, and A. Sarabakha, C. Fu, and E. Kayacan, "Double-Input Interval Type-2 Fuzzy Logic Controllers: Analysis and Design," in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, Italy, 2017, pp. 1–6. doi:10.1109/FUZZ-IEEE.2017.8015485. The contributions of the co-authors are as follows:

- I developed the research idea and prepared the manuscript drafts.
- Dr Fu assisted during the laboratory work.
- A/Prof Kayacan suggested the research area and edited the manuscript draft.

Chapter 6 is published as S. Patel, A. Sarabakha, D. Kircali, and E. Kayacan, "An Intelligent Hybrid Artificial Neural Network-Based Approach for Control of Aerial Robots," Journal of Intelligent & Robotic Systems, pp. 1–12, May 2019. doi:10.1007/s10846-019-01031-z. The contributions of the co-authors are as follows:

- I developed the control algorithms.
- Mr Patel performed experiments and prepared the manuscript draft.
- Mr Kircali prepared the experimental setup and performed experiments.
- A/Prof Kayacan suggested the research area and revised the manuscript.

Chapter 7 is published as S. Zhou, A. Sarabakha, E. Kayacan, M. K. Helwa, and A. P. Schoellig, "Knowledge Transfer Between Robots with Similar Dynamics for High-Accuracy Impromptu Trajectory Tracking," in 2019 European Control Conference (ECC), Naples, Italy, 2019, pp. 1–8. doi:10.23919/ECC.2019.8796140. The contributions of the co-authors are as follows:

- I provided the initial ideas.
- Ms Zhou performed the laboratory work and prepared the manuscript draft.
- A/Prof Kayacan revised the manuscript.
- Dr Helwa edited the manuscript draft.
- A/Prof Schoellig suggested the research area.

Chapter 8 is published as A. Sarabakha, N. Imanberdiyev, E. Kayacan, M. A. Khanesar, and H. Hagras, "Novel Levenberg–Marquardt Based Learning Algorithm for Unmanned Aerial Vehicles," Information Sciences, vol. 417, pp. 361–380, Nov. 2017. doi:10.1016/j.ins.2017.07.020. The contributions of the co-authors are as follows:

- I coded the control algorithms and prepared the manuscript draft.
- Mr Imanberdiyev performed experiments and prepared the manuscript draft.

- A/Prof Kayacan suggested the research area and edited the manuscript draft.
- Dr Khanesar provided the theoretical framework.
- Prof Hagras revised the manuscript.

Chapter 9 is published as A. Sarabakha, and E. Kayacan, "Online Deep Fuzzy Learning for Control of Nonlinear Systems Using Expert Knowledge," IEEE Transactions on Fuzzy Systems. doi:10.1109/TFUZZ.2019.2936787, and A. Sarabakha, and E. Kayacan, "Online Deep Learning for Improved Trajectory Tracking of Unmanned Aerial Vehicles Using Expert Knowledge," in 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, 2019, pp. 7727–7733. doi:10.1109/ICRA.2019.8794314. The contributions of the co-authors are as follows:

- I performed the laboratory work and prepared the manuscript drafts.
- A/Prof Kayacan revised the manuscript drafts.

02-01-2020 Andriy Sarabakha

Date Andriy Sarabakha

Abstract

In recent years, many research activities have focused on the developments for unmanned aerial vehicles (UAVs) due to their usefulness in providing cost-effective solutions to dangerous, dirty and dull tasks. In many applications, it is crucial for UAVs to be able to fly autonomously in uncertain environments under variable operating conditions. In such circumstances, an intelligent capability of the flight controller is a must rather than a choice. Model-free controllers propose alternative solutions to the model-based controllers without requiring a precise system's model which is often either unavailable or time-consuming to obtain. One branch of model-free methods is composed by fuzzy logic controllers (FLCs) due to their capability of delivering excellent control in the presence of uncertainties. However, one weakness of FLCs is that their parameters have to be tuned to deal efficiently with uncertainties. On the other hand, neural networks are computing models which progressively improve their performance by learning from training examples. Hence, artificial neural networks (ANNs) and deep neural networks (DNNs) propose learning approaches to enhance control strategies. Nevertheless, the main disadvantage of neural networks is that their inner workings are difficult to interpret. The limitations of fuzzy logic and neural networks were a driving force behind the creation of hybrid systems where the combination of DNN and FLC can overcome the drawbacks of each individual method.

This thesis focuses on the aforementioned artificial intelligence-based control methods that enable UAVs to accurately track 3D trajectories. The investigation starts from the simplest static type-1 FLC, through interval type-2 FLC, to the most efficient novel fuzzy mapping-based controllers. In this thesis, it was demonstrated that the analytical representation of the fuzzy mapping facilitates the tuning of the parameters in FLCs. Next, the controllers based on ANNs and DNNs with learning capabilities were investigated. In this thesis, it was verified experimentally that the proposed approaches can improve real-time control performance. Finally, a novel deep fuzzy neural network framework which profoundly fuses DNN and FLC for online training was proposed and validated under a variety of operating conditions.

"A helicopter is a mechanical engineer's dream and an aerodynamicist's nightmare."

—**John Watkinson**, British teacher

"If you are in trouble anywhere, an airplane can fly over and drop flowers, but a helicopter can land and save your life."

—**Igor Sikorsky**, Ukrainian American aviation pioneer

"Once you have tasted flight, you will forever walk the earth with your eyes turned skyward, for there you have been, and there you will always long to return."

—**Leonardo da Vinci**, Italian engineer, scientist and painter

Acknowledgements

This thesis would not have been possible without the help and support of many people. I would like to thank all those who have inspired and assisted me during my PhD period.

First of all, I would like to thank my family for their continuous and priceless support. To my parents, without you it would not have been possible to achieve this great goal. To my sister, Olena, thank you for supporting me all this time and checking all my papers. To my girlfriend, Ana, you have always encouraged the best of me and I will be forever grateful.

Moreover, I would like to thank my supervisor, Prof. Erdal Kayacan, for selecting me as a PhD student, providing me with motivating academic guidance and giving me the freedom to pursue many interesting ideas. I would also like to thank my supervisor, Prof. Domenico Campolo, for the appreciated advice and feedback. This thesis is the result of many fruitful collaborations and scientific discussions with Prof. Changhong Fu, Prof. Yiqun Dong, Prof. Robert John, Prof. Christian Wagner, Prof. Tufan Kumbasar, Prof. Angela Schoellig and Prof. Giuseppe Loianno. Many thanks to my colleagues, Nursultan Imanberdiyev, Mohit Mehndiratta, Efe Camci, Ilker Bozcan, SiQi Zhou and many others, for insightful conversations. I would like to extend my gratitude to Dogan Kircali and Siddharth Patel for technical discussions and practical suggestions which have helped me to improve the quality of my work. During my time at NTU, I learned a lot, gathered many valuable experiences, and had much fun time. Thus, I am grateful to all the amazing people who contributed to any of those things. Last but not least, I would like to thank the examiners of my thesis who gave me very detailed and useful comments.

Singapore,
June 2020 *Andriy*

Contents

List of Figures

List of Tables

List of Symbols

Unless stated differently, the conventions are defined as follows:

- scalars are represented by lower case or upper case italic letters, e.g., s or S;
- vectors are represented by lower case bold letters, e.g., $\mathbf{v}$;
- matrixes are represented by upper case bold letters, e.g., $\mathbf{M}$;
- desired values are indicated with star superscripts, e.g., a^*;
- estimated values are indicated with hat overline, e.g., $\hat{a}$;
- predicted values are indicated with tilde overline, e.g., $\tilde{a}$;
- measured values are indicated with bar overline, e.g., $\bar{a}$.

A	antecedent fuzzy set	
$\mathbf{a}$	linear acceleration in world frame	$[\mathrm{m/s^2}]$
$\mathbf{a}_{\mathcal{B}}$	linear acceleration in body frame	$[\mathrm{m/s^2}]$
b	propeller's thrust coefficient	$[\mathrm{N \cdot s^2}]$
C	consequent fuzzy set	
d	propeller's drag coefficient	$[\mathrm{N \cdot m \cdot s^2}]$
$\mathbf{e_p}$	position error	[m]
$\mathbf{e_R}$	attitude error	[rad]
$\mathbf{e_v}$	velocity error	[m/s]
$\mathbf{e}_{\boldsymbol{\omega}}$	angular velocity error	[rad/s]

$\mathcal{F}_B$	body reference frame	
$\mathcal{F}_W$	world fixed reference frame	
$\mathbf{f}_e$	external forces	[N]
f_i	force generated by the i^{th} motor	[N]
g	gravitational acceleration constant	[m/s^2]
$\mathbf{I}$	inertia matrix	[kg $\cdot$ m^2]
$\mathcal{I}$	input to neural network	
I_x	moment of inertia about x-axis	[kg $\cdot$ m^2]
I_y	moment of inertia about y-axis	[kg $\cdot$ m^2]
I_z	moment of inertia about z-axis	[kg $\cdot$ m^2]
k	time step	
l	arm length	[m]
m	mass	[kg]
N_D	number of data samples	
N_H	number of neurons in the hidden layer	
N_I	number of system's inputs	
N_M	number of UAV's motors	
N_O	number of system's outputs	
N_R	number of rules	
N_S	number of system's states	
$\mathcal{O}$	output from neural network	
$\mathbf{p}$	position	[m]
R	rule in the rule-base	
$\mathbf{R}$	rotation matrix	
R	rule-base	
$\mathbf{r}$	relative degree of a system	
$\mathbf{S}$	similarity factor	
$\mathbf{T}$	transformation matrix	
T	thrust force	[N]
t	time	[s]
$\mathbf{u}$	control input to the system	

$\mathbf{v}$	linear velocity in world frame	[m/s]
$\mathbf{v}_{\mathcal{B}}$	linear velocity in body frame	[m/s]
v_x	linear velocity in world x-axis	[m/s]
v_y	linear velocity in world y-axis	[m/s]
v_z	linear velocity in world z-axis	[m/s]
$\mathbf{W}$	weights in the neural network	
$\mathbf{x}$	state of the system	
x	position along world frame x-axis	[m]
$\mathbf{y}$	output from the system	
y	position along world frame y-axis	[m]
z	position along world frame z-axis	[m]
α	learning rate	
ϵ	aggressiveness factor	
θ	pitch angle	[rad]
$\boldsymbol{\theta}$	attitude	[rad]
$\boldsymbol{\Sigma}$	fuzzy input set	
$\boldsymbol{\sigma}$	crisp inputs to FLS	
$\boldsymbol{\tau}_e$	external torques	[N · m]
τ_i	torque generated by the i^{th} motor	[N · m]
τ_θ	pitch torque in body frame	[N · m]
τ_ϕ	roll torque in body frame	[N · m]
τ_ψ	yaw torque in body frame	[N · m]
$\boldsymbol{\Phi}$	fuzzy output set	
ϕ	roll angle	[rad]
$\boldsymbol{\varphi}$	crisp outputs from FLS	
ψ	yaw angle	[rad]
$\boldsymbol{\Omega}$	speed of the motors	[rad/s]
$\boldsymbol{\omega}$	angular velocity in world frame	[rad/s]
$\boldsymbol{\omega}_{\mathcal{B}}$	angular velocity in body frame	[rad/s]
ω_i	speed of the i^{th} motor	[rad/s]

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BIBS	Bounded-Input Bounded-State
BIBO	Bounded-Input Bounded-Output
COM	Center Of Mass
CPU	Central Processing Unit
DFNN	Deep Fuzzy Neural Network
DNN	Deep Neural Network
DI-IT2-FLC	Double-Input IT2-FLC
DI-IT2-FLS	Double-Input IT2-FLS
DI-IT2-FPD	Double-Input Interval Type-1 Fuzzy PD
DI-T1-FLC	Double-Input T1-FLC
DI-T1-FLS	Double-Input T1-FLS
DI-T1-FPD	Double-Input Type-1 Fuzzy PD
FLC	Fuzzy Logic Control
FLS	Fuzzy Logic System
FNN	Fuzzy Neural Network
FOU	Footprint Of Uncertainty
FM	Fuzzy Mapping
FS	Fuzzy Set
GP	Gaussian Process
GPS	Global Positioning System
GPU	Graphics Processing Unit

IMU	Inertial Measurement Unit
IT2-FLC	Interval Type-2 FLC
IT2-FLS	Interval Type-2 FLS
IT2-FNN	Interval Type-2 FNN
KM	Karnik-Mendel
LM	Levenberg-Marquardt
MAE	Mean Absolute Error
MAX	Maximum Absolute Error
MF	Membership Function
MIMO	Multiple-Input Multiple-Output
MSE	Mean Squared Error
MVCS	Mean Variation of Control Signal
NSFLC	Non-Singleton FLC
NSFLS	Non-Singleton FLS
PD	Proportional-Derivative
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PS	Parameter Setting
RAM	Random Access Memory
RMSE	Root Mean Squared Error
ROS	Robot Operating System
RTK-GPS	Real-Time Kinematic GPS
SFLC	Singleton FLC
SFLS	Singleton FLS
SMC	Sliding Mode Control
T1-FLC	Type-1 FLC
T1-FLS	Type-1 FLS
T1-FNN	Type-1 FNN
T2-FLC	Type-2 FLC
TSK	Takagi-Sugeno-Kang
UAV	Unmanned Aerial Vehicle

Part I

Introduction and Background

Chapter 1

Introduction

Since the beginning of time flying objects have exerted a great fascination on mankind. The last decades have seen many exciting developments in the area of unmanned aerial vehicles (UAVs) or drones. Moreover, UAVs are gaining increasing interest due to a wide area of applications from military to civilian fields. An attractive group of flying robots is composed of multicopter aircraft. A *multicopter*, e.g., quadcopter or hexacopter, is a UAV which has vertical take-off and landing characteristics. Due to its simple mechanical structure, it is capable of flying without all those complex linkages appearing in typical helicopters. However, like a classical helicopter, multicopters belong to a group of dynamical systems with non-linear dynamics. Additionally, it is really hard to model all second-order effects. Thus, a control system capable of dealing with non-linearity, unmodelled dynamics and disturbances is needed [1]. Furthermore, integrating the sensors, actuators and intelligence into a lightweight flying system is not trivial.

Designing nonlinear controllers for real-world systems to achieve high-accuracy tracking is typically difficult, due to nonlinear relations combined with parameter uncertainties, external disturbance, noisy measurements and other nonidealities in the systems. However, designing and tuning conventional model-based controllers to achieve satisfactory performance can be a time-consuming and difficult task, due to non-linear dynamics, aerodynamic effects and various flight operating regimes. In the presence of the aforementioned conditions, a model-free controller may be preferred over model-based controllers.

Fuzzy logic is a form of many-valued reasoning paradigm in which the truth values of variables may assume any real number between 0 and 1 both inclusive [2]. Consequently, fuzzy logic systems (FLSs) are employed to handle the concept of partial truth. Thereupon, fuzzy logic controllers (FLCs), which inherit FLSs, are alternative solutions to the model-based controllers without the requirement for a precise mathematical model of the system which is often either unavailable or time-consuming to obtain. Moreover, FLCs can improve the robustness of the control system in the presence of uncertainties and noise. Type-1 FLCs (T1-FLCs) are the most widely used types of FLCs, due to their limited complexity from design and computation perspectives [3].

Though T1-FLCs are widely used, type-1 fuzzy sets (FSs), described by type-1 membership functions (MFs), can effectively handle only bounded levels of uncertainty, while real-world applications frequently have to deal with high levels and multiple sources of uncertainty [4]. Therefore, there has been a growing interest in a more advanced form of FLC, namely type-2 FLC (T2-FLC) [5]. Better handling of the uncertainty using T2-FLCs is provided by an additional degree of freedom benefiting from the footprint of uncertainty (FOU) in their FSs [6]. However, also an additional complexity arises from the inclusion of FOU as well as the third dimension [7]. Therefore, the research has tended to focus on interval T2-FLCs (IT2-FLCs) [8], rather than on general T2-FLCs [9], because the mathematical formulation of general T2-FLCs is much more complex than that of IT2-FLCs [10]. The adoption of IT2-FLC allows reducing the computational complexity which is an immense benefit in real-time applications [11].

However, modern computers can perform the basic algebraic operations, e.g., additions, subtractions, multiplication and divisions, much more efficiently than the operations of FSs, e.g., unions, intersections and implications, needed in fuzzy logic [12]. Therefore, the availability of an analytical form of fuzzy mapping (FM), which represents FLS, can open new doors to the use of FLCs in real-time applications. Still, one weakness of FLCs is that their parameters have to be tuned to deal with uncertainties.

By definition, artificial neural networks (ANNs) are computing models which progressively improve their performance by learning from training examples [13]. Similarly to biological brains, ANNs are built by many simple processing elements, called neurons, which are interconnected by links, called synapses [14]. Hence, ANN

learns from the training samples by adjusting the synaptic weights of the connections between neurons [15]. Moreover, ANNs reduce the need for feature engineering which is one of the most time-consuming tasks in machine learning, for the training data [16]. Therefore, ANNs are ideal for situations that require approximating a function that depends on a huge number of inputs which nonlinearly connects to the output [17]. Given the ability of ANNs to generalise knowledge from training samples, ANN-based controllers are suitable to control nonlinear systems [18].

Though ANN can generalise knowledge from training samples, common single-hidden-layer ANNs are able to approximate effectively only simple nonlinear functions, while real-world systems are frequently highly nonlinear [19]. On the other hand, deep neural networks (DNNs) which are distinguished from single-hidden-layer ANNs by their depth that is the number of layers through which data must pass in a multi-step process [20]. Thus, DNNs can effectively be used to solve advanced tasks similar to or even better than human experts [21]. Moreover, DNNs can approximate non-linear functions with an exponentially lower number of training parameters and higher sample complexity when compared to ANNs [22]. Therefore, DNNs propose a novel approach to enhance the control strategies for nonlinear systems [23]. After training the DNN module on collected flight samples, it can be used in real-time to provide the control signal [24].

The fuzzy logic has an exceptional ability to handle the uncertainties in the system [25]. However, one weakness of FLCs is that their parameters have to be tuned to deal with uncertainties [26]. On the other hand, ANNs are a family of supervised learning models that mimics human brain [17]. Yet, the main weakness of ANNs is that their inner workings are difficult to interpret [27]. The combination of FLC and ANN, called fuzzy neural network (FNN), fuses the reasoning ability of FLC to handle uncertain information with the training capability of ANN to learn from the controlled process [28]. Consequently, FNN adopts the advantages of both FLC and ANN [29].

It has been shown that DNNs are good at approximating knowledge but they do not explain how they take their decisions [30]. On the other hand, FLSs are good at explaining their decisions but they are not good at acquiring new information [26]. The limitations of these two techniques have been a driving force behind the creation of hybrid systems where the combination of DNN and FLS, called deep fuzzy neural network (DFNN), can overcome the drawbacks of each method [31].

This thesis focuses on the aforementioned artificial intelligence-based control methods that enable UAVs to accurately track 3D trajectories. First, the investigation considers the simplest static type-1 FLC, interval type-2 FLC, and the most efficient FM-based controllers. Consequently, the controllers with learning capabilities are studied based on ANNs and DNNs. Finally, the efficacy of combining fuzzy logic and neural networks to adopt the advantages of both techniques is discussed.

1.1 Related Works

As being one of the fastest-growing sectors in the aerospace industry, UAVs can provide an inexpensive solution to time-consuming, dull, dirty and dangerous missions, such as emergency evacuation [32], traffic surveillance [33], aircraft detection [34], orchard monitoring [35] and environment mapping [36]. There are two control paradigms for UAVs: model-based [37], which needs an exact model of the system, and model-free [38], which does not need an exact model of the system. The examples of the most widely used model-based controllers are proportional-integral-derivative (PID) [39], dynamic feedback linearization [40], linear-quadratic regulator [41], and model predictive control [42].

On the other hand, FLCs have been proposed as an alternative approach to conventional model-based controllers when it is challenging to obtain the precise mathematical model of the system [43–45]. This is due to several characteristics of FLCs, inter-alia, improving the robustness and of the nonlinear control system in the presence of uncertainties and external disturbances by using the expert knowledge [25]. Therefore, FLCs have become one of the most popular model-free approaches to control mobile robots [46, 47], especially UAVs [48], since their precise mathematical model is challenging to obtain.

There are several types of control methods that use FLS as an essential component. The majority of applications belong to the class of fuzzy PID controllers, where FLS is placed within the feedback control loop and computes the PID control signal through fuzzy inference [49]. In [50], more systematic analysis and design for conventional double-input T1-FLC (DI-T1-FLC) are presented. In [51], a fuzzy variable structure control is introduced for designing and tuning of DI-T1-FLC based on variable structure control theory. The fuzzy PID controller derived in [52]

successfully demonstrated better performance than the conventional PID controller for many cases, particularly for nonlinear plants. In [53], a function-based evaluation approach is proposed for a systematic study of type-1 fuzzy PID controllers. In [54], a general technique is developed for rigorously deriving analytical input-output structure for fuzzy controllers that use Zadeh fuzzy AND-operator. In [55], an analytical structure for fuzzy PID controllers has been derived using L-type and G-type input FSs, trapezoidal output FSs, Mamdani minimum inference method, algebraic product triangular norm, bounded sum triangular co-norm and center of sums defuzzification method.

In the literature, singleton FLCs (SFLCs) are the most well-known and widely used types of FLCs [56–58]. On the other hand, it is reported that non-singleton FLCs (NSFLCs) give more promising results when compared to their singleton counterparts for non-linear servo systems [59], active suspension systems [60], and UAV control [61], where nonlinearities and uncertainties are more visible in the system. Although both singleton and non-singleton T1-FLCs use the same fuzzy rule base, inference engine and defuzzifier, there is a different fuzzifier in NS-T1-FLCs which treats the inputs as FSs to deal with input uncertainties better [62–64].

Recently, many researchers have put significant attention toward more advanced forms of fuzzy logic, such as T2-FLCs [65–67]. The transition from T1-FLCs to T2-FLCs has been inspired by the observation that type-1 FSs can only deal with a limited level of uncertainty whereas real-world control applications are often confronted with high levels and multiple sources of uncertainty [68]. T2-FLCs can be used to handle uncertainties better in the system, e.g., noisy measurements, due to the additional degree of freedom provided by FOU in their FSs [69]. IT2-FLCs have received more consideration [70–73], because the mathematics that is needed for IT2-FLCs – primarily interval arithmetic – is much simpler than that of general T2-FLCs [74]. The use of IT2-FLC helps to decrease the computation time which is a big advantage in real-time on-board control applications [75].

Several studies have been presented to analyse the effect of the FOU on the type-2 FM [76]. In [77], the analytical structure of a special class of IT2 fuzzy proportional-derivative (PD) and proportional-integral (PI) controllers that uses the Karnik-Mendel (KM) iterative algorithm for the type-reduction has been presented. In [78], the mathematical input-output structure of Mamdani IT2 fuzzy PI controllers

is derived for centroid and averaged defuzzifiers. Instead of using common type-reduction methods, IT2-FLC analysed in [78] approximates the type-reduced set by averaging embedded IT2-FLCs. In [3], some recent research results are summarized on understanding the fundamental differences between T1-FLCs and IT2-FLCs. It has been shown in [3] that IT2-FLC can implement complex control surfaces that cannot be achieved by T1-FLC using the same rule-base. In [79], a technique is developed which is capable of deriving the analytical structure for a wide class of IT2-FLCs. In [80], an explicit solution is proposed to determine optimal switching points of KM method for double-input IT2-FLCs (DI-IT2-FLC). In [81], the analytical structural analysis of the simplest DI-IT2-FLC is presented. In [82], an approach to derive the analytical structure of a class of double-input Takagi-Sugeno FLCs is presented. Recently, in [83], the authors analysed the input-output relationship of various IT2-FLCs with trapezoidal FSs, and compared the difference in control performance via analytical structure approach. Nevertheless, an exhaustive analysis of FM for Mamdani DI-IT2-FLCs and real-time validation of the theoretical claims are still missing in the literature [84]. The continuity of T1-FLCs and IT2-FLCs have been introduced in [76]. Moreover, the study of other properties, such as symmetry and monotonicity, of FM of double-input FLCs is missing in the literature.

In the literature, neural networks have successfully been integrated within control systems to improve tracking performance [85]. In [86], the unknown part of the dynamical model of a quadcopter is modelled by DNN. In [87], a robust direct inverse control of a quadrotor is learnt by recurrent DNN in simulation. In [88] and [89], DNNs are used to learn the dynamics of helicopter and multicopter, respectively. In [90], DNN pre-cascaded module is used to improve the performance a quadrotor in tracking arbitrary hand-drawn trajectory. However, in all these works, DNNs are trained offline and, then, used in real-time without further learning.

In the literature, there are attempts to integrate strengths of learning capability of neural networks and reasoning ability provided by fuzzy logic, called fuzzy neural network (FNN), for various applications, such as emission prediction [91], movie classification [92] and robot control [93]. However, in all these approaches, the sequential learning paradigms is implemented [94]. For example, in [91, 93], first, the original inputs are fuzzified and, then, the fuzzy numbers are fed into the neural network. Contrarily, the method in [92], first, transforms the original data by using DNN and, then, the deep representation is fuzzified at the output layer.

1.2 Contribution

The major and minor contributions achieved in this thesis are listed below.

Major contributions:

- Single input interval type-2 fuzzy PID controllers are derived and elaborated in terms of their interpretability (video: tiny.cc/SI-IT2-FLC).
- A fuzzy mapping for double-input interval type-2 FLC has been explicitly derived and analysed (video: tiny.cc/FM-FLC).
- A novel DFNN-based framework has been developed for learning online the inverse dynamics of a system (video: tiny.cc/DFNN).

Minor contributions:

- The simulation setup has been implemented for quadcopter and coaxial hexacopter UAV in ROS and Gazebo environment (video: tiny.cc/QLearning).
- Three different type-1 FLCs have been implemented for a quadcopter UAV control problem (video: tiny.cc/T1-FLC).
- Type-1 FLC has been used with monocular visual-inertial SLAM in the long-term navigation of quadrotor UAV (video: tiny.cc/SLAM-FLC).
- An ANN-based control method, which enables both fast flight and agile manoeuvres, has been developed and tested (video: tiny.cc/fast_ANN).
- An ANN-based controller has been applied for the fault-tolerant control for actuator failure in coaxial hexacopter (video: tiny.cc/fault_ANN).
- A DNN-based controller for transfer learning has been analytically derived and tested on two different UAVs (video: tiny.cc/DNN).
- A novel LM theory-based algorithm for type-1 FNN has been presented to control a quadcopter UAV (video: tiny.cc/LM-FNN).

1.3 Outline

The outline of this work is as follows. After this introductory chapter, in Chapter 2, the research problem related to the control of nonlinear systems in general – and UAV in particular – is defined. In Chapter 3, potentials of different singleton and non-singleton T1-FLCs are explored in simulation and real-time experiments by using monocular keyframe-based visual-inertial localization for position estimation. In Chapter 4, capabilities of different IT2-FLCs with Gaussian and elliptic MFs are investigated for changing speed trajectories. In Chapter 5, an alternative method to derive and analyse FM for T1-FLCs and IT2-FLCs is proposed and validated in simulation and real-time experiments. In Chapter 6, potentials of ANNs-based controllers are explored under fast and agile flights with a motor failure case for a hexacopter UAV. In Chapter 7, DNN-based controller is proposed to solve a transfer learning problem between similar robots and validated in simulation and real-time experiments on two similar UAVs. In Chapter 8, potentials of FNN-based controllers with sliding mode control (SMC) theory-based and Levenberg-Marquardt (LM) theory-based training algorithms are investigated in the presence of periodic wind gust in simulation and real-time experiments. In Chapter 9, capabilities of a novel DFNN-based controller are explored under various operational conditions in simulation and real-time experiments. Finally, some conclusions and possible future works are drawn in Section 10. Besides, the derivations of the 3D rotation matrix and of the rate transformation matrix are reported in Appendix A; while the hat and vee mapping operators are defined in Appendix B.

A schematic overview of this thesis is depicted in Fig. 1.1 to show the relations between different chapters and systems.

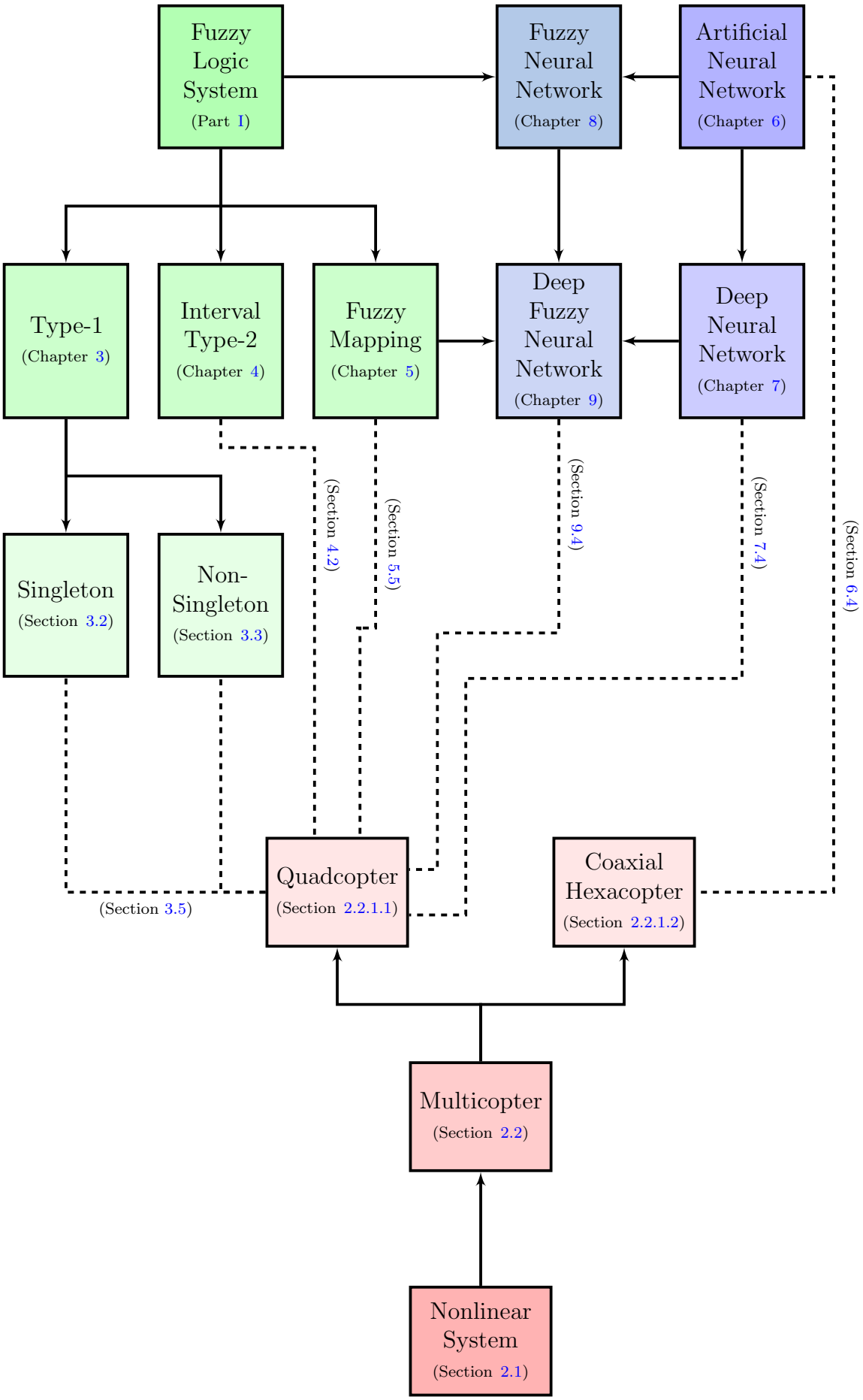


Figure 1.1: Schematic overview of this thesis.

Chapter 2

Problem Definition

The term system, which comes from the Greek word "*systema*", defines a collection of inter-related elements pursuing a particular objective [95]. Nearly all physical systems are inherently nonlinear in nature since most of the real-world relationships are nonlinear [96]. A nonlinear system is a system that does not satisfy the superposition principle [97]. Nonlinear dynamical systems, describing changes in system variables over time, may appear chaotic, counterintuitive and unpredictable, contrasting with much simpler linear systems. The dynamical model of a UAV is inherently nonlinear since the aerodynamic laws are highly nonlinear. Designing nonlinear controllers for real-world systems to achieve high-accuracy tracking is typically difficult, due to nonlinear relations combined with parameter uncertainties, external disturbance, noisy measurements and other nonidealities in the systems.

In this chapter, the research problem related to the control of nonlinear systems in general – and UAV in particular – is defined. First, Section 2.1 presents general nonlinear discrete-time multi-input multi-output (MIMO) systems with internal uncertainties, external disturbances and noisy measurements. Then, Section 2.2 describes dynamical models and control schemes for quadcopter and coaxial hexacopter UAVs.

Supplementary Material:
Matlab model of a nonlinear system: github.com/andriyukr/nonlinear_system.
Gazebo and Matlab models of UAV: github.com/andriyukr/uav_model.

2.1 Nonlinear Systems

First, consider a general nonlinear discrete-time MIMO dynamical system represented by its state-space model:

$$\begin{cases} \mathbf{x}(k+1) & = f(\mathbf{x}(k)) + g(\mathbf{x}(k))\mathbf{u}(k) \\ \mathbf{y}(k) & = h(\mathbf{x}(k)), \end{cases} \tag{2.1}$$

where $k \in \mathbb{N}^+$ is the time step, $\mathbf{x} \in \mathbb{R}^{N_S}$ is the state of the system, $\mathbf{u} \in \mathbb{R}^{N_I}$ is the input to the system, $\mathbf{y} \in \mathbb{R}^{N_O}$ is the output from the system, and $f : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S}$, $g : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S} \times \mathbb{R}^{N_I}$ and $h : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O}$ are system functions.

Definition 2.1.1. Let $\mathbf{r} \in \mathbb{N}_0^{N_O}$ be the vector of relative degrees of the system, which is the number of times one has to differentiate the output to have at least one of the inputs explicitly appearing [98], i.e.:

$$\arg\max_{\mathbf{r}_i} \frac{\partial}{\partial \mathbf{u}} \left[h_i \left(f^{\mathbf{r}_i - 1} \left(f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \right) \right) \right] \neq \mathbf{0}, \quad i \in \{1, \ldots, N_O\}. \tag{2.2}$$

Assumption 1. The system in (2.1) has well-defined relative degrees in (2.2) [99].

Assumption 2. The system in (2.1) is minimum-phase [100].

Assumption 3. The system in (2.1) is input-to-output stable [101], i.e.:

$$\|\mathbf{y}(k)\| \leq \gamma \left(\|\mathbf{u}(k)\| \right) \quad \forall k, \tag{2.3}$$

where γ is a gain function. In other words, γ is a scalar continuous function which is nondecreasing and $\gamma(0) = 0$.

The input and the output of the system are related by

$$\mathbf{y}_i(k + \mathbf{r}_i) = h_i \left(f^{\mathbf{r}_i - 1} \left(f(\mathbf{x}(k)) + g(\mathbf{x}(k))\mathbf{u}(k) \right) \right), \quad i \in \{1, \ldots, N_O\}. \tag{2.4}$$

If $\mathbf{y}$ is affine in $\mathbf{u}$, then (2.4) becomes

$$\mathbf{y}_i(k + \mathbf{r}_i) = \mathcal{F}_i(\mathbf{x}(k)) + \mathcal{G}_i(\mathbf{x}(k))\mathbf{u}(k), \quad i \in \{1, \ldots, N_O\}, \tag{2.5}$$

where

$$\mathcal{F}_i(\mathbf{x}(k)) = h_i \left(f^{\mathbf{r}_i}(\mathbf{x}(k)) \right) : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O} \tag{2.6}$$

and

$$\mathcal{G}_i(\mathbf{x}(k)) = \frac{\partial}{\partial \mathbf{u}(k)} \left[h_i \left(f^{\mathbf{r}_i - 1} \left(f(\mathbf{x}(k)) + g(\mathbf{x}(k))\mathbf{u}(k) \right) \right) \right] : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O} \times \mathbb{R}^{N_I} \quad (2.7)$$

are decoupling functions. Finally, to track the desired output of the system $\mathbf{y}^* \in \mathbb{R}^{N_O}$, the control law at time k can be written as in [102]:

$$\mathbf{u}_i(k) = \left[\mathcal{G}(\mathbf{x}(k)) \right]^{-1} \left(\mathbf{y}^*(k + \mathbf{r}_i) - \mathcal{F}(\mathbf{x}(k)) \right), \quad i \in \{1, \ldots, N_O\}. \quad (2.8)$$

Assumption 4. The desired output of the system in (2.1) is available and bounded, i.e.:

$$\forall k \quad \exists \mathbf{y}^*(k) \in \mathbb{R}^{N_O} \quad | \quad \|\mathbf{y}^*(k)\|_\infty \leq c_{\mathbf{y}^*}, \quad (2.9)$$

where $c_{\mathbf{y}^*}$ is some positive constant [103].

If a precise model of the system exists, the inversion of the system can be computed. However, the system's parameters might be unknown and difficult to estimate (e.g., moments of inertia). Besides, these parameters might change during the operation of the system (e.g., mass). In addition, it is difficult to predict the external disturbance term (e.g., wind gust). Furthermore, measurements from the system might come from a noisy sensor (e.g., monocular camera). Therefore, the control law in (2.8) cannot always be calculated precisely.

2.1.1 Internal Uncertainties

Consider a general nonlinear discrete-time MIMO dynamical system with internal uncertainties:

$$\begin{cases} \mathbf{x}(k+1) & = \tilde{f}(\mathbf{x}(k)) + \tilde{g}(\mathbf{x}(k))\mathbf{u}(k) \\ \mathbf{y}(k) & = \tilde{h}(\mathbf{x}(k)), \end{cases} \quad (2.10)$$

where $\tilde{f} : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S}$, $\tilde{g} : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S} \times \mathbb{R}^{N_I}$ and $\tilde{h} : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O}$ are new system functions. To track the desired output of the system $\mathbf{y}^*$, the control law is

$$\mathbf{u}_i(k) = \left[\tilde{\mathcal{G}}(\mathbf{x}(k)) \right]^{-1} \left(\mathbf{y}^*(k + \mathbf{r}_i) - \tilde{\mathcal{F}}(\mathbf{x}(k)) \right), \quad (2.11)$$

where

$$\tilde{\mathcal{F}}_i(\mathbf{x}_k) = \tilde{h}_i \left(\tilde{f}^{\mathbf{r}_i}(\mathbf{x}(k)) \right) : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O} \quad (2.12)$$

and

$$\tilde{\mathcal{G}}_i(\mathbf{x}(k)) = \frac{\partial}{\partial \mathbf{u}(k)} \left[\tilde{h}_i \left(\tilde{f}^{\mathbf{r}_i - 1} \left(\tilde{f}(\mathbf{x}(k)) + \tilde{g}(\mathbf{x}(k)) \mathbf{u}(k) \right) \right) \right] : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O} \times \mathbb{R}^{N_I} \tag{2.13}$$

are new decoupling functions. Therefore, for an exact tracking of $\mathbf{y}^*$, the exact values of new system functions $\tilde{f}$, $\tilde{g}$ and $\tilde{h}$ are required.

2.1.2 External Disturbance

Consider a general nonlinear discrete-time MIMO dynamical system with external disturbances:

$$\begin{cases} \mathbf{x}(k+1) & = f(\mathbf{x}(k)) + g(\mathbf{x}(k)) \mathbf{u}(k) + d(k) \\ \mathbf{y}(k) & = h(\mathbf{x}(k)), \end{cases} \tag{2.14}$$

where $d : \mathbb{R} \to \mathbb{R}^{N_I}$ is the disturbance to the system. To track the desired output of the system $\mathbf{y}^*$, the control law at time k is

$$\mathbf{u}_i(k) = \left[\mathcal{G}(\mathbf{x}(k)) \right]^{-1} \left(\mathbf{y}^*(k + \mathbf{r}_i) - \mathcal{F}(\mathbf{x}(k) - \mathcal{D}(k)) \right), \tag{2.15}$$

where $\mathcal{D}(k) = h_i \left(f^{\mathbf{r}_i - 1}(d(k)) \right) \in \mathbb{R}^{N_O}$ is the disturbance decoupling matrix. Therefore, for an exact tracking of $\mathbf{y}^*$, the exact value of disturbance $d(k)$ is required.

2.1.3 Noisy Measurement

Consider a general nonlinear discrete-time MIMO dynamical system with noisy measurements:

$$\begin{cases} \mathbf{x}(k+1) & = f(\mathbf{x}(k)) + g(\mathbf{x}(k)) \mathbf{u}(k) \\ \mathbf{y}(k) & = h(\mathbf{x}(k)) + \mathcal{N}(k), \end{cases} \tag{2.16}$$

where $\mathcal{N} : \mathbb{R}^2 \to \mathbb{R}^{N_O}$ is an additive noise, e.g., additive white Gaussian noise, at time step k. To track the desired output of the system $\mathbf{y}^*$, the control law is

$$\mathbf{u}_i(k) = \left[\mathcal{G}(\mathbf{x}(k)) \right]^{-1} \left(\mathbf{y}^*(k + \mathbf{r}_i) - \mathcal{N}(k + \mathbf{r}_i) - \mathcal{F}(\mathbf{x}(k)) \right). \tag{2.17}$$

Therefore, for an exact tracking of $\mathbf{y}^*$, the exact model of noise $\mathcal{N}$ is required.

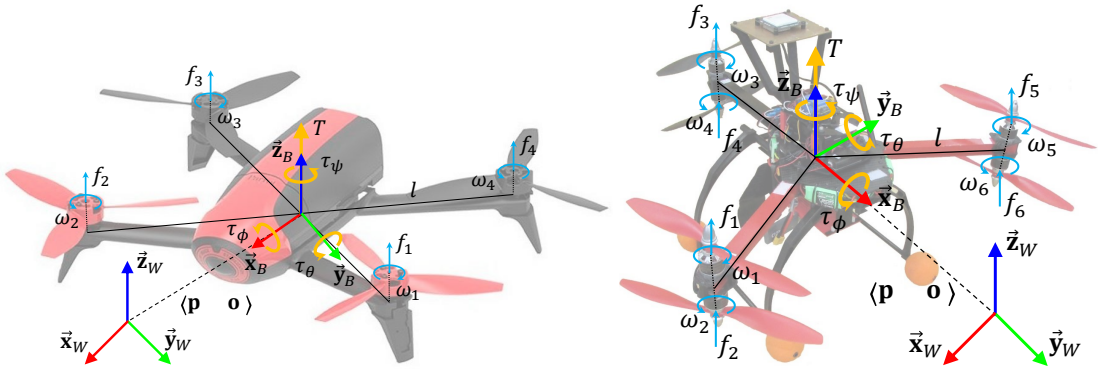
2.2 Multicopter Unmanned Aerial Vehicles

Nowadays, the use of UAVs by amateurs with minimal piloting skills, skilled hobbyists, and licensed pilots for purposes of aerial photography, drone racing, and hobby, continues to grow [104]. There are two major types of users which lead to the innovation and development of cutting-edge technology in this sector – military and consumer industry. The former is mainly interested in long-range endurance missions, and – thus, usually prefers fixed-wing UAVs, while the latter leans more toward vertical take-off and landing capable multicopter UAVs. This modern-day marvel is improving progressively and it is becoming accessible to most of the people. Furthermore, besides leisure activities, these UAVs have also started to assume an important role in a wide range of applications like – transportation [105], surveying and mapping [106], and search and rescue [107].

2.2.1 Multicopter Unmanned Aerial Vehicle's Dynamics

Let the world fixed inertial reference frame be $\mathcal{F}_W = \{\vec{\mathbf{x}}_W, \vec{\mathbf{y}}_W, \vec{\mathbf{z}}_W\}$ and the body frame be $\mathcal{F}_B = \{\vec{\mathbf{x}}_B, \vec{\mathbf{y}}_B, \vec{\mathbf{z}}_B\}$. The origin of the body frame is located at the center of mass (COM) of the UAV. Two UAV configurations with the corresponding reference frames for X4 quadrotor and Y6 coaxial hexacopter UAV are illustrated in Fig. 2.1.

The N_M motors' rotations generate N_M forces f_i, $i \in \{1, \ldots, N_M\}$, directed along the axis of rotation $\vec{\mathbf{z}}_B$ and with module proportional to the speed of rotation, and



(a) X4 quadrotor UAV. (b) Y6 coaxial hexacopter UAV.

Figure 2.1: UAVs configurations with their references frames.

N_M torques τ_i, $i \in \{1, \ldots, N_M\}$, around the axis of rotation $\vec{\mathbf{z}}_B$ and with module proportional to the speed of rotation [108]:

$$\begin{cases} f_i &= b\omega_i^2 \\ \tau_i &= d\omega_i^2 \end{cases}, \quad i \in \{1, \ldots, N_M\}, \tag{2.18}$$

where b is the propeller thrust coefficient, d is the propeller drag coefficient and ω_i is the rotational speed of the i^{th} propeller.

The UAV electric motors are velocity controlled, so the vector of control inputs $\mathbf{u}$ may bu considered directly as

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T, \tag{2.19}$$

where T is the total thrust and acts along $\vec{\mathbf{z}}_B$ axis, whereas τ_ϕ, τ_θ and τ_ψ are the moments acting around $\vec{\mathbf{x}}_B$, $\vec{\mathbf{y}}_B$ and $\vec{\mathbf{z}}_B$ axes, respectively.

The absolute position of the UAV $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is described by three Cartesian coordinates of its COM in $\mathcal{F}_W$. While the attitude of the UAV $\boldsymbol{\theta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ is described by three Euler's angles. These three angles are respectively called roll ϕ, pitch θ and yaw ψ.

The time derivative of the position (x, y, z) gives the linear velocity

$$\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T, \tag{2.20}$$

of the UAV's COM expressed in $\mathcal{F}_W$. Let $\mathbf{v}_B \in \mathbb{R}^3$ be the absolute velocity of the UAV expressed in $\mathcal{F}_B$. So, $\mathbf{v}$ and $\mathbf{v}_B$ are related by

$$\mathbf{v} = \mathbf{R}\mathbf{v}_B, \tag{2.21}$$

where $\mathbf{R} \in \mathsf{SO}(3)$ is the rotation matrix from $\mathcal{F}_B$ to $\mathcal{F}_W$ and is computed in Appendix A:

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{2.22}$$

Similarly, the time derivative of the attitude (ϕ, θ, ψ) gives the angular velocity expressed in $\mathcal{F}_W$:

$$\boldsymbol{\omega} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T, \tag{2.23}$$

and the angular velocity expressed in $\mathcal{F}_B$ is

$$\boldsymbol{\omega}_B = \begin{bmatrix} \omega_\phi & \omega_\theta & \omega_\psi \end{bmatrix}^T. \tag{2.24}$$

The relation between $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_B$ is given by

$$\boldsymbol{\omega} = \mathbf{T}\boldsymbol{\omega}_B, \tag{2.25}$$

in which $\mathbf{T}$ is the transformation matrix and is computed in Appendix A.1:

$$\mathbf{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}, \tag{2.26}$$

which depends only on the UAV's attitude.

Correspondingly, let $\mathbf{a} \in \mathbb{R}^3$ be the absolute acceleration of the UAV expressed in $\mathcal{F}_W$ and $\mathbf{a}_B \in \mathbb{R}^3$ be the absolute acceleration of the UAV expressed in $\mathcal{F}_B$. So, the relation between $\mathbf{a}$ and $\mathbf{a}_B$ is computed by the analytical derivative of (2.21):

$$\mathbf{a} = \dot{\mathbf{R}}\mathbf{v}_B + \mathbf{R}\mathbf{a}_B. \tag{2.27}$$

Using the Newton-Euler equations about UAV's COM, the dynamical model of the UAV body is the following [109]:

$$\begin{cases} m\dot{\mathbf{v}} & = \mathbf{f}_E \\ \mathbf{I}\dot{\boldsymbol{\omega}}_B & = -\boldsymbol{\omega}_B \times \mathbf{I}\boldsymbol{\omega}_B + \boldsymbol{\tau}_E, \end{cases} \tag{2.28}$$

where m is the UAV mass, $\mathbf{I}$ is the inertia matrix, and $\mathbf{f}_E$ is the vector of external forces and $\boldsymbol{\tau}_E$ is the vector of external torques. The products of inertia can be considered to be 0 due to the symmetries of the system, and $\mathbf{I}$ becomes a diagonal

matrix given by

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}. \tag{2.29}$$

Some calculations yield the following form for $\mathbf{f}_E$ and $\boldsymbol{\tau}_E$:

$$\begin{cases} \mathbf{f}_E &= \begin{bmatrix} (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\,T \\ (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\,T \\ (\cos\phi\cos\theta)\,T - mg \end{bmatrix} \\ \boldsymbol{\tau}_E &= \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}, \end{cases} \tag{2.30}$$

in which g is the gravitational acceleration constant ($g = 9.81\text{m/s}^2$). Finally, using dynamic and kinematic differential equations (2.21), (2.25), (2.28) and (2.30), the following system of non-linear differential equations is obtained:

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{\phi} = \omega_\phi + (\sin\phi\tan\theta)\,\omega_\theta + (\cos\phi\tan\theta)\,\omega_\psi \\ \dot{\theta} = (\cos\phi)\,\omega_\theta - (\sin\phi)\,\omega_\psi \\ \dot{\psi} = \frac{\sin\phi}{\cos\theta}\omega_\theta + \frac{\cos\phi}{\cos\theta}\omega_\psi \\ \dot{v}_x = \frac{1}{m}(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi)\,T \\ \dot{v}_y = \frac{1}{m}(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi)\,T \\ \dot{v}_z = \frac{1}{m}(\cos\phi\cos\theta)\,T - g \\ \dot{\omega}_\phi = \frac{I_y - I_z}{I_x}\omega_\theta\omega_\psi + \frac{1}{I_x}\tau_\phi \\ \dot{\omega}_\theta = \frac{I_z - I_x}{I_y}\omega_\phi\omega_\psi + \frac{1}{I_y}\tau_\theta \\ \dot{\omega}_\psi = \frac{I_x - I_y}{I_z}\omega_\phi\omega_\theta + \frac{1}{I_z}\tau_\psi. \end{cases} \tag{2.31}$$

which can be described in state space form in (2.1) with

$$\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & v_x & v_y & v_z & \omega_\phi & \omega_\theta & \omega_\psi \end{bmatrix}^T, \tag{2.32}$$

$$f(\mathbf{x}) = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_\phi + \sin\phi\tan\theta\omega_\theta + \cos\phi\tan\theta\omega_\psi \\ \cos\phi\omega_\theta - \sin\phi\omega_\psi \\ \frac{\sin\phi}{\cos\theta}\omega_\theta + \frac{\cos\phi}{\cos\theta}\omega_\psi \\ 0 \\ 0 \\ -g \\ \frac{I_y - I_z}{I_x}\omega_\theta\omega_\psi \\ \frac{I_z - I_x}{I_y}\omega_\phi\omega_\psi \\ \frac{I_x - I_y}{I_z}\omega_\phi\omega_\theta \end{bmatrix}, \tag{2.33}$$

$$g(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right) & 0 & 0 & 0 \\ \frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right) & 0 & 0 & 0 \\ \frac{1}{m}\cos\phi\cos\theta & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}, \tag{2.34}$$

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T \tag{2.35}$$

and

$$\mathbf{y} = \mathbf{x}. \tag{2.36}$$

Remark 2.1. If the attitude controller as in [110] is included in the dynamical model, then the virtual control inputs are.

$$\mathbf{u} = \begin{bmatrix} v_z & \phi & \theta & \omega_\psi \end{bmatrix}^T. \tag{2.37}$$

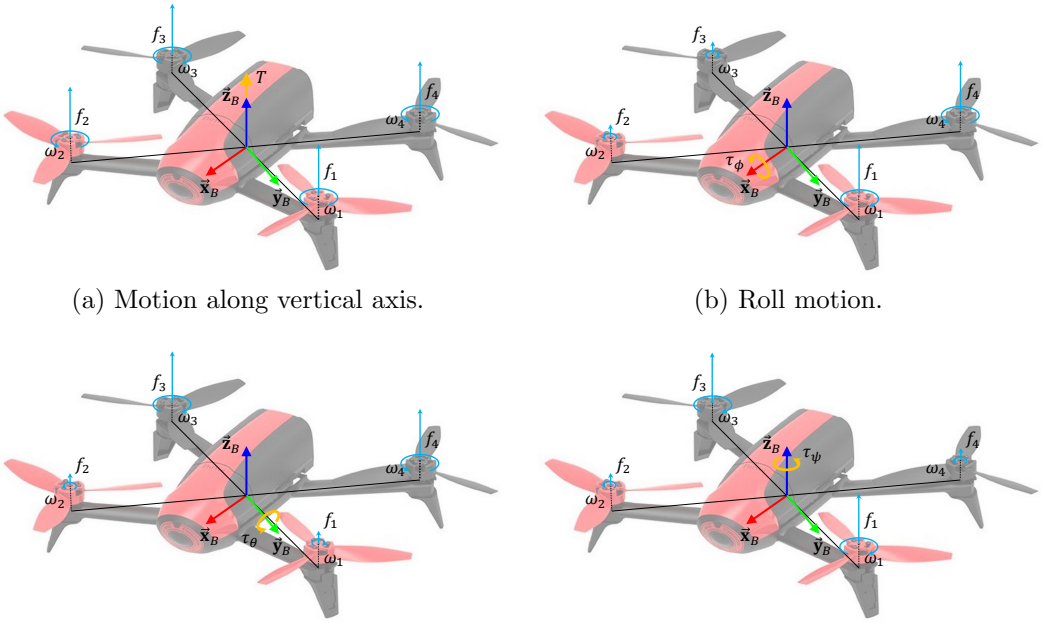
2.2.1.1 X4 Quadcopter

A quadcopter, or quadrotor, shown in Fig. 2.1a, is an aerial vehicle actuated by modulating the speed commands of each of the four motors ($N_M = 4$). It consists of four identical rotors and propellers located at the extremities of an X-shaped frame. In a quadcopter, the two axes $\vec{\mathbf{x}}_B$ and $\vec{\mathbf{y}}_B$ lie in the plane defined by the centres of the four rotors and point between rotor 1 and rotor 2 and between rotor 1 and rotor 4, respectively, as illustrated in Fig. 2.1a. The axis $\vec{\mathbf{z}}_B$ points upward as the direction of the total thrust, defining the body-up configuration.

In a quadcopter, all the movements are the consequence of the propellers' speed (as shown in Fig. 2.2): two propellers rotate in a clockwise direction (the second and the fourth propellers), while the other two rotate in a counter-clockwise direction (the first and the third propellers). Changing simultaneously the throttle of all motors, while the vehicle is horizontal, produces vertical motion (Fig. 2.2a). A difference of speed between the blades on the same axis carries a rotation of the aircraft about the other axis. Roll moment is produced by adjusting the thrust of the left motor with respect to the right one (Fig. 2.2b). Pitch moment is produced similarly by increasing the thrust of the front motor while decreasing that of the rear motor or vice versa (Fig. 2.2c). Yaw moment is slightly more subtle: if the front and rear motors (which spin clockwise) spin faster than the left and right motors (which spin counter-clockwise), yawing results due to the difference in rotor drag moments on the respective motors (Fig. 2.2d). Therefore, the quadcopter is a highly non-linear dynamical system with four control inputs (angular speed of four rotors) and six degrees of freedom (position and orientation in space), resulting in a MIMO under-actuated system.

The relation between $\mathbf{u}$ in (2.19) and ω_i, $i \in \{1, \ldots, 4\}$, is algebraic [111]:

$$\begin{cases} T & = f_1 + f_2 + f_3 + f_4 \\ \tau_\phi & = l \sin \frac{\pi}{4} (f_1 - f_2 - f_3 + f_4) \\ \tau_\theta & = l \cos \frac{\pi}{4} (-f_1 - f_2 + f_3 + f_4) \\ \tau_\psi & = -\tau_1 + \tau_2 - \tau_3 + \tau_4, \end{cases} \tag{2.38}$$



(a) Motion along vertical axis.

(b) Roll motion.

(c) Pitch motion.

(d) Yaw motion.

Figure 2.2: The quadcopter concept. The length of the arrows is proportional to the corresponding forces and torques.

where l is the arm length. Applying (2.18) to (2.38):

$$\begin{cases} T &= b\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right) \\ \tau_\phi &= \frac{\sqrt{2}}{2} bl\left(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2\right) \\ \tau_\theta &= \frac{\sqrt{2}}{2} bl\left(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2\right) \\ \tau_\psi &= d\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right), \end{cases} \tag{2.39}$$

and writing it in matrix form:

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ \frac{\sqrt{2}}{2}bl & -\frac{\sqrt{2}}{2}bl & -\frac{\sqrt{2}}{2}bl & \frac{\sqrt{2}}{2}bl \\ -\frac{\sqrt{2}}{2}bl & -\frac{\sqrt{2}}{2}bl & \frac{\sqrt{2}}{2}bl & \frac{\sqrt{2}}{2}bl \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}. \tag{2.40}$$

The relation in 2.40 is always invertible, when $l \neq 0$, $b \neq 0$ and $d \neq 0$:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \frac{1}{4bdl} \begin{bmatrix} dl & \sqrt{2}d & -\sqrt{2}d & -bl \\ dl & -\sqrt{2}d & -\sqrt{2}d & bl \\ dl & -\sqrt{2}d & \sqrt{2}d & -bl \\ dl & \sqrt{2}d & \sqrt{2}d & bl \end{bmatrix} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}. \tag{2.41}$$

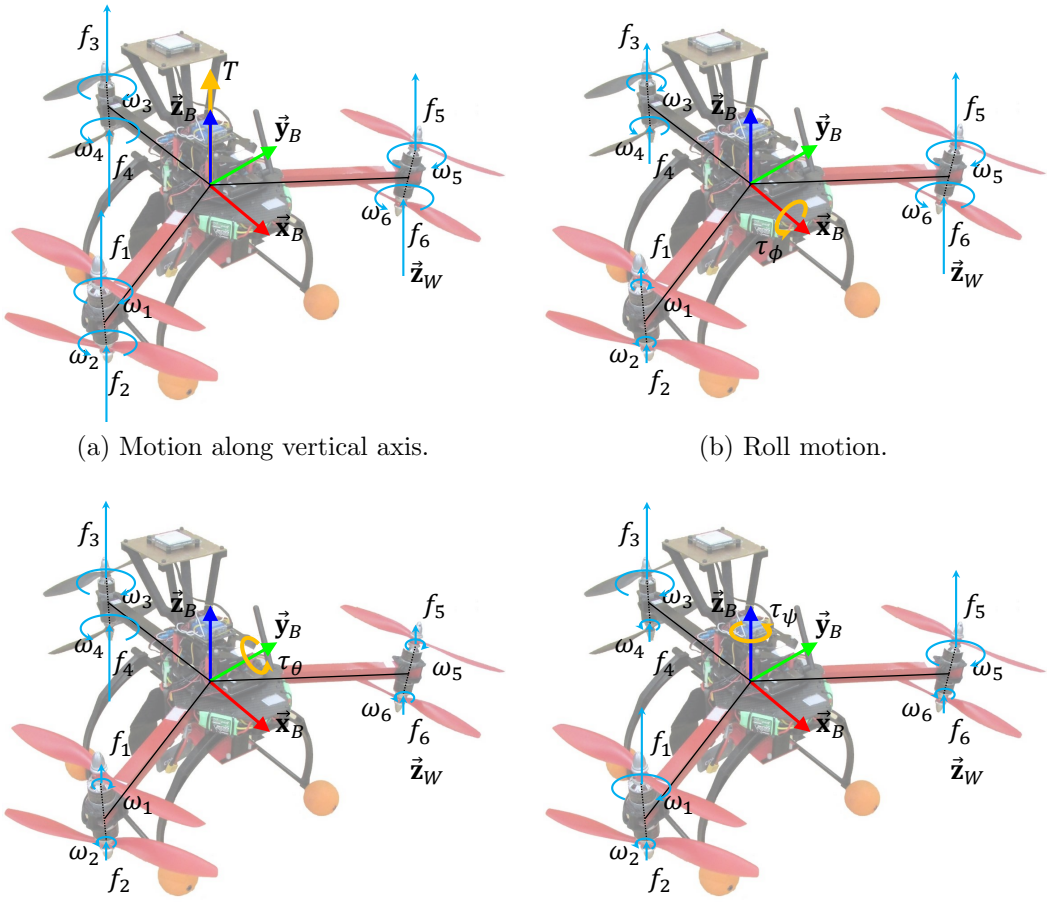
Therefore, the control inputs can be brought back to the speed of the motors.

Remark 2.2. For the dynamical simulations, the model of the quadcopter is implemented in the robot operating system (ROS) and Gazebo simulator. The UAV's position is measured by a simulated global positioning system (GPS); while the UAV's attitude and angular velocities are provided by a simulated inertial measurement unit (IMU).

2.2.1.2 Y6 Coaxial Hexacopter

A Y6 coaxial hexacopter, shown in Fig. 2.1b, is an aerial vehicle actuated by modulating the speed commands of each of the six motors ($N_M = 6$). It consists of six identical rotors and propellers located at the extremities of a Y-shaped frame with two motors per arm (top and bottom). In a coaxial hexacopter, the two axes $\vec{\mathbf{x}}_B$ and $\vec{\mathbf{y}}_B$ lie in the plane defined by the centres of the six rotors, as illustrated in Fig. 2.1b. The axis $\vec{\mathbf{z}}_B$ points upward as the direction of the total thrust, defining the body-up configuration.

In a coaxial hexacopter, as well as in a quadcopter, all the movements are the consequence of the propellers' speed (as shown in Fig. 2.3). The top three propellers rotate in a clockwise direction, while the bottom three rotate in a counter-clockwise direction. Changing simultaneously the throttle of all motors, while the vehicle is horizontal, produces vertical motion (Fig. 2.3a). A difference of speed between the blades on the same axis carries a rotation of the aircraft about the other axis. Roll moment is produced by adjusting the thrust of the left motor with respect to the right one (Fig. 2.3b). Similarly, pitch moment is produced by increasing the thrust of the front motor while decreasing that of the rear motor or vice versa (Fig. 2.3c). Yaw moment is slightly more subtle: if the top motors (which spin clockwise) spin faster than the bottom motors (which spin counter-clockwise), yawing results due to the difference in rotor drag moments on the respective motors (Fig. 2.3d).



(a) Motion along vertical axis.

(b) Roll motion.

(c) Pitch motion.

(d) Yaw motion.

Figure 2.3: The coaxial hexacopter concept. The length of the arrows is proportional to the corresponding forces and torques.

The relation between $\mathbf{u}$ in (2.19) and ω_i, $i \in \{1, \dots, 6\}$, is algebraic:

$$\begin{cases} T &= f_1 + f_2 + f_3 + f_4 + f_5 + f_6 \\ \tau_\phi &= l\left(\sin\frac{\pi}{3}(f_3 + f_4) - \sin\frac{\pi}{3}(f_5 + f_6)\right) \\ \tau_\theta &= l\left(-(f_1 + f_2) + \cos\frac{\pi}{3}(f_3 + f_4 + f_5 + f_6)\right) \\ \tau_\psi &= -\tau_1 + \tau_2 - \tau_3 + \tau_4 - \tau_5 + \tau_6. \end{cases} \tag{2.42}$$

Applying (2.18) to (2.42):

$$\begin{cases} T &= b\left(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2+\omega_5^2+\omega_6^2\right) \\ \tau_\phi &= \frac{\sqrt{3}}{2}bl\left(\omega_3^2+\omega_4^2-\omega_5^2-\omega_6^2\right) \\ \tau_\theta &= \frac{1}{2}bl\left(-2\omega_1^2-2\omega_2^2+\omega_3^2+\omega_4^2+\omega_5^2+\omega_6^2\right) \\ \tau_\psi &= d\left(-\omega_1^2+\omega_2^2-\omega_3^2+\omega_4^2-\omega_5^2+\omega_6^2\right), \end{cases} \tag{2.43}$$

and writing it in matrix form:

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} b & b & b & b & b & b \\ 0 & 0 & \frac{\sqrt{3}}{2}bl & \frac{\sqrt{3}}{2}bl & -\frac{\sqrt{3}}{2}bl & -\frac{\sqrt{3}}{2}bl \\ -bl & -bl & \frac{1}{2}bl & \frac{1}{2}bl & \frac{1}{2}bl & \frac{1}{2}bl \\ -d & d & -d & d & -d & d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ \omega_5^2 \\ \omega_6^2 \end{bmatrix}. \tag{2.44}$$

The relation in (2.44) is always invertible, when $l \neq 0$, $b \neq 0$ and $d \neq 0$. The inverse of (2.44) is

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ \omega_5^2 \\ \omega_6^2 \end{bmatrix} = \frac{1}{6bdl} \begin{bmatrix} dl & 0 & -2d & -bl \\ dl & 0 & -2d & bl \\ dl & \sqrt{3}d & d & -bl \\ dl & \sqrt{3}d & d & bl \\ dl & -\sqrt{3}d & d & -bl \\ dl & -\sqrt{3}d & d & bl \end{bmatrix} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}. \tag{2.45}$$

Therefore, the control inputs can be brought back to the speed of the individual motors.

Remark 2.3. For the dynamical simulations, the model of the coaxial hexacopter is implemented in ROS and Gazebo simulator. The UAV's position is measured by a simulated GPS; while the UAV's attitude and angular velocities are provided by a simulated IMU.

2.2.2 Control Scheme

The overall structure of the closed-loop control scheme for the UAV's dynamical system (2.31) is illustrated in Fig. 2.4. It consists of five main blocks: high-level position controller, mid-level velocity and attitude controllers, low-level motor speed controller, and UAV itself.

2.2.2.1 Position Control

The position controller in Fig. 2.4 consists of three identical and independent sub-controllers for x, y and z axes, as shown in Fig. 2.5. If $\mathbf{p}^* = \begin{bmatrix} x^* & y^* & z^* \end{bmatrix}^T$ is the desired position of the UAV, then the position error is

$$\mathbf{e_p} = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}^T = \mathbf{p}^* - \mathbf{p}. \tag{2.46}$$

The position controller computes the desired linear velocity $\mathbf{v}^* = \begin{bmatrix} v_x^* & v_y^* & v_z^* \end{bmatrix}^T$, in order to reach the desired position $\mathbf{p}^*$ from the current position $\mathbf{p}$. Each sub-controller takes the corresponding position error, as the input, and returns the corresponding control signal, as the output. For the x-axis controller, the input is e_x and the output is v_x^*. For the y-axis controller, the input is e_y and the output is v_y^*. For the z-axis controller, the input is e_z and the output is v_z^*.

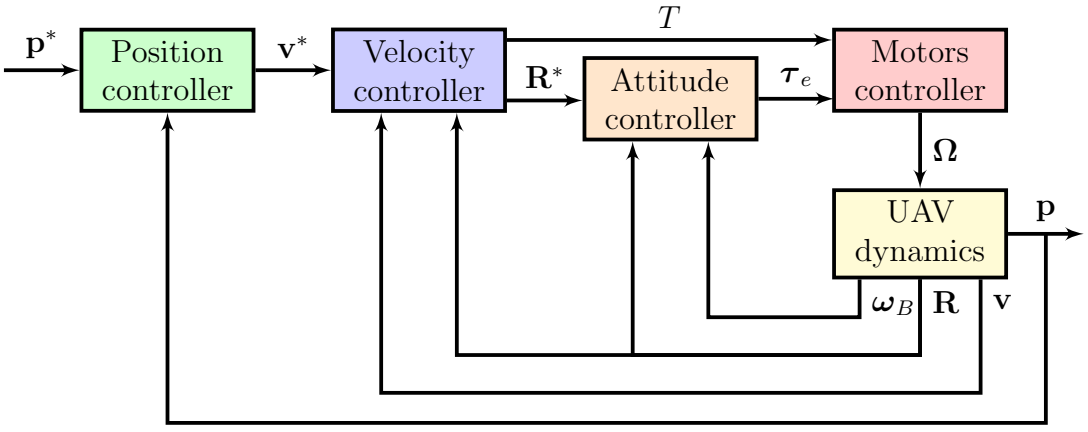


Figure 2.4: Block diagram of the control system for a multicopter UAV.

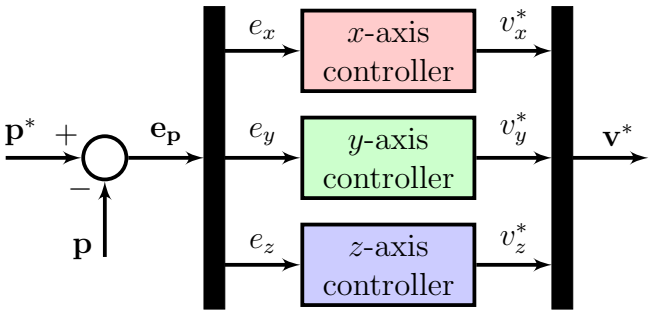


Figure 2.5: Block diagram of the position controller for a multicopter UAV.

2.2.2.2 Velocity Control

For the velocity tracking, the nonlinear geometric controller on the special Euclidean group $\mathsf{SE}(3)$ is used [32]. If $\mathbf{v}^*$ is the vector of desired linear velocities provided by the velocity controller, the velocity error is given by

$$\mathbf{e_v} = \mathbf{v} - \mathbf{v}^*. \tag{2.47}$$

To fly always forward, UAV has to point towards the direction of the movement. Therefore, the desired direction of the first body axis is

$$\vec{\mathbf{x}}_B^* = \frac{1}{\left\| v_x^* \quad v_y^* \quad 0 \right\|} \begin{bmatrix} v_x^* & v_y^* & 0 \end{bmatrix}^T. \tag{2.48}$$

Now, the desired direction of the second and third body axes can be computed:

$$\begin{cases} \vec{\mathbf{z}}_B^* &= \frac{-k_\mathbf{v}\mathbf{e_v} - mg\mathbf{e}_3}{\|-k_\mathbf{v}\mathbf{e_v} - mg\mathbf{e}_3\|} \\ \vec{\mathbf{y}}_B^* &= \frac{\vec{\mathbf{z}}_B^* \times \vec{\mathbf{x}}_B^*}{\|\vec{\mathbf{z}}_B^* \times \vec{\mathbf{x}}_B^*\|}, \end{cases} \tag{2.49}$$

where $k_\mathbf{v}$ is some positive constant and $\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. It can be also assumed that $\vec{\mathbf{x}}_B^* \nparallel \vec{\mathbf{z}}_B^*$. The rotation matrix for the desired attitude $(\vec{\mathbf{x}}_B^*, \vec{\mathbf{y}}_B^*, \vec{\mathbf{z}}_B^*)$ is given by

$$\mathbf{R}^* = \begin{bmatrix} \vec{\mathbf{y}}_B^* \times \vec{\mathbf{z}}_B^* & \vec{\mathbf{y}}_B^* & \vec{\mathbf{z}}_B^* \end{bmatrix} \in \mathsf{SO}(3). \tag{2.50}$$

Finally, the first control input in (2.19) – thrust – is chosen as follows:

$$T = \left(k_\mathbf{v}\mathbf{e_v} + mg\mathbf{e}_3\right)^T \mathbf{R}\mathbf{e}_3, \tag{2.51}$$

where $k_\mathbf{v}$ is some positive constant gain.

2.2.2.3 Attitude Control

For the attitude tracking, the nonlinear geometric controller on the special Euclidean group $\mathsf{SE}(3)$ is used [112]. If $\mathbf{R}^*$ is the desired rotation matrix provided by the velocity controller, the attitude error is given by

$$\mathbf{e}_{\mathbf{R}} = \frac{1}{2}\left[\mathbf{R}^{*T}\mathbf{R} - \mathbf{R}^T\mathbf{R}^*\right]^\vee, \tag{2.52}$$

where $[\cdot]^\vee$ is the vee map: $\mathsf{SO}(3) \to \mathbb{R}^3$, defined in (B.3). Then, the error for the angular velocity is given by

$$\mathbf{e}_{\boldsymbol{\omega}} = \boldsymbol{\omega}_B - \mathbf{R}^T\mathbf{R}^*\boldsymbol{\omega}_B^*, \tag{2.53}$$

where $[\boldsymbol{\omega}_B^*]^\wedge = \mathbf{R}^{*T}\dot{\mathbf{R}}^*$, $[\cdot]^\wedge$ is the hat map: $\mathbb{R}^3 \to \mathsf{SO}(3)$, defined in (B.1), and the derivative of the rotation matrix $\dot{\mathbf{R}}$ is defined in (**??**). Finally, the remaining control inputs in (2.19) are chosen as follows:

$$\boldsymbol{\tau} = -k_{\mathbf{R}}\mathbf{e}_{\mathbf{R}} - k_{\boldsymbol{\omega}}\mathbf{e}_{\boldsymbol{\omega}} + \boldsymbol{\omega}_B \times \mathbf{I}\boldsymbol{\omega}_B, \tag{2.54}$$

where $\boldsymbol{\tau} = \begin{bmatrix}\tau_\phi & \tau_\theta & \tau_\psi\end{bmatrix}^T$ is desired torque, $k_{\mathbf{R}}$ and $k_{\boldsymbol{\omega}}$ are some positive constants.

2.2.2.4 Motors Speed Control

The motor speed controller is a static controller and it maps control inputs in (2.19) computed with (2.51) and (2.54) to the desired motor speed $\boldsymbol{\Omega}$. In case of the quadcopter, $\boldsymbol{\Omega} = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3 & \omega_4\end{bmatrix}^T$ and is computed with (2.41). In case of the coaxial hexacopter, $\boldsymbol{\Omega} = \begin{bmatrix}\omega_1 & \omega_2 & \omega_3 & \omega_4 & \omega_5 & \omega_6\end{bmatrix}^T$ and is computed with (2.45).

2.2.2.5 Real-World Control Scheme

The architecture of the real-time implementation of the control scheme is shown in Fig. 2.6. The trajectory generator provides high-level navigation commands which are interpreted by the position controller as the desired position. Once the desired thrust and attitude are computed, they are sent to the onboard velocity-attitude controller which converts them to the control inputs. Different sensors are used to estimate the actual position of the UAV to feed it back to the position controller.

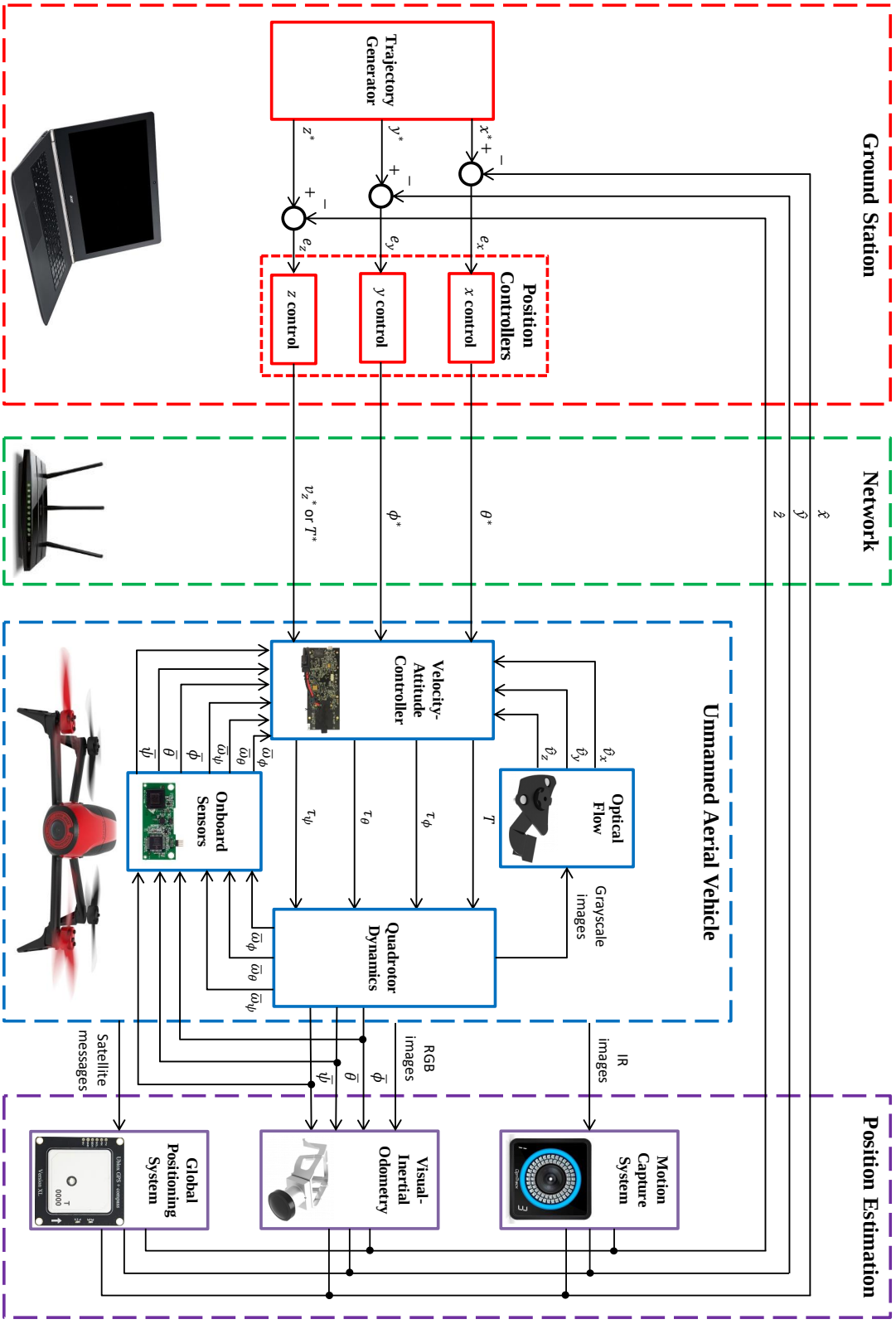


Figure 2.6: Architecture of the real-world implementation of the control scheme for UAV.

Part II

Fuzzy Logic-Based Control

Chapter 3

Type-1 Fuzzy Logic-Based Control

Fuzzy logic is a form of many-valued reasoning paradigm in which the truth values of variables may assume any real number between 0 and 1 [2]. Fuzzy logic is employed to deal with the concept of partial truth by using expert knowledge throughout its design. Thereupon, FLCs are alternative solutions to the model-based controllers without the requirement for the precise mathematical model of the system which is often either unavailable or highly time-consuming to obtain. Hence, FLCs have been extensively used for the control of nonlinear systems, like in (2.1), due to their capability of delivering excellent control in the presence of uncertainties and noise. Generally, T1-FLCs are the most widely used types of FLCs, due to their limited complexity from design and computation perspectives [3].

In this chapter, potentials of different T1-FLCs are explored under various operational conditions. First, Sections 3.1 revises the definition of T1-FLCs. Sections 3.2 and 3.3 present traditional singleton and enhanced non-singleton T1-FLCs, respectively. Then, Sections 3.4 and 3.5 show simulation and experimental results on quadrotor UAV, respectively. Finally, some conclusions are drawn in Section 3.6.

Supplementary Material:
ROS package for the proposed T1-FLCs: github.com/andriyukr/controllers.
Video for the simulation results: tiny.cc/T1-FLC.
Video for the experimental results: tiny.cc/SLAM-FLC.

3.1 Mathematical Preliminaries

Commonly, T1-FLS consists of four elements: fuzzifier, rule-base, inference engine and defuzzifier. All these blocks are interconnected, and Fig. 3.1 shows the general structure of T1-FLS. Generally, T1-FLS can be seen as a mapping from crisp input $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 & \cdots & \sigma_{N_I} \end{bmatrix}^T$, where N_I is the number of crisp inputs, to crisp output $\boldsymbol{\varphi}^{\mathrm{T1}} = \begin{bmatrix} \varphi_1^{\mathrm{T1}} & \cdots & \varphi_{N_O}^{\mathrm{T1}} \end{bmatrix}^T$, where N_O is the number of crisp outputs. Initially, the fuzzifier transforms crisp inputs $\boldsymbol{\sigma}$ into fuzzy input sets $\boldsymbol{\Sigma}^{\mathrm{T1}}$.

Definition 3.1.1. If σ is a crisp input to T1-FLS, a type-1 *fuzzy set* (FS) A is described by a type-1 *membership function* (MF) $\mu_A(\sigma) \in [0,1]$, i.e.:

$$A = \{(\sigma, \mu_A(\sigma)) \mid \mu_A(\sigma) \in [0,1] \quad \forall \sigma \in \mathbb{R}\}. \tag{3.1}$$

In other words, FSs are associated with the fuzzy inputs $\boldsymbol{\Sigma}^{\mathrm{T1}} = \begin{bmatrix} \Sigma_1^{\mathrm{T1}} & \cdots & \Sigma_{N_I}^{\mathrm{T1}} \end{bmatrix}^T$ and fuzzy outputs $\boldsymbol{\Phi}^{\mathrm{T1}} = \begin{bmatrix} \Phi_1^{\mathrm{T1}} & \cdots & \Phi_{N_O}^{\mathrm{T1}} \end{bmatrix}^T$; while MFs μ are used to describe these FSs. An example of three Gaussian type-1 FSs are illustrated in Fig. 3.2.

Remark 3.1. If MFs in (3.1) assume only 0 or 1, i.e., $\mu(\sigma) \in \{0,1\}$; then, type-1 FSs degenerate into singleton FSs.

Three singleton FSs are illustrated in Fig. 3.3. Based on how the inputs are handled in the fuzzifier, two types of fuzzification exist: singleton and non-singleton.

Definition 3.1.2. If the input to FLS is type-0, i.e., crisp input, then FLS is called *singleton* FLS (SFLS).

Definition 3.1.3. If the input to FLS is type-1 FS, then FLS is called *non-singleton* FLS (NSFLS).

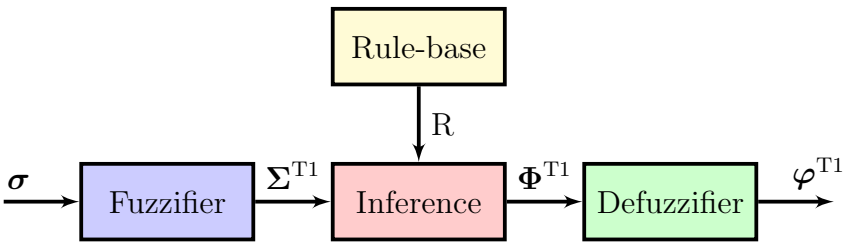


Figure 3.1: Structure of the type-1 fuzzy logic system.

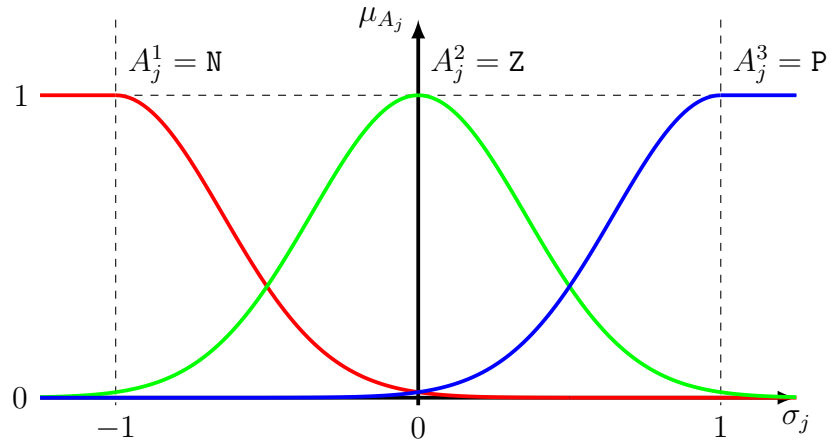


Figure 3.2: "Negative" (N), "zero" (Z) and "positive" (P) FSs represented by three Gaussian type-1 MFs.

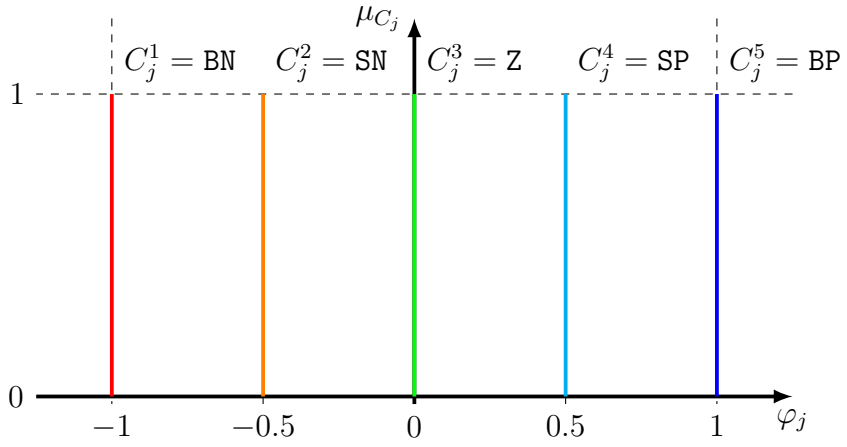


Figure 3.3: "Big negative" (BN), "small negative" (SN), "zero" (Z), "small positive" (SP) and "big positive" (BP) FSs represented by five singleton MFs.

The rule-base R is the core of any FLS. Every single rule $R_i \in \mathrm{R}$ can be expressed as an IF − THEN statement. The IF-part is the antecedent, while the THEN-part is the consequent.

Definition 3.1.4. If N_R is the number of rules in a rule-base R, then the i^{th} *rule* $R_i \in \mathrm{R}$, $i \in \{1, \ldots, N_R\}$, is indicated as IF − THEN statement, i.e.:

$$R_i: \begin{array}{l} \text{IF } \sigma_1 \text{ is } A_{1,i} \text{ and } \ldots \text{ and } \sigma_{N_I} \text{ is } A_{N_I,i}, \\ \text{THEN } \varphi_1 \text{ is } C_{1,i} \text{ and } \ldots \text{ and } \varphi_{N_O} \text{ is } C_{N_O,i} \end{array}, \quad i \in \{1, \ldots, N_R\}, \tag{3.2}$$

where $A_{j,i}$, $j \in \{1, \ldots, N_I\}$, represents antecedent FS; while $C_{h,i}$, $h \in \{1, \ldots, N_O\}$, represents consequent FS.

A typical 9 and 27 rules rule-bases are shown in Table 3.1 [113] and Table 3.2 [114], respectively.

Remark 3.2. It has to be emphasised that A_j^k, $j \in \{1,\ldots,N_I\}, k \in \{1,\ldots,N_F\}$, represents the k^{th} antecedent FS of the j^{th} input; while, $A_{j,i}$, $j \in \{1,\ldots,N_I\}, i \in \{1,\ldots,N_R\}$, represents the antecedent FS of the j^{th} input in the i^{th} rule. The same convention is used for the consequent FSs, C_j^k, $j \in \{1,\ldots,N_I\}, k \in \{1,\ldots,N_F\}$, represents the k^{th} consequent FS of the j^{th} output; while, $C_{h,i}$, $h \in \{1,\ldots,N_O\}, i \in \{1,\ldots,N_R\}$, represents the consequent FS of the h^{th} output in the i^{th} rule.

Once crisp inputs are fuzzified, FSs activate the inference engine. The inference engine implies type-1 fuzzy output sets $\boldsymbol{\Phi}^{\text{T1}}$ from type-1 fuzzy input sets $\boldsymbol{\Sigma}^{\text{T1}}$.

Definition 3.1.5. The *firing strength* $f_i(\boldsymbol{\sigma}) \in [0,1]$, $i \in \{1,\ldots,N_R\}$, of the i^{th} rule can be computed with the product t-norm, i.e.:

$$f_i(\boldsymbol{\sigma}) = \prod_{j=1}^{N_I} \mu_{A_{j,i}}(\sigma_j), \quad i \in \{1,\ldots,N_R\}. \tag{3.3}$$

Table 3.1: A typical 9 rules rule-base of FLC.

σ_1	σ_2		
	N	Z	P
N	R_1 : BN	R_2 : BN	R_3 : Z
Z	R_4 : BN	R_5 : Z	R_6 : BP
P	R_7 : Z	R_8 : BP	R_9 : BP

Table 3.2: A typical 27 rules rule-base of FLC.

σ_1	σ_2	σ_3		
		N	Z	P
N	N	R_1 : BN	R_2 : BN	R_3 : SN
	Z	R_4 : BN	R_5 : SN	R_6 : Z
	P	R_7 : SN	R_8 : Z	R_9 : SP
Z	N	R_{10} : BN	R_{11} : SN	R_{12} : Z
	Z	R_{13} : SN	R_{14} : Z	R_{15} : SP
	P	R_{16} : Z	R_{17} : SP	R_{18} : BP
P	N	R_{19} : SN	R_{20} : Z	R_{21} : SP
	Z	R_{22} : Z	R_{23} : SP	R_{24} : BP
	P	R_{25} : SP	R_{26} : BP	R_{27} : BP

Remark 3.3. From Definitions 3.1.1 and 3.1.5, it can be seen that $f_i \in [0, 1]$, $i \in \{1, \ldots, N_R\}$.

In other words, the inference engine manages which rules are fired. Finally, the output of FLCs must be crisp numbers. This is accomplished by the defuzzification.

Definition 3.1.6. The h^{th} *defuzzified output*, $h \in \{1, \ldots, N_O\}$, can be computed with the centroid defuzzification, i.e.:

$$\varphi_h^{\text{T1}}(\boldsymbol{\sigma}) = \frac{\sum_{i=1}^{N_R} f_i(\boldsymbol{\sigma}) C_{h,i}}{\sum_{i=1}^{N_R} f_i(\boldsymbol{\sigma})}, \quad h \in \{1, \ldots, N_O\}. \tag{3.4}$$

Remark 3.4. By substituting (3.3) into (3.4), the h^{th} defuzzified output, $h \in \{1, \ldots, N_O\}$, can be computed as follows:

$$\varphi_h^{\text{T1}}(\boldsymbol{\sigma}) = \frac{\sum_{i=1}^{N_R} \left(\prod_{j=1}^{N_I} \mu_{A_{j,i}}(\sigma_j) \right) C_{h,i}}{\sum_{i=1}^{N_R} \left(\prod_{j=1}^{N_I} \mu_{A_{j,i}}(\sigma_j) \right)}, \quad h \in \{1, \ldots, N_O\}. \tag{3.5}$$

The structure of a triple-input type-1 fuzzy PID controller, which inherits triple-input T1-FLS, is shown in Fig. 3.4. The input scaling factors k_p, k_i and k_d are chosen to normalize e, $\int e$ and $\dot{e}$ to the universe of discourse of the antecedent MFs, i.e., $[-1, 1]$. So, e, $\int e$ and $\dot{e}$ are transformed into σ_1, σ_2 and σ_3, respectively, before inputting them to triple-input T1-FLS. Consequently, the output φ^{T1} from triple-input T1-FLS is transformed into the control signal u by an unscaling gain k_o such that the output is denormalized to the domain of the control signal. In the adopted control structure, only one parameter has to be tuned, i.e., k_o.

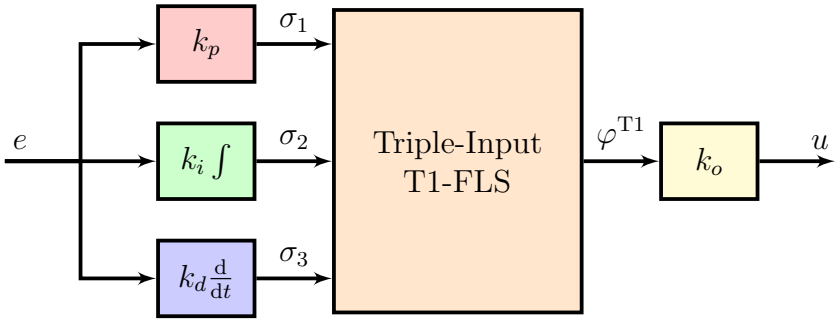


Figure 3.4: Structure of a triple-input type-1 fuzzy PID controller.

3.2 Singleton Fuzzy Logic Control

The input to SFLS is a singleton or crisp input:

$$\mu_\sigma(\sigma_j) = \begin{cases} 1, & \sigma_j = c_j \\ 0, & \sigma_j \neq c_j, \end{cases} \tag{3.6}$$

where c_j is the center of j^{th} input MF. Namely, in SFLS, the fuzzifier maps a crisp input σ_j into a fuzzy set Σ_j with support c_j, as shown in Fig. 3.5a.

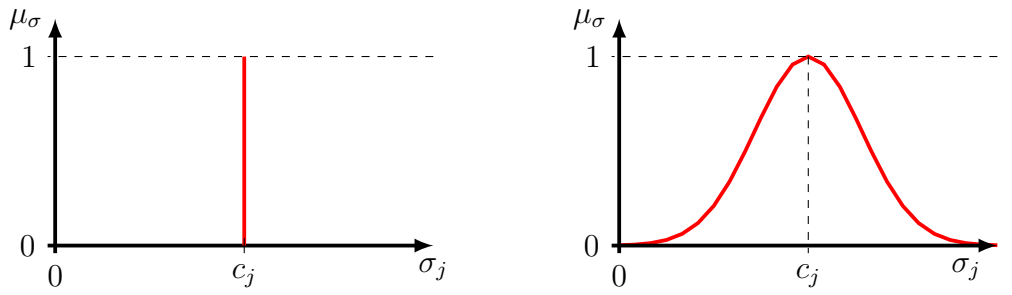
Remark 3.5. The fuzzifier of the SFLS does not model any vagueness for the input. Therefore, it does not make full use of the modelling capability.

3.3 Non-Singleton Fuzzy Logic Control

On the other hand, NSFLC is T1-FLS whose inputs are modelled as type-1 FSs by prefiltering unit, as shown in Fig. 3.6. Namely, NSFLS can be used to cope better with noisy, imprecise or inaccurate input measurements. In NSFLS, the prefilter maps a crisp input σ_j into MF μ_σ, e.g., Gaussian MF, as shown in Fig. 3.5b:

$$\mu_\sigma(\sigma_j) = \exp\left[-\frac{(\sigma_j - c_j)^2}{2d_j^2}\right], \tag{3.7}$$

where d_j is the standard deviation of the j^{th} FS.



(a) Singleton prefiltering. (b) Non-singleton Gaussian prefiltering.

Figure 3.5: Singleton and non-singleton prefiltering.

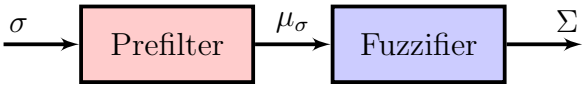


Figure 3.6: Fuzzification in NSFLS.

Remark 3.6. The non-singleton fuzzifier implies that the given input value c_j is the most likely value to be correct one. Moreover, larger values of the spread d_j represent that more uncertainties, e.g., noise, are inherent within the input data.

In the literature, NSFLSs are categorised into three types, based on the type of the prefilter:

i) NSFLC with standard prefilter [115], i.e., Sta-NSFLS;

ii) NSFLC with centroid-based prefilter [62], i.e., Cen-NSFLS;

iii) NSFLC with similarity-based prefilter [63], i.e., Sim-NSFLS.

Fig. 3.7 shows the differences among various prefilters.

3.3.1 Standard Non-Singleton Fuzzy Logic Control

In Sta-NSFLS, for calculating the input FS, the maximum of the intersection between prefiltered input and antecedent MF is utilised:

$$\mu_{sta}(\sigma) = \sup \min \left(\mu_\sigma(\sigma), \mu_A(\sigma)\right). \tag{3.8}$$

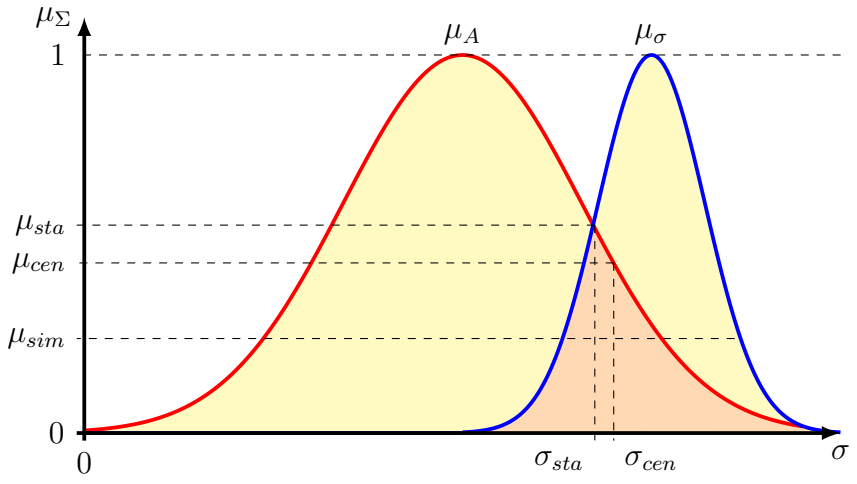


Figure 3.7: Examples of NSFLS prefiltering with standard, centroid-based and similarity-based prefilters.

3.3.2 Centroid Non-Singleton Fuzzy Logic Control

In Fig. 3.8, two different prefiltered inputs are shown which are intersected with an antecedent MF. Although the actual prefiltered inputs are different, the input FSs calculated by the standard approach are the same in both cases. Thus, two different inputs, or more specifically inputs with a different associated uncertainty distribution, result in the same input FS and, thus, the same output from FLS. Hence, a method with a more detailed capture of input uncertainty and its intersection with the respective antecedent FS is desirable, i.e., a new method should have a higher sensitivity to the shape of the intersection. Fig. 3.8 shows how in Sta-NSFLSs the maximum of the intersection between different prefiltered inputs and antecedent MF result in the same σ_{sta} and, thus, the same input FS. However, in Cen-NSFLS, $\sigma_{1,cen}$ and $\sigma_{2,cen}$ are different when the centroid of each intersection has been applied instead of their maximum. In Cen-NSFLS, for calculating the input FS, the centroid of the intersection between the fuzzified input and antecedent FS is used rather than the maximum of the intersection utilized in Sta-NSFLSs:

$$\sigma_{cen} = \frac{\int \sigma \mu_\sigma(\sigma) \mathrm{d}\sigma}{\int \mu_\sigma(\sigma) \mathrm{d}\sigma}. \tag{3.9}$$

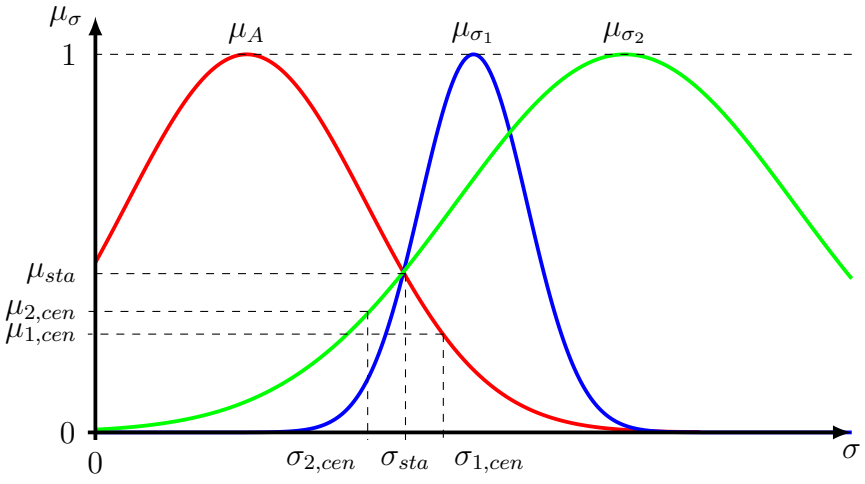


Figure 3.8: Difference of NSFLS prefiltering with standard and centroid-based prefilters.

3.3.3 Similarity Non-Singleton Fuzzy Logic Control

In Fig. 3.9, intersections of two different prefiltered inputs with an antecedent FS are shown. Although, these two fuzzified inputs have two different associated uncertainty distributions, the standard and centroid-based fuzzification for both σ_1 and σ_2 are the same, i.e, $\mu_{\sigma_1}(\sigma_{1,cen}) \equiv \mu_{\sigma_2}(\sigma_{2,cen}) \equiv \mu_{\sigma_1}(\sigma_{1,sta}) \equiv \mu_{\sigma_2}(\sigma_{2,sta})$. Hence, a new NSFLS, which is more sensitive to the input uncertainty, is desirable. In [63], a novel NSFLS with the similarity-based inference engine, i.e., Sim-NSFLS, was presented and used for the well-known problem of Mackey-Glass time series predictions. The results showed that Sim-NSFLS outperformed Sta-NSFLS and Cen-NSFLS under different noise conditions. As shown in Fig. 3.9, $\mu_{\sigma_{1,sim}}$ and $\mu_{\sigma_{2,sim}}$ are two different MFs for two different inputs based on the similarity-based approach. Considering the prefiltered input σ and antecedent FS A with MFs $\mu_\sigma(\sigma)$ and $\mu_A(\sigma)$, respectively, the similarity between σ and A is defined based on the Jaccard similarity:

$$\mu_{sim} = \frac{\int \min(\mu_\sigma(\sigma), \mu_A(\sigma))\mathrm{d}\sigma}{\int \max(\mu_\sigma(\sigma), \mu_A(\sigma))\mathrm{d}\sigma}. \tag{3.10}$$

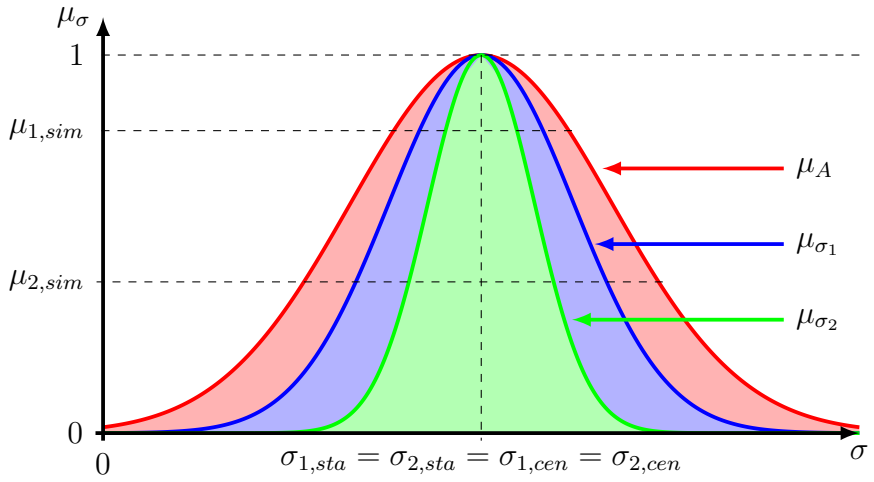


Figure 3.9: Difference of NSFLS prefiltering with standard, centroid-based and similarity-based prefilters.

3.4 Simulation Results

The 3D trajectory is defined according to the minimum snap property [116] which enables the real-time generation of an optimal trajectory through a sequence of 3D positions, thereby ensuring safe passage through specified environments as well as maintaining the constraints on accelerations and velocities. Some manoeuvrable flights were generated, e.g., descending and climbing straight lines as well as curves, the sharp turns between the straight lines and curves, to test the control performance of each NSFLC controller. It is noted that the generated 3D trajectory is sent to all NSFLCs at the same time.

Different Gaussian distributions are employed in (3.7) for the input and output MFs of FLC. Each input variable, i.e., position error $\mathbf{e_p}$ in (2.46), its derivative $\dot{\mathbf{e}}_\mathbf{p}$ and its integral $\int \mathbf{e_p}$, has three MFs, depicted in Fig. 3.2; while the output variable, i.e., control signal $\mathbf{u}$, has five MFs, depicted in Fig. 3.3. The rule-base of FLC is defined based on the expert experience, and the rules are summarized in Table 3.2.

3.4.1 Sources of Uncertainties

In order to produce the noisy measurements $\bar{\mathbf{p}} = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} \end{bmatrix}^T$, the white Gaussian noise is added to the true position:

$$\begin{cases} \bar{x} = \mathcal{N}(x, \sigma_N^2) \\ \bar{y} = \mathcal{N}(y, \sigma_N^2) \\ \bar{z} = \mathcal{N}(z, \sigma_N^2), \end{cases} \tag{3.11}$$

where σ_N is the standard deviation of the position noise and $\mathcal{N}(\mu, \sigma^2)$ is the Gaussian distribution with mean μ and variance σ^2.

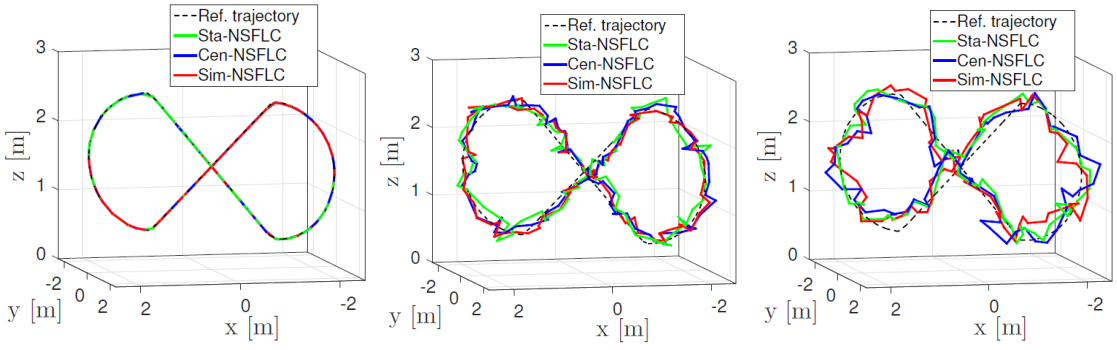
3.4.2 Discussion

The control performance evaluation is carried out in terms of the mean squared error (MSE) of the 3D position:

$$\text{MSE} = \frac{1}{N_D} \sum_{i=1}^{N_D} \|\mathbf{p}_i^* - \mathbf{p}_i\|^2 , \tag{3.12}$$

where N_D is the number of data samples, $\mathbf{p}_i^* = \begin{bmatrix} x_i^* & y_i^* & z_i^* \end{bmatrix}^T$ and $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ are the desired and actual positions for the i-th sample, respectively.

To test and evaluate the quadrotor UAV control performances, six levels of noise ($\sigma_N = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$) and three NSFLCs (Sta-NSFLC, Cen-NSFLC and Sim-NSFLS) with five input fuzzifications ($\sigma_F = \{0.2, 0.4, 0.6, 0.8, 1.0\}$), i.e., different standard deviations for input MFs, are provided. Fig. 3.10 shows the example of three UAV flights with the same level of fuzzifier ($\sigma_F = 1.0$) under three different levels of noise ($\sigma_N = 0.0$, $\sigma_N = 0.5$ and $\sigma_N = 1.0$). As can be seen from Fig. 3.11, Cen-NSFLCs outperform Sta-NSFLCs, and the control performances of Sim-NSFLCs are better than both Cen-NSFLCs and Sta-NSFLCs. In addition, the larger values of the σ_F for the fuzzifier can assist the NSFLCS to achieve better performances. Fig. 3.11 also clearly shows the control performance differences among Sta-NSFLC, Cen-NSFLC and Sim-NSFLC.



(a) $\sigma_N = 0.0$ (b) $\sigma_N = 0.5$ (c) $\sigma_N = 1.0$

Figure 3.10: Trajectory tracking under three different levels of noise ($\sigma_N = 0.0$, $\sigma_N = 0.5$ and $\sigma_N = 1.0$) with the same level of fuzzifier ($\sigma_F = 1.0$).

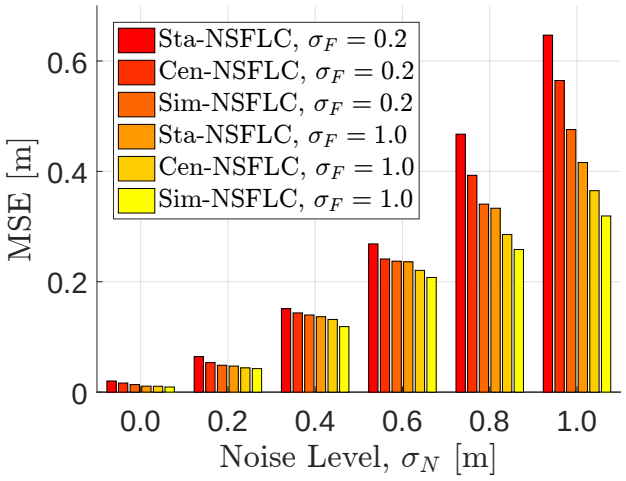


Figure 3.11: Control performances of Sta-NSFLC, Cen-NSFLC and Sim-NSFLC.

3.5 Experimental Results

Real-world quadrotor UAV flight experiments are conducted and evaluated in the OptiTrack motion capture system laboratory at Nanyang Technological University, Singapore. The OptiTrack system can provide real-time rigid body position measurement, i.e., ground truth, in a three-dimensional space with an update rate of 240Hz and accuracy of 0.1mm. All the controllers, i.e., conventional PID, SFLC, Tra-NSFLC, and Cen-NSFLC, are developed in C++ and integrated into ROS. To evaluate different levels of input uncertainty affecting the control inputs of the quadrotor UAV, four different types of flight experiments with different flight speeds are carried out:

- Test 1: hovering at fixed position ($\begin{bmatrix}0 & 0 & 1\end{bmatrix}$ m);
- Test 2: following an eight-shaped trajectory at 1.0m/s;
- Test 3: following an eight-shaped trajectory at 1.5m/s;
- Test 4: following an eight-shaped trajectory at 2.0m/s.

Remark 3.7. In Test 1, the UAV flight speed can be considered as 0m/s.

To evaluate the robustness of each controller, the trajectory is generated using the minimum snap property and several flight manoeuvres, such as ascending and descending straight lines as well as curves. To evaluate the performances, all

controllers were designed and iteratively tuned, with an emphasis on investing equal amounts of design effort for each controller. The flight data collected from one hundred experiments are analysed.

Remark 3.8. For the localisation of UAV, the monocular keyframe-based visual-inertial SLAM algorithm is used. The monocular visual-inertial SLAM produces noisy odometry because of the motion blure at higher flight speeds [117].

3.5.1 Monocular Visual-Inertial SLAM Performance

The relationship between the flight speed and uncertainty level is shown in Fig. 3.12. To evaluate the monocular keyframe-based visual-inertial SLAM performance, the root mean squared error (RMSE) between the ground truth and the SLAM estimation, i.e., RMSE_{GE}, is used. In Fig. 3.12, the average SLAM performance results are shown with different UAV flight speeds. As can be seen from Fig. 3.12, the SLAM algorithm obtains the best position estimation result during the UAV hovering flight tests. As the UAV flight speed is increasing, the position estimation accuracy is decreasing. The average RMSE_{GE}s for Tests 2, 3 and 4 have increased by 13.9cm, 16.0cm and 19.9cm when compared to the UAV hovering flight tests. Fig. 3.12 also shows that increased flight speed results in higher position input uncertainty. Similarly, experiments showed that variation amount in illumination, reduction in detected features rotation and translation speeds are all proportionally correlated to increasing uncertainty/noise levels in the position estimation inputs.

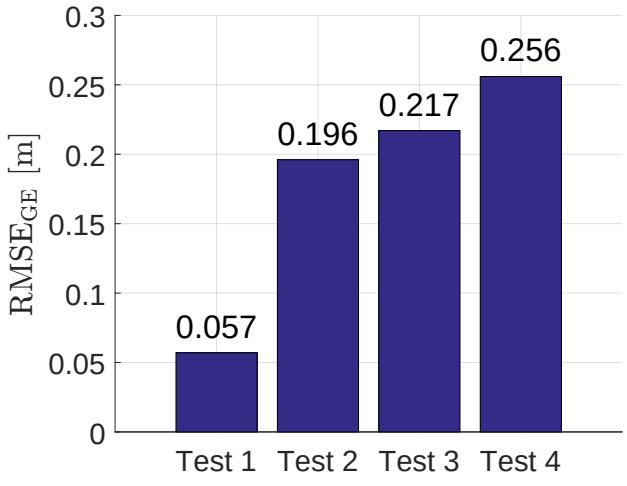
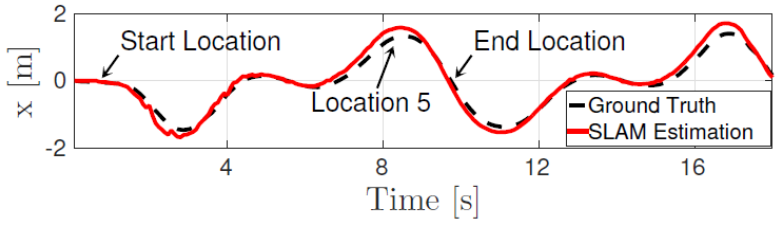


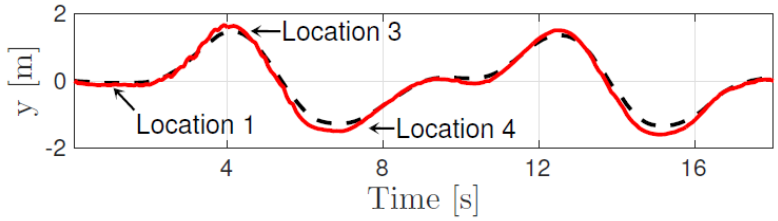
Figure 3.12: SLAM performances under different UAV flight speeds.

For example, the quadrotor UAV during the hovering flights always looks at the same scene, i.e., same illumination and number of detected features, without capturing the blurred image frames.

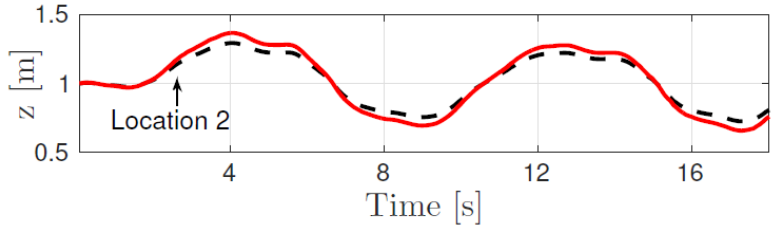
Fig. 3.13 shows x, y and z translation estimations (red colour) of two rounds of the trajectory following application which is controlled by the Cen-NSFLC with a maximum flight speed of 2.0m/s. The ground truths (black colour) of the x, y and z translations from the OptiTrack system are used for performance comparisons. As can be seen from Fig. 3.13, although faster flights result in more challenging pose estimations in the monocular keyframe-based visual-inertial SLAM, the pose estimations can match the ground truths fairly well. Therefore, the SLAM estimations are accurate enough to be used as the control inputs in the longterm navigation of the real-world quadrotor UAV.



(a) x translation with maximum flight speed 2.0m/s



(b) y translation with maximum flight speed 2.0m/s



(c) z translation with maximum flight speed 2.0m/s

Figure 3.13: SLAM results with maximum flight speed 2.0m/s.

3.5.2 Discussion

The control performances of PID, SFLC, Sta-NSFLC and Cen-NSFLC are evaluated based on the RMSE between the ground truth and the reference trajectory, i.e., RMSE_{GR}. Fig. 3.14 shows the control performance results, i.e., average RMSE_{GR}s calculated from one hundred flight tests. From Fig. 3.14, it can be observed that the Cen-NSFLC consistently has the best performance across all speed levels. The control performances of the FLCs are better than those of the conventional PID controller. On the other hand, NSFLCs can obtain superior control performance compared to the SFLC, while Cen-NSFLC outperforms Sta-NSFLC.

Figs. 3.15 and 3.16 show the control performances of all the controllers in one round of the trajectory following application with the maximum flight speed of 2.0m/s. As can be seen from these three figures, all the controllers can navigate the quadrotor UAVs to follow the online generated trajectory, but the control performance ranking is Cen-NSFLC, Sta-NSFLC, SLFC, and PID controller. Although the Euclidean errors of SFLC and Sta-NSFLC in some parts of the trajectory are less than the one of Cen-NSFLC, the overall control performance of Cen-NSFLC is better than the ones of SFLC and Sta-NSFLC. Meantime, Cen-NSFLC outperforms PID controller during all parts of the trajectory following.

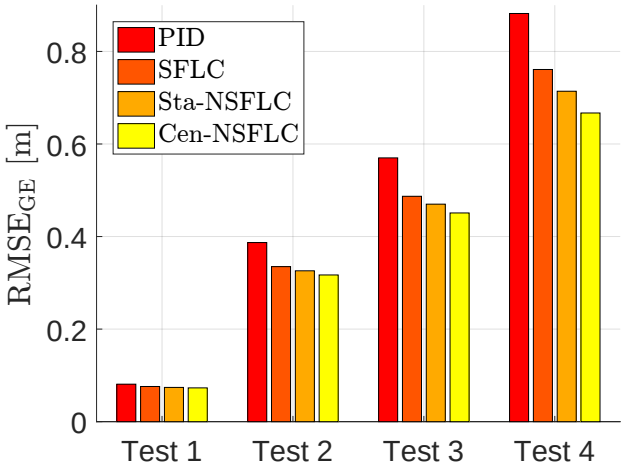
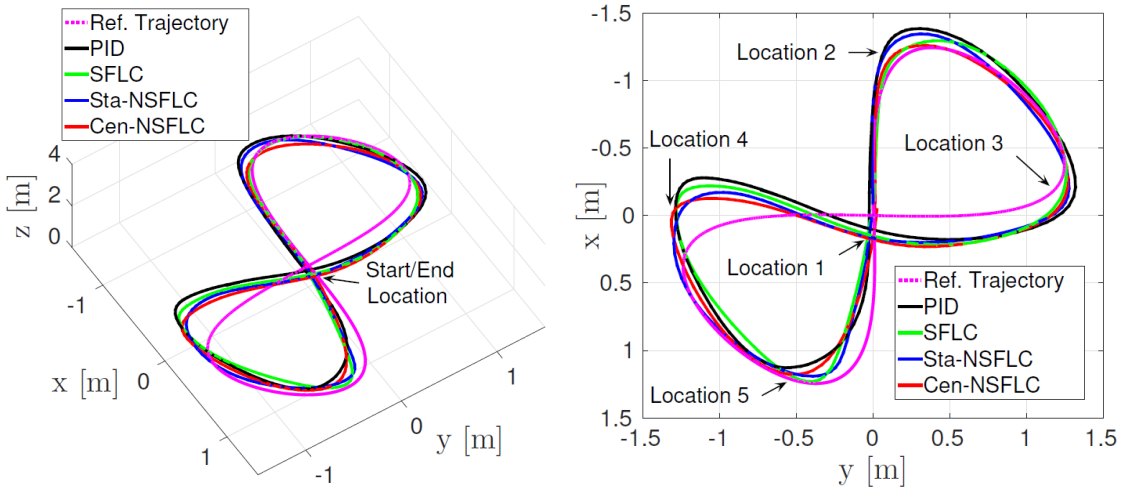


Figure 3.14: Control performances of four controllers (PID, SFLC, Sta-NSFLC and Cen-NSFLC) in four different scenarios.



(a) 3D view (b) 2D xy-view

Figure 3.15: Trajectory following performances of all controllers in Test 4.

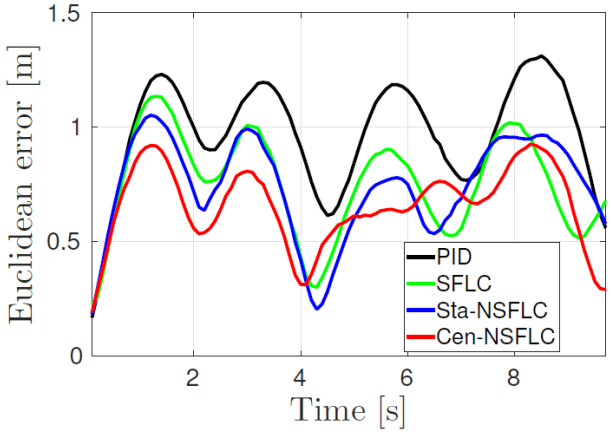


Figure 3.16: Euclidean error evolution of all controllers in Test 4.

3.6 Conclusion

In this chapter, two novel NSFLC with center-based and similarity-based prefiltering were developed and deployed to control a quadcopter UAV for the 3D trajectory tracking application. A comprehensive comparison and evaluation were carried out with three different types of NSFLCs, i.e., Sta-NSFLC, Cen-NSFLC and Sim-NSFLC, under different levels of input uncertainty, i.e., measurement noise. Extensive simulation and experimental tests show that Sim-NSFLC can obtain better control performances compared to Sta-NSFLC and Cen-NSFLC, especially at the higher input noise levels. Moreover, the higher input fuzzification has more capability to a handle higher level of input noise.

Chapter 4

Interval Type-2 Fuzzy Logic-Based Control

Though T1-FLCs are widely used, type-1 FSs are able to deal effectively only with bounded levels of uncertainty, while real-world applications frequently have to deal with high levels and multiple sources of uncertainties [4, 118]. Therefore, there has been a growing interest in a more advanced form of FLCs, namely T2-FLCs [65]. Better handling of the uncertainties using T2-FLCs is provided by the additional degree of freedom benefiting from FOU in their FSs [6, 119]. However, the additional complexity arises from the inclusion of FOU as well as the third dimension [7]. Therefore, the research has tended to focus on IT2-FLCs [8], rather than on general T2-FLCs [9], because the mathematical formulation of general T2-FLCs is much more complex than that of IT2-FLCs [10, 74]. The adoption of IT2-FLC allows reducing the computational complexity which is an immense benefit in real-time applications [11].

In this chapter, potentials of different IT2-FLCs are explored under various operational conditions. First, Sections 4.1 revises the definition of IT2-FLCs. Then, Sections 4.2 shows experimental results on quadrotor UAV. Finally, some conclusions are drawn in Section 4.3.

Supplementary Material:
ROS package for the proposed IT2-FLCs: github.com/andriyukr/controllers.

4.1 Mathematical Preliminaries

Commonly, T2-FLS consists of five elements: fuzzifier, rule base, inference engine, type reducer and defuzzifier. All these components are interconnected, and Fig. 4.1 shows the general structure of T2-FLS. Similarly to T1-FLS, T2-FLS can be seen as a mapping from the crisp input $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 & \cdots & \sigma_{N_I} \end{bmatrix}^T$, where N_I is the number of crisp inputs, to the crisp output $\boldsymbol{\varphi}^{\mathrm{T2}} = \begin{bmatrix} \varphi_1^{\mathrm{T2}} & \cdots & \varphi_{N_O}^{\mathrm{T2}} \end{bmatrix}^T$, where N_O is the number of crisp outputs. Initially, the fuzzifier transforms crisp inputs $\boldsymbol{\sigma}$ into fuzzy input sets $\boldsymbol{\Sigma}^{\mathrm{T2}}$.

Definition 4.1.1. If σ is a crisp input to T2-FLS, and $U_{\tilde{A}}(\sigma)$ is the universe of the secondary variable υ, a *type-2 FS* $\tilde{A}$ is described by a type-2 MF, $\mu_{\tilde{A}}(\sigma, u) \in [0, 1]$, where $\sigma \in \mathbb{R}$ and $\upsilon \in U_{\tilde{A}}(\sigma) \subseteq [0, 1]$, i.e.:

$$\tilde{A} = \{(\sigma, \upsilon, \mu_{\tilde{A}}(\sigma, \upsilon)) \mid \mu_{\tilde{A}}(\sigma, \upsilon) \in [0, 1] \quad \forall \sigma \in \mathbb{R} \quad \forall \upsilon \in U_{\tilde{A}}(\sigma) \subseteq [0, 1]\}. \tag{4.1}$$

Definition 4.1.2. If σ is a crisp input to IT2-FLS, and $U_{\tilde{A}}(\sigma)$ is the universe of the secondary variable υ, an *interval type-2 FS* $\tilde{A}$ is described by an interval type-2 MF, $\mu_{\tilde{A}}(\sigma, \upsilon) = 1$, where $\sigma \in \mathbb{R}$ and $\upsilon \in U_{\tilde{A}}(\sigma) \subseteq [0, 1]$, i.e.:

$$\tilde{A} = \{(\sigma, \upsilon, 1) \quad \forall \sigma \in \mathbb{R} \quad \forall \upsilon \in U_{\tilde{A}}(\sigma) \subseteq [0, 1]\}. \tag{4.2}$$

In other words, in IT2-FSs, all the third dimension values are equal to one. An example of three elliptic interval type-2 FSs are illustrated in Fig. 4.2.

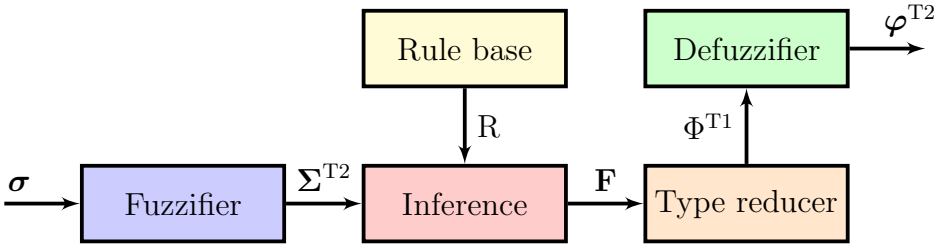


Figure 4.1: Structure of the type-2 fuzzy logic system.

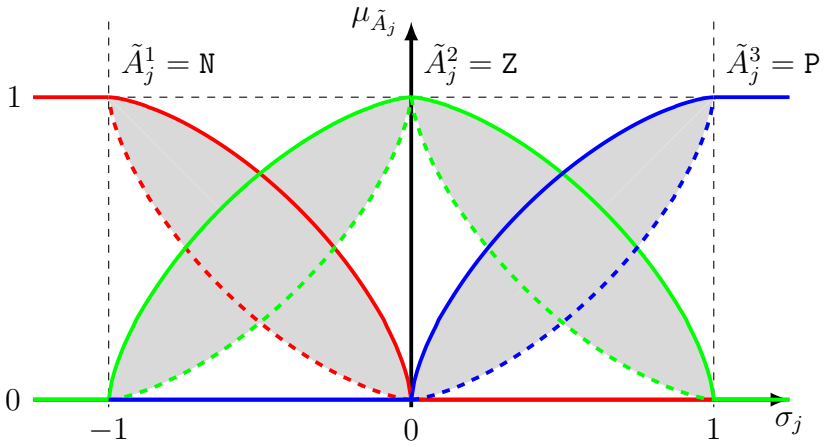


Figure 4.2: "Negative" (N), "zero" (Z) and "positive" (P) FSs represented by three elliptic interval type-2 MFs.

Definition 4.1.3. If σ is a crisp input to T2-FLS, *FOU* of FS $\tilde{A}$ is a bounded region (grey areas in Fig. 4.2) which is defined by the union of all $\mu_{\tilde{A}}(\sigma, \upsilon)$, i.e.:

$$\mathrm{FOU}(\tilde{A}) = \bigcup_{\sigma \in \mathbb{R}} U_{\tilde{A}}(\sigma). \tag{4.3}$$

Definition 4.1.4. If σ is a crisp input to T2-FLS, the *upper MF* $\overline{\mu}_{\tilde{A}}(\sigma)$ is MF which confines from top $\mathrm{FOU}(\tilde{A})$ (coloured solid lines in Fig. 4.2); while the *lower MF* $\underline{\mu}_{\tilde{A}}(\sigma)$ is MF which confines from bottom $\mathrm{FOU}(\tilde{A})$ (coloured dashed lines in Fig. 4.2), i.e.:

$$\begin{cases} \overline{\mu}_{\tilde{A}}(\sigma) = \overline{\mathrm{FOU}(\tilde{A})} & \forall \sigma \in \mathbb{R} \\ \underline{\mu}_{\tilde{A}}(\sigma) = \underline{\mathrm{FOU}(\tilde{A})} & \forall \sigma \in \mathbb{R}. \end{cases} \tag{4.4}$$

Remark 4.1. If $\overline{\mu}_{\tilde{A}}(\sigma) \equiv \underline{\mu}_{\tilde{A}}(\sigma) \quad \forall \sigma \in \mathbb{R}$, then the type-2 FS $\tilde{A}$ will degenerate to the type-1 FS A with MF $\mu_A(\sigma) \equiv \overline{\mu}_{\tilde{A}}(\sigma) \equiv \underline{\mu}_{\tilde{A}}(\sigma) \quad \forall \sigma \in \mathbb{R}$.

Remark 4.2. A common way to extend a type-1 FS to an interval type-2 FS is:

$$\begin{cases} \overline{\mu}_{\tilde{A}}(\sigma) &= \mu_A(\sigma) \\ \underline{\mu}_{\tilde{A}}(\sigma) &= \alpha \mu_A(\sigma), \end{cases} \tag{4.5}$$

where α is the height of the lower MFs [120].

Remark 4.3. If $\alpha = 1$ in (4.5), then according to Remark 4.1 the interval type-2 FS $\tilde{A}$ will degenerate to the type-1 FS A with MF $\mu_A(\sigma) \equiv \overline{\mu}_{\tilde{A}}(\sigma) \equiv \underline{\mu}_{\tilde{A}}(\sigma) \quad \forall \sigma \in \mathbb{R}$.

Definition 4.1.5. If N_R is the number of rules in a rule-base R, then the i^{th} *rule* $R_i \in \text{R}$, $i \in \{1, \ldots, N_R\}$, is indicated as IF $-$ THEN statement, i.e.:

$$R_i : \begin{matrix} \text{IF } \sigma_1 \text{ is } \tilde{A}_{1,i} \text{ and } \ldots \text{ and } \sigma_{N_I} \text{ is } \tilde{A}_{N_I,i}, \\ \text{THEN } \varphi_1 \text{ is } C_{1,i} \text{ and } \ldots \text{ and } \varphi_{N_O} \text{ is } C_{N_O,i} \end{matrix}, \quad i \in \{1, \ldots, N_R\}, \tag{4.6}$$

where $\tilde{A}_{j,i}$, $j \in \{1, \ldots, N_I\}$, represents antecedent FS; while $C_{h,i}$, $h \in \{1, \ldots, N_O\}$, represents consequent FS.

Remark 4.4. In T2-FLS, the structure of the rule-base remains exactly the same as in T1-FLS.

Remark 4.5. Similarly to T1-FLS, $\tilde{A}^k_j$, $j \in \{1, \ldots, N_I\}, k \in \{1, \ldots, N_F\}$, represents the k^{th} antecedent FS of the j^{th} input; while, $\tilde{A}_{j,i}$, $j \in \{1, \ldots, N_I\}, i \in \{1, \ldots, N_R\}$, represents the antecedent FS of the j^{th} input in the i^{th} rule.

Once the crisp inputs are fuzzified, FSs activate the inference engine. The inference engine implies type-2 fuzzy output sets $\mathbf{F}$ from type-2 fuzzy input sets $\mathbf{\Sigma}^{\text{T2}}$.

Definition 4.1.6. The *set of firing strengths* $F_i(\boldsymbol{\sigma}) \in [0,1]^2$, $i \in \{1, \ldots, N_R\}$, of the i-th rule can be computed with the product t-norm [121], i.e.:

$$F_i(\boldsymbol{\sigma}) = \begin{bmatrix} \overline{f}_i(\boldsymbol{\sigma}) = \prod_{j=1}^{N_I} \overline{\mu}_{\tilde{A}_{j,i}}(\sigma_j) \\ \underline{f}_i(\boldsymbol{\sigma}) = \prod_{j=1}^{N_I} \underline{\mu}_{\tilde{A}_{j,i}}(\sigma_j) \end{bmatrix}, \quad i \in \{1, \ldots, N_R\}. \tag{4.7}$$

Consequently, the type reducer maps a type-2 FS $\mathbf{F}$ into a type-1 FS Φ^{T1} [122].

Definition 4.1.7. The h^{th} left and right *end-points of the type-reduced set* $\varphi_{L,h}$ and $\varphi_{R,h}$, respectively, $h \in \{1, \ldots, N_O\}$, can be computed with KM centroid type-reduction algorithm [123]:

$$\begin{cases} \varphi_{L,h}(\boldsymbol{\sigma}) & = \frac{\sum_{i=1}^{L} \overline{f}_i(\boldsymbol{\sigma}) C_{h,i} + \sum_{i=L+1}^{N} \underline{f}_i(\boldsymbol{\sigma}) C_{h,i}}{\sum_{i=1}^{L} \overline{f}_i(\boldsymbol{\sigma}) + \sum_{i=L+1}^{N} \underline{f}_i(\boldsymbol{\sigma})} \\ \varphi_{R,h}(\boldsymbol{\sigma}) & = \frac{\sum_{i=1}^{R} \underline{f}_i(\boldsymbol{\sigma}) C_{h,i} + \sum_{i=R+1}^{N} \overline{f}_i(\boldsymbol{\sigma}) C_{h,i}}{\sum_{i=1}^{R} \underline{f}_i(\boldsymbol{\sigma}) + \sum_{i=R+1}^{N} \overline{f}_i(\boldsymbol{\sigma})} \end{cases}, \quad h \in \{1, \ldots, N_O\}, \tag{4.8}$$

in which L and R are the left and right switching points, respectively. Usually, L and R are computed by an iterative algorithm.

The output of FLCs must be a crisp number, this is accomplished by the defuzzifier.

Definition 4.1.8. The h^{th} *defuzzified output*, $h \in \{1, \ldots, N_O\}$, can be computed with the average defuzzification, i.e.:

$$\varphi_h^{\text{IT2}}(\boldsymbol{\sigma}) = \frac{\varphi_{L,h}(\boldsymbol{\sigma}) + \varphi_{R,h}(\boldsymbol{\sigma})}{2}, \quad h \in \{1, \ldots, N_O\}. \tag{4.9}$$

Remark 4.6. By combining (4.7), (4.8) and (4.9), the h^{th} defuzzified output, $h \in \{1, \ldots, N_O\}$, can be computed as follows:

$$\begin{aligned}\varphi_h^{\text{IT2}}(\boldsymbol{\sigma}) = & \frac{\sum_{i=1}^{L}\left(\prod_{j=1}^{N_I}\overline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right)C_{h,i} + \sum_{i=L+1}^{N}\left(\prod_{j=1}^{N_I}\underline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right)C_{h,i}}{2\sum_{i=1}^{L}\left(\prod_{j=1}^{N_I}\overline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right) + \sum_{i=L+1}^{N}\left(\prod_{j=1}^{N_I}\underline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right)} \\ & + \frac{\sum_{i=1}^{R}\left(\prod_{j=1}^{N_I}\underline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right)C_{h,i} + \sum_{i=R+1}^{N}\left(\prod_{j=1}^{N_I}\overline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right)C_{h,i}}{2\sum_{i=1}^{R}\left(\prod_{j=1}^{N_I}\underline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right) + \sum_{i=R+1}^{N}\left(\prod_{j=1}^{N_I}\overline{\mu}_{\tilde{A}_{j,i}}(\sigma_j)\right)}, \\ & h \in \{1, \ldots, N_O\}.\end{aligned} \tag{4.10}$$

The structure of a double-input interval type-2 fuzzy PD controller, which inherits double-input IT2-FLS, is shown in Fig. 4.3. The input scaling factors k_p and k_d are chosen to normalize e and $\dot{e}$ to the universe of discourse of the antecedent MFs, i.e., $[-1, 1]$. So, e and $\dot{e}$ are transformed into σ_1 and σ_2, respectively, before inputting them to double-input IT2-FLS. Consequently, the output φ^{T2} from double-input IT2-FLS is transformed into the control signal u by using an unscaling gain k_o such that the output is denormalized to the domain of the control signal. In the adopted control structure, only one parameter has to be tuned, i.e., k_o.

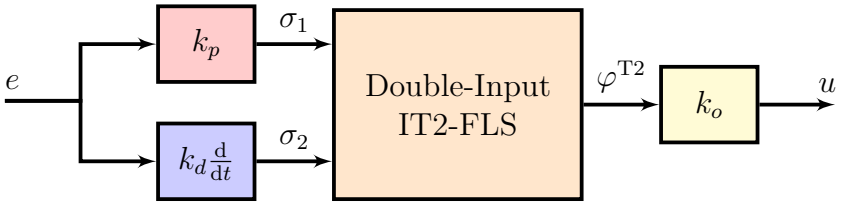


Figure 4.3: Structure of a double-input single-output interval type-2 fuzzy PD controller.

4.2 Experimental Results

The antecedent MFs are defined as novel elliptic IT2-FSs which are depicted in Fig. 4.2. The conventional way to represent elliptic IT2-FSs is

$$\begin{cases} \overline{\mu}_{\tilde{A}_j^k}(\sigma_j) & = \begin{cases} \sqrt[a_1]{1-\left|\frac{\sigma_j-c_{j,k}}{d_{j,k}}\right|^{a_1}} & , c_{j,k}-d_{j,k} \leq \sigma_j \leq c_{j,k}+d_{j,k} \\ 0 & , \sigma_j < c_{j,k}-d_{j,k}, c_{j,k}+d_{j,k} < \sigma_j \end{cases} \\ \underline{\mu}_{\tilde{A}_j^k}(\sigma_j) & = \begin{cases} \sqrt[a_2]{1-\left|\frac{\sigma_j-c_{j,k}}{d_{j,k}}\right|^{a_2}} & , c_{j,k}-d_{j,k} \leq \sigma_j \leq c_{j,k}+d_{j,k} \\ 0 & , \sigma_j < c_{j,k}-d_{j,k}, c_{j,k}+d_{j,k} < \sigma_j, \end{cases} \end{cases} \tag{4.11}$$

where $c_{j,k}$ and $d_{j,k}$ are centers and widths of MFs, respectively. The parameters a_1 and a_2 determine the width of FOU of each MF, and these parameters should be selected in the following form:

$$\begin{cases} & a_1 \geq 1 \\ 0 \leq & a_2 \leq 1. \end{cases} \tag{4.12}$$

4.2.1 Setup

The experimental flight tests for the trajectory tracking problem were conducted in the indoor environment. The laboratory environment is designed to use a set of eight OptiTrack Prime 13 cameras, to provide real-time pose (position and attitude) of the UAV with an update rate of 240Hz and accuracy around 0.1mm. The pose data are routed to the controller through the local network. The aircraft used for the experimental flight tests is Parrot Bebop 2 UAV, which is an attitude controlled commercial quadrotor. The Bebop Autonomy ROS package is used to communicate the output velocity from the controller to the quadrotor via a Wi-Fi connection.

4.2.2 Trajectory

In the experimental scenario, a circle with radius 2m is chosen for the trajectory tracking problem. To test the stability and robustness of the controllers, the reference speed varies along the trajectory. The profile of the desired velocity is designed in a way to have different velocity conditions for each quarter of the circle,

as shown in Fig. 4.4. In the first quarter ($t \in [0, \pi/2)$), the speed gradually increases from 0m/s to 2m/s. In the second quarter ($t \in [\pi/2, \pi)$), the speed is kept constant at 2m/s. In the third quarter ($t \in [\pi, 3\pi/2)$), the speed is increased again to reach 4m/s. Finally, in the fourth quarter ($t \in [3\pi/2, 2\pi)$), the speed is decreased to 0m/s. Then, the new circle starts and the velocity profile is repeated.

4.2.3 Discussion

The experimental results are evaluated through the most commonly used error-based measures, i.e., root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{N_D}\sum_{i=1}^{N_D}(\mathbf{p}_i^* - \mathbf{p}_i)^2}, \tag{4.13}$$

maximum absolute error (MAX):

$$MAX = \max_{i \in N_D} \|\mathbf{p}_i^* - \mathbf{p}_i\|, \tag{4.14}$$

and mean absolute error (MAE):

$$MAE = \frac{1}{N_D}\sum_{i=1}^{N_D} \|\mathbf{p}_i^* - \mathbf{p}_i\|, \tag{4.15}$$

where N_D is the number of data samples, $\mathbf{p}_i^* = \begin{bmatrix} x_i^* & y_i^* & z_i^* \end{bmatrix}^T$ and $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ are desired and actual position for the i-th sample, respectively.

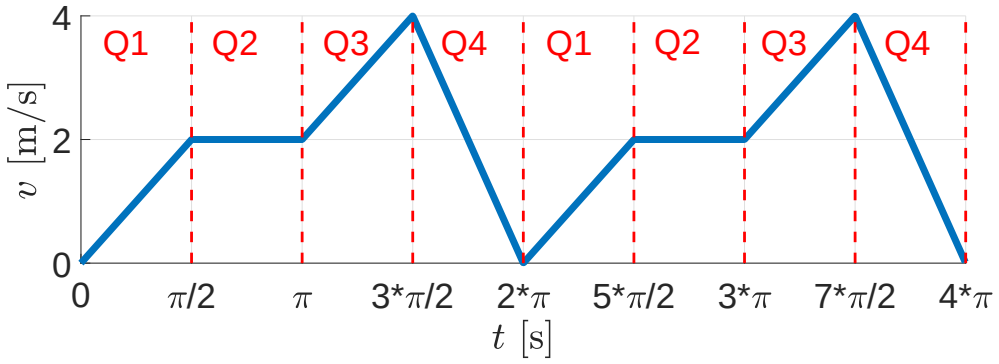


Figure 4.4: Velocity profile of the desired trajectory.

In Fig. 4.5, the trajectory tracking performance of interval type-2 fuzzy logic controller with elliptic MFs is compared with the performances of conventional PD controller, type-1 fuzzy logic controllers with Gaussian and elliptic MFs, and type-2 fuzzy logic controllers with Gaussian MFs. The projections of the trajectory on x, y and z axes of five complete circles are shown in Fig. 4.6. As can be seen, the steady-state error is reduced because of the filtering capabilities of fuzzy logic controllers. This can also be seen from the Euclidean error and the average RMSE values from ten experiments which are shown in Fig. 4.7 and Table 4.1.

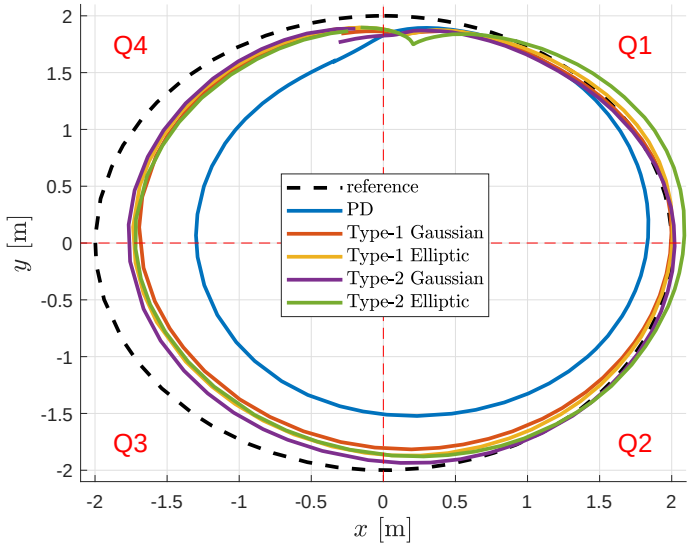
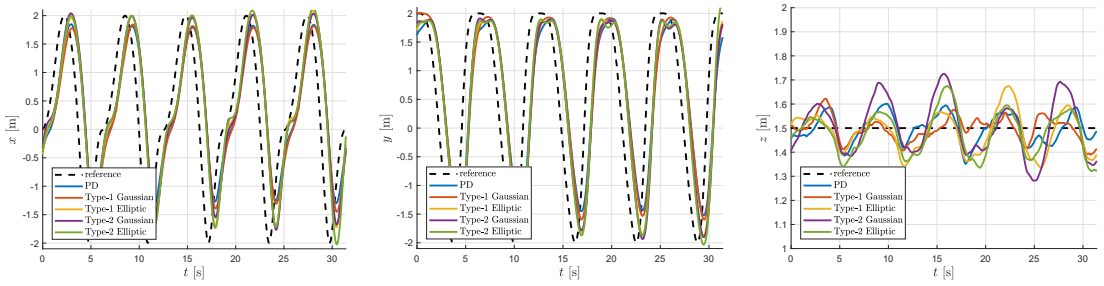


Figure 4.5: 3D trajectory tracking by five different controllers.



(a) x-axis tracking. (b) y-axis tracking. (c) z-axis tracking.

Figure 4.6: Projection of trajectory tracking along x, y and z axes by five different controllers.

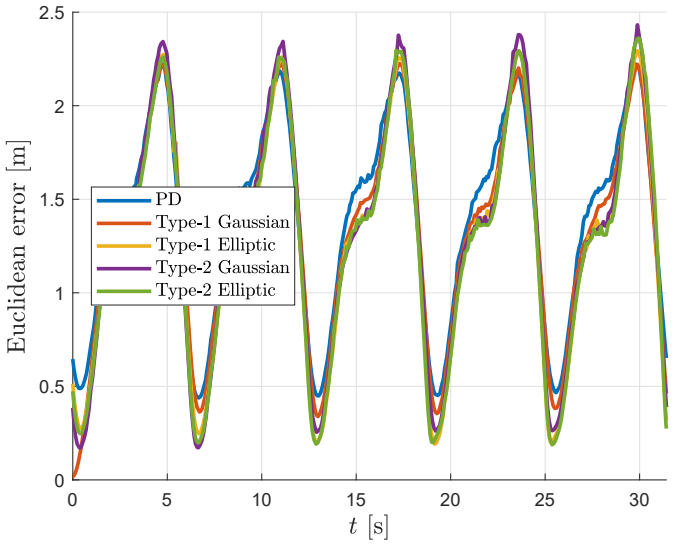


Figure 4.7: Euclidean error of different controllers.

4.3 Conclusion

Whereas type-1 and type-2 MFs are the core of any fuzzy logic system, there are no performance criteria available to evaluate the goodness or correctness of fuzzy MFs. In this chapter, an extensive analysis in terms of the capability of interval type-2 elliptic fuzzy MFs in modelling uncertainty has been done. Having decoupled parameters for its support and width, elliptic MFs are unique amongst existing type-2 fuzzy MFs. In this investigation, the uncertainty distribution along the elliptic MF support is studied, and a detailed analysis is given to compare and contrast its performance with existing type-2 fuzzy MFs. Moreover, to test the performance of FLC with elliptic interval type-2 MFs, extensive real-time experiments are conducted for the 3D trajectory tracking problem of a quadrotor. The results of this study might open the doors to wider use of elliptic MFs for real-world identification and control applications as the proposed MF is easy to interpret in addition to its unique features.

Table 4.1: Comparison results for the error from different controllers.

Controller	**RMSE**	**MAX**	**MAE**
PD	1.595	2.224	1.490
T1-FLC with Gaussian MFs	1.455	2.326	1.339
T1-FLC with elliptic MFs	1.418	2.296	1.277
IT2-FLC with Gaussian MFs	1.445	2.432	1.309
IT2-FLC with elliptic MFs	1.416	2.361	1.270

Chapter 5

Fuzzy Mapping-Based Control

The mathematical expression of FM provides an efficient tool to analytically study FLCs [124]. In addition, modern computers can perform the basic algebraic operations, e.g., additions, subtractions, multiplication and divisions, much more efficiently than the operations on FSs, e.g., unions, intersections and implications, needed in fuzzy logic [12]. Therefore, the availability of an analytical form of FM will open new doors to the use of FLCs in real-time applications. Recently, FM for the single-input IT2-FLC case has been derived in [120]. Nevertheless, an exhaustive analysis of FM for Mamdani double-input FLCs and real-time validation of the theoretical claims are still missing in the literature [84]. Moreover, the study of some properties of FM for double-input FLCs, such as symmetry and monotonicity, are missing in the literature.

In this chapter, an alternative method to derive and analyse FM for FLCs is proposed. First, Sections 5.1 revises the definition of FM. Sections 5.2 and 5.3 present derivation and analysis of FM for T1-FLCs and for IT2-FLCs, respectively. Then, Sections 5.4 and 5.5 show simulation and experimental results quadrotor UAV, respectively. Finally, some conclusions are drawn in Section 5.6.

Supplementary Material:
Matlab code for the FM generation: github.com/andriyukr/FM.
ROS package for the proposed FM controllers: github.com/andriyukr/controllers.
Video for the simulation results: tiny.cc/FM-FLC_simulation.
Video for the experimental results: tiny.cc/FM-FLC.

5.1 Mathematical Preliminaries

Generally, FLS can be seen as a mapping from crisp input $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 & \cdots & \sigma_{N_I} \end{bmatrix}^T$, where N_I is the number of crisp inputs, to crisp output $\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 & \cdots & \varphi_{N_O} \end{bmatrix}^T$, where N_O is the number of crisp outputs [115].

Definition 5.1.1. In a general FLS, FM from $\boldsymbol{\sigma} \in \mathbb{R}^{N_I}$ to $\boldsymbol{\varphi} \in \mathbb{R}^{N_O}$ is a function $\boldsymbol{\varphi}(\boldsymbol{\sigma}) : \mathbb{R}^{N_I} \to \mathbb{R}^{N_O}$.

In order to facilitate the analytical derivation, FM of double-input FLSs is considered. However, the presented approach can be applied to FLSs with an arbitrary number of inputs.

Definition 5.1.2. In a double-input FLS, FM from $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 & \sigma_2 \end{bmatrix}^T \in \mathbb{R}^2$ to $\varphi \in \mathbb{R}$ is a function $\varphi(\boldsymbol{\sigma}) : \mathbb{R}^2 \to \mathbb{R}$.

Remark 5.1. The unit double-input FM $\varphi^{\mathrm{T0}}(\boldsymbol{\sigma})$ is defined as:

$$\varphi^{\mathrm{T0}}(\boldsymbol{\sigma}) = \frac{\sigma_1 + \sigma_2}{2}. \tag{5.1}$$

Definition 5.1.3. The *aggressiveness* ε of FM is the value of its gradient $\delta(\boldsymbol{\sigma}) = \nabla\varphi(\boldsymbol{\sigma})$ in the neighbourhood of the equilibrium point $(0, 0)$ and in the direction of the unit vector $\hat{\mathbf{w}}$, i.e.:

$$\varepsilon = \hat{\mathbf{w}}^T \delta(0, 0). \tag{5.2}$$

Remark 5.2. If the gradient of $\varphi^{\mathrm{T0}}(\boldsymbol{\sigma})$ is $\delta^{\mathrm{T0}}(\boldsymbol{\sigma}) = \nabla\varphi^{\mathrm{T0}}(\boldsymbol{\sigma})$, and $\hat{\mathbf{w}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T$, which is the unit vector in the direction of $\begin{bmatrix} \sigma_1 & \sigma_2 \end{bmatrix}^T$, then the aggressiveness of the unit mapping in (5.1) is calculated as follows:

$$\varepsilon^{\mathrm{T0}} = \hat{\mathbf{w}}^T \delta^{\mathrm{T0}}(0, 0) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \frac{\sqrt{2}}{2}. \tag{5.3}$$

5.2 Type-1 Fuzzy Mapping

Definition 5.2.1. In double-input T1-FLS (DI-T1-FLS), FM from $\boldsymbol{\sigma} = \begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}^T \in \mathbb{R}^2$ to $\varphi^{\mathrm{T1}} \in \mathbb{R}$ is a function $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma}) : \mathbb{R}^2 \to \mathbb{R}$.

In order to facilitate the analytical derivation, the antecedent MFs are designed to be triangular type-1 FSs, as illustrated in Fig. 5.1. The typical representation of a triangular MF is [125]:

$$\mu_{A_j^k}(\sigma_j) = \begin{cases} 0 & , \sigma_j < a_{k-1} \\ \frac{\sigma_j - a_{k-1}}{a_k - a_{k-1}} & , a_{k-1} \leq \sigma_j \leq a_k \\ \frac{a_{k+1} - \sigma_j}{a_{k+1} - a_k} & , a_k < \sigma_j \leq a_{k+1} \\ 0 & , \sigma_j > a_{k+1}, \end{cases} \tag{5.4}$$

where $k \in \{1, 2, 3\}$ and $j \in \{1, 2\}$. From Fig. 5.1 can be seen that $a_0 = -\infty$, $a_1 = -1$, $a_2 = 0$, $a_3 = 1$ and $a_4 = +\infty$. Besides, the consequent MFs are designed to be singleton, as illustrated in Fig. 3.3. Moreover, the rule-base in Table 3.1 is used.

For the analytical analysis, complex symbolic computations are needed. Thus, an equivalent definition of (5.4) is used:

$$\mu_{A_j^k}(\sigma_j) = \max\left(\min\left(\frac{\sigma_j - a_{k-1}}{a_k - a_{k-1}}, \frac{a_{k+1} - \sigma_j}{a_{k+1} - a_k}\right), 0\right), \tag{5.5}$$

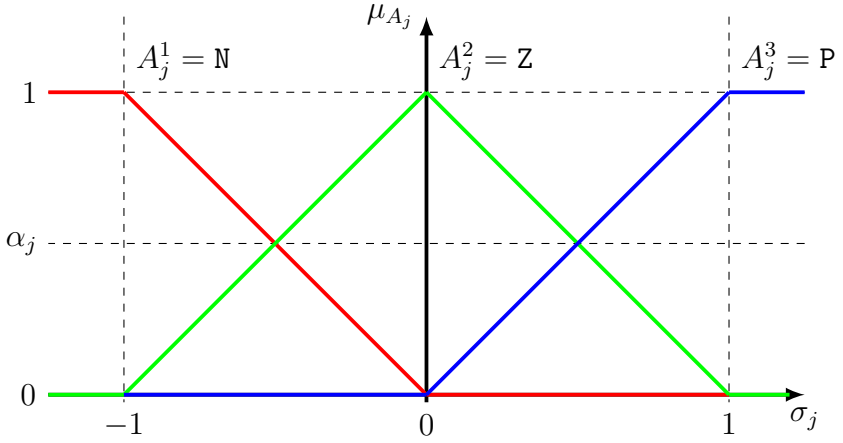


Figure 5.1: "Negative" (N), "zero" (Z) and "positive" (P) FSs represented by three triangular type-1 MFs.

where max() and min() functions are reformulated as algebraical functions:

$$\begin{cases} \max(a,b) &= \frac{a+b+|a-b|}{2} \\ \min(a,b) &= \frac{a+b-|a-b|}{2} \end{cases} \quad \forall a \in \mathbb{R} \quad \forall b \in \mathbb{R}. \tag{5.6}$$

5.2.1 Derivation of Fuzzy Mapping for DI-T1-FLS

The rule-base in Table 3.1 contains nine rules; consequently, $N_R = 9$. By using (3.5), the output from FLS is as follows:

$$\varphi^{\mathrm{T1}}(\boldsymbol{\sigma}) = \frac{\sum_{i=1}^{9} \mu_{A_{1,i}}(\sigma_1) \cdot \mu_{A_{2,i}}(\sigma_2) \cdot C_i}{\sum_{i=1}^{9} \mu_{A_{1,i}}(\sigma_1) \cdot \mu_{A_{2,i}}(\sigma_2)}. \tag{5.7}$$

Then, combining Definition 3.1.4, Table 3.1 and Fig. 5.1, it is clear that $\mu_{A_{1,1}} = \mu_{A_{1,2}} = \mu_{A_{1,3}} = \mu_{A_{2,1}} = \mu_{A_{2,4}} = \mu_{A_{2,7}} = \mu_{A^1}$, $\mu_{A_{1,4}} = \mu_{A_{1,5}} = \mu_{A_{1,6}} = \mu_{A_{2,2}} = \mu_{A_{2,5}} = \mu_{A_{2,8}} = \mu_{A^2}$ and $\mu_{A_{1,7}} = \mu_{A_{1,8}} = \mu_{A_{1,9}} = \mu_{A_{2,3}} = \mu_{A_{2,6}} = \mu_{A_{2,9}} = \mu_{A^3}$, which are defined in (5.5). Combining Definition 3.1.4, Table 3.1 and Fig. 3.3, it is clear that $C_1 = C_2 = C_4 = C^1 = -1$, $C_3 = C_5 = C_7 = C^3 = 0$, and $C_6 = C_8 = C_9 = C^5 = 1$. Hence, after performing some simplifications in (5.7), $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is obtained:

$$\varphi^{\mathrm{T1}}(\boldsymbol{\sigma}) = \sigma_1 + \sigma_2 - \frac{|\sigma_1|\sigma_2 + \sigma_1|\sigma_2|}{2}. \tag{5.8}$$

The expression in (5.8) describes DI-T1-FLS in an analytical form. Therefore, instead of considering DI-T1-FLS as a grey-box, its symbolic representation, i.e., $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$, can be used. The generated fuzzy surface, which maps the two inputs σ_1 and σ_2 to the output φ, is plotted in Fig. 5.2.

5.2.2 Analysis of Fuzzy Mapping for DI-T1-FLS

From the asymptotic computational analysis, the runtime complexity for T1-FLS represented by (3.5) is $O(2R_I N_R R_O)$ which is linear w.r.t. N_I, N_R and N_O. While the runtime complexity of FM for T1-FLS represented by (5.8) is $O(1)$ which is constant. Therefore, independently on the number of inputs, rules and outputs in T1-FLS, the computational complexity of FM for T1-FLS is constant.

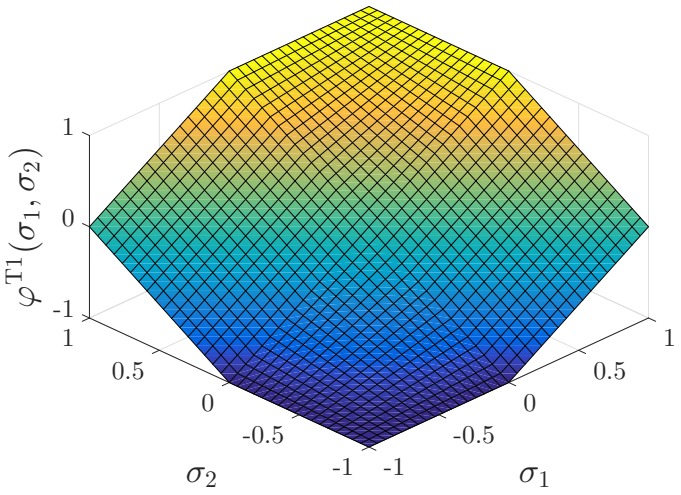


Figure 5.2: Fuzzy surface generated by DI-T1-FLS.

The gradient of $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is $\delta^{\mathrm{T1}}(\boldsymbol{\sigma}) = \nabla\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$. If $\hat{\mathbf{w}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T$ which is the unit vector in the direction of $\begin{bmatrix} \sigma_1 & \sigma_2 \end{bmatrix}$, by using Definition 5.1.3, the aggressiveness of φ^{T1} becomes:

$$\varepsilon^{\mathrm{T1}} = \hat{\mathbf{w}}^T \delta^{\mathrm{T1}}(0,0) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \sqrt{2}. \tag{5.9}$$

Therefore, the aggressiveness of DI-T1-FLS is constant. From (5.9) and (5.3), it can be observed that $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is more aggressive than $\varphi^{0}(\boldsymbol{\sigma})$ in the neighbourhood of $(0,0)$, since $\varepsilon^{\mathrm{T1}} > \varepsilon^{\mathrm{T0}}$.

Theorem 5.2.1 (Symmetry of type-1 FM)**.** If $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ indicates FM of DI-T1-FLS, then

i) $\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2)$ is an even symmetric function w.r.t. the bisection of the first ($\sigma_1 > 0$, $\sigma_2 > 0$) and third ($\sigma_1 < 0$, $\sigma_2 < 0$) quadrants in the Cartesian plane, i.e., $\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2) = \varphi^{\mathrm{T1}}(\sigma_2,\sigma_1) \quad \forall\sigma_1 \in [-1,1] \quad \forall\sigma_2 \in [-1,1]$;

ii) $\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2)$ is an odd symmetric function w.r.t. the bisection of the second ($\sigma_1 > 0$, $\sigma_2 < 0$) and fourth ($\sigma_1 < 0$, $\sigma_2 > 0$) quadrants in the Cartesian plane, i.e., $\varphi^{\mathrm{T1}}(-\sigma_1,-\sigma_2) = -\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2) \quad \forall\sigma_1 \in [-1,1] \quad \forall\sigma_2 \in [-1,1]$.

Proof. Using (5.8), it follows:

i) $\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2) = \sigma_1 + \sigma_2 - \frac{|\sigma_1|\sigma_2+\sigma_1|\sigma_2|}{2}$ and $\varphi^{\mathrm{T1}}(\sigma_2,\sigma_1) = \sigma_2 + \sigma_1 - \frac{|\sigma_2|\sigma_1+\sigma_2|\sigma_1|}{2}$; therefore, $\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2) = \varphi^{\mathrm{T1}}(\sigma_2,\sigma_1) \quad \forall\sigma_1 \in [-1,1] \quad \forall\sigma_2 \in [-1,1]$;

ii) $\varphi^{\mathrm{T1}}(-\sigma_1,-\sigma_2) = -\sigma_1 - \sigma_2 - \frac{-|\sigma_1|\sigma_2-\sigma_1|\sigma_2|}{2} = -\sigma_1 - \sigma_2 + \frac{|\sigma_1|\sigma_2+\sigma_1|\sigma_2|}{2} = -\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2)$; therefore, $\varphi^{\mathrm{T1}}(-\sigma_1,-\sigma_2) = -\varphi^{\mathrm{T1}}(\sigma_1,\sigma_2) \quad \forall\sigma_1 \in [-1,1] \quad \forall\sigma_2 \in [-1,1]$.

□

Corollary 5.2.1.1. Using the second property in Theorem 5.2.1, $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ can be rewritten as:

$$\varphi^{\mathrm{T1}}(\boldsymbol{\sigma}) = -\varphi^{\mathrm{T1}}(-\boldsymbol{\sigma}). \tag{5.10}$$

Theorem 5.2.2 (Continuity of type-1 FM)**.** If $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ indicates FM of DI-T1-FLS, then $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is a continuous function in the region $[-1,1]^2$ w.r.t. its input variable $\boldsymbol{\sigma}$, i.e., $\varphi^{\mathrm{T1}} \in \mathcal{C}^0([-1,1]^2)$.

Proof. First, (5.8) is decomposed into two components: $\varphi^{\mathrm{T1},1}(\boldsymbol{\sigma}) = \sigma_1 + \sigma_2$ and $\varphi^{\mathrm{T1},2}(\boldsymbol{\sigma}) = \frac{|\sigma_1|\sigma_2+\sigma_1|\sigma_2|}{2}$. Since $\varphi^{\mathrm{T1},1}(\boldsymbol{\sigma})$ is a polynomial function, it is continuous on $\mathbb{R}^2$. On the other side, it can be observed that $\lim_{\sigma_1\to\mathbf{c}} \varphi^{\mathrm{T1},2}(\boldsymbol{\sigma}) = \varphi^{\mathrm{T1},2}(\mathbf{c}) \quad \forall\mathbf{c} \in \mathbb{R}^2$. Therefore, $\varphi^{\mathrm{T1},2}(\boldsymbol{\sigma})$ is also continuous on $\mathbb{R}^2$. Since $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is a linear combination of continuous functions, i.e., $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma}) = \varphi^{\mathrm{T1},1}(\boldsymbol{\sigma}) - \varphi^{\mathrm{T1},2}(\boldsymbol{\sigma})$, $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is also a continuous function.

□

Corollary 5.2.2.1. If the control inputs to the double-input type-1 fuzzy PD (DI-T1-FPD) controller are continuous, then the control output from DI-T1-FPD controller is also continuous.

Theorem 5.2.3 (Monotonicity of type-1 FM)**.** If $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ indicates FM of DI-T1-FLS, then $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ is a monotonic increasing function in the region $[-1,1]^2$ w.r.t. its input variables $\boldsymbol{\sigma}$, i.e., $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_1} \geq 0 \wedge \frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_2} \geq 0 \quad \forall\boldsymbol{\sigma} \in [-1,1]^2$.

Proof. From 5.8, $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_1} = 1 - \frac{\sigma_2\mathrm{sign}(\sigma_1)-|\sigma_2|}{2}$. By observing that $|\sigma_2| = \sigma_2\mathrm{sign}(\sigma_2)$, $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_1} = 1 - \frac{\sigma_2(\mathrm{sign}(\sigma_1)-\mathrm{sign}(\sigma_2))}{2}$, in which $\frac{\sigma_2(\mathrm{sign}(\sigma_1)-\mathrm{sign}(\sigma_2))}{2} \in [-1,1] \quad \forall\boldsymbol{\sigma} \in [-1,1]^2$. Consequently, $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_1} \in [0,2] \quad \forall\boldsymbol{\sigma} \in [-1,1]^2$, and, thus, $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_1} \geq 0 \quad \forall\boldsymbol{\sigma} \in [-1,1]^2$. From which follows that $\varphi(\boldsymbol{\sigma})$ is an increasing function w.r.t. σ_1 $\forall\boldsymbol{\sigma} \in [-1,1]^2$. From the first result in Theorem 5.2.1, if $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_1} \geq 0 \quad \forall\boldsymbol{\sigma} \in [-1,1]^2$, then $\frac{\partial\varphi^{\mathrm{T1}}}{\partial\sigma_2} \geq 0 \quad \forall\boldsymbol{\sigma} \in [-1,1]^2$. Therefore, φ^{T1} is a monotonic increasing in the region $[-1,1]^2$ w.r.t. its input variables σ_1 and σ_2.

□

5.3 Interval Type-2 Fuzzy Mapping

Definition 5.3.1. In double-input IT2-FLS (DI-IT2-FLS), FM from $\boldsymbol{\sigma} = \begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}^T \in \mathbb{R}^2$ to $\varphi^{\text{IT2}} \in \mathbb{R}$ is a function $\varphi^{\text{IT2}}(\boldsymbol{\sigma}) : \mathbb{R}^2 \to \mathbb{R}$.

As described in Remark 4.2, type-1 FSs in Fig. 5.1 can be extended to interval type-2 FSs in depicted Fig. 5.3.

5.3.1 Derivation of Fuzzy Mapping for DI-IT2-FLS

By observing the structure of the rule-base for double-input FLS in Table 3.1, each consequent MF (`BN`, `Z`, `BP`) can be implied from exactly three rules. Consequently, L and R in Definition 4.1.7 are multiples of 3 in the interval $(1, N_R = 9)$, i.e., $L \in \{3, 6\} \wedge R \in \{3, 6\}$. By using the constraint that $L \leq R$, three distinct cases for the switching points can be determined, i.e., $< \{L = 3, R = 3\}, \{L = 3, R = 6\}, \{L = 6, R = 6\} >$. Each of these cases defines a region $(\Omega_1, \Omega_2, \Omega_3)$ on $[\sigma_1 \times \sigma_2]$ plane, as shown in Fig. 5.4. Hence, Ω_1, Ω_2 and Ω_3 are analytically defined as:

$$\begin{cases} \Omega_1 = \{\{\sigma_1, \sigma_2\} \in [-1, 1]^2 \mid \sigma_2 \geq -1, \sigma_2 \leq \omega_{12}(\sigma_1)\} \\ \Omega_2 = \{\{\sigma_1, \sigma_2\} \in [-1, 1]^2 \mid \sigma_2 > \omega_{12}(\sigma_1), \sigma_2 < \omega_{23}(\sigma_1)\} \\ \Omega_3 = \{\{\sigma_1, \sigma_2\} \in [-1, 1]^2 \mid \sigma_2 \geq \omega_{23}(\sigma_1), \sigma_2 \leq 1\}, \end{cases} \tag{5.11}$$

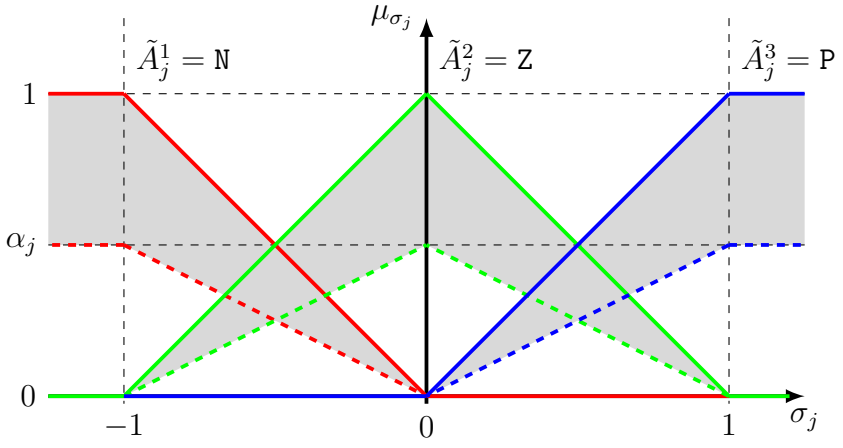


Figure 5.3: "Negative" (`N`), "zero" (`Z`) and "positive" (`P`) FSs represented by three triangular interval type-2 MFs.

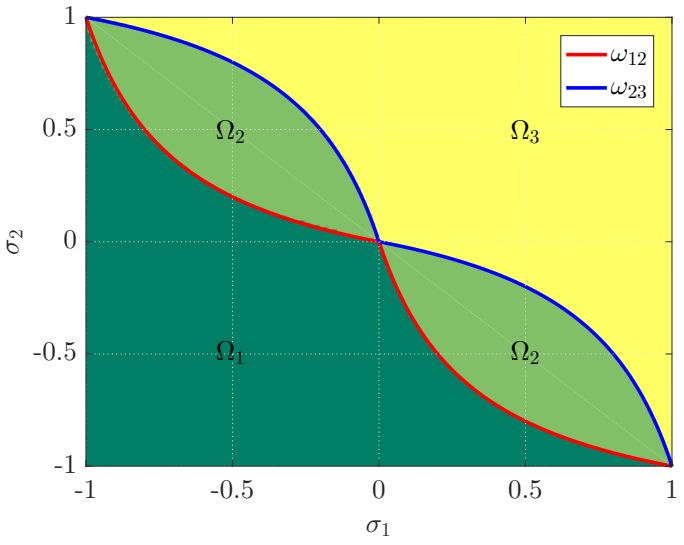


Figure 5.4: Three regions of DI-IT2-FLC FM and two contours between these regions.

where ω_{12} and ω_{23} are the contours which separate Ω_1 from Ω_2 and Ω_2 from Ω_3, respectively. For each region corresponds FM, i.e., $\varphi_1^{\text{IT2}}(\boldsymbol{\sigma})$, $\varphi_2^{\text{IT2}}(\boldsymbol{\sigma})$ and $\varphi_3^{\text{IT2}}(\boldsymbol{\sigma})$. Thus, $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$ can be broken down by using (4.9):

$$\varphi^{\text{IT2}}(\boldsymbol{\sigma}) = \begin{cases} \varphi_1^{\text{IT2}}(\boldsymbol{\sigma}) = \frac{\varphi_{L=3}(\boldsymbol{\sigma})+\varphi_{R=3}(\boldsymbol{\sigma})}{2}, & \boldsymbol{\sigma} \in \Omega_1 \\ \varphi_2^{\text{IT2}}(\boldsymbol{\sigma}) = \frac{\varphi_{L=3}(\boldsymbol{\sigma})+\varphi_{R=6}(\boldsymbol{\sigma})}{2}, & \boldsymbol{\sigma} \in \Omega_2 \\ \varphi_3^{\text{IT2}}(\boldsymbol{\sigma}) = \frac{\varphi_{L=6}(\boldsymbol{\sigma})+\varphi_{R=6}(\boldsymbol{\sigma})}{2}, & \boldsymbol{\sigma} \in \Omega_3. \end{cases} \tag{5.12}$$

The determination of the left and right end-points allows to derive the output of DI-IT2-FLS in a closed-form. Therefore, it is possible to find FM $\varphi(\boldsymbol{\sigma})$ by substituting (4.7) into (4.8):

$$\begin{cases} \varphi_{L=3}(\boldsymbol{\sigma}) &= \frac{\sigma_1(\sigma_2+1)-\sigma_1\sigma_2+\sigma_2(\sigma_1+1)}{\alpha_1\alpha_2(\sigma_1+1)(\sigma_2+1)-\sigma_1-\sigma_2(\sigma_1+1)} \\ \varphi_{L=6}(\boldsymbol{\sigma}) &= \frac{\alpha_1\alpha_2\sigma_2-\alpha_1\alpha_2\sigma_1(\sigma_2-1)}{(\sigma_1-1)(\sigma_2-1)+\alpha_1\alpha_2(\sigma_1+\sigma_2-\sigma_2\sigma_1)} \\ \varphi_{R=3}(\boldsymbol{\sigma}) &= \frac{\alpha_1\alpha_2\sigma_1+\alpha_1\alpha_2\sigma_2(\sigma_1+1)}{(\sigma_1+1)(\sigma_2+1)-\alpha_1\alpha_2(\sigma_1\sigma_2+\sigma_1+\sigma_2)} \\ \varphi_{R=6}(\boldsymbol{\sigma}) &= \frac{\sigma_1(1-\sigma_2)+\sigma_1\sigma_2-\sigma_2(1-\sigma_1)}{\alpha_1\alpha_2(\sigma_1-1)(\sigma_2-1)-\sigma_1-\sigma_2(\sigma_1-1)}, \end{cases} \tag{5.13}$$

Now, $\varphi_1^{\mathrm{IT2}}(\boldsymbol{\sigma})$, $\varphi_2^{\mathrm{IT2}}(\boldsymbol{\sigma})$ and $\varphi_3^{\mathrm{IT2}}(\boldsymbol{\sigma})$ are computed with (5.12) and (5.13):

$$\begin{cases} \varphi_1^{\mathrm{IT2}}(\boldsymbol{\sigma}) & = \frac{1}{2}\left(\frac{\alpha_1\alpha_2(\sigma_1\sigma_2-\sigma_1-\sigma_2)}{(\sigma_1-1)(\sigma_2-1)+\alpha_1\alpha_2(\sigma_1+\sigma_2-\sigma_2\sigma_1)} + \frac{\sigma_1\sigma_2-\sigma_1-\sigma_2}{\alpha_1\alpha_2(\sigma_1-1)(\sigma_2-1)+\sigma_1+\sigma_2-\sigma_1\sigma_2}\right) \\ \varphi_2^{\mathrm{IT2}}(\boldsymbol{\sigma}) & = \frac{1}{2}\left(\frac{\sigma_2(\sigma_1+1)-\alpha_1\alpha_2\sigma_1(\sigma_2-1)}{\sigma_2(\sigma_1+1)-\alpha_1\alpha_2\sigma_1-\alpha_1\alpha_2(\sigma_1+1)(\sigma_2-1)} - \frac{\sigma_1(\sigma_2-1)-\alpha_1\alpha_2\sigma_2(\sigma_1+1)}{\sigma_1(\sigma_2-1)+\alpha_1\alpha_2\sigma_2-\alpha_1\alpha_2(\sigma_1+1)(\sigma_2-1)}\right) \\ \varphi_3^{\mathrm{IT2}}(\boldsymbol{\sigma}) & = \frac{1}{2}\left(\frac{\alpha_1\alpha_2(\sigma_1+\sigma_2-\sigma_1\sigma_2)}{(\sigma_1-1)(\sigma_2-1)+\alpha_1\alpha_2\sigma_1-\alpha_1\alpha_2\sigma_2(\sigma_1-1)} - \frac{\sigma_1\sigma_2-\sigma_1-\sigma_2}{\sigma_1+\sigma_2-\sigma_1\sigma_2+\alpha_1\alpha_2(\sigma_1-1)(\sigma_2-1)}\right). \end{cases} \tag{5.14}$$

Finally, by definition the contours which separate Ω_1 from Ω_2 and Ω_2 from Ω_3, respectively, are:

$$\begin{cases} \omega_{12} & = \{\boldsymbol{\sigma} \in [-1,1]^2 \mid \varphi_{R=3}(\boldsymbol{\sigma}) = \varphi_{R=6}(\boldsymbol{\sigma})\} \\ \omega_{23} & = \{\boldsymbol{\sigma} \in [-1,1]^2 \mid \varphi_{L=3}(\boldsymbol{\sigma}) = \varphi_{L=6}(\boldsymbol{\sigma})\}, \end{cases} \tag{5.15}$$

ω_{12} and ω_{23} are computed as functions of only σ_1 with (5.13):

$$\omega_{12}(\sigma_1) = \begin{cases} \frac{-\alpha_1\alpha_2\sigma_1}{\sigma_1-\alpha_1\alpha_2\sigma_1+1}, & \sigma_1 < 0 \\ \frac{-\sigma_1}{\sigma_1+\alpha_1\alpha_2-\alpha_1\alpha_2\sigma_1}, & \sigma_1 \geq 0 \end{cases} \tag{5.16}$$

and

$$\omega_{23}(\sigma_1) = \begin{cases} \frac{-\sigma_1}{\alpha_1\alpha_2-\sigma_1+\alpha_1\alpha_2\sigma_1}, & \sigma_1 < 0 \\ \frac{-\alpha_1\alpha_2\sigma_1}{\alpha_1\alpha_2\sigma_1-\sigma_1+1}, & \sigma_1 \geq 0. \end{cases} \tag{5.17}$$

The expression in (5.14) describes DI-IT2-FLS in an analytical form. Therefore, instead of considering DI-IT2-FLC as a gray-box, its explicit representation in (5.14), i.e., $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$, can be used.

Remark 5.3. If $\alpha_1 = 1$ and $\alpha_2 = 1$, then $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$ in (5.14) will degenerate to $\varphi^{\mathrm{T1}}(\boldsymbol{\sigma})$ in (5.8).

Remark 5.4. It can be observed from (5.13), (5.14), (5.16) and (5.17) that α_1 and α_2 are always coupled, i.e., $\alpha_1\alpha_2$. Therefore, it makes sense to perform the analysis only w.r.t. $\alpha = \alpha_1\alpha_2$. The reason why α_1 and α_2 are always coupled is because the meet operation used to compute the lower firing strengths $\underline{f}_i(\boldsymbol{\sigma})$, $i \in [1, N_R]$, in (4.7), is the product t-norm.

5.3.2 Analysis of Fuzzy Mapping for DI-IT2-FLS

From the asymptotic computational analysis, the runtime complexity for IT2-FLC represented by (4.10) is $O(4N_I N_R N_O)$ which is linear w.r.t. N_I, N_R and N_O. While the runtime complexity of FM for IT2-FLC represented by (5.14) is $O(3)$ which is constant. Therefore, independently on the number of inputs, rules and outputs, the computational complexity of FM for IT2-FLC is constant.

The gradient of $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$ is $\delta^{\text{IT2}}(\boldsymbol{\sigma}) = \nabla\varphi^{\text{IT2}}(\boldsymbol{\sigma})$. If $\hat{\mathbf{w}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T$ which is the unit vector in the direction of $\begin{bmatrix} \sigma_1 & \sigma_2 \end{bmatrix}$, by using Definition 5.1.3, the aggressiveness of φ^{IT2} becomes:

$$\varepsilon^{\text{IT2}} = \hat{\mathbf{w}}^T \delta^{\text{IT2}}(0,0) = \frac{\sqrt{2}}{2}\left(\alpha + \frac{1}{\alpha}\right). \tag{5.18}$$

This relation is depicted in Fig. 5.5. For small values of α, the behaviour of DI-IT2-FLC becomes more aggressive around $(0,0)$; while, for high values of α, the behaviour of DI-IT2-FLC becomes less aggressive around $(0,0)$.

Remark 5.5. It is noted that $\varphi^{\text{T1}}(\boldsymbol{\sigma})$ is not more aggressive than $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$, since $\varepsilon^{\text{IT2}} \geq \varepsilon^{\text{T1}} \quad \forall\alpha$ and $\varepsilon^{\text{IT2}} = \varepsilon^{\text{T1}}$ only when $\alpha = 1$. Consequently, it is noted that $\varphi^{\text{T0}}(\boldsymbol{\sigma})$ is less aggressive than $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$, since $\varepsilon^{\text{IT2}} > \varepsilon^{\text{T0}} \quad \forall\alpha$.

The generated FSs, which map σ_1 and σ_2 to $\varphi^{\text{IT2}}(\sigma_1, \sigma_2)$, are plotted in Fig. 5.6 for $\alpha \in \{0.01, 0.04, 0.09, 0.16, 0.25, 0.36, 0.49, 0.64, 0.81\}$. Distinct FSs can be generated by simply varying only one parameter of FOU, i.e., α.

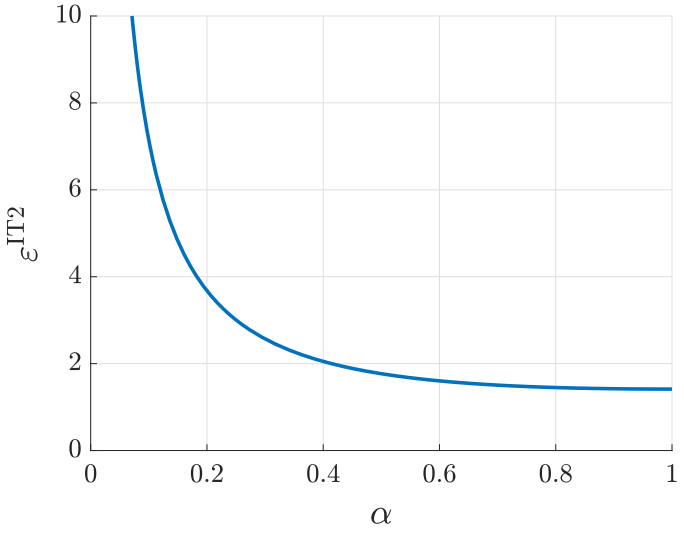
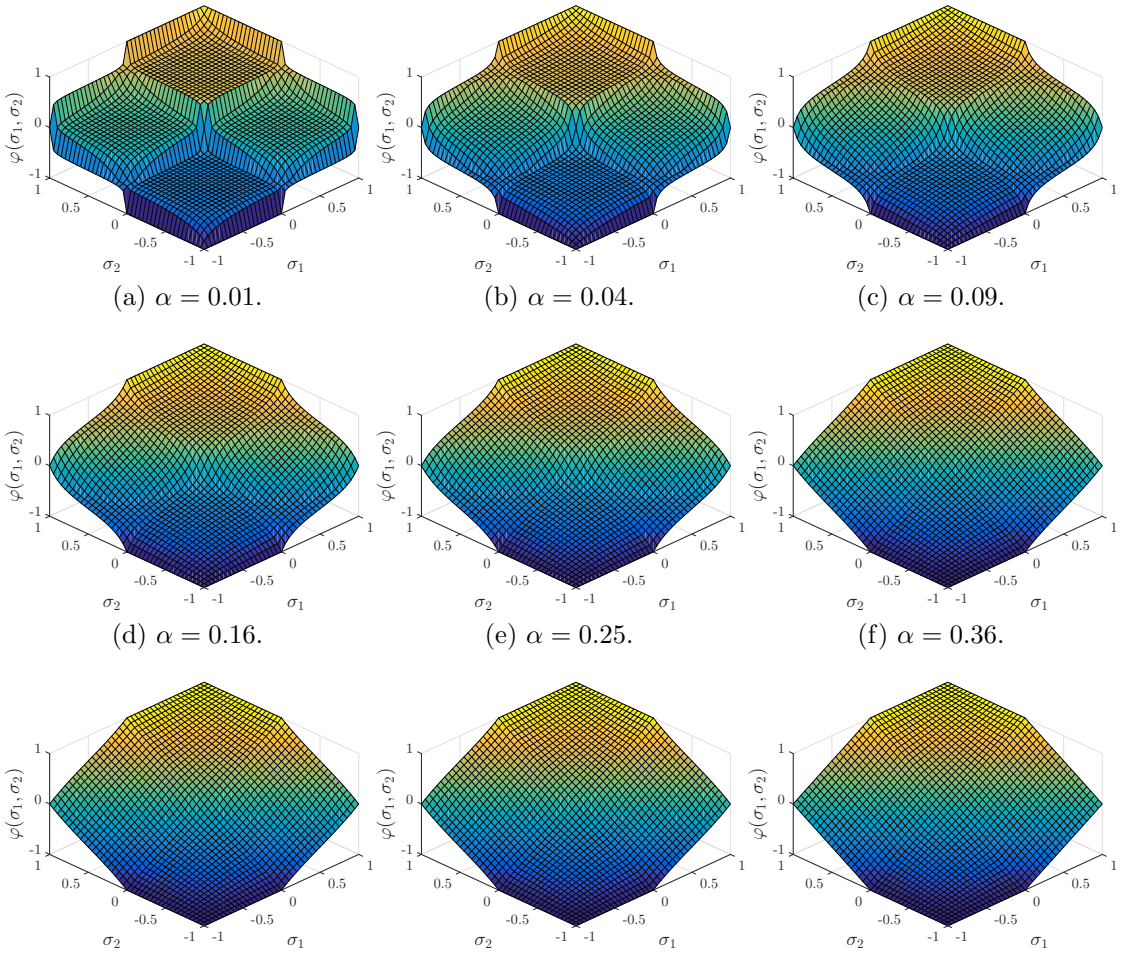


Figure 5.5: Relation between aggressiveness of $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$ and α.



(a) $\alpha = 0.01$. (b) $\alpha = 0.04$. (c) $\alpha = 0.09$.

(d) $\alpha = 0.16$. (e) $\alpha = 0.25$. (f) $\alpha = 0.36$.

(g) $\alpha = 0.49$. (h) $\alpha = 0.64$. (i) $\alpha = 0.81$.

Figure 5.6: Fuzzy surface generated by DI-IT2-FLS for different values of α.

Theorem 5.3.1 (Symmetry of interval type-2 FM)**.** If $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$ indicates FM of DI-IT2-FLS, then

i) $\varphi^{\mathrm{IT2}}(\sigma_1, \sigma_2)$ is an even symmetric function w.r.t. the bisection of the first ($\sigma_1 > 0$, $\sigma_2 > 0$) and third ($\sigma_1 < 0$, $\sigma_2 < 0$) quadrants in the Cartesian plane, i.e., $\varphi^{\mathrm{IT2}}(\sigma_1, \sigma_2) = \varphi^{\mathrm{IT2}}(\sigma_2, \sigma_1) \quad \forall \sigma_1 \in [-1, 1] \quad \forall \sigma_2 \in [-1, 1]$;

ii) $\varphi^{\mathrm{IT2}}(\sigma_1, \sigma_2)$ is an odd symmetric function w.r.t. the bisection of the second ($\sigma_1 > 0$, $\sigma_2 < 0$) and fourth ($\sigma_1 < 0$, $\sigma_2 > 0$) quadrants in the Cartesian plane, i.e., $\varphi^{\mathrm{IT2}}(-\sigma_1, -\sigma_2) = -\varphi^{\mathrm{IT2}}(\sigma_1, \sigma_2) \quad \forall \sigma_1 \in [-1, 1] \quad \forall \sigma_2 \in [-1, 1]$.

Proof.

i) To prove this property, the sufficient and necessary conditions are $\varphi_1^{\text{IT2}}(\sigma_1, \sigma_2) = \varphi_1^{\text{IT2}}(\sigma_2, \sigma_1)$, $\varphi_2^{\text{IT2}}(\sigma_1, \sigma_2) = \varphi_2^{\text{IT2}}(\sigma_2, \sigma_1)$ and $\varphi_3^{\text{IT2}}(\sigma_1, \sigma_2) = \varphi_3^{\text{IT2}}(\sigma_2, \sigma_1)$, which is immediate from (5.14).

ii) To prove this property, the sufficient and necessary conditions are $\varphi_1^{\text{IT2}}(-\sigma_1, -\sigma_2) = -\varphi_1^{\text{IT2}}(\sigma_1, \sigma_2)$ and $\varphi_2^{\text{IT2}}(-\sigma_1, -\sigma_2) = -\varphi_2^{\text{IT2}}(\sigma_1, \sigma_2)$, which is immediate from (5.14).

□

Corollary 5.3.1.1. Using the second property in Theorem 5.3.1, $\varphi_3(\boldsymbol{\sigma})$ can be rewritten as:

$$\varphi_3(\boldsymbol{\sigma}) = -\varphi_1(-\boldsymbol{\sigma}). \tag{5.19}$$

Lemma 5.3.2. If $\varphi_{L=3}(\boldsymbol{\sigma})$, $\varphi_{L=6}(\boldsymbol{\sigma})$, $\varphi_{R=3}(\boldsymbol{\sigma})$ and $\varphi_{R=6}(\boldsymbol{\sigma})$ indicate left and right FMs of the type-reduced set, ω_{12} and ω_{23} are the switching borders between $\varphi_{R=3}(\boldsymbol{\sigma})$ and $\varphi_{R=6}(\boldsymbol{\sigma})$ and between $\varphi_{L=3}(\boldsymbol{\sigma})$ and $\varphi_{L=6}(\boldsymbol{\sigma})$, respectively, then

$$\begin{cases} \varphi_{L=3}(\boldsymbol{\sigma}) = \varphi_{L=6}(\boldsymbol{\sigma}) = 0 \mid \sigma_2 = \omega_{23}(\sigma_1) & \forall \sigma_1 \\ \varphi_{R=3}(\boldsymbol{\sigma}) = \varphi_{R=6}(\boldsymbol{\sigma}) = 0 \mid \sigma_2 = \omega_{12}(\sigma_1) & \forall \sigma_1. \end{cases} \tag{5.20}$$

Proof. By substituting (5.16) and (5.17) into (5.13), it is possible to observe that $\varphi_{R=3}(\sigma_1, \omega_{12}(\sigma_1)) = 0 \wedge \varphi_{R=6}(\sigma_1, \omega_{12}(\sigma_1)) = 0 \wedge \varphi_{L=3}(\sigma_1, \omega_{23}(\sigma_1)) = 0 \wedge \varphi_{L=6}(\sigma_1, \omega_{23}(\sigma_1)) = 0$.

□

Theorem 5.3.3 (Continuity of interval type-2 FM)**.** If $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$ indicates FM of DI-IT2-FLS, then $\varphi^{\text{IT2}}(\boldsymbol{\sigma})$ is a continuous function in the region $[-1, 1]^2$ w.r.t. its input variable $\boldsymbol{\sigma}$, i.e., $\varphi^{\text{IT2}} \in \mathcal{C}^0([-1, 1]^2)$.

Proof. As can be observed from (5.13), no vertical asymptotes exist in $\varphi_{L=3}(\boldsymbol{\sigma})$, $\varphi_{L=6}(\boldsymbol{\sigma})$, $\varphi_{R=3}(\boldsymbol{\sigma})$ and $\varphi_{R=6}(\boldsymbol{\sigma})$ in their domains of definition $\Omega_1 \cup \Omega_2$, Ω_3, Ω_1 and $\Omega_2 \cup \Omega_3$, respectively. Namely, $\lim_{\boldsymbol{\sigma} \to \mathbf{c}} \varphi_{L=3}(\boldsymbol{\sigma}) = \varphi_{L=3}(\mathbf{c}) \quad \forall \boldsymbol{\sigma} \in \Omega_1 \cup \Omega_2 \wedge \lim_{\boldsymbol{\sigma} \to \mathbf{c}} \varphi_{L=6}(\boldsymbol{\sigma}) = \varphi_{L=6}(\mathbf{c}) \quad \forall \boldsymbol{\sigma} \in \Omega_3 \wedge \lim_{\boldsymbol{\sigma} \to \mathbf{c}} \varphi_{R=3}(\boldsymbol{\sigma}) = \varphi_{R=3}(\mathbf{c}) \quad \forall \boldsymbol{\sigma} \in \Omega_1 \wedge \lim_{\boldsymbol{\sigma} \to \mathbf{c}} \varphi_{R=6}(\boldsymbol{\sigma}) = \varphi_{R=6}(\mathbf{c}) \quad \forall \boldsymbol{\sigma} \in \Omega_2 \cup \Omega_3$. Therefore, $\varphi_{L=3}(\boldsymbol{\sigma})$ is continuous on $\Omega_1 \cup \Omega_2$, $\varphi_{L=6}(\boldsymbol{\sigma})$ is continuous on Ω_3, $\varphi_{R=3}(\boldsymbol{\sigma})$ is continuous on Ω_1 and $\varphi_{L=6}(\boldsymbol{\sigma})$ is continuous on $\Omega_2 \cup \Omega_3$.

Besides, by using Lemma 5.3.2, $\lim_{\boldsymbol{\sigma}\to\mathbf{c}} \varphi_{L=3}(\boldsymbol{\sigma}) = \lim_{\boldsymbol{\sigma}\to\mathbf{c}} \varphi_{L=6}(\boldsymbol{\sigma}) = 0 \quad \forall \mathbf{c} = [c_1, c_2] \mid c_2 = \omega_{23}(c_1) \wedge \lim_{\boldsymbol{\sigma}\to\mathbf{c}} \varphi_{R=3}(\boldsymbol{\sigma}) = \lim_{\boldsymbol{\sigma}\to\mathbf{c}} \varphi_{R=6}(\boldsymbol{\sigma}) = 0 \quad \forall \mathbf{c} = [c_1, c_2] \mid c_2 = \omega_{12}(c_1)$. Thus, also the continuity on the border ω_{23} for $\varphi_L(\boldsymbol{\sigma})$ and on the border ω_{12} for $\varphi_R(\boldsymbol{\sigma})$ is proven. Therefore, $\varphi_L(\boldsymbol{\sigma})$ and $\varphi_R(\boldsymbol{\sigma})$ are continuous in the region $[-1, 1]^2$, i.e., $\varphi_L \in \mathcal{C}^0([-1, 1]^2) \wedge \varphi_R \in \mathcal{C}^0([-1, 1]^2)$.

Lastly, the Theorem of Continuous Functions states that "the sum of a finite number of continuous functions is a continuous function". From (5.12), $\varphi_1^{\mathrm{IT2}}(\boldsymbol{\sigma})$, $\varphi_2^{\mathrm{IT2}}(\boldsymbol{\sigma})$ and $\varphi_3^{\mathrm{IT2}}(\boldsymbol{\sigma})$ are sums of continuous functions $\varphi_{L=3}(\boldsymbol{\sigma})$, $\varphi_{L=6}(\boldsymbol{\sigma})$, $\varphi_{R=3}(\boldsymbol{\sigma})$ and $\varphi_{R=6}(\boldsymbol{\sigma})$. Then, also $\varphi_1^{\mathrm{IT2}}(\boldsymbol{\sigma})$, $\varphi_2^{\mathrm{IT2}}(\boldsymbol{\sigma})$ and $\varphi_3^{\mathrm{IT2}}(\boldsymbol{\sigma})$ are all continuous in the region $[-1, 1]^2$, i.e., $\varphi_1^{\mathrm{IT2}} \in \mathcal{C}^0([-1, 1]^2) \wedge \varphi_2^{\mathrm{IT2}} \in \mathcal{C}^0([-1, 1]^2) \wedge \varphi_3^{\mathrm{IT2}} \in \mathcal{C}^0([-1, 1]^2)$. From (5.12), $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$ is a combination of continuous functions $\varphi_1^{\mathrm{IT2}}(\boldsymbol{\sigma})$, $\varphi_2^{\mathrm{IT2}}(\boldsymbol{\sigma})$ and $\varphi_3^{\mathrm{IT2}}(\boldsymbol{\sigma})$. Therefore, $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$ is also a continuous function in the region $[-1, 1]^2$.

□

Corollary 5.3.3.1. If the control inputs to the double-input interval type-2 fuzzy PD (DI-IT2-FPD) controller are continuous, then the control output from the DI-IT2-FPD controller is also continuous.

Theorem 5.3.4 (Monotonicity of interval type-2 FM)**.** If $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$ indicates FM of DI-IT2-FLS, then $\varphi^{\mathrm{IT2}}(\boldsymbol{\sigma})$ is a monotonic increasing function in the region $[-1, 1]^2$ w.r.t. its input variables $\boldsymbol{\sigma}$, i.e., $\frac{\partial \varphi^{\mathrm{IT2}}}{\partial \sigma_1} \geq 0 \wedge \frac{\partial \varphi^{\mathrm{IT2}}}{\partial \sigma_2} \geq 0 \quad \forall \boldsymbol{\sigma} \in [-1, 1]^2 \quad \forall \alpha \in [0, 1]$.

Proof. Firstly, let's show that $\varphi(\boldsymbol{\sigma})$ is an increasing function w.r.t. σ_1 $\forall \boldsymbol{\sigma} \in [-1, 1]^2$. From 5.14, $\frac{\partial \varphi_1^{\mathrm{IT2}}}{\partial \sigma_1} = \frac{\alpha(\sigma_2+1)}{(\sigma_1+\sigma_2+\sigma_1\sigma_2-\alpha\sigma_1-\alpha\sigma_2-\alpha\sigma_1\sigma_2+1)^2} \wedge \frac{\partial \varphi_2^{\mathrm{IT2}}}{\partial \sigma_1} = \frac{\alpha\left(\alpha{\sigma_2}^2-{\sigma_2}^2+1\right)}{2(\sigma_1+\alpha+\sigma_1\sigma_2-\alpha\sigma_1-\alpha\sigma_1\sigma_2)^2} + \frac{\alpha\left(\alpha-2\sigma_2-{\sigma_2}^2+2\alpha\sigma_2+\alpha{\sigma_2}^2\right)}{2(\alpha-\sigma_2+\sigma_1\sigma_2+\alpha\sigma_2-\alpha\sigma_1\sigma_2)^2} \wedge \frac{\partial \varphi_3^{\mathrm{IT2}}}{\partial \sigma_1} = \frac{\alpha(1-\sigma_2)}{(\sigma_1\sigma_2-\sigma_2-\sigma_1+\alpha\sigma_1+\alpha\sigma_2-\alpha\sigma_1\sigma_2+1)^2}$. Consequently, $\frac{\partial \varphi_1^{\mathrm{IT2}}}{\partial \sigma_1} \geq 0 \quad \forall \sigma_1 \in [0, 1] \quad \forall \sigma_2 \in [0, 1] \quad \forall \alpha \in [0, 1]$ and $\frac{\partial \varphi_2^{\mathrm{IT2}}}{\partial \sigma_1} \geq 0 \quad (\forall \sigma_1 \in [-1, 0] \quad \forall \sigma_2 \in [0, 1] \vee \forall \sigma_1 \in [0, 1] \quad \forall \sigma_2 \in [-1, 0]) \wedge \forall \alpha \in [0, 1]$, which is its definition domain, and $\frac{\partial \varphi_3^{\mathrm{IT2}}}{\partial \sigma_1} \geq 0 \quad \forall \sigma_1 \in [0, 1] \quad \forall \sigma_2 \in [0, 1] \quad \forall \alpha \in [0, 1]$. Therefore, φ^{T2} is a monotonic increasing in the region $[-1, 1]^2$ w.r.t. σ_1. From the first result in Theorem 5.3.1, if φ^{T2} is a monotonic increasing in the region $[-1, 1]^2$ w.r.t. σ_1, then φ^{T2} is a monotonic increasing in the same region also w.r.t. σ_2. Therefore, φ^{T2} is a monotonic increasing in the region $[-1, 1]^2$ w.r.t. both σ_1 and σ_2.

□

5.4 Simulation Results

For the dynamical simulations, the Y6 coaxial hexacopter is implemented in ROS environment and Gazebo simulator which provides a seamless connection for the developed algorithms between the simulation and real-world applications. Various FOU parameters settings (PSs) are investigated to validate the theoretical analysis. The following PSs are selected: PS-1: $\alpha = 0.09$, PS-2: $\alpha = 0.25$, and PS-3: $\alpha = 0.81$. For PS-1, it is expected that the resulting DI-IT2-FPD controller will have a fast response time. However, the control system might not be robust against nonlinearities and uncertainties. For PS-2, it is expected that the resulting DI-IT2-FPD controller controller will increase the damping when the error is small which will enhance the system response. On the other hand, the controller should decrease the damping when the error is relatively large. For PS-3, it is expected that the resulting DI-IT2-FPD controller will be potentially more robust against parameter variations and disturbances. However, the controller might have a slower response time.

5.4.1 Trajectory

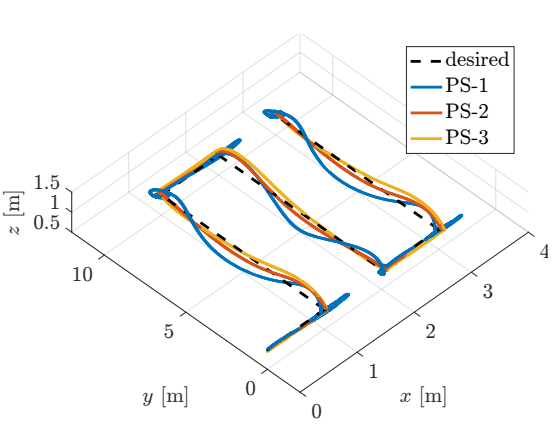
In the simulation scenario, a square-wave 3D trajectory is chosen to test the stability and robustness of each controller with different PSs. The navigation of the UAV combines long and short straight lines path as well as hovering:

$$\begin{cases} x_k^* &= \left\lfloor \frac{k}{2} \right\rfloor \\ y_k^* &= 10 \left\lfloor \frac{(k-1) \bmod 4}{2} \right\rfloor \\ z_k^* &= 1, \end{cases} \tag{5.21}$$

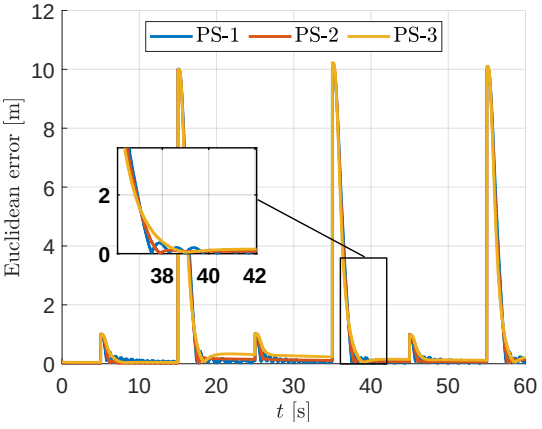
where $k \in \mathbb{N}^+$ and $\lfloor \star \rfloor$ is the largest integer not greater than value $\star$. First, UAV hovers at $[0, 0, 1]$m. After that, UAV flies to the next way-point at $[1, 0, 1]$m and hovers for 10s before flying to the next way-point. This type of trajectory is often used in autonomous UAV mapping and exploration scenarios.

5.4.2 Discussion

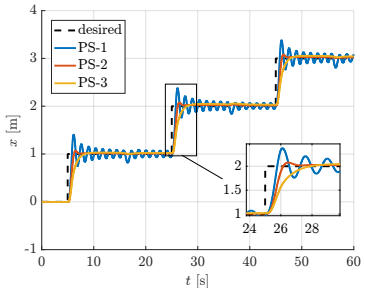
The 3D trajectory tracking of DI-IT2-FPD controllers with PS-1, PS-2 and PS-3 is shown in Fig. 5.7a; while the Euclidean error is shown in Fig. 5.7b. The position responses projected on x, y and z axes are shown in Figs. 5.7c–5.7e. As can be seen from Fig. 5.7, the controller with PS-1 has an oscillatory behaviour, while the controller with PS-3 is relatively slow in converging to the desired value. On the other hand, the controller with PS-2 combines the characteristics of both controllers with PS-1 and PS-3; it is fast with smaller overshoot and no oscillations. The response properties are also given in Table 5.1. As can be seen from Table 5.1, DI-IT2-FPD with PS-1 has shorter rising time but longer settling time and overshoot, while DI-IT2-FPD with PS-3 has smaller overshoot but longer rise time. What is more, DI-IT2-FPD with PS-2 results in the lowest mean squared error value and settling time.



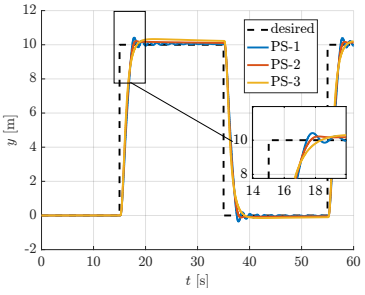
(a) 3D trajectory tracking.



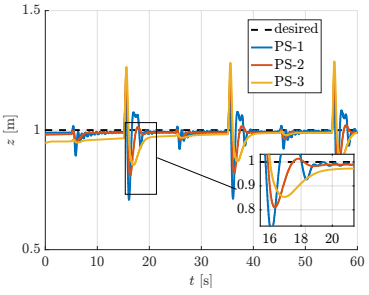
(b) Euclidean error.



(c) x-axis tracking.



(d) y-axis tracking.



(e) z-axis tracking.

Figure 5.7: Trajectory tracking of DI-IT2-FPD position controllers with different PSs.

Table 5.1: Properties of different controllers.

DI-IT2-FPD controller	**PS-1**	**PS-2**	**PS-3**
Mean squared error, [m]	0.777	**0.748**	0.821
Overshoot, [m]	0.405	0.069	**0.031**
Rise time, [s]	**0.80**	1.18	3.32
Settling time (5%), [s]	-	**1.71**	2.53

On the other hand, from the analysis in Fig. 5.5, PSs with small value of α, e.g., PS-1, should result in a more aggressive behaviour. Moreover, PSs with high value of α, e.g., PS-3, should result in a smoother behaviour. Furthermore, PSs with intermediate value of α, e.g., PS-2, should result in a moderate behaviour. Therefore, it can be concluded that the simulation results match with the theoretical expectations.

5.5 Experimental Results

The experimental flight tests are conducted in the motion capture system, which provides in real-time the quadcopter's position: x, y and z coordinates. The OptiTrack cameras are able to recognise a particular object according to the pattern of the reflective markers fixed on the object. The cameras provide the estimated position at a rate of 100Hz. Next, the control signal is computed by the ground station (CPU: 2.6GHz, 64bit, quad-core; GPU: 4GB; RAM: 16GB DDR4) and sent to the quadrotor at a rate of 100Hz.

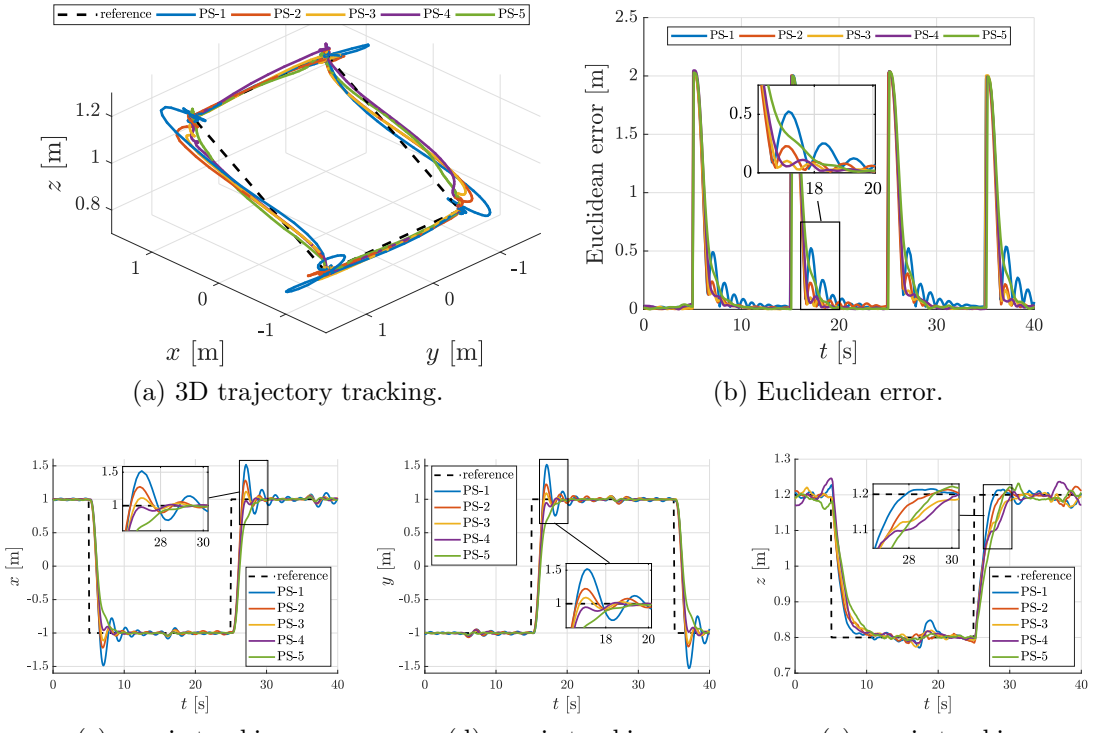
5.5.1 Trajectory

In the experimental scenario, a slanted square-shaped 3D trajectory with 2m square's side, shown in Fig. 5.8a, is chosen to test different controllers. This trajectory is designed to combine several manoeuvres which include hovering, straight line path, climbing and descending motion. The trajectory includes four way-points, located at $\{[1.0, -1.0, 1.2], [-1.0, -1.0, 0.8], [-1.0, 1.0, 0.8], [1.0, 1.0, 1.2]\}$m. Initially, UAV hovers at $[1, -1, 1.2]$m. Then, it starts flying towards the next way-point located at $[-1.0, -1.0, 0.8]$m where it hovers for 10s before moving to the next way-point.

Remark 5.6. In the considered case study, the maximum position error is 2m, therefore, the proportional input scaling factor is $k_p = \frac{1}{2}$; while the maximum speed error is 2m/s, so the derivative input scaling factor is $k_d = \frac{1}{2}$. In addition, the denormalization gain is tuned by trial-and-error method and set to $k_o = 3$.

5.5.2 Discussion

The results of 3D trajectory tracking of the designed DI-T1-FPD controller and DI-IT2-FPD controllers with PS-1, PS-2, PS-3, PS-4 and PS-5 are plotted in Fig. 5.8a; while the Euclidean error is shown in Fig. 5.8b. The position responses projected on x, y and z axes are shown in Figs. 5.8c–5.8e. These figures show that DI-IT2-FPD with low α (PS-1) has a high overshoot with an oscillatory action, while DI-IT2-FPD with high α (PS-5) has no overshooting with relatively slow convergence to the desired value. At the same time, DI-IT2-FPDs with intermediate α (PS-2 and PS-3) combine the aspects of both smooth and aggressive controllers. They are fast in converging with low overshoots and small oscillations.



(a) 3D trajectory tracking.

(b) Euclidean error.

(c) x-axis tracking.

(d) y-axis tracking.

(e) z-axis tracking.

Figure 5.8: Trajectory tracking of different DI-IT2-FPD controllers in absence of wind.

Repeating the experiments ten times for each controller, Table 5.2 shows the calculated Euclidean MAE, MVCS for x and y axes, mean overshoot, mean rise time and mean settling time at 5% of the desired value. The MVCS is computed as

$$\text{MVCS} = \frac{1}{N_D - 1} \sum_{i=1}^{N_D - 1} \frac{|\theta^*_{i+1} - \theta^*_i| + |\phi^*_{i+1} - \phi^*_i|}{2}, \tag{5.22}$$

where N_D is the number of data samples, and θ^*_i and ϕ^*_i are commanded pitch and roll angles of UAV for the i-th sample, respectively. As can be observed from Table 5.2, DI-IT2-FPD controller with PS-3 has the lowest MAE value since it benefits of the best combination of aggressiveness, when UAV is far from the desired position, and smoothness, when UAV is close to the desired position. At the same time, DI-IT2-FPD controller with PS-5 has the lowest MVCS value and smallest overshoot since it generates smooth control commands. On the other hand, DI-IT2-FPD controller with PS-1 has higher overshoot but the smallest rise time. The settling time at 5% of the final value is the lowest for DI-IT2-FPD controller with PS-4 which undershoots the desired position and is fast to stabilize UAV.

To check the robustness of the designed controllers, wind disturbances have been introduced. The maximum wind gust is around 5m/s. Table 5.3 shows the average properties of DI-IT2-FPD controllers after ten experiments for each case. As can be observed, in the presence of wind the Euclidean MAE increases for all the controllers. However, DI-IT2-FPD controller with PS-3 has a good capability to capture the wind disturbance, and it results again in the lowest MAE value. The intensity of the control signal again is higher for more aggressive controllers. At the same time, DI-IT2-FPD controller with PS-4 has the smallest overshoot because this controller, similarly to DI-IT2-FPD controller with PS-3, has a good capability to capture the wind disturbance. The rise time increases for more aggressive controllers because the headwind hampers fast flight and the tailwind does not help to fly faster. While

Table 5.2: Properties of DI-IT2-FPD controllers in absence of wind.

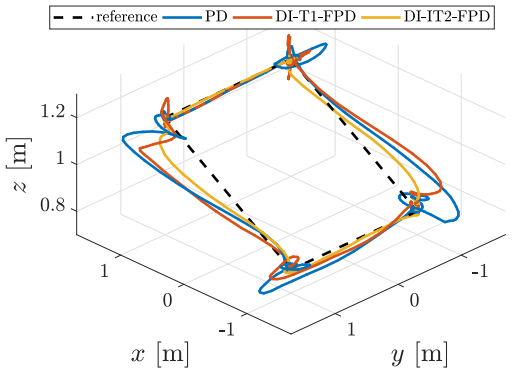
DI-IT2-FPD controller	**PS-1**	**PS-2**	**PS-3**	**PS-4**	**PS-5**
MAE, [m]	0.299	0.241	**0.228**	0.240	0.259
MVCS, [°]	0.314	0.077	0.072	0.041	**0.030**
Overshoot, [m]	0.515	0.230	0.115	0.023	**0.001**
Rise time, [s]	**1.58**	1.60	1.78	3.25	4.63
Settling time, [s]	4.70	2.58	2.13	**2.08**	3.00

Table 5.3: Properties of DI-IT2-FPD controllers in presence of wind.

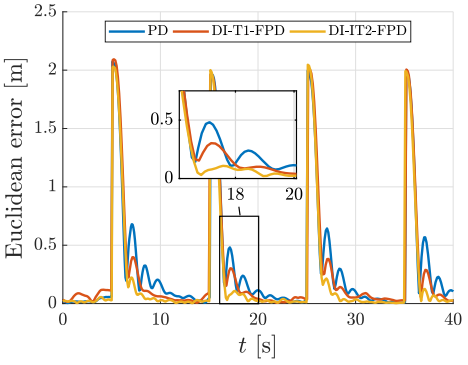
DI-IT2-FPD controller	**PS-1**	**PS-2**	**PS-3**	**PS-4**	**PS-5**
MAE, [m]	0.305	0.259	**0.238**	0.250	0.282
MVCS, [°]	0.379	0.105	0.090	0.068	**0.061**
Overshoot, [m]	0.500	0.268	0.120	**0.018**	0.028
Rise time, [s]	**1.60**	1.68	1.70	2.98	4.20
Settling time, [s]	4.88	2.83	2.10	**2.00**	2.78

the rise time decreases for smoother controllers because the headwind does not reduce the flight speed and the tailwind help to fly faster. For a similar reason, the settling time is larger for aggressive controllers and smaller for smooth controllers.

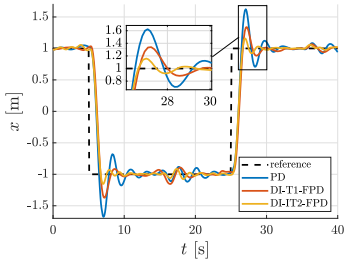
In addition, DI-T1-FPD and DI-IT2-FPD with PS-3 controllers are compared with the conventional PD controller. The results of 3D trajectory tracking of PD, designed DI-T1-FPD and DI-IT2-FPD with PS-3 position controllers are shown in Fig. 5.9a. The Euclidean error is shown in Fig. 5.9b for different controllers. The position (x, y and z) responses are shown in Figs. 5.9c–5.9e.



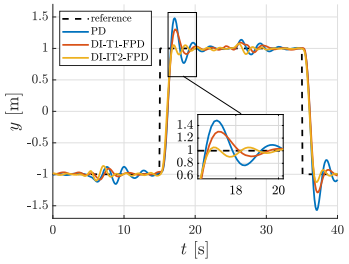
(a) 3D trajectory tracking.



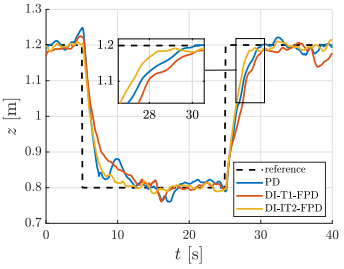
(b) Euclidean error.



(c) x-axis tracking.



(d) y-axis tracking.



(e) z-axis tracking.

Figure 5.9: Trajectory tracking of three different position controllers in presence of wind.

For the statistical analysis of control performances, the experiments are repeated ten times for each controller. To compare the trajectory tracking performances, a box-plot is presented in Figs. 5.10. It is possible to observe that on average DI-IT2-FPD controller with PS-3 has the lowest MAE and standard deviation on the tested trajectory when compared to other controllers.

Table 5.4 compares the characteristics of five controllers: standard PD, DI-T1-FPD which uses the standard type-1 fuzzy logic process, DI-T1-FPD* which uses directly FM in (5.8), DI-IT2-FPD which uses the standard interval type-2 fuzzy logic process and DI-IT2-FPD* which uses directly FM in (5.12). The average computation time for the traditional DI-T1-FPD and DI-IT2-FPD controllers is larger when compared to that of PD. Since in conventional FLCs, first, the input is fuzzified, then, it goes through the inference engine and, in the end, it is defuzzified. Moreover, in IT2-FLC, the FSs have to be reduced from type-2 to type-1 before the defuzzification. However, in DI-T1-FPD* and DI-IT2-FPD*, a direct FM is used which drastically reduces the computation time. The design, implementation and tuning of PD controllers are easy since it has only two parameters (k_p and k_d). On the other hand, DI-T1-FPD controller has three parameters (k_p, k_d and k_o) and DI-IT2-FPD controller has four parameters (k_p, k_d, k_o and α). Finally, DI-IT2-FPD controller with PS-3 results in the lowest MAE value computed from ten experiments.

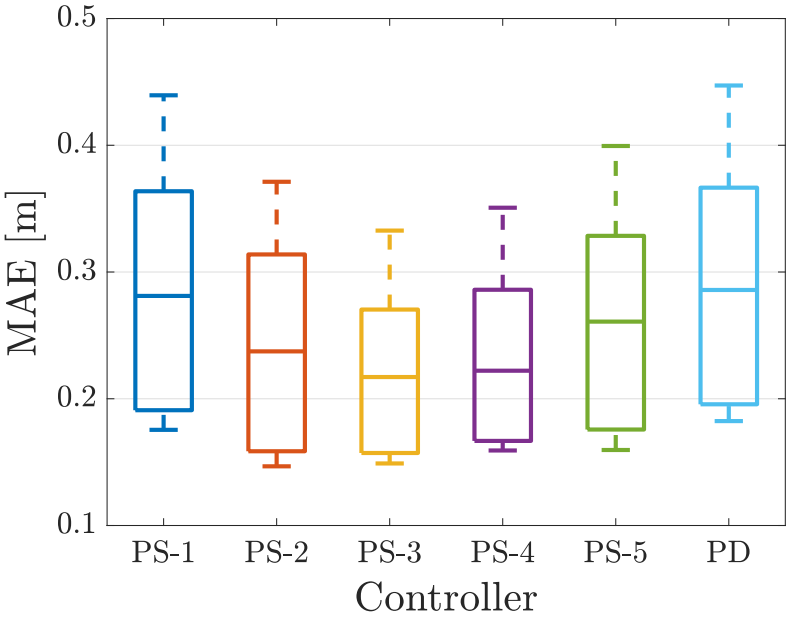


Figure 5.10: Box-plot of the tracking performances of six different controllers in presence of wind.

Table 5.4: Characteristics of different types of controllers.

Controller	**Computation time (average),** [ms]	**Number of parameters**	**MAE with wind** [m]
PD	**0.008**	**2**	0.314
DI-T1-FPD	1.356	3	0.282
DI-T1-FPD*	0.015		
DI-IT2-FPD	1.759	4	**0.238**
DI-IT2-FPD*	0.017		

From the experimental tests, it can be observed the following:

- For low values of α, DI-IT2-FPD controllers generate more aggressive control inputs. Consequently, in a real physical system, it results in higher overshoot but lower rising time. If the system is underdamped, it might result in oscillatory behaviour which increases the settling time.
- For high values of α, DI-IT2-FPD controllers generate smoother control inputs. Consequently, in a real physical system, it results in undershoot and higher rising time but no oscillations occur. Nevertheless, since the control action is not strong, in disturbed systems, the response will be strongly affected by these disturbances causing more overshoot/undershoot.
- For moderate values of α, DI-IT2-FPD controllers combines the characteristics of two cases above.

To summarize, the behaviour of DI-IT2-FPD controllers with small values of α, i.e., $0 < \alpha \ll 1$, is more aggressive around the desired position; while, the behaviour of DI-IT2-FPD controllers with big values of α, i.e., $0 \ll \alpha \leq 1$, is less aggressive around the desired position. These observations fully confirm the theoretical analysis in Subsection 5.3.2. Therefore, α can be called the *aggressiveness parameter*. Lastly, there is no universally good value of α which can satisfy all the cases. The optimal value of α depends on the specific application, controlled system and working environment.

5.6 Conclusion

In this chapter, the main focus is to design, deploy and analyse DI-T1-FLC and DI-IT2-FLC with various PSs. First of all, an alternative systematic approach to explicitly derive the mathematical input-output relationships of DI-T1-FLC and DI-IT2-FLC has been presented. These nonlinear closed-form relationships allowed to verify some important characteristics of both DI-T1-FLC and DI-IT2-FLC, like symmetry, continuity and monotonicity. Then, the design method for DI-IT2-FLC has been presented where only one parameter of FOU has to be selected, i.e., aggressiveness parameter α. By only modifying this parameter, DI-IT2-FLC controllers can be designed in an easy manner to have more aggressive or smoother behaviour. Besides, the developed controllers are computationally faster than the traditional FLCs. To prove these theoretical claims, different DI-IT2-FPD controllers with various PSs have been implemented in ROS. Then, the developed controllers have been tested, in simulation and experimental case studies, for the way-points tracking control of a quadcopter aircraft. Finally, it has been shown that the theoretical claims and expectations match the results in the case studies.

Part III

Neural Network-Based Control

Chapter 6

Artificial Neural Network-Based Control

By definition, ANNs are computing models which progressively improve their performance by learning from training examples [13]. Similarly to biological neural networks, ANNs are built by many simple processing elements, called neurons, which are interconnected by links, called synapses [14]. Hence, ANN learns from the training samples by adjusting the synaptic weights of the connections between neurons [15]. Moreover, ANNs reduce the need for feature engineering, which is one of the most time-consuming tasks in machine learning, for the training data [16]. Therefore, ANNs are ideal for situations that require approximating a function that depends on a huge number of inputs, which nonlinearly connects to the output [17]. Given the ability of ANNs to generalise knowledge from training samples, an ANN-based controller is suitable to control nonlinear systems [18].

In this chapter, potentials of ANNs are explored under various operational conditions. First, Section 6.1 revises the definition of ANN. Then, Sections 6.3 and 6.4 show simulation and experimental results for fast and agile flight with a motor failure case for a hexacopter UAV. Finally, the conclusions are drawn in Section 6.5.

Supplementary Material:
ROS package for the proposed ANN controllers: github.com/andriyukr/controllers.
Video for the experimental results: tiny.cc/fast_ANN.
Video for the experimental results: tiny.cc/failure_ANN.

6.1 Mathematical Preliminaries

In a general single-hidden-layer ANN, the neurons are organised in input layer with $(N_I + 1)$ neurons, hidden layer with $(N_H + 1)$ neurons, and output layer with N_O neurons. First, the input $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_{N_I}, 1\}$ is fed into the hidden layer of ANN through the network weights $\mathbf{W}_1 = \begin{bmatrix} w_{1,1,1} & \cdots & w_{1,1,N_H} \\ \vdots & \ddots & \vdots \\ w_{1,N_I+1,1} & \cdots & w_{1,N_I+1,N_H} \end{bmatrix} \in \mathbb{R}^{(N_I+1)\times N_H}$. Then, the output $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_{N_O}\}$ is computed by applying the network weights $\mathbf{W}_2 = \begin{bmatrix} w_{2,1,1} & \cdots & w_{2,1,N_O} \\ \vdots & \ddots & \vdots \\ w_{2,N_H+1,1} & \cdots & w_{2,N_H+1,N_O} \end{bmatrix} \in \mathbb{R}^{(N_H+1)\times N_O}$ to the output from the hidden layer. The weights in ANN are updated following a set of rules during the learning process.

Assumption 5. The network weights $\mathbf{W}_1$ and $\mathbf{W}_2$ are bounded, i.e.:

$$\begin{cases} \|\mathbf{W}_1(t)\|_\infty \leq c_{\mathbf{W}_1} \\ \|\mathbf{W}_2(t)\|_\infty \leq c_{\mathbf{W}_2} \end{cases} \quad \forall t, \tag{6.1}$$

where $c_{\mathbf{W}_1}$ and $c_{\mathbf{W}_2}$ are some positive constants.

In the proposed approach, the input is $\mathcal{I} = \{e, \dot{e}\}$, i.e., the position feedback error and its time derivative; while the output is $\mathcal{O} = \{u_{\text{ANN}}\}$, i.e., the control signal, as shown in Fig 6.1. Therefore, there are three input neurons ($N_I = 3$) and one output neuron ($N_O = 1$). The number of neurons in the hidden layer defines the learning capabilities of ANN. The neural networks with a single hidden layer are universal approximators [126], i.e., with a sufficient number of neurons, the network can learn any measurable function [127]. Typically, a smaller number of neurons may result in better generalisation in terms of different situations observed during tests; while a large number of neurons provides better convergency [128]. Therefore, an optimal number of neurons should be selected.

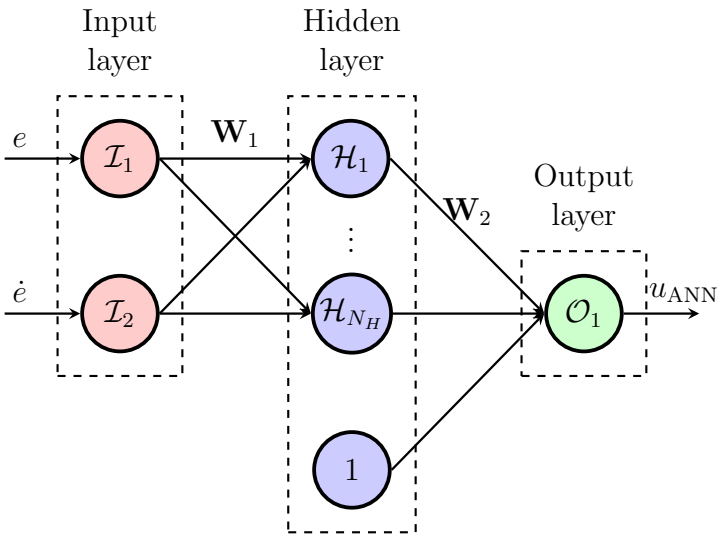


Figure 6.1: Structure of the proposed artificial neural network organised in input layer with two neurons, hidden layers with $N_H + 1$ neurons, and output layer with one neuron.

Assumption 6. The two input signals $e(t)$ and $\dot{e}(t)$, and their respective time derivatives $\dot{e}(t)$ and $\ddot{e}(t)$ are bounded [129], i.e.:

$$\begin{cases} |e(t)| \leq c_e \\ |\dot{e}(t)| \leq c_{\dot{e}} \\ |\ddot{e}(t)| \leq c_{\ddot{e}} \end{cases} \qquad \forall t, \tag{6.2}$$

where c_e, $c_{\dot{e}}$ and $c_{\ddot{e}}$ are some positive constants.

The control signal from ANN is computed as a linear combination of each input:

$$u_{\text{ANN}} = \frac{\sum_{j=1}^{N_H} \mathcal{H}_j w_{2,j,1}}{\sum_{j=1}^{N_H} \mathcal{H}_j} = \sum_{j=1}^{N_H} \bar{\mathcal{H}}_j w_{2,j,1}, \tag{6.3}$$

where $\bar{\mathcal{H}}_j$ is the normalized output from the j^{th} neuron in the hidden layer:

$$\bar{\mathcal{H}}_j = \frac{\mathcal{H}_j}{\sum_{h=1}^{N_H} \mathcal{H}_h}, \tag{6.4}$$

and

$$\mathcal{H}_j = \phi\left(\sum_{i=1}^{N_I} \mathcal{I}_i w_{1,i,j}\right), \tag{6.5}$$

where $j \in \{1, \ldots, N_H\}$ and ϕ is the scalar activation function. From Assumption 5 and (6.3), it is evident that $u_{\text{ANN}}(t)$ and $\dot{u}_{\text{ANN}}(t)$ are also bounded signals [130]:

$$\begin{cases} |u_{\text{ANN}}(t)| \leq c_u \\ |\dot{u}_{\text{ANN}}(t)| \leq c_{\dot{u}} \end{cases} \quad \forall t, \tag{6.6}$$

where c_u and $c_{\dot{u}}$ are some positive constants [131].

In the proposed control scheme, ANN works in parallel with a conventional PD controller, as shown in Fig. 6.2. The PD controller ensures the stability of the system in the initial phase of the learning process and acts as a feedback part of the controller providing sufficient time for ANN to initialize its learning process [93]. Thus, ANN will learn the control parameters and take over the control of the system. With its adaptive learning rates, ANN is very fast to learn and can instantaneously contribute to better performance, i.e., trajectory tracking accuracy.

Remark 6.1. One may note from Fig. 6.2 that the control output u is one-dimensional. The same control structure is used to generate all the four control signals described in (2.37), but only one is shown here for the sake of simplicity and to avoid repetition.

The overall control input u to the controlled system is defined by:

$$u = u_{\text{PD}} - u_{\text{ANN}}, \tag{6.7}$$

where u_{PD} and u_{ANN} are the control signals generated by PD and ANN controllers, respectively. The general PD control law is described as follows:

$$u_{\text{PD}} = k_p e + k_d \dot{e}, \tag{6.8}$$

where k_p and k_d are proportional and derivative gains, respectively.

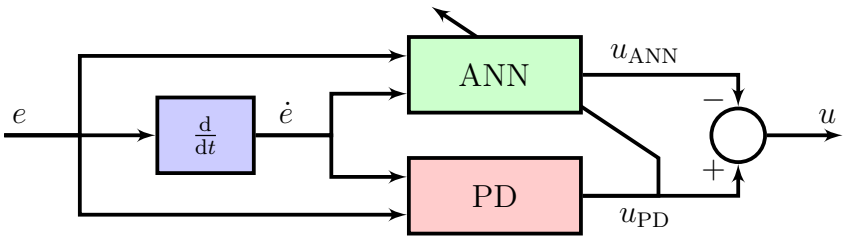


Figure 6.2: Control scheme: ANN in parallel with PD controller.

6.2 Sliding Mode Control-Based Learning

For the learning process of ANN, an SMC-based parameter adaptation scheme is used. The SMC framework is designed by selecting a suitable sliding manifold that will ensure the desired system dynamics. Moreover, to fulfil the sliding mode constraints/conditions, a dynamic feedback adaptation mechanism or, in other words, an online learning algorithm for ANN parameters has to be designed.

The difference between the measured output of the system and the output of the ANN can be defined as a time-varying sliding surface [132]. A time-varying sliding surface S can describe the zero value of the learning error coordinate $u_{\mathrm{PD}}(t)$ by using the theory of SMC [133]:

$$S(t) = u_{\mathrm{PD}}(t) = u_{\mathrm{ANN}}(t) + u(t) = 0. \tag{6.9}$$

Using the condition in (6.9), ANN is trained to obtain the desired response such that it becomes a nonlinear regulator that assists the conventional PD controller. Thus, the sliding surface for the nonlinear system under control is [134]:

$$S(e) = \left(\frac{\mathrm{d}}{\mathrm{d}t} + \lambda\right) e = \dot{e} + \lambda e, \tag{6.10}$$

where $\lambda > 0$ is a constant which determines the slope of the sliding surface. A sliding motion will occur on the sliding manifold $S(t) = u_{\mathrm{PD}}(t) = 0$ after a finite time t_h, if the condition $S(t)\dot{S}(t) = u_{\mathrm{PD}}(t)\dot{u}_{\mathrm{PD}}(t) < 0$ is satisfied for all t such that $[t, t_h) \subset (-\infty, t_h)$ in some nontrivial semi-open subinterval of time [135]. Consequently, $u_{\mathrm{ANN}}(t)$ is constrained to perfectly follow the desired output signal $u(t)$ for all $t > t_h$. The time instant t_h is the hitting time for the learning error $u_{\mathrm{PD}}(t) = 0$. For an arbitrary initial condition $u_{\mathrm{PD}}(0)$, $u_{\mathrm{PD}}(t)$ will eventually converge to a small neighbourhood of zero during a finite time t_h. Therefore, the adaptation laws for the parameters of ANN are given as follows:

$$\begin{cases} \dot{w}_{2,j,1} & = -\alpha \frac{\mathcal{H}_j}{\sum_{h=1}^{N_H} \mathcal{H}_h^2} \mathrm{sign}(u_{\mathrm{PD}}) \quad \forall j \in \{1, \ldots, N_H\} \\ \dot{\alpha} & = \gamma |u_{\mathrm{PD}}| - \gamma \nu \alpha, \end{cases} \tag{6.11}$$

where $\alpha > 0$ is an adaptive learning rate, $\gamma \geq 0$ and $\nu \geq 0$ are learning parameter. The pseudo-code of the ANN training is presented in Algorithm 1.

Algorithm 1: Online adaptation of ANN.

Input: e, $\dot{e}$, u_{PD}
Output: u_{ANN}
Data: N_H, α_0, γ, ν
Result: ANN controls the system online
begin
  ANN $\leftarrow$ ConstructNetworkLayers(2, N_H, 1)
  $\mathbf{W}_1 \leftarrow$ InitializeWeights()
  $\mathbf{W}_2 \leftarrow$ InitializeWeights()
  $\alpha \leftarrow \alpha_0$
  repeat
    Get e, $\dot{e}$ and u_{PD}
    $\mathcal{H}_j \leftarrow \phi\left(\sum_{i=1}^{N_I} \mathcal{I}_i w_{1,i,j}\right) \quad \forall j \in \{1, \ldots, N_H\}$
    $\bar{\mathcal{H}}_j \leftarrow \frac{\mathcal{H}_j}{\sum_{h=1}^{N_H} \mathcal{H}_h} \quad \forall j \in \{1, \ldots, N_H\}$
    $\dot{w}_{2,j,1} \leftarrow -\alpha \frac{\mathcal{H}_j}{\sum_{h=1}^{N_H} \mathcal{H}_h^2} \mathrm{sign}(u_{\mathrm{PD}}) \quad \forall j \in \{1, \ldots, N_H\}$
    $\dot{\alpha} \leftarrow \gamma |u_{\mathrm{PD}}| - \gamma \nu \alpha$
    $u_{\mathrm{ANN}} \leftarrow \sum_{j=1}^{N_H} \bar{\mathcal{H}}_j w_{2,j,1}$
    Send u_{ANN} to the system
  until *Stop*
end

Remark 6.2. In the adaptation laws in (6.11), the learning rate α is variable and its value evolves during the learning process. This adaptation law allows choosing a small initial value for α which, consequently, grows during the training phase.

Theorem 6.2.1 (Stability of ANN)**.** If the adaptation laws for the parameters are chosen as in (6.11), then learning error u_{PD} will converge to a small neighbourhood of zero in a finite time t_h for any arbitrary initial condition.

Proof. Let's consider the following Lyapunov function:

$$V = \frac{1}{2} u_{\mathrm{PD}}^2 + \frac{1}{2\gamma} (\alpha - \alpha^*)^2. \tag{6.12}$$

Hence, $V > 0$ for $(u_{\mathrm{PD}} \neq 0) \vee (\alpha - \alpha^* \neq 0)$ and $V = 0$ for $(u_{\mathrm{PD}} = 0) \wedge (\alpha - \alpha^* = 0)$.

Taking the time derivative of V:

$$\dot{V} = u_{\mathrm{PD}} \dot{u}_{\mathrm{PD}} + \frac{1}{\gamma} (\alpha - \alpha^*) \dot{\alpha} = u_{\mathrm{PD}} (\dot{u} + \dot{u}_{\mathrm{ANN}}) + \frac{1}{\gamma} (\alpha - \alpha^*) \dot{\alpha}, \tag{6.13}$$

in which

$$\dot{u}_{\text{ANN}} = \sum_{j=1}^{N_H} \left(\dot{\bar{\mathcal{H}}}_j w_{2,j,1} + \bar{\mathcal{H}}_j \dot{w}_{2,j,1} \right). \tag{6.14}$$

By replacing (6.14) into (6.13):

$$\dot{V} = u_{\text{PD}} \left(\dot{u} + \sum_{j=1}^{N_H} \dot{\bar{\mathcal{H}}}_j w_{2,j,1} + \sum_{j=1}^{N_H} \bar{\mathcal{H}}_j \dot{w}_{2,j,1} \right) + \frac{1}{\gamma}(\alpha - \alpha^*)\dot{\alpha}. \tag{6.15}$$

Considering the fact that $\sum_{j=1}^{N_H} \bar{\mathcal{H}}_j = 1 \implies \sum_{j=1}^{N_H} \dot{\bar{\mathcal{H}}}_j = 0$:

$$\dot{V} = u_{\text{PD}} \left(\dot{u} + \sum_{j=1}^{N_H} \bar{\mathcal{H}}_j \dot{w}_{2,j,1} \right) + \frac{1}{\gamma}(\alpha - \alpha^*)\dot{\alpha}. \tag{6.16}$$

By considering the adaptation laws of $\mathbf{W}_2$ in (6.11):

$$\begin{aligned} \dot{V} &= u_{\text{PD}} \left(\dot{u} - \alpha \text{sign}(u_{\text{PD}}) \right) + \frac{1}{\gamma}(\alpha - \alpha^*)\dot{\alpha} \\ &= |u_{\text{PD}}| B_{\dot{u}} - \alpha |u_{\text{PD}}| + \frac{1}{\gamma}(\alpha - \alpha^*)\dot{\alpha} \\ &= |u_{\text{PD}}| B_{\dot{u}} - (\alpha - \alpha^*)|u_{\text{PD}}| - \alpha^* |u_{\text{PD}}| + \frac{1}{\gamma}(\alpha - \alpha^*)\dot{\alpha}. \end{aligned} \tag{6.17}$$

By considering the adaptation laws of α in (6.11):

$$\dot{V} = |u_{\text{PD}}| B_{\dot{u}} - \alpha^* |u_{\text{PD}}| - \nu \left(\alpha - \frac{1}{2}\alpha^* \right)^2 + \frac{1}{4}\nu \alpha^{*2}. \tag{6.18}$$

Assuming $B_{\dot{u}} \leq \frac{1}{2}\alpha^*$, the following inequality is obtained:

$$\dot{V} \leq -\frac{1}{2}\alpha^* |u_{\text{PD}}| + \frac{1}{4}\nu \alpha^{*2}, \tag{6.19}$$

which implies that the Lyapunov function decreases until $u_{\text{PD}} < \frac{1}{4}\nu\alpha^*$ and u_{PD} remains bounded. Furthermore, ν is a design parameter and it is possible to select this value as small as desired. □

6.3 Simulation Results

At first, the controller is tested in simulation to determine the ANN's meta-parameters, like learning rate and number of neurons in the hidden layer. The Gazebo simulation environment is used to simulate and model the flying UAV, including a motor failure, because of its powerful and robust physics engine. A thorough analysis of the different number of neurons in the hidden layer is done to observe its effects on computation times and learning performance. The computation time is calculated for the different number of neurons, in a simple circular trajectory of 5m radius at reference speed of 1m/s, as shown in Table 6.1. Note that the time given in the table is the average computation time taken to run one loop of the ANN controller. It is evident from the study that no significant change is observed in terms of tracking improvement with the increase in the number of neurons, albeit the computation time multiplies.

As aforementioned, ANN begins to learn online from a pre-set learning rate, each time it is initialized and applies a correction to the model-based techniques. This allows keeping the original benefits of the control, including stability properties, while the proposed algorithm adds effort to improve performance metrics. Thus, any particular data set for a scenario is not fed to the controller to learn any specific trajectory, rather the controller is designed to perform better in every arbitrary condition. The main goal for the ANN-assisted controller is to learn in a very short time and perform better than the commonly used conventional controllers. Moreover, implementing a simple neural network with just a single hidden layer with few neurons is not sufficient to learn the complex system dynamics of UAV.

Table 6.1: Comparison of computation times and mean Euclidean error for different number of neurons in hidden layer.

Number of neurons	3	9	20	50	100	500
Computation time, [ms]	0.092	0.127	0.143	0.245	0.427	2.57
Mean Euclidean error, [ms]	1.55	1.52	1.56	1.55	1.58	1.69

6.4 Experimental Results

To validate the performance of the proposed controller, comprehensive real-time tests are conducted on a custom-made coaxial hexacopter, depicted in Fig. 2.1b. An additional motor failure relay is added to UAV, which helps to trigger the motor failure on-demand from the radio transmitter. The tests are performed in an outdoor environment with the use of the real-time kinematic (RTK)-GPS, which provides the x, y and z position information with an accuracy of approximately $5 - 20$cm at 5Hz. It is to be noted that the experiments were conducted with average wind gusts of 5m/s.

The onboard computer running all the codes in C++ on ROS is a low-cost and low-power Odroid XU4 – thus, allowing the system to be autonomous. The main constraint being the computation power available on the onboard computer, desirable three sets of neurons are selected from hardware-in-the-loop simulation to ensure that sensible computation is utilised in real-time. The position information from RTK-GPS together with the data from the inertial measurement unit is fed into the local position estimator which estimates the pose of UAV. This information is used by the controller to compute the control signal and provide it to UAV through a 5GHz wireless network.

The goal is to achieve a great performance using the proposed controller in challenging and previously unknown trajectories, for which a perfectly tuned set of PID gains can not be determined. The ANN-assisted conservative PD controller is employed to perform trajectory tracking. The advantage is that since the ANN starts learning online each time from scratch when initialized, it converges faster and better when compared to other controllers as shown in the results. The results of the ANN controller are compared with two widely used position controllers. One is the position controller of the Pixhawk autopilot stack [136, 137], referred as PID_{FCU}; while the other is a standard PID position controller, referred as PID_{pos}, which sends attitude-setpoints – roll, pitch, and yaw angles – and thrust commands. The same experimental scenario is repeated with a different controller each time. Note that all the iterations with different controllers were carried out in similar outdoor conditions.

6.4.1 Fast and Agile Flight

A trajectory with two segments – zig-zag and straight line – is chosen to make UAV experience both agile and fast manoeuvres at high speeds. In the first segment of the trajectory, UAV follows a zig-zag path for 55m along x-axis and a periodic change of $\pm$5m along y-axis at a target speed of 5m/s. Then, in the second segment, UAV follows a straight-line path at the target speed of 15m/s for another 70m.

Table 6.2 gives a brief overview of the experiments carried out for the different controllers numerous times on various trajectories. In particular, a zig-zag path at high speed and a simple circular trajectory at nominal speed are traced. The zig-zag path is a pattern stretching 30m along x-axis and $\pm$5m along y-axis, while the latter is a circle of 2m radius circling three times. The experiments are performed for each of the three controllers in discussion and are repeated twice for the sake of repeatability. An overall improvement of the proposed ANN-based controller can be observed and that its performance is independent on the chosen trajectory.

The results plotted in Fig. 6.3 show the trajectory tracking of the UAV in 3D space over time. The wind gusts and the high speeds of the UAV exert huge stresses on the rotors, thus slight deviations from the trajectory are inevitable. The slight deviation along z-axis towards the end of the trajectoryis because of the tilting thrust vector of the UAV, reducing the vertical component of thrust compared to the weight of the UAV. As seen from the top view of the trajectory in Fig. 6.4, the maximum deviation from the trajectory in case of ANN is about 1m and 5m in any direction for the zig-zag and straight-line parts, respectively. However, for PID_{FCU} and PID_{pos} it is about 2.5m and 2m for zig-zag and 13m and 9m for straight-line parts, respectively. Moreover, even at high speeds and very sharp turns, ANN tracks the trajectory to the closest point on the bends.

Table 6.2: Statistical comparison of three controllers.

Trajectory	**Controller**	**Mean Euc. error**, [m]	**MAE**, [m]	σ
Zig-zag	PID_{FCU}	1.331	5.378	2.178
	PID_{pos}	1.022	4.942	1.834
	ANN-PD	0.861	4.550	1.721
Circle	PID_{FCU}	1.147	1.776	0.396
	PID_{pos}	1.042	1.617	0.542
	ANN-PD	0.511	0.757	0.299

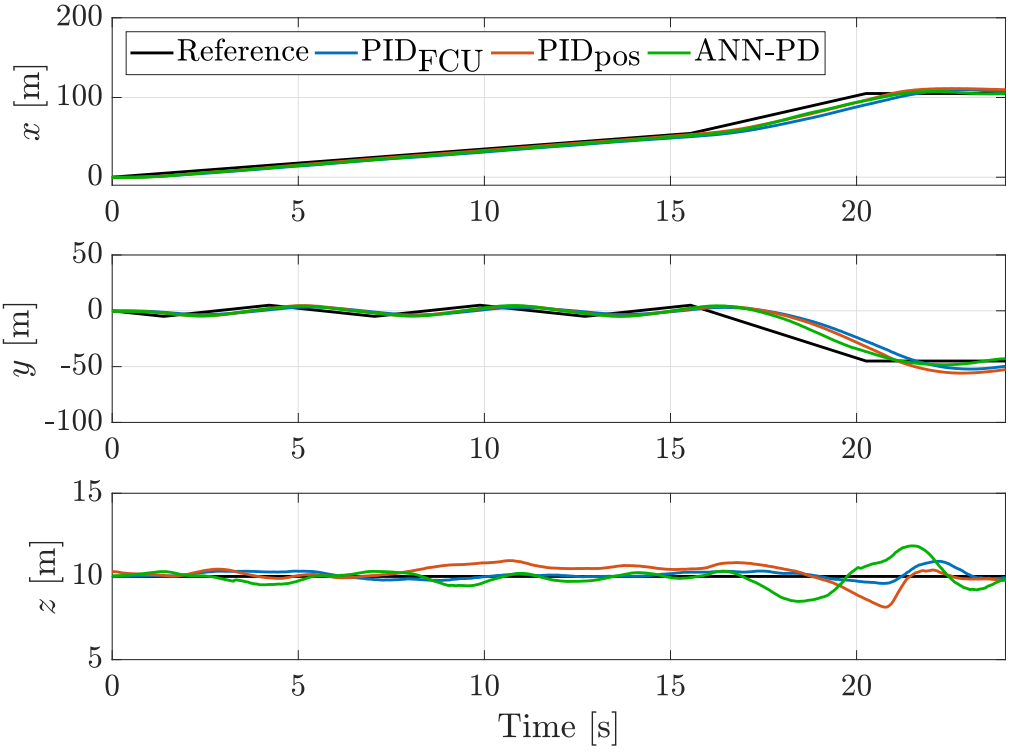


Figure 6.3: Real-time trajectory tracking of the UAV.

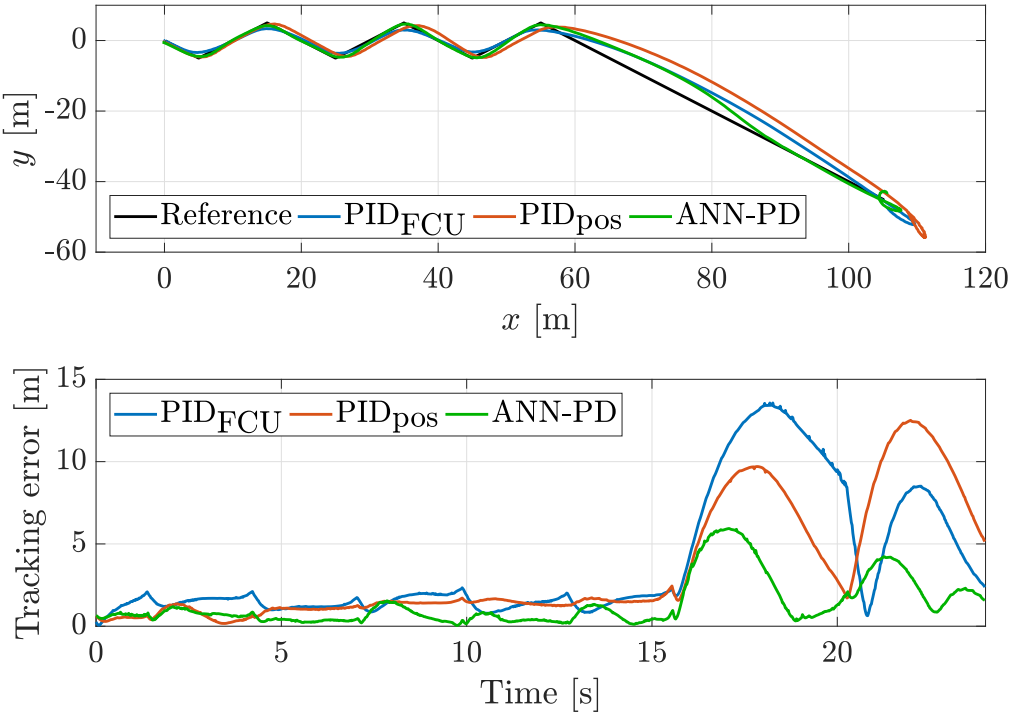


Figure 6.4: Top view and tracking error of the considered controllers.

A common parameter usually calculated as a performance comparison metric is the Euclidean error in (3.12). Specifically in the trajectory tracking problem, using the

Euclidean error alone can penalize the algorithm because the delay in following the trajectory is taken into account and not how accurately it is following the command [138]. Thus, the overall tracking error in x, y, and z axes, plotted in Fig. 6.4, shows how closely or accurately the actual path is followed despite such tight constraints. The improvement achieved by ANN in terms of tracking error is 63% and 60% compared to PID_{FCU} and PID_{pos}, respectively. Even on the straight-line segment of the trajectory, it converges to the actual trajectory despite the initial deviation. Considering the high speeds and attitude angles attained during the entire 155m long trajectory, the error for ANN is significantly smaller. At the end of the straight path, when UAV is travelling close to 18m/s, it has to come to a halt, which physically it is not possible to stop in an instant – thus, the overshoot at the end and then UAV converges to hover states. The ground speed achieved for the different controllers is compared in Fig. 6.5. As shown in the acceleration plot in Fig. 6.5, ANN is the fastest to accelerate and complete the trajectory. The ANN-assisted controller is able to maintain stable flight while reaching peak velocities of 18m/s and attitude angles of 45° during the trajectory. Highest average speeds are observed for ANN during the zig-zag path as ANN follows the trajectory to minimize the tracking error. The control output signals of

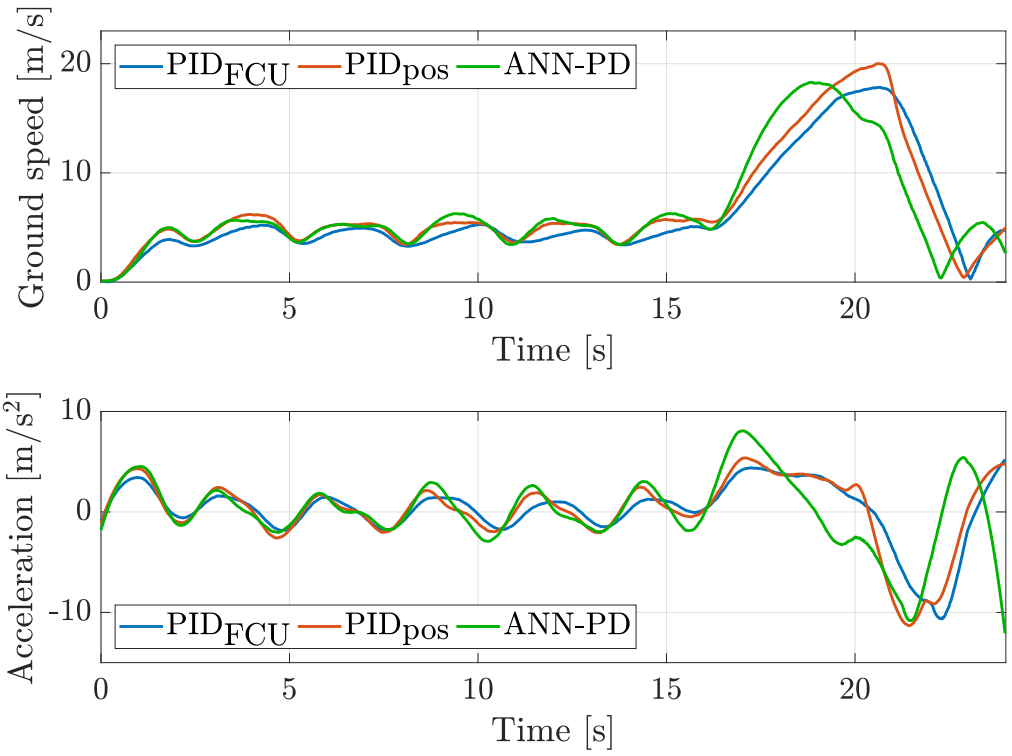


Figure 6.5: Ground speed and acceleration of the considered controllers.

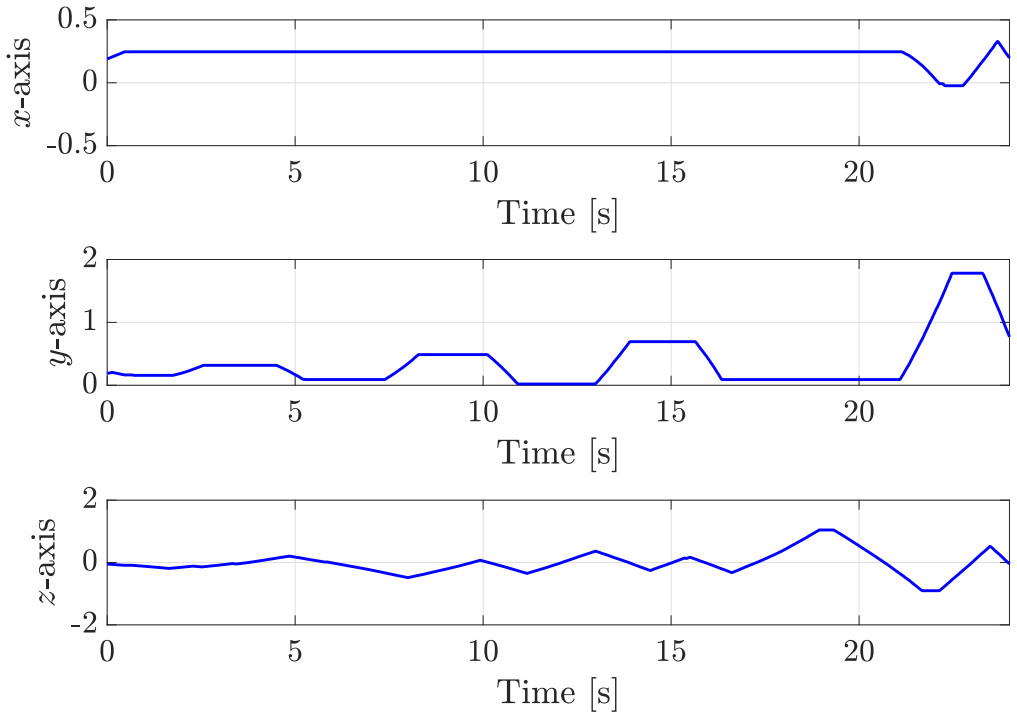


Figure 6.6: Control signals of ANN for x, y and z axes.

the ANN controller working in parallel with PD controller are presented in Fig. 6.6 for the x, y and z axes.

To summarise, it is shown that ANN accelerates faster to follow the desired trajectory and results in the best trajectory tracking among the three considered controllers. The learning capability of ANN helps to minimize the tracking error over time and provide superior performance. The mean absolute tracking error for each of the three controllers along with the maximum speed and acceleration attained are given in Table 6.3.

Table 6.3: MAE, maximum speed and maximum acceleration achieved for the considered controllers.

Controller	**MAE**, [m]	**Max. speed**, [m/s]	**Max. acceleration**, [m/s^2]
PID_{FCU}	3.749	17.8	5.222
PID_{pos}	3.429	20.0	5.373
ANN-PD	1.356	18.3	8.065

6.4.2 Motor Failure

Safe operation of UAVs has a high priority as most of them operate in human populated areas [139]. A long straight manoeuvre is chosen to track the motion at high-speeds reaching 20m/s for the fast flight of UAV. Then, UAV starts the mapping trajectory with 3m × 5m dimensions, after reaching the destination area, at a speed of 2m/s. During this part of the trajectory, UAV experiences a motor failure and, yet, continues to complete the trajectory and land safely at the end.

The plot in Fig. 6.7 shows the trajectory tracking of the UAV in 3-dimensional space over time. The plots are shaded in two colours, the initial green phase shows the motors are running properly and the red phase starting at 21s mark shows the flight with motor failure. The motor failure is triggered when the UAV is following the mapping part of the trajectory. A slight change in z axis seen from the plot is the initial drop in height due to the instantaneous loss of thrust. The ANN controller learns it as a disturbance and compensates for the loss. The integral term of the PID controller also tries to minimize the steady-state error but ANN-PD is more effective. The UAV lands at the end with the motor failure state and a huge lag in the PID_{FCU}'s capability to land can be seen. On the other hand, the ANN-based controller and PID_{pos} are more effective at landing compared to the former. It is to be noted that the landing height is not exactly 0m, but slightly below that, as

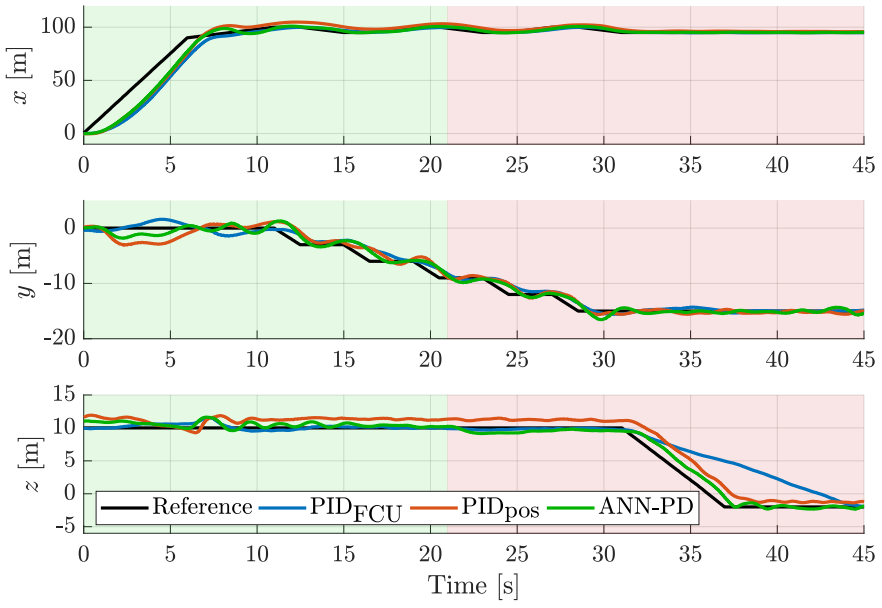


Figure 6.7: Position tracking performance of various controllers

the field where the experiments are carried out is not an even surface – thus, the landing point is below the datum of the take-off point.

The high speeds, experiencing motor failure, and the wind gusts exert tremendous stresses on the rotors and the UAV inertial dynamics, and thus small deviations from the trajectory are inevitable. The overview of the trajectory as seen from the top is depicted in Fig. 6.8. It can be noted that the PID_{pos} has more deviations from the actual path and the ANN controller follows the sharp bends more effectively, thus minimizing the overall error. In the trajectory tracking problem, the Euclidean error is usually calculated to determine the controller's performance, but it may penalize the algorithm as it takes into account the time delay in following the trajectory and not how accurately it is following [140]. Thus, the overall tracking error in the x, y, and z axes showing the UAV's capability of following the actual path is shown in Fig. 6.9. The ANN controller is able to achieve an overall improvement of 41% and 55% when compared to the PID_{FCU} and PID_{pos}, respectively, for the entire stretch of trajectory. Keeping in mind the high speeds and the high attitude angles achieved during the trajectory, the error for ANN is quite small.

The ground speed of the UAV, during the trajectory, for the various considered controllers is shown in Fig. 6.10. The ANN accelerates faster and thus tracks the trajectory better than the other two controllers. The acceleration plot is also shown in Fig. 6.10. The UAV follows the mapping part of the trajectory at a speed of 2m/s with a motor failure. The ANN controller working in parallel with a PD controller is able to stabilize the flight during this scenario and still able to follow the desired path closely. The UAV lands at the end of the trajectory at a defined location.

The control output of the ANN controller for the x, y, and z axes is shown in

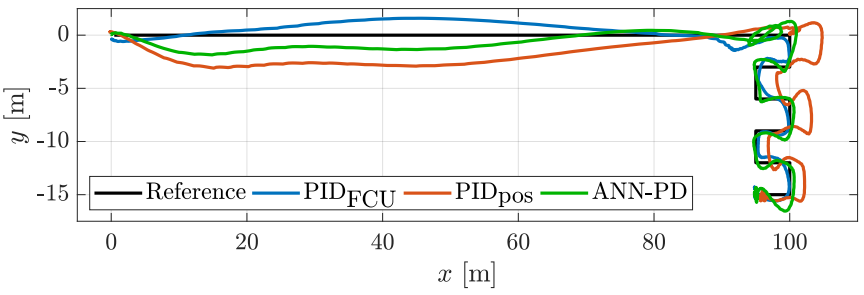


Figure 6.8: Top view comparison of various controllers

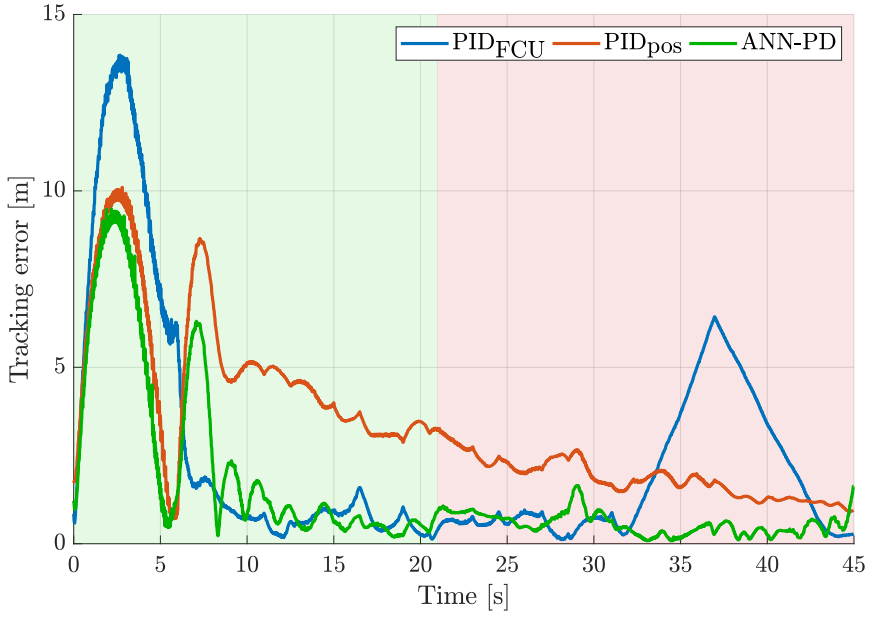


Figure 6.9: Tracking error comparison of various controllers

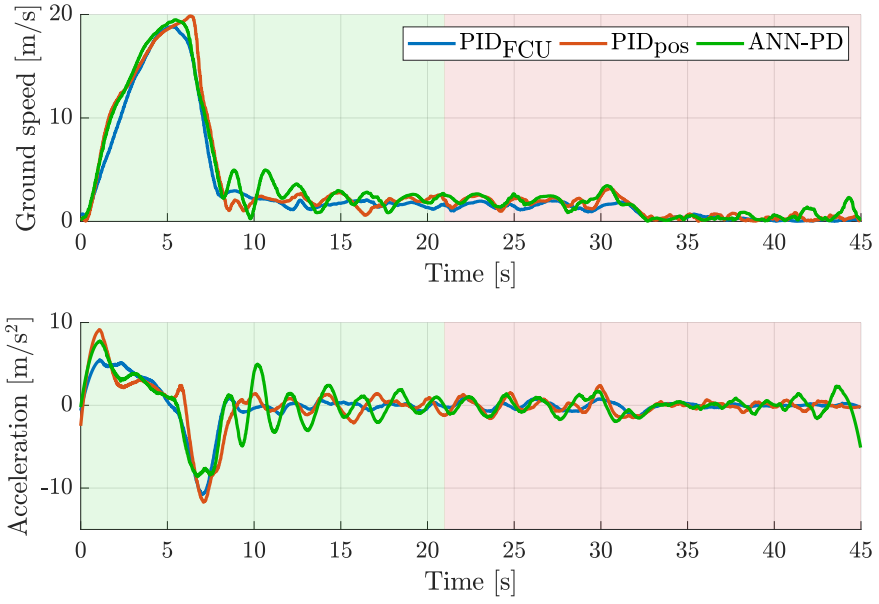


Figure 6.10: Ground speed and acceleration of different controllers

Fig. 6.11. Note the sudden increase in the control output of z axis as the motor is failed at the instant of 21s. The performance characteristics of the controllers are given in Table 6.4.

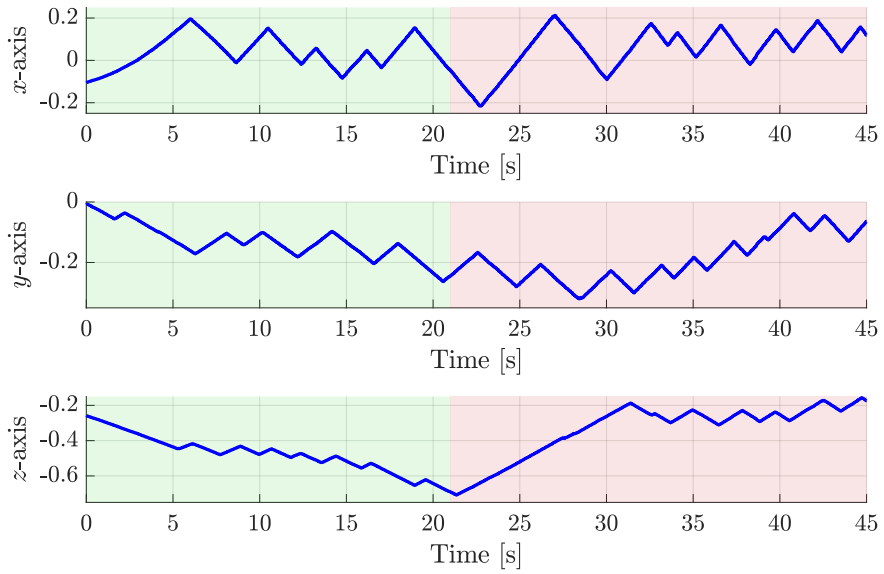


Figure 6.11: Control output of ANN controller for x, y, and z axes

6.5 Conclusion

In this chapter, an ANN-assisted PD controller is proposed for the UAV's control in various challenging conditions. A fast flight manoeuvre at speeds of 18m/s is performed to show the superior performance of the proposed controller. While performing a predefined task, the UAV experiences a single motor failure, and the proposed controller handles the failure ensuring the safety of the mission as well as the UAV. The model-free nature of the controller helps in accurate trajectory tracking even for high speed and agile manoeuvres. The advantage of the proposed controller is that it does not need a well-tuned set of PD gains as it learns online and improves the performance metrics while following the trajectory. Moreover, the proposed controller is computationally cheap to be implemented on the onboard computer. The real-time experiments show that for all the phases of the considered scenario the proposed controller outperforms the conventional PID controllers.

Table 6.4: MAE, maximum speed and maximum acceleration achieved for the considered controllers.

Controller	**MAE**, [m]	**Max. speed**, [m/s]	**Max. acceleration**, [m/s^2]
PID_{FCU}	5.564	18.81	5.50
PID_{pos}	6.653	19.86	9.13
ANN-PD	4.288	19.49	7.76

Chapter 7

Deep Neural Network-Based Control

Though ANN are able to generalise knowledge from training samples, common single-hidden-layer ANNs can approximate effectively only simple nonlinear functions, while real-world systems are frequently highly nonlinear [141]. On the other hand, DNNs which are distinguished from the more commonplace single-hidden-layer ANNs by their depth that is the number of layers through which data must pass in a multi-step process [20]. Thus, DNNs can approximate non-linear functions with an exponentially lower number of training parameters and higher sample complexity when compared to ANNs [22]. Therefore, DNNs propose a novel approach to enhance the control strategies for nonlinear systems [23]. After training the DNN module on collected flight samples, it can be used in real-time to provide the control signal [24].

In this chapter, the potentials of DNNs are explored under various operational conditions. First, Section 7.1 revises the definition of DNN. Next, Section 7.2 provides theoretical results related to the transfer learning problem. Then, Sections 7.3 and 7.4 show simulation and experimental results for linear systems and two types of quadrotor UAVs, respectively. Finally, the conclusions are drawn in Section 7.5.

Supplementary Material:
Video for the experimental results: tiny.cc/DNN.

7.1 Mathematical Preliminaries

In a general DNN, the neurons are organised in input layer with N_I neurons, N_L hidden layers with $N_{H,h}$, $h \in \{1,\dots,N_L\}$, neurons in each layer, and output layer with N_O neurons, as shown in Fig 7.1. First, the input $\mathcal{I} = \{\mathcal{I}_1,\dots,\mathcal{I}_{N_I}\}$ is forwarded to the first hidden layer of the network through the network weights $\mathbf{W}_1 = \begin{bmatrix} w_{1,1,1} & \cdots & w_{1,1,(N_{H,1}-1)} \\ \vdots & \ddots & \vdots \\ w_{1,N_I,1} & \cdots & w_{1,N_I,(N_{H,1}-1)} \end{bmatrix} \in \mathbb{R}^{N_I \times (N_{H,1}-1)}$. Commonly, the hidden layers of DNN are systematised in a fully connected structure with the network weights $\mathbf{W}_h = \begin{bmatrix} w_{h,1,1} & \cdots & w_{h,1,(N_{H,h}-1)} \\ \vdots & \ddots & \vdots \\ w_{h,N_{H,h-1},1} & \cdots & w_{h,N_{H,h-1},(N_{H,h}-1)} \end{bmatrix} \in \mathbb{R}^{N_{H,h-1}\times(N_{H,h}-1)}$, $h \in \{2,\dots,N_L\}$. Finally, the output $\mathcal{O} = \{\mathcal{O}_1,\dots,\mathcal{O}_{N_O}\}$ is computed by using the network weights $\mathbf{W}_{N_L+1} = \begin{bmatrix} w_{N_L+1,1,1} & \cdots & w_{N_L+1,1,N_O} \\ \vdots & \ddots & \vdots \\ w_{N_L+1,N_{H,N_L},1} & \cdots & w_{N_L+1,N_{H,N_L},N_O} \end{bmatrix} \in \mathbb{R}^{N_{H,N_L}\times N_O}$ to the output from the last hidden layer. The weights in DNN are updated following a set of rules during the learning process.

Assumption 7. The network weights $\mathbf{W}_i$, $i \in \{1,\dots,N_L+1\}$, are bounded, i.e.:

$$\|\mathbf{W}_i(k)\|_\infty \leq c_{\mathbf{W}_i}, \quad i \in \{1,\dots,N_L+1\} \quad \forall k, \tag{7.1}$$

where $c_{\mathbf{W}_i}$, $i \in \{1,\dots,N_L+1\}$, are some positive constants.

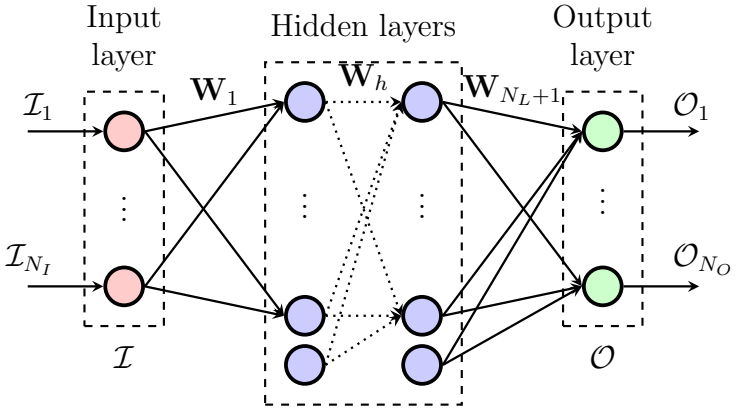


Figure 7.1: Structure of DNN organised in input layer with N_I neurons, N_L fully-connected hidden layers with $N_{H,h}$, $h \in \{1,\dots,N_L\}$, neurons in each layer, and output layer with N_O neurons.

7.2 Transfer Learning

Due to the cost associated with data collection and training, approaches have been proposed to transfer knowledge between robots and thereby increase the efficiency of robot learning. These transferred learning approaches are expected to speed up the training of the target robot and enhance its performance in untrained tasks [142]. The knowledge transfer method that allows the DNN module trained on a source robot system to enhance the impromptu tracking performance of a target robot system that has different dynamics can be implemented. The source and target robot systems can be considered as closed-loop systems whose dynamics are represented by (2.1).

Remark 7.1. Assumptions 1, 2 and 3 are necessary for safe operations and for applying the DNN inverse learning [102].

Assumption 8. The source and target systems have the same relative degree r.

Remark 7.2. Assumption 8 holds, for instance, if the two robots have similar structures but different parameters, e.g., mass and dimension.

The DNN module represents the inverse dynamics of a source system and is previously trained offline with a sufficiently rich dataset. During the testing phase, the DNN module is leveraged to enhance the tracking performance of a target system that shares some dynamic similarities with the source system. The online learning module (trained based on small sets of real-time data) further adjusts the reference generated by the DNN module to allow the target system to achieve high-accuracy tracking on arbitrary trajectories from the first attempt, i.e., impromptu tracking. The proposed control architecture is depicted in Fig. 7.2.

Definition 7.2.1. Let $\mathbf{u}_1 \in \mathbb{R}^{N_I}$ be the reference from the DNN module trained on the source system, and $\mathbf{u}_2 \in \mathbb{R}^{N_I}$ be the reference from the online learning module.

The overall reference to the target baseline system $\mathbf{u}(k) \in \mathbb{R}^{N_I}$ is given by

$$\mathbf{u}(k) = \mathbf{u}_1(k) + \mathbf{u}_2(k). \tag{7.2}$$

The online learning approach is considered that adapts the reference of the DNN module $\mathbf{u}_1(k)$ based on the tracking error. In particular, the reference $\mathbf{u}_2(k)$ can be

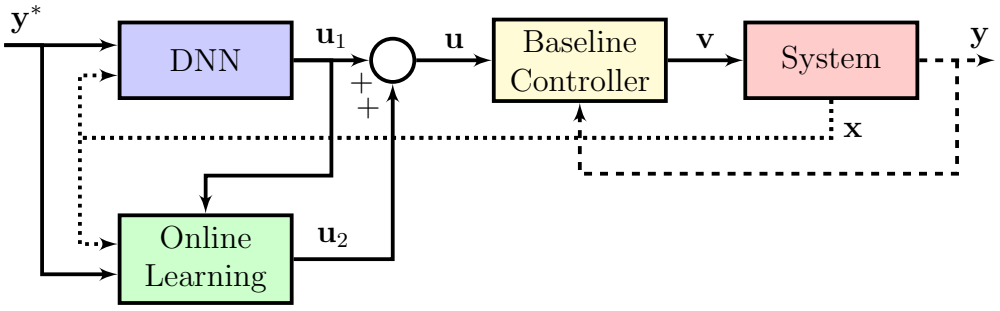


Figure 7.2: Block diagram of the DNN-enhanced control architecture with online learning module (solid lines represent calculated quantities, dashed lines represent measured quantities, dotted lines represent estimated quantities).

approximated by

$$\mathbf{u}_2(k) = \alpha \tilde{\mathbf{e}}(k+r), \tag{7.3}$$

where α is an adaptation gain, and $\tilde{\mathbf{e}}(k+r)$ is a prediction of the tracking error r time steps ahead.

Let consider a nonlinear source system, on which the DNN module is trained, similarly as in (2.5):

$$\mathbf{y}(k+r) = \mathcal{F}_S(\mathbf{x}(k)) + \mathcal{G}_S(\mathbf{x}(k))\mathbf{u}(k), \tag{7.4}$$

where

$$\mathcal{F}_S(\mathbf{x}(k)) = h_S\left(f_S^r(\mathbf{x}(k))\right) \tag{7.5}$$

and

$$\mathcal{G}_S(\mathbf{x}(k)) = \frac{\partial}{\partial \mathbf{u}(k)}\left[h_S\left(f_S^{r-1}\left(f_S(\mathbf{x}(k)) + g_S(\mathbf{x}(k))\mathbf{u}(k)\right)\right)\right] \tag{7.6}$$

are decoupling functions, in which $f_S : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S}$, $g_S : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S} \times \mathbb{R}^{N_I}$ and $h_S : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O}$ are source system functions. In addition to the source system, let consider a nonlinear target system similarly as in (2.5):

$$\mathbf{y}(k+r) = \mathcal{F}_T(\mathbf{x}(k)) + \mathcal{G}_T(\mathbf{x}(k))\mathbf{u}(k), \tag{7.7}$$

where

$$\mathcal{F}_T(\mathbf{x}(k)) = h_T\left(f_T^r(\mathbf{x}(k))\right) \tag{7.8}$$

and

$$\mathcal{G}_T(\mathbf{x}(k)) = \frac{\partial}{\partial \mathbf{u}(k)}\left[h_T\left(f_T^{r-1}\left(f_T(\mathbf{x}(k)) + g_T(\mathbf{x}(k))\mathbf{u}(k)\right)\right)\right] \tag{7.9}$$

are decoupling functions, in which $f_T : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S}$, $g_T : \mathbb{R}^{N_S} \to \mathbb{R}^{N_S} \times \mathbb{R}^{N_I}$ and $h_T : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O}$ are target system functions. As discussed in [102], the DNN module approximates

$$\hat{\mathbf{u}}_1(k) = [\mathcal{G}_S(\mathbf{x}(k))]^{-1} \left(\mathbf{y}^*(k+r) - \mathcal{F}_S(\mathbf{x}(k))\right). \tag{7.10}$$

By substituting (7.2) and (7.10) into (7.8), one can see that the ideal reference $\mathbf{u}_2(k)$ for achieving exact tracking is

$$\mathbf{u}_2(k) = \alpha^* \mathbf{e}^*(k+r), \tag{7.11}$$

where

$$\alpha^* = [\mathcal{G}_T(\mathbf{x}(k))]^{-1} \tag{7.12}$$

and

$$\mathbf{e}^*(k+r) = \mathbf{y}^*(k+r) - \mathcal{F}_T(\mathbf{x}(k)) - \mathcal{G}_T(\mathbf{x}(k))\mathbf{u}_1(k). \tag{7.13}$$

Remark 7.3. To achieve exact tracking, the online learning module should predict the tracking error of the target system that would result from applying $\mathbf{u}_1(k)$.

The error prediction in (7.13) depends on the current state $\mathbf{x}(k)$, the reference $\mathbf{u}_1(k)$ from the DNN module, and the future desired output $\mathbf{y}^*(k+r)$. When the dynamics of the source and the target systems are not known, one may use supervised learning to train a model online to approximate (7.13).

Remark 7.4. For training an online model to approximate (7.13), at each time step k, one may construct a dataset with paired inputs $\mathcal{I}_2 = \{\mathbf{x}(h-r), \mathbf{u}(h-r), \mathbf{y}^*(h)\}$ and outputs $\mathcal{O}_2 = \{\mathbf{y}^*(h) - \mathbf{y}(h)\}$ over the past N_P time steps $h \in \{k - N_P, ..., k\}$, where N_P is the size of the dataset. Then, the error $\tilde{\mathbf{e}}(k+r)$ can be predicted using the online model with input $\mathcal{I}_2 = \{\mathbf{x}(k), \mathbf{u}_1(k), \mathbf{y}^*(k+r)\}$.

Given the predicted error $\tilde{\mathbf{e}}(k+r)$, another component to be determined for computing $\mathbf{u}_2(k)$ is the gain α. With an online model $F\left(\mathbf{x}(k), \mathbf{u}_1(k), \mathbf{y}^*(k+r)\right)$ approximating (7.13), it can be shown that α^* can be obtained from $\hat{\alpha} = -\left[\frac{\partial F}{\partial \mathbf{u}_1}\right]^{-1}$. In practice, due to noise in the systems, the online estimation of α^* can be non-trivial.

7.2.1 System Similarity

Definition 7.2.2. Two systems are similar, if at any given state $\mathbf{x}(k)$, the application of an input $\mathbf{u}(k)$ to the systems results in similar outputs $\mathbf{y}(k+r)$ [143].

For the similarity discussion, let assume source and target systems are linearised (or linear) to simplify the analysis:

$$\begin{cases} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k), \end{cases} \tag{7.14}$$

where $\mathbf{A} \in \mathbb{R}^{N_S \times N_S}$, $\mathbf{B} \in \mathbb{R}^{N_S \times N_I}$ and $\mathbf{C} \in \mathbb{R}^{N_O \times N_S}$ are constant system matrices. It can be shown that the input and output of system (7.14) are related by

$$\mathbf{y}(k+r) = \mathcal{A}\mathbf{x}(k) + \mathcal{B}\mathbf{u}(k), \tag{7.15}$$

where r is the relative degree of system (7.14),

$$\mathcal{A} = \mathbf{C}\mathbf{A}^r \tag{7.16}$$

and

$$\mathcal{B} = \mathbf{C}\mathbf{A}^{r-1}\mathbf{B}. \tag{7.17}$$

From (7.15), the input-output relationship is fully characterized by $\mathcal{A}$ and $\mathcal{B}$, which can be thought as the state-to-output gain vector and the input-to-output gain, respectively.

Definition 7.2.3. If $\{\mathcal{A}_S, \mathcal{B}_S\}$ and $\{\mathcal{A}_T, \mathcal{B}_T\}$ are gain matrices defined as in (7.16) and (7.17) related to the source and target systems, respectively, the similarity factor $\mathbf{S}$ between the source and target systems can be defined as:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \end{bmatrix} = \begin{bmatrix} 1 - \frac{\mathcal{B}_T}{\mathcal{B}_S} \\ \mathcal{A}_T - \frac{\mathcal{B}_T}{\mathcal{B}_S}\mathcal{A}_S \end{bmatrix}. \tag{7.18}$$

The terms $\mathbf{S}_1$ and $\mathbf{S}_2$ characterize the differences in the input-to-output gain and state-to-output gain vector between the source and target systems, respectively. Note that $\mathbf{S} \equiv \mathbf{0}$, iff $\mathcal{A}_T \equiv \mathcal{A}_S$ and $\mathcal{B}_T \equiv \mathcal{B}_S$, i.e., the state-to-output and input-to-output gains of two systems are identical.

Assumption 9. The output of the offline DNN $\mathbf{u}_1(k)$ corresponds to the inverse of the source system

$$\mathbf{u}_1(k) = \frac{1}{\mathcal{B}_S}\left(\mathbf{y}^*(k+r) - \mathcal{A}_S\mathbf{x}(k)\right), \tag{7.19}$$

where $\mathcal{A}_S$ and $\mathcal{B}_S$ are the gains of the source system, and $\mathbf{x}(k)$ and $\mathbf{y}^*(k+r)$ are the state and desired output of the target system.

Assumption 10. The error in the prediction can be bounded:

$$\begin{aligned}\Lambda &= \|\mathbf{e}^*(k+r) - \tilde{\mathbf{e}}(k+r)\| \\ &\leq \beta_1\|\mathbf{y}^*(k+r)\| + \beta_2\|\mathbf{x}(k)\| + \beta_3,\end{aligned} \tag{7.20}$$

where β_1, β_2, and β_3 are some positive constants.

In addition, by Assumption 3 the target system is input-to-state stable. It can be shown that the state of system (7.14) can be bounded as follows:

$$\|\mathbf{x}\|_\infty \leq L_1\|\mathbf{u}\|_\infty + L_2\|\mathbf{x}_0\|, \tag{7.21}$$

L_1 and L_2 are some positive constants.

Theorem 7.2.1 (Stability of DNN)**.** Consider a target system represented by (7.14) and the control architecture in Fig. 7.2, where the reference of the online learning module $\mathbf{u}_2(k)$ has the form of (7.3). Under Assumptions 3, 4, 9, and 10, the overall system is bounded-input bounded-state (BIBS) stable if

$$|\alpha|\left(\|\mathbf{S}_2\| + \beta_2\right) < \frac{\beta_4}{L_1}, \tag{7.22}$$

where $\beta_4 = 1 - L_1\left\|\frac{\mathcal{A}_S}{\mathcal{B}_S}\right\|$.

Proof. At time step k, the output of the online learning module is $\mathbf{u}_2(k) = \alpha\tilde{\mathbf{e}}(k+r)$, where α is a constant gain and $\tilde{\mathbf{e}}(k+r)$ is the predicted tracking error. The adjusted reference $\mathbf{u}(k)$ sent to the target baseline system is $\mathbf{u}(k) = \mathbf{u}_1(k) + \alpha\tilde{\mathbf{e}}(k+r)$, where $\mathbf{u}_1(k)$ is the output of the offline DNN module. By Assumptions 9 and 10, $\mathbf{u}(k)$ can be written as

$$\mathbf{u}(k) = \frac{1}{\mathcal{B}_S}\left(\mathbf{y}^*(k+r) - \mathcal{A}_S\mathbf{x}(k)\right) + \alpha\left(\mathbf{e}^*(k+r) - \Lambda\right). \tag{7.23}$$

For a target system represented by (7.14), $\mathbf{e}^*(k+r)$ in (7.13) can be written as

$$\begin{aligned}\mathbf{e}^*(k+r) &= \mathbf{y}^*(k+r) - \mathcal{A}_T\mathbf{x}(k) - \mathcal{B}_T\mathbf{u}_1(k) \\ &= \mathbf{y}^*(k+r) - \mathcal{A}_T\mathbf{x}(k) - \frac{\mathcal{B}_T}{\mathcal{B}_S}\left(\mathbf{y}^*(k+r) - \mathcal{A}_S\mathbf{x}(k)\right).\end{aligned} \tag{7.24}$$

By substituting the expression of $\mathbf{e}^*(k+r)$ into (7.23),

$$\mathbf{u}(k) = \left(\frac{1}{\mathcal{B}_S} + \alpha S_1\right)\mathbf{y}^*(k+r) - \left(\frac{\mathcal{A}_S}{\mathcal{B}_S} + \alpha\mathbf{S}_2\right)\mathbf{x}(k) - \alpha\Lambda. \tag{7.25}$$

Moreover, by Assumptions 3 and 10, $\|\mathbf{x}\|_\infty$ and $\|\mathbf{y}^*\|_\infty$ can be related by the following inequality:

$$\begin{aligned}\|\mathbf{x}\|_\infty \leq\ & L_1\left(\left\|\frac{1}{\mathcal{B}_S}\right\| + |\alpha|\|\mathbf{S}_1\| + \beta_1|\alpha|\right)\|\mathbf{y}^*\|_\infty \\ & + L_1\left(\left\|\frac{\mathcal{A}_S}{\mathcal{B}_S}\right\| + |\alpha|\|\mathbf{S}_2\| + \beta_2|\alpha|\right)\|\mathbf{x}\|_\infty + L_1\beta_3|\alpha| + L_2\|\mathbf{x}_0\|.\end{aligned} \tag{7.26}$$

From (7.26), if

$$1 - L_1\left(\left\|\frac{\mathcal{A}_S}{\mathcal{B}_S}\right\| + |\alpha|\|\mathbf{S}_2\| + \beta_2|\alpha|\right) > 0, \tag{7.27}$$

or equivalently

$$|\alpha|\left(\|\mathbf{S}_2\| + \beta_2\right) < \frac{\beta_4}{L_1}, \tag{7.28}$$

then the state of the system can be bounded as follows:

$$\|\mathbf{x}\|_\infty \leq \frac{L_1\left(\left\|\frac{1}{\mathcal{B}_S}\right\| + |\alpha|\|\mathbf{S}_1\| + \beta_1|\alpha|\right)\|\mathbf{y}^*\|_\infty + L_1\beta_3|\alpha| + L_2\|\mathbf{x}_0\|}{1 - L_1\left(\left\|\frac{\mathcal{A}_S}{\mathcal{B}_S}\right\| + |\alpha|\|S_2\| + \beta_2|\alpha|\right)}. \tag{7.29}$$

By Assumption 4, $\mathbf{y}^*$ is bounded, and hence $\|\mathbf{y}^*\|_\infty$ is also bounded, then the system state is bounded, and the overall system is BIBS stable. □

7.3 Simulation Results

The source system is selected as:

$$\begin{cases} \mathbf{x}(k+1) &= \begin{bmatrix} 0 & 1 \\ -0.15 & 0.8 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} -0.2 & 1 \end{bmatrix} \mathbf{x}(k). \end{cases} \tag{7.30}$$

while the target system is selected as:

$$\begin{cases} \mathbf{x}(k+1) &= \begin{bmatrix} 0 & 1 \\ -0.24 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} -0.1 & 1 \end{bmatrix} \mathbf{x}(k). \end{cases} \tag{7.31}$$

A DNN module can be designed to enable the system to achieve exact tracking on untrained trajectories. The offline DNN module is trained to control the source system in (7.30) but is used to enhance the tracking performance of the target system in (7.31).

Remark 7.5. The source system in (7.30) and the target system in (7.31) are minimum phase and have relative degrees $r = 1$. The source system has two poles at $\{0.3, 0.5\}$ and a zero at 0.2; while the target system has two poles at $\{0.4, 0.6\}$ and a zero at 0.1. When implementing the learning modules, the controlled systems are considered to be black boxes for which only input-output data and some basic properties, e.g., relative degree, are known.

The offline inverse module is trained on a source system, from which abundant data has been collected. The collected data can often be compactly represented by parametric regression techniques. For the source system (7.30), the DNN module is trained and used to enhance the target system (7.31) with the proposed online learning approach. The DNN module of the source system is a three-layers feedforward network with 20 hyperbolic tangent neurons in each hidden layer. The input and output of the DNN module are $\mathcal{I}_1 = \{x_1(k), x_2(k), y^*(k+1)\}$ and $\mathcal{O}_1 = \{u_1(k)\}$. The training dataset is constructed from the source system's response on 25 sinusoidal trajectories with different combinations of frequencies and amplitudes.

The online error prediction module is a local model trained on a small dataset constructed from the latest observations of the target system. The objective of incorporating the online module is to achieve fast adaptation to the dynamic differences between the source and target systems. In the simulation, a Gaussian process (GP) regression model is utilized for learning the error prediction module online. Based on Remark 7.4, the input and output of the online module are selected to be $\mathcal{I}_2 = \{x_1(k), x_2(k), u_1(k), y^*(k+1)\}$ and $\mathcal{O}_2 = \{\tilde{e}(k+1)\}$, respectively. At each time step k, a fixed-sized training dataset is built based on the latest 15 observations; in particular, the input and output are $\mathcal{I}_2 = \{x_1(p-1), x_2(p-1), u(p-1), y^*(p)\}$ and $\mathcal{O}_2 = \{y^*(p) - y(p)\}$ for $p \in \{k-15, \ldots, k\}$, respectivelly. For the simulation, the GP model uses the squared-exponential kernel $K(\xi, \xi') = \sigma_1^2 \exp\left(-\frac{1}{2}\sum_i \frac{(\xi_i - \xi_i')^2}{l_i^2}\right)$ and polynomial explicit basis functions $\{1, \xi_i, \xi_i^2\}$, where ξ denotes the input to the module and ξ_i denotes the i-th component of ξ, l_i is the length scale associated with the input dimension ξ_i, and σ_1^2 is the prior variance [144]. The length scales l_i are identical for all input dimensions in the simulation. The gain α^* is estimated based on the online error prediction module as $\hat{\alpha} = -\left[\frac{\partial F}{\partial u_1}\right]^{-1}$, where F denotes the function represented by the GP regression model.

7.3.1 Discussion

The performances of the proposed approach are tested on the target systems in (7.31). The desired test trajectory is:

$$y^*(t) = \sin\left(\frac{\pi}{4}t\right) + \cos\left(\frac{\pi}{8}t\right) - 1, \tag{7.32}$$

where $t = 1.5 \times 10^{-3}k$ is the continuous-time variable. This test trajectory is not previously used in the training of the offline learning module.

Fig. 7.3 compares the predicted error from the online module and the analytical error prediction of the target system computed based on (7.13). It can be seen that the online module designed based on Remark 7.4 is able to accurately predict the error of the target system that would result from applying the reference u_1 alone. On the test trajectory, RMSE of the online module prediction is approximately 2.9×10^{-7}.

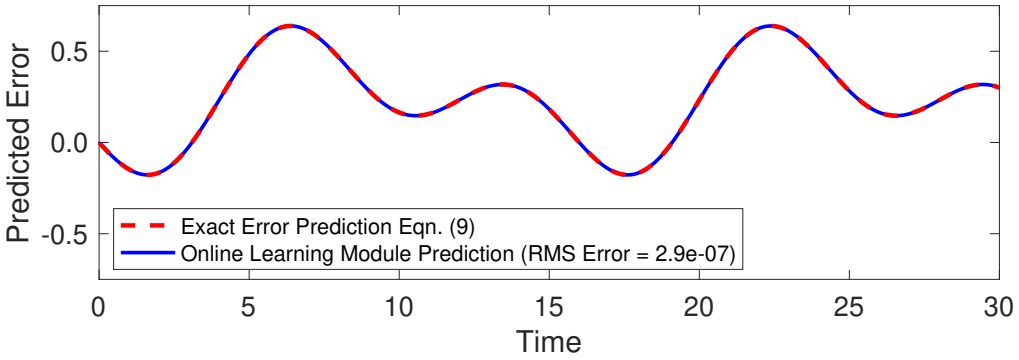


Figure 7.3: Plot of the error prediction from the online learning module.

Fig. 7.4 shows the outputs of the target system when the baseline controller is applied (grey curve), the baseline system is enhanced by the offline module alone (green curve), and the baseline system is enhanced by both the online and the offline modules (blue curve). As compared to the baseline system, the offline module alone reduces the tracking RMSE of the target system from 3.97 to 0.44. The online module further reduces the tracking RMSE to 9×10^{-5}. Applying the offline and the online learning modules jointly allows the target system to achieve approximately exact tracking on a test trajectory that is not seen by the source or the target system a-priori.

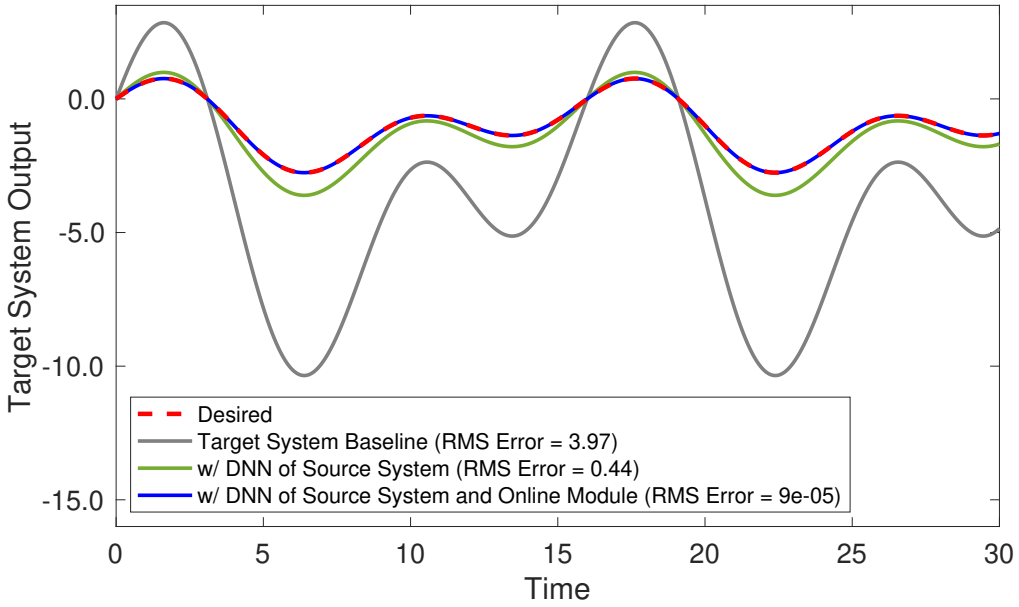


Figure 7.4: Output of the target system controlled by three different approaches.

7.4 Experimental Results

With impromptu tracking of hand-drawn trajectories as the benchmark problem [145], the proposed online learning approach is used for transferring the DNN module trained on a source quadrotor system – Parrot ARDrone 2.0 – to a target quadrotor system – Parrot Bebop 2. With the ARDrone as the testing platform, it is shown that a DNN module trained offline can effectively enhance the impromptu tracking performance of the quadrotor on arbitrary hand-drawn trajectories. The DNN module trained on ARDrone is leveraged to enhance the impromptu tracking performance of Bebop and further apply the proposed online learning approach to achieve high-accuracy tracking. The offboard position controller receives the reference position $\mathbf{p}_r$ and reference velocity $\mathbf{v}_r$, and computes roll and pitch commands ϕ and θ, yaw rate command ω_ψ, and z-velocity command v_z.

A DNN module is trained offline to approximate the inverse of the ARDrone baseline system dynamics. The input and output of the DNN module are $\mathcal{I}_1 = \{x^*(k+4) - x(k), y^*(k+4) - y(k), z^*(k+3) - z(k), v_x^*(k+3) - v_x(k), v_y^*(k+3) - v_y(k), v_z^*(k+2) - v_z(k), \boldsymbol{\theta}(k), \boldsymbol{\omega}(k)\}$ and $\mathcal{O}_1 = \{\mathbf{p}_r(k) - \mathbf{p}(k), \mathbf{v}_r(k) - \mathbf{v}(k)\}$, respectively. The DNN module consists of fully-connected feedforward networks with 4 hidden layers of 128 rectified linear units. The training dataset of the DNN module is constructed from the ARDrone baseline system response on a 400s 3-dimensional sinusoidal trajectory. At a sampling rate of 7Hz, approximately $2'800$ pairs of data points are collected for training. For 30 hand-drawn test trajectories, this offline DNN module is able to reduce the impromptu tracking error of the ARDrone baseline system by 43% on average.

Based on Remark 7.4, the input and output of the online learning module are $\mathcal{I}_2 = \{\mathbf{p}(k), \mathbf{v}(k), \boldsymbol{\theta}(k), \boldsymbol{\omega}(k), \mathbf{p}_r(k), \mathbf{v}_r(k), x^*(k+4), y^*(k+4), z^*(k+3), v_x^*(k+3), v_y^*(k+3), v_z^*(k+2)\}$ and $\mathcal{O}_2 = \{\tilde{x}(k+4), \tilde{y}(k+4), \tilde{z}(k+3), \tilde{v}_x(k+3), \tilde{v}_x(k+3), \tilde{v}_x(k+2)\}$, respectively. In the experiment, to make the online learning more efficient, instead of predicting the position and velocity errors directly, a GP model has been trained to predict the position of UAV, i.e., $\mathbf{p}(k+r) = [x(k+4), y(k+4), z(k+3)]$. Then, the predicted error is computed by subtracting the predicted position from future desired position, i.e., $\mathbf{p}^*(k+r) - \mathbf{p}(k+r)$, where $\mathbf{p}^*(k+r) = [x^*(k+4), y^*(k+4), z^*(k+3)]$. The predicted position errors are used to compute the corrections for the position components; while the velocity reference corrections are numerically approximated

with a first-order finite difference scheme. Due to the measurement noise in the experiment, instead of estimating the parameter α online, constant gains $\alpha = (5, 5, 0.5)$ for the x, y, and z axes are used.

7.4.1 Discussion

Figs. 7.5 and 7.6 compares the tracking performance of three control strategies on Bebop on one of the test hand-drawn trajectories. When comparing the performance of Bebop system enhanced by ARDrone DNN (green curve) and the performance of Bebop baseline system (grey curve), ARDrone DNN reduces the delay and the amplitude errors in Bebop tracking response. Along this particular trajectory, the DNN module alone reduces the tracking RMSE of Bebop from approximately 0.42m to 0.26m. When further comparing with the performance of the DNN-enhanced system with the addition of the online learning module (blue curve), the tracking of Bebop, especially in the x-direction, is brought close to the desired trajectory. With the online learning module, the tracking RMSE is reduced to approximately 0.14m. When the online learning module is applied, there are small overshoots at the locations with larger curvatures. The overshoots may be reduced with the online tuning of the hyperparameters of GP and online estimations of α.

Fig. 7.7 summarises the performance errors of three control strategies on 10 hand-drawn trajectories. When compared with Bebop baseline system performance (grey bars), the direct application of the transferred DNN module (green bars) reduces the tracking RMSE of Bebop baseline system by an average of 46%. With the addition of the online learning module (blue bars), an average of 74% tracking RMSE

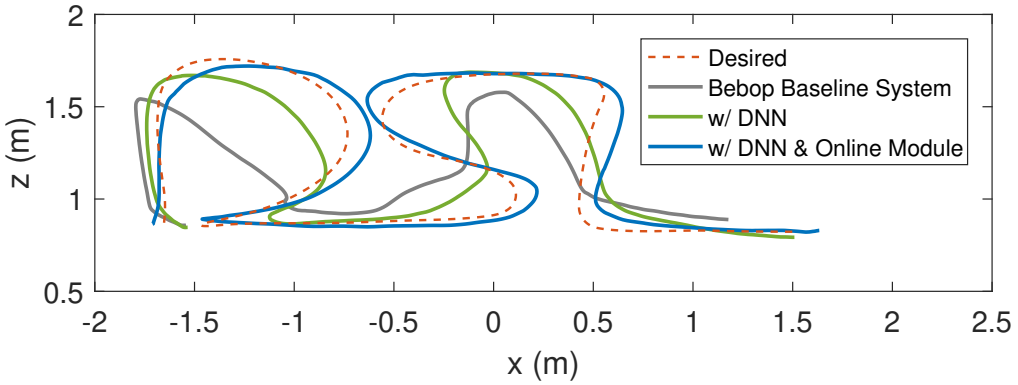


Figure 7.5: Trajectory tracking of the target system in the xz-plane by three control strategies.

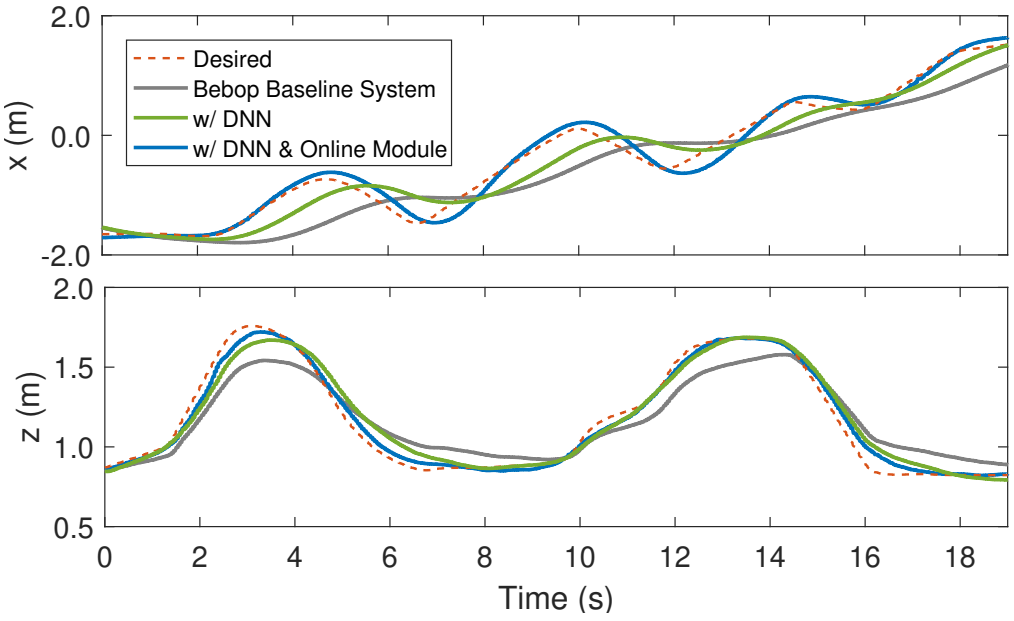


Figure 7.6: Position trajectories of the target system controlled by three control strategies.

reduction is achieved. Two additional sets of results are included for comparison: the performance of ARDrone enhanced by the DNN module trained on ARDrone system (yellow bars), and the performance of Bebop enhanced by a DNN module trained on Bebop system (light-blue bars). Without requiring further data collection and offline training, the inclusion of the online learning module effectively reduces the tracking RMSE of Bebop to values that are comparable to those of the cases where the quadrotors are enhanced by their own offline DNN modules. These results demonstrate the efficiency of the proposed online learning module to leverage past experience and reduce data re-collection and training.

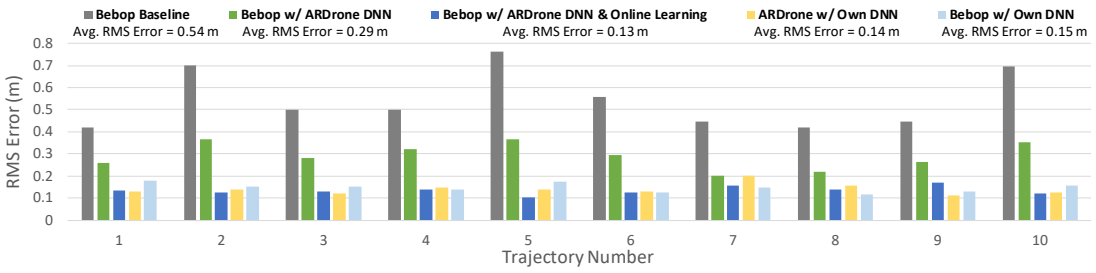


Figure 7.7: Tracking performance of the target system on 10 hand-drawn trajectories.

7.5 Conclusion

In this chapter, the impromptu tracking problem is considered, and an online learning approach is proposed to efficiently transfer a DNN module trained on a source robot system to a target robot system. In the theoretical analysis, an expression of the online module is derived for achieving exact tracking. Then, based on a linear system formulation, an approach for characterising system similarity is proposed, and insights on the impact of the system similarity on the stability of the overall system are provided in the knowledge transfer problem. The approach is verified experimentally by applying the proposed online learning approach to transfer a DNN inverse dynamics module across two quadrotor platforms (Parrot ARDrone 2.0 and Parrot Bebop 2). On 10 arbitrary hand-drawn trajectories, the DNN module of the source system reduces the tracking error of the target system by an average of 46%. The incorporation of the online module further reduces the tracking error and leads to an average of 74% error reduction. These experimental results show that the proposed online learning and knowledge transfer approach can efficaciously circumvent data recollection on the target robot, and thus, the costs and risks associated with training new robots to achieve higher performance in impromptu tasks.

Part IV

Fuzzy Neural Network-Based Control

Chapter 8

Neural Fuzzy-Based Control

It has been shown that fuzzy logic has an exceptional ability to handle the uncertainties in the system [25]. Therefore, FLC is one of the most popular model-free approaches to control nonlinear systems when their precise mathematical model is challenging to obtain [146]. However, one weakness of FLCs is that their parameters have to be tuned to deal with uncertainties [26]. On the other hand, ANNs are a family of supervised learning models that mimics human brain [17]. Therefore, ANNs are widely used in many applications due to their ability to learn from input-output data [147]. However, the main weakness of ANNs is that their inner workings are difficult to interpret [27]. The combination of FLS and ANN – FNN – fuses the reasoning ability of FLS to handle uncertain information with the training capability of ANN to learn from the controlled process [28]. Hence, FNN adopts the advantages of both FLS and ANN [29].

In this chapter, potentials of FNNs are explored under various operational conditions. First, Section 8.1 revises the definition of FNN. Next, Sections 8.2 and 8.3 proposes SMC theory-based and LM theory-based training algorithms, respectively. Then, Sections 8.4 and 8.5 show simulation and experimental results, respectively, for a quadcopter UAV in the presence of periodic wind gust. Finally, the conclusions are drawn in Section 8.6.

Supplementary Material:
ROS package for the proposed FNN controllers: github.com/andriyukr/controllers.
Video for the experimental results: tiny.cc/FNN.

8.1 Mathematical Preliminaries

In a general FNN, the input $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_{N_I}\}$ to FNN is fuzzified by N_F MFs. Identically to type-1 FLS, FS and MF in FNN are defined as in Definition 3.1.1, where the parameters of MFs are among the tunable parameters of FNN. Typically, FNN employs Takagi-Sugeno-Kang (TSK) fuzzy model in which the antecedent part is FS, while the consequent part is the function of input variables.

Definition 8.1.1. If N_R is the number of rules in a rule-base R, then the j^{th} *rule* $R_j \in \text{R}$, $j \in \{1, \ldots, N_R\}$, is indicated as IF $-$ THEN statement, i.e.:

$$R_j : \begin{array}{l} \text{IF } \mathcal{I}_1 \text{ is } A_{1,j} \text{ and } \ldots \text{ and } \mathcal{I}_{N_I} \text{ is } A_{N_I,j}, \\ \text{THEN } c_j = \sum_{i=1}^{N_I} w_{i,j}\mathcal{I}_j + w_{0,j} \end{array}, \quad i \in \{1, \ldots, N_R\}, \tag{8.1}$$

where $A_{j,i}$ represents antecedent FS and $w_{i,j}$ are weights in FNN.

Assumption 11. The network weights $\mathbf{W} = \begin{bmatrix} w_{0,1} & \cdots & w_{0,N_R} \\ \vdots & \ddots & \vdots \\ w_{N_I,1} & \cdots & w_{N_I,N_R} \end{bmatrix} \in \mathbb{R}^{(N_I+1)\times N_R}$ are bounded, i.e.:

$$\|\mathbf{W}(t)\|_\infty \leq c_{\mathbf{W}} \quad \forall t, \tag{8.2}$$

where $c_{\mathbf{W}}$ is some positive constants.

Remark 8.1. Often, the weights $w_{i,j} = 0$, $i \in \{1, \ldots, N_I\} \wedge j \in \{1, \ldots, N_R\}$, which results in a zero-order TSK-FNN model, i.e., $c_j = w_{0,j}$, $j \in \{1, \ldots, N_R\}$.

The weights $w_{i,j}$, $i \in \{1, \ldots, N_R\} \wedge j \in \{0, \ldots, N_I\}$, are updated following a set of rules during the learning process. The firing strength f_i, $i \in \{1, \ldots, N_R\}$, of each rule is calculated with the multiplication t-norm as in Definition 3.1.5. The output signal of FNN u_{FNN} is computed as the weighted average of each rule's output [148]:

$$u_{\text{FNN}} = \frac{\sum_{i=1}^{N_R} f_i c_i}{\sum_{i=1}^{N_R} f_i} = \sum_{i=1}^{N_R} \overline{f}_i c_i, \tag{8.3}$$

where $\overline{f}_i$ is the normalized value of the i^{th} firing strength:

$$\overline{f}_i = \frac{f_i}{\sum_{i=1}^{N_R} f_i}. \tag{8.4}$$

In the proposed approach, the input is $\mathcal{I} = \{e, \dot{e}\}$, i.e., the position feedback error and its time derivative; while the output is $\mathcal{O} = \{u_{\text{FNN}}\}$, i.e., the control signal. Therefore, there are two input neurons ($N_I = 2$) and one output neuron ($N_O = 1$). To fuzzify two inputs, N_F Gaussian MF in (3.7) are adopted.

In the proposed control scheme, FNN works in parallel with a conventional PD controller, as shown in Fig. 8.1. The conventional PD controller is utilized as an ordinary feedback controller to ensure the global asymptotic stability of the system and provide sufficient time for the initialization of the learning process of FNN. Thus, FNN learns the control parameters and takes over the control responsibility of the system. The overall control input u to the controlled system is defined by:

$$u = u_{\text{PD}} - u_{\text{FNN}}, \tag{8.5}$$

where u_{PD} and u_{FNN} are the control signals produced by the PD controller in (6.8) and the FNN controller in (8.3), respectively.

By Assumptions 6 and 11, the consequent c_i, $i \in \{1, \ldots, N_R\}$, in (8.1) is bounded, i.e.:

$$|c_i(t)| \leq c_c, \quad i \in \{1, \ldots, N_R\} \quad \forall t, \tag{8.6}$$

where c_c is some positive constant. From Remark 3.3 and (8.4), it is evident that

$$\overline{f}_i \in [0, 1], \quad i \in \{1, \ldots, N_R\}. \tag{8.7}$$

In addition, from (8.3), (8.6) and (8.7), $u_{\text{ANN}}(t)$ and $\dot{u}_{\text{ANN}}(t)$ are also bounded signals, i.e.:

$$\begin{cases} |u_{\text{FNN}}(t)| \leq c_u \\ |\dot{u}_{\text{FNN}}(t)| \leq c_{\dot{u}} \end{cases} \quad \forall t, \tag{8.8}$$

where c_u and $c_{\dot{u}}$ are some positive constants.

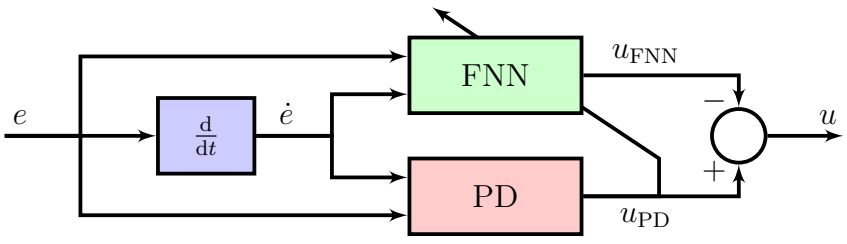


Figure 8.1: Control scheme: FNN in parallel with PD controller.

8.2 Sliding Mode Control-Based Learning

The zero dynamics of the learning error coordinate $u_{\mathrm{PD}}(t)$ can be described as a time-varying sliding surface S_{PD} by utilizing the principles of the SMC theory [149]:

$$S_{\mathrm{PD}}(u_{\mathrm{FNN}}, u) = u_{\mathrm{PD}}(t) = u_{\mathrm{FNN}}(t) + u(t) = 0. \tag{8.9}$$

By using this condition, the FNN structure is trained to become a nonlinear regulator which assists the conventional parallel controller to obtain the desired response. Hence, the sliding surface for the nonlinear system under control is as follows:

$$S(e, \dot{e}) = \dot{e} + \lambda e. \tag{8.10}$$

where $\lambda > 0$ is a parameter which determines the slope of the sliding surface. A sliding motion will appear on the sliding manifold $S_{\mathrm{PD}}(u_{\mathrm{FNN}}, u) = 0$ after a finite time t_h, if the condition $S_{\mathrm{PD}}(t)\dot{S}_{\mathrm{PD}}(t) = u_{\mathrm{PD}}(t)\dot{u}_{\mathrm{PD}}(t) < 0$ is satisfied for all t in some nontrivial semi-open subinterval $[t, t_h) \subset (-\infty, t_h)$. Since it is desired to design a dynamical feedback adaptation mechanism, or an on-line learning algorithm, for the FNN parameters such that the sliding mode condition above is enforced.

Theorem 8.2.1 (Stability of FNN with SMC)**.** If the SMC theory-based parameter update rules for FNN are as follows:

$$\begin{cases} \dot{c}_{1,i} & = \dot{\mathcal{I}}_1 \\ \dot{c}_{2,i} & = \dot{\mathcal{I}}_2 \\ \dot{d}_{1,i} & = -\alpha \frac{d_{1,i}^3}{(\mathcal{I}_1 - c_{1,i})^2} \mathrm{sign}(u_{\mathrm{PD}}) \quad \forall i \in \{1, \ldots, N_F\} \\ \dot{d}_{2,i} & = -\alpha \frac{d_{2,i}^3}{(\mathcal{I}_2 - c_{2,i})^2} \mathrm{sign}(u_{\mathrm{PD}}) \quad \forall j \in \{1, \ldots, N_R\}, \\ \dot{w}_{0,j} & = -\alpha \frac{\overline{f}_j}{\overline{\mathbf{F}}^T \overline{\mathbf{F}}} \mathrm{sign}(u_{\mathrm{PD}}) \\ \dot{\alpha} & = \gamma |u_{\mathrm{PD}}| - \gamma \nu \alpha \end{cases} \tag{8.11}$$

where $\overline{\mathbf{F}} = \begin{bmatrix} \overline{f}_{1,1} & \cdots & \overline{f}_{1,N_F} \\ \vdots & \ddots & \vdots \\ \overline{f}_{N_F,1} & \cdots & \overline{f}_{N_F,N_F} \end{bmatrix}$, $\alpha > 0$ is an adaptive learning rate, $\gamma \geq 0$ and $\nu \geq 0$ are learning parameters; then, the control signal $u_{\mathrm{PD}}(t)$ will converge to a small neighbourhood of zero during a finite time t_h for any arbitrary initial condition $u_{\mathrm{PD}}(0)$.

Proof. To fuzzify two inputs, the following Gaussian MF in (3.7) are adopted:

$$\begin{cases} \mu_{1,j}(\mathcal{I}_1) = \exp\left[-\frac{(\mathcal{I}_1 - c_{1,j})^2}{2d_{1,j}^2}\right] \\ \mu_{2,j}(\mathcal{I}_2) = \exp\left[-\frac{(\mathcal{I}_2 - c_{2,j})^2}{2d_{2,j}^2}\right] \end{cases} \quad \forall j \in \{1, \ldots, N_F\}. \tag{8.12}$$

The time derivatives of (8.12) is as follows:

$$\begin{cases} \dot{\mu}_{1,j}(\mathcal{I}_1) = -2A_{1,j}(A_{1,j})'\mu_{1,i}(\mathcal{I}_1) \\ \dot{\mu}_{2,j}(\mathcal{I}_2) = -2A_{2,j}(A_{2,j})'\mu_{2,j}(\mathcal{I}_2) \end{cases} \quad \forall j \in \{1, \ldots, N_F\}, \tag{8.13}$$

where $A_{1,j} = \frac{\mathcal{I}_1 - c_{1,j}}{d_{1,j}}$ and $A_{2,j} = \frac{\mathcal{I}_2 - c_{2,j}}{d_{2,j}}$. The time derivative of (8.4) can be obtained as follows:

$$\dot{\overline{f}}_{i,j} = -\overline{f}_{i,j}\dot{K}_{i,j} + \overline{f}_{i,j}\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\overline{f}_{i,j}\dot{K}_{i,j} \quad \forall i \in \{1, \ldots, N_F\} \quad \forall j \in \{1, \ldots, N_F\}, \tag{8.14}$$

where $\dot{K}_{i,j} = 2\left(A_{1,i}A'_{1,i} + A_{2,j}A'_{2,j}\right)$.

The Lyapunov function is selected as follows:

$$V = \frac{1}{2}u_{\mathrm{PD}}^2(t) + \frac{1}{2\gamma}(\alpha - \alpha^*)^2. \tag{8.15}$$

The time derivative of (8.15) is given by:

$$\dot{V} = u_{\mathrm{PD}}\left(\dot{u}_{\mathrm{FNN}} + \dot{u}\right) + \frac{1}{\gamma}\dot{\alpha}\left(\alpha - \alpha^*\right). \tag{8.16}$$

The time derivative of (8.3) is given by:

$$\dot{u}_{\mathrm{FNN}} = \sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\left(\dot{c}_{i,j}\overline{f}_{i,j} + c_{i,j}\dot{\overline{f}}_{i,j}\right). \tag{8.17}$$

By replacing (8.17) to (8.16), the following equation is obtained:

$$\begin{aligned}\dot{V} &= u_{\mathrm{PD}}\left(\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\left(\dot{c}_{i,j}\overline{f}_{i,j}+c_{i,j}\left(-\overline{f}_{i,j}\dot{K}_{i,j}+\overline{f}_{i,j}\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\overline{f}_{i,j}\dot{K}_{i,j}\right)\right)+\dot{u}\right)\\ &\quad+\frac{1}{\gamma}\dot{\alpha}(\alpha-\alpha^*)\\ &= u_{\mathrm{PD}}\left(\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\dot{c}_{i,j}\overline{f}_{i,j}-2\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\overline{f}_{i,j}\left(A_{1,i}(A_{1,i})'+A_{2,j}(A_{2,j})'\right)c_{i,j}\right.\\ &\quad\left.+2\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\left(\overline{f}_{i,j}c_{i,j}\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\overline{f}_{i,j}\left(A_{1,i}(A_{1,i})'+A_{2,j}(A_{2,j})'\right)\right)+\dot{u}\right)\\ &\quad+\frac{1}{\gamma}\dot{\alpha}(\alpha-\alpha^*),\end{aligned} \tag{8.18}$$

where

$$\begin{cases}\dot{A}_{1,j} &= \frac{(\dot{\mathcal{I}}_1-\dot{c}_{1,j})d_{1,j}-(\mathcal{I}_1-c_{1,j})\dot{d}_{1,j}}{d_{1,j}^2}\\ \dot{A}_{2,j} &= \frac{(\dot{\mathcal{I}}_2-\dot{c}_{2,j})d_{2,j}-(\mathcal{I}_2-c_{2,j})\dot{d}_{2,j}}{d_{2,j}^2}\end{cases}\quad \forall j\in\{1,\ldots,N_F\}. \tag{8.19}$$

From (8.11), follows:

$$A_{1,j}\dot{A}_{1,j}=A_{2,j}\dot{A}_{2,j}=\alpha\mathrm{sign}(u_{\mathrm{PD}}). \tag{8.20}$$

Consequently, (8.18) becomes

$$\begin{aligned}\dot{V} &= u_{\mathrm{PD}}\left(\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\dot{c}_{i,j}\overline{f}_{i,j}-4\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}c_{i,j}\overline{f}_{i,j}\alpha\mathrm{sign}(u_{\mathrm{PD}})\right.\\ &\quad\left.+4\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\left(\overline{f}_{i,j}c_{i,j}\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\overline{f}_{i,j}\alpha\mathrm{sign}(u_{\mathrm{PD}})\right)+\dot{u}\right)+\frac{1}{\gamma}\dot{\alpha}(\alpha-\alpha^*).\end{aligned} \tag{8.21}$$

Since $\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\overline{f}_{i,j}=1$; then, (8.21) becomes

$$\dot{V}=u_{\mathrm{PD}}\left(\sum_{i=1}^{N_F}\sum_{j=1}^{N_F}\dot{c}_{i,j}\overline{f}_{i,j}+\dot{u}\right)+\frac{1}{\gamma}\dot{\alpha}(\alpha-\alpha^*), \tag{8.22}$$

where

$$\dot{c}_{i,j}=-\frac{\overline{W}_{ij}}{\overline{W}^T\overline{W}}\alpha\mathrm{sign}(u_{\mathrm{PD}}). \tag{8.23}$$

Furthermore,

$$\begin{aligned}\dot{V} &= u_{\mathrm{PD}}\left(-\alpha \mathrm{sign}(u_{\mathrm{PD}}) + \dot{u}\right) + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*) \\ &= -\alpha|u_{\mathrm{PD}}| + |u_{\mathrm{PD}}|c_{\dot{u}} + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*) \\ &= -(\alpha - \alpha^*)|u_{\mathrm{PD}}| - \alpha^*|u_{\mathrm{PD}}| + |u_{\mathrm{PD}}|c_{\dot{u}} + \frac{1}{\gamma}\dot{\alpha}(\alpha - \alpha^*).\end{aligned} \tag{8.24}$$

Considering the adaptation law of α as follows,

$$\dot{\alpha} = \gamma|u_{\mathrm{PD}}| - \gamma\nu\alpha; \tag{8.25}$$

then, (8.24) becomes:

$$\dot{V} = |u_{\mathrm{PD}}|c_{\dot{u}} - \alpha^*|u_{\mathrm{PD}}| - \nu(\alpha - \frac{1}{2}\alpha^*)^2 + \frac{\nu}{4}\alpha^{*2}. \tag{8.26}$$

By taking α^* as $c_{\dot{u}} \leq \frac{\alpha^*}{2}$, it follows:

$$\dot{V} \leq -\frac{\alpha^*}{2}|u_{\mathrm{PD}}| + \frac{\nu}{4}\alpha^{*2}, \tag{8.27}$$

which implies that the Lyapunov function decreases until $|u_{\mathrm{PD}}| < \frac{\nu\alpha^*}{2}$. So that u_{PD} will stay bounded. Furthermore ν is a design parameter and it is possible to take this value as small as desired. The relation between the sliding function S_p and the zero adaptive learning error level S_{PD} is as follows:

$$S_{\mathrm{PD}} = u_{\mathrm{PD}} = k_p e = k_p S_p. \tag{8.28}$$

□

8.3 Levenberg-Marquardt-Based Learning

Theorem 8.3.1 (Stability of FNN with LM)**.** If the LM theory-based parameter update rules for FNNs are as follows:

$$\begin{cases} \dot{c}_{1,i} & = \dot{\mathcal{I}}_1 \\ \dot{c}_{2,i} & = \dot{\mathcal{I}}_2 \\ \dot{d}_{1,i} & = -\alpha \frac{d_{1,i}^3}{(\mathcal{I}_1 - c_{1,i})^2} \text{sign}(u_{\text{PD}}) \\ \dot{d}_{2,i} & = -\alpha \frac{d_{2,i}^3}{(\mathcal{I}_2 - c_{2,i})^2} \text{sign}(u_{\text{PD}}) \\ \dot{w}_{0,j} & = -\gamma \left(\overline{\mathbf{F}}^T \overline{\mathbf{F}} + \delta \mathbf{I} \right)^{-1} \overline{\mathbf{F}} \text{sign}(u_{\text{PD}}) \end{cases} \quad \begin{array}{l} \forall i \in \{1, \ldots, N_F\} \\ \forall j \in \{1, \ldots, N_R\}, \end{array} \tag{8.29}$$

where δ is the adaptive parameter and is selected equal to

$$\delta = \max \left(\overline{\mathbf{F}}^T \overline{\mathbf{F}}, \alpha \right), \tag{8.30}$$

in which α has a constant value; then, the learning error $u_{\text{PD}}(t)$ will converge to a small neighbourhood of zero during a finite time t_h for any arbitrary initial condition $u_{\text{PD}}(0)$.

Proof. The Lyapunov function is selected as follows:

$$V = \frac{1}{2} u_{\text{PD}}^2(t). \tag{8.31}$$

The time derivative of (8.31) is given by:

$$\dot{V} = u_{\text{PD}} \left(\sum_{i=1}^{N_F} \sum_{j=1}^{N_F} \dot{w}_{i,j} \overline{f}_{i,j} + \dot{u} \right). \tag{8.32}$$

From (8.29), follows:

$$\begin{aligned}
\dot{V} &= u_{\mathrm{PD}}\left(\gamma\overline{\mathbf{F}}^{T}\left(-\delta^{-1}+\delta^{-1}\overline{\mathbf{F}}\left(\mathbf{I}+\overline{\mathbf{F}}^{T}\delta^{-1}\overline{\mathbf{F}}\right)^{-1}\overline{\mathbf{F}}^{T}\delta^{-1}\right)\overline{\mathbf{F}}\mathrm{sign}(u_{\mathrm{PD}})+\dot{u}_{\mathrm{PD}}\right)\\
&= u_{\mathrm{PD}}\left(\gamma\overline{\mathbf{F}}^{T}\left(-\delta^{-1}+\delta^{-1}\overline{\mathbf{F}}\left(\delta+\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\right)^{-1}\overline{\mathbf{F}}^{T}\right)\overline{\mathbf{F}}\mathrm{sign}(u_{\mathrm{PD}})+\dot{u}_{\mathrm{PD}}\right)\\
&= -\delta^{-1}\gamma\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}|u_{\mathrm{PD}}|+|u_{\mathrm{PD}}|\delta^{-1}\gamma\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\left(\delta+\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\right)^{-1}\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}+c_{\dot{u}}|u_{\mathrm{PD}}|\\
&= -\delta^{-1}\gamma\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}|u_{\mathrm{PD}}|+|u_{\mathrm{PD}}|\delta^{-1}\gamma\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\left(\delta+\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\right)^{-1}\left(\delta+\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}-\delta\right)+c_{\dot{u}}|u_{\mathrm{PD}}|\\
&= -\delta^{-1}\gamma\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}|u_{\mathrm{PD}}|+\gamma|u_{\mathrm{PD}}|\delta^{-1}\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}-\gamma|u_{\mathrm{PD}}|\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\left(\delta+\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\right)^{-1}+c_{\dot{u}}|u_{\mathrm{PD}}|\\
&= -\gamma|u_{\mathrm{PD}}|\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\left(\delta+\overline{\mathbf{F}}^{T}\overline{\mathbf{F}}\right)^{-1}+c_{\dot{u}}|u_{\mathrm{PD}}|.
\end{aligned} \tag{8.33}$$

By considering $\delta = \overline{\mathbf{F}}^{T}\overline{\mathbf{F}}$, (8.33) becomes:

$$\dot{V} = -0.5\gamma|u_{\mathrm{PD}}|+c_{\dot{u}}|u_{\mathrm{PD}}|. \tag{8.34}$$

It is further assumed that $\gamma > 4c_{\dot{u}}$, so that:

$$\dot{V} \leq -\frac{1}{4}\gamma|u_{\mathrm{PD}}|. \tag{8.35}$$

□

8.4 Simulation Results

The control gains for the PD controller are chosen as follows:

$$\begin{cases} k_p &= 2.5\\ k_d &= 0.005. \end{cases} \tag{8.36}$$

The initial control parameters of the FNN controller are:

$$\begin{cases} c_{1,x} = c_{1,y} = c_{1,z} & = \begin{bmatrix} -10 & 0 & 10 \end{bmatrix}^T \\ c_{2,x} = c_{2,y} = c_{2,z} & = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T \\ d_{1,x} = d_{1,y} = d_{1,z} & = \begin{bmatrix} 5 & 5 & 5 \end{bmatrix}^T \\ d_{2,x} = d_{2,y} = d_{2,z} & = \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix}^T \\ \alpha_x = \alpha_y = \alpha_z & = 0.001, \end{cases} \tag{8.37}$$

while the initial condition of time variable weight coefficients $w_{i,j}(0)$ are chosen randomly to be sufficiently small, i.e., $w_{i,j}(0) \in [0, 0.001]$. The adaptive learning parameters for FNN with SMC theory-based rules are chosen as:

$$\begin{cases} \gamma_x = \gamma_y = \gamma_z & = 1 \\ \nu_x = \nu_y = \nu_z & = 0.9; \end{cases} \tag{8.38}$$

FNN with LM theory-based rules:

$$\begin{cases} \gamma_x = \gamma_y = \gamma_z & = 1.5 \\ \alpha_x = \alpha_y = \alpha_z & = 1; \end{cases} \tag{8.39}$$

with the update rate $dt = 0.01$s.

In the classical UAV flight missions, climbing, hovering, ascending and descending curves as well as level flight are considered as typical flight manoeuvres. Therefore, in the experimental scenario, all aforementioned manoeuvres will be included during the flight of quadrotor in order to evaluate the robustness of proposed controllers under the flight sequence which resembles the actual UAV flights. At the beginning, the quadrotor UAV is hovering at the initial position. Then, it starts to make a smooth eight-shaped trajectory. Throughout the simulation, the module of the desired linear velocity is kept constant and equal to 2.5m/s. Besides, the feasibility of flight under the dynamic constraints of quadrotor is ensured by saturating the control input signals and defining trajectory as a time-based trajectory.

In the tested scenario, to evaluate the efficacy of the proposed control strategy, the periodic wind gust is added. The wind gust blows with the speed $v_w = 3.0$m/s

along $[-1, -1, 0]$ direction for the first 20s. Then, its orientation changes to the opposite direction, to change back to the original direction after 40s.

8.4.1 Discussion

Fig. 8.2 shows the trajectory tracking in the presence of periodic wind for PD controller and two FNN controllers which are tuned by SMC and LM methods which operate in parallel with the conventional PD controller. From Fig. 8.2, it is evident that the PD controller has a notable steady-state error due to internal uncertainties such as lack of modelling as well as external disturbance from the wind. Besides, when the PD controller works alone, it cannot eliminate the steady-state error. However, in the case of SMC-FNN and LM-FNN controllers, the steady-state error is notably reduced because of adaptive learning capabilities of FNN structure. As a result, trajectory tracking performance of quadrotor which uses intelligent FNN structures becomes significantly better compared to the normal case, when the PD controller is only used.

As for output control signals, Fig. 8.3 and Fig. 8.4 present control signals for x, y and z axes in case when the PD controller is operating in parallel with FNN controller which are tuned by SMC and LM approach, respectively. As can be seen from these figures, the FNN controller is taking over the control responsibilities from PD controller, and therefore, after some time the output control signal from

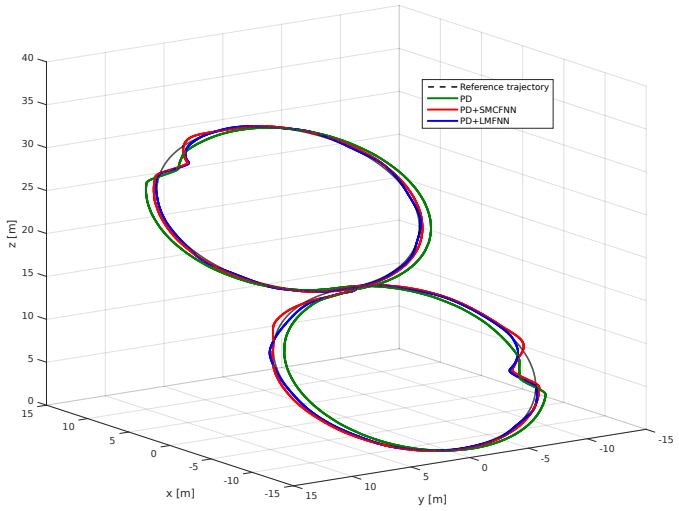
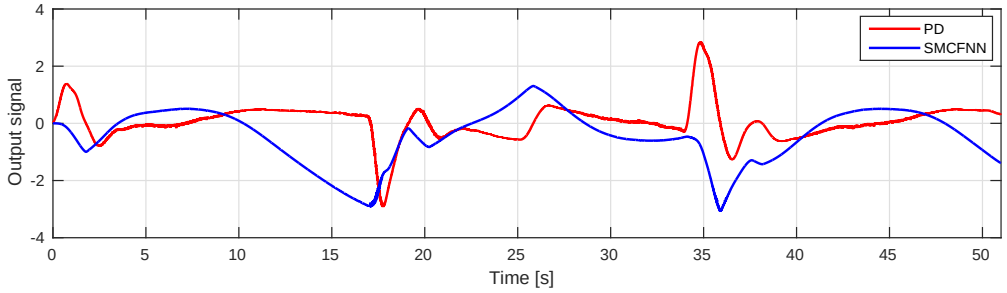
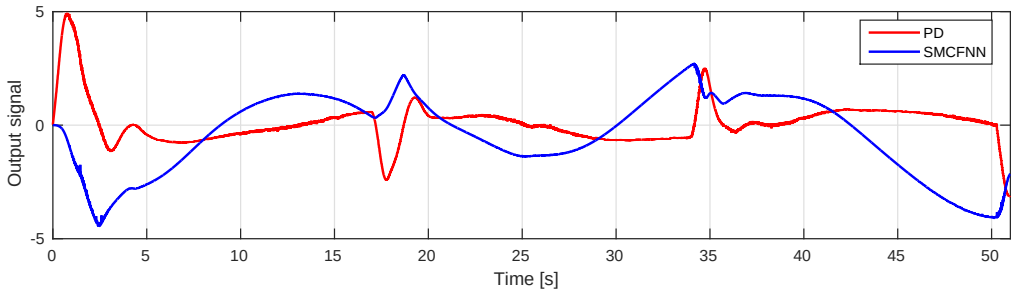


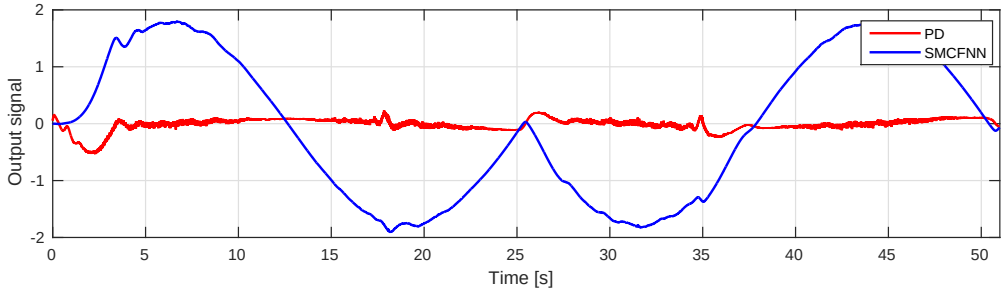
Figure 8.2: Trajectory tracking of different FNN position controllers in presence of wind.



(a) x-axis



(b) y-axis

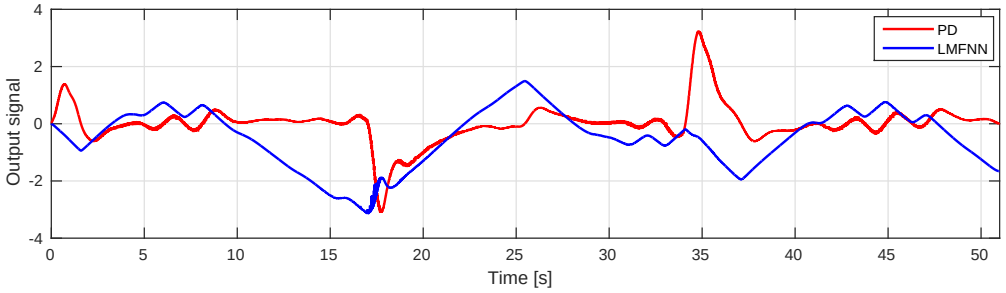


(c) z-axis

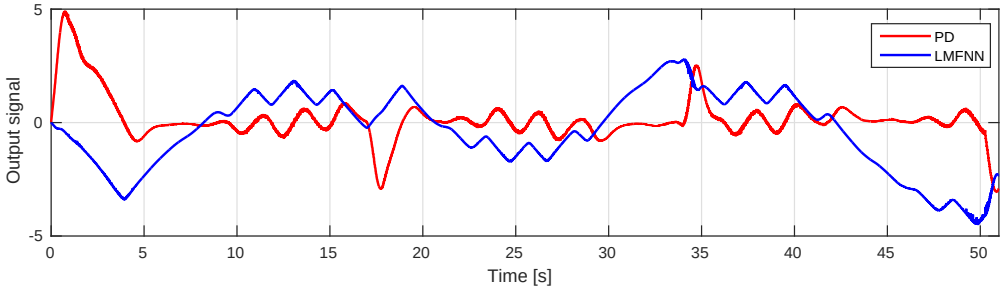
Figure 8.3: Control signals for x, y and z axes generated by PD and SMC-FNN.

PD controller approaches to zero neighbourhood, and then only FNN controls the system as it is supposed to do in such kind of control schemes. It should be noted that when trajectory sequence changes or some disturbances occur the output control signals from the PD controller become non-zero. In such case, FNN restarts the learning process and takes over control responsibilities again as shown in Fig. 8.3 and Fig. 8.4.

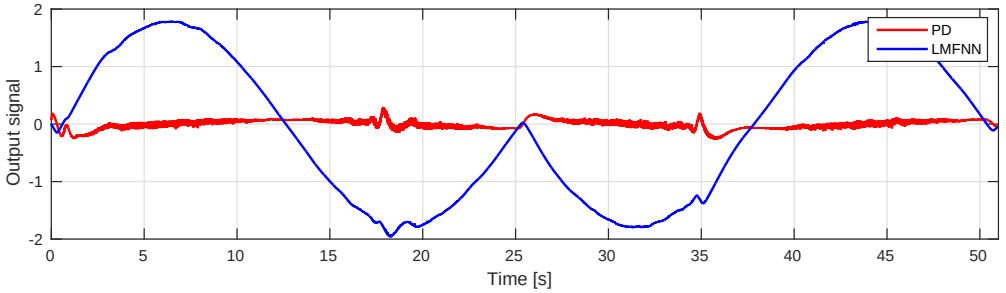
The Euclidean errors in presence and absence of wind are shown in Fig. 8.5. In both cases, the combination of PD and FNN controllers, PD + SMC-FNN and PD + LM-FNN, give a significantly less error than the conventional PD controller when it works alone. On the other hand, the PD controller can be tuned in a more aggressive way to achieve superior results, although this is not practical in real life due to the lack of modelling and unknown disturbances in real-time applications.



(a) x-axis



(b) y-axis



(c) z-axis

Figure 8.4: Control signals for x, y and z axes generated by PD and LM-FNN.

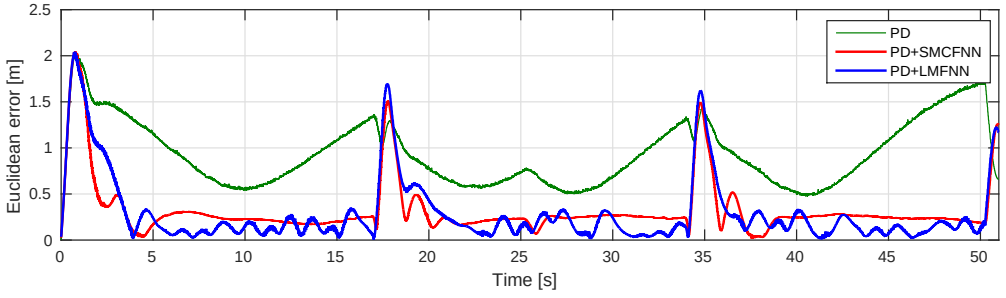


Figure 8.5: Euclidean MSE of different FNN position controllers in presence of wind.

Furthermore, aggressive tuning tends to be the case dependent, and therefore, it cannot give the comparable performance in different conditions; while adaptive learning capabilities of FNN structure are essential for real-world applications.

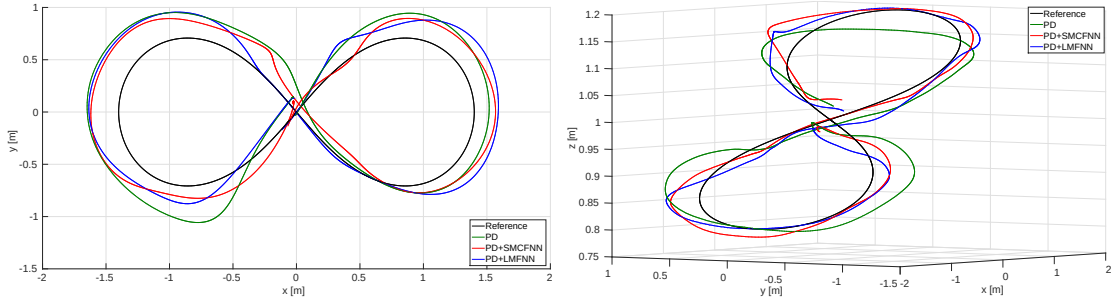
8.5 Experimental Results

The experimental flight tests for the trajectory tracking problem were conducted in the indoor environment. The aircraft used for the experiments is Parrot AR.Drone 2.0 Power Edition UAV, which is a velocity controlled commercial quadrotor. The control approaches are implemented in Linux using ROS and C++ environment.

8.5.1 Discussion

The performance of the PD controller alone and FNN controller trained by SMC and LM working in parallel with the conventional PD controller for the eight-shaped time-based trajectory is presented. The maximum speed along the trajectory is kept to be 1m/s. To evaluate the efficacy of the proposed control strategy, an industrial fan, which generates maximum wind gust of 2m/s, is used to imitate the external disturbances. The wind gust blows along the $[1, -1, 0]$ direction.

In Fig. 8.6, the trajectory tracking is shown for the PD controller and FNN controllers which are tuned by SMC and LM. As can be seen, the steady-state error is notably reduced because of the learning capabilities of FNN structure. This can also be seen from the Euclidean error and the average RMSE values from ten experiments which are shown in Fig. 8.7 and Table 8.1, respectively. The combination of PD and FNN controller gives a significantly less error than the conventional PD controller when it works alone. It should be noted that FNN controllers decrease the PD controller RMSE error by about 36%.



(a) in xy-plane (b) in 3D view

Figure 8.6: Trajectory tracking for $v_w = 2$m/s.

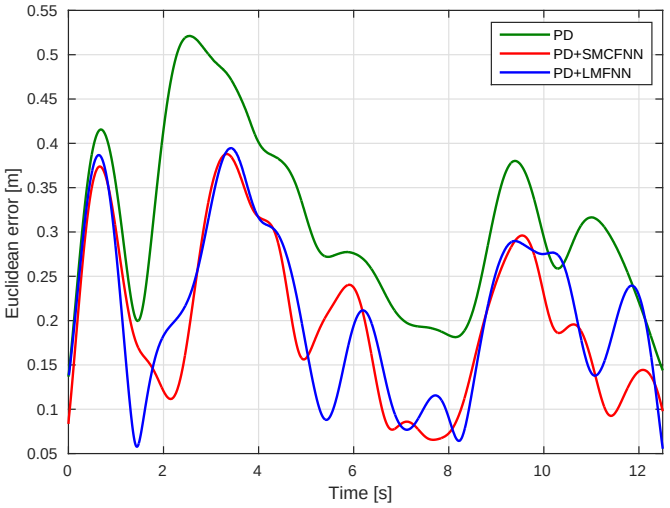
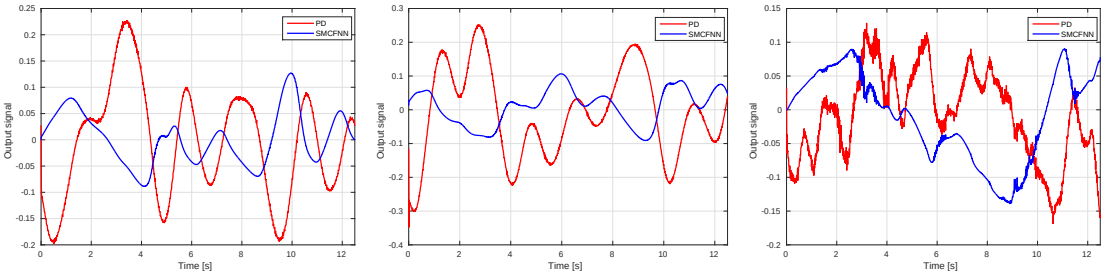


Figure 8.7: Euclidean error with wind gust $v_w = 2\text{m/s}$.

Hence, the trajectory tracking performance of the quadrotor which uses intelligent FNN structure becomes significantly better compared to the normal case when the PD controller is only utilized. However, SMC-based and LM-based FNN controllers were not able completely to take over the control responsibilities from the PD controller as shown in Fig. 8.8 and 8.9. Whereas both FNN controllers were dominating in simulation studies, it was not the case for the real-time tests. The reason for this is the space limitation of the Motion Capture Lab which measures $5 \times 7\text{m}^2$. Therefore, the active area for UAV is around $3 \times 5\text{m}^2$. In such a small area, also by having challenging trajectory, the FNN controller does not have enough time to learn. Besides, there exists a communication delay between computer and AR.Drone UAV. This latency is mainly caused by the WiFi protocol delay as well as the down-sampling/buffering step performed by AR.Drone firmware prior to sending the feedback over WiFi.

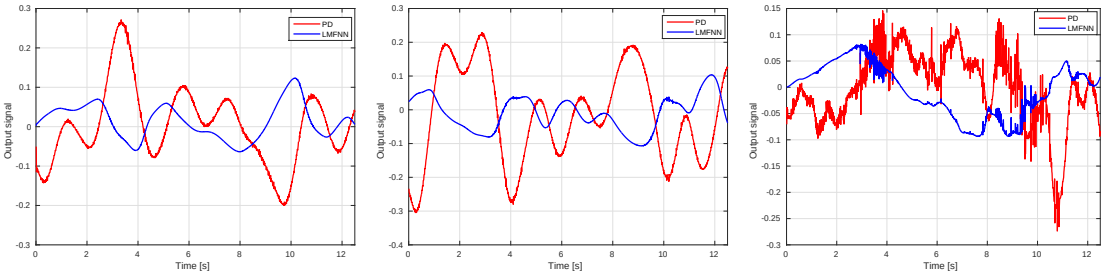
Table 8.1: Average Euclidean RMSE for considered controllers [m].

Controller	**PD**	**SMC-based FNN**	**LM-based FNN**
RMSE	0.327	0.210	0.228



(a) in x-axis (b) in y-axis (c) in z-axis

Figure 8.8: Control signals for x, y and z axes generated by PD and SMC-FNN.



(a) in x-axis (b) in y-axis (c) in z-axis

Figure 8.9: Control signals for x, y and z axes generated by PD and LM-FNN.

8.6 Conclusion

In this chapter, SMC theory and LM-based learning algorithm for intelligent FNN controller are proposed for the control and stabilisation of the quadrotor UAV along a predefined trajectory in the presence of wind gust conditions. The stability analysis of the proposed parameter update rules is presented. It was also demonstrated that proposed methods are capable of significantly reducing the steady-state errors and overcome the disturbances and existed uncertainties which are generated by lack of modelling. Extensive simulations in ROS and Gazebo environment are conducted to evaluate the performance of the proposed controllers with the conventional PD controller. In order to further test the proposed methods, the real-time experiments have been also performed by using OptiTrack Motion Capture System. Experimental results show that the combination of PD and FNN which is tuned by SMC and LM algorithms gives a significantly lower steady-state error than the conventional PD controller when it works alone.

Chapter 9

Deep Fuzzy Neural Network-Based Control

It has been shown that DNNs are good at approximating knowledge, but they do not explain how they take their decisions [30]. On the other hand, FLSs are good at explaining their decisions, but generally, they are not good at acquiring new information [26]. The limitations of these two techniques have been a driving force behind the creation of hybrid systems where the combination of DNN and FLS can overcome the drawbacks of each individual method [31]. In the literature, there are attempts to integrate strengths of learning capability of neural networks and reasoning ability provided by fuzzy logic, called FNN [91]. However, these approaches usually utilise sequential learning paradigms [94]. Correspondingly, one may ask whether a joint learning framework exists that fuses wisely these two methods [150].

In this chapter, potentials of DFNNs are explored under various operational conditions. First, Section 9.1 revises the definition of DFNN. Then, Sections 9.3 and 9.4 show simulation and experimental results for linear system and quadrotor UAVs, respectively. Finally, the conclusions are drawn in Section 9.5.

Supplementary Material:
Video for the experimental results: tiny.cc/DFNN.

9.1 Mathematical Preliminaries

To deal with the problems described in Chapter 2, an adaptive controller which can learn system dynamics online and deal with uncertainties is required. It has been shown that DNNs are able to learn from input-output data, whereas fuzzy logic has an ability to handle noise and uncertainties. The DNN with one antecedent fuzzification layer, also called DFNN, can be used for learning control of nonlinear systems. The DFNN neurons are organised in input layer with N_I neurons, fuzzification layer with $(N_I \times N_F)$ neurons, N_L fully-connected hidden layers with $N_{H,h}$, $h \in \{1, \ldots, N_L\}$, neurons in each layer, and output layer with N_O neurons, as shown in Fig 9.1. The input $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_{N_I}\}$ to DFNN is fuzzified by N_F MFs. Then, the fuzzified input $\{\mu_{F_1}(\mathcal{I}_1), \ldots, \mu_{F_{N_F}}(\mathcal{I}_1), \ldots, \mu_{F_1}(\mathcal{I}_{N_I}), \ldots, \mu_{F_{N_F}}(\mathcal{I}_{N_I})\}$ is forwarded to the first hidden layer of DFNN through the network weights $\mathbf{W}_1$. The hidden layers in DFNN are organized in a fully-connected structure with the network weights $\mathbf{W}_h$, $h = 2, \ldots, N_L$. Finally, the output $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_{N_O}\}$ is computed by applying the network weights $\mathbf{W}_{N_L+1}$ to the output from the last hidden layer.

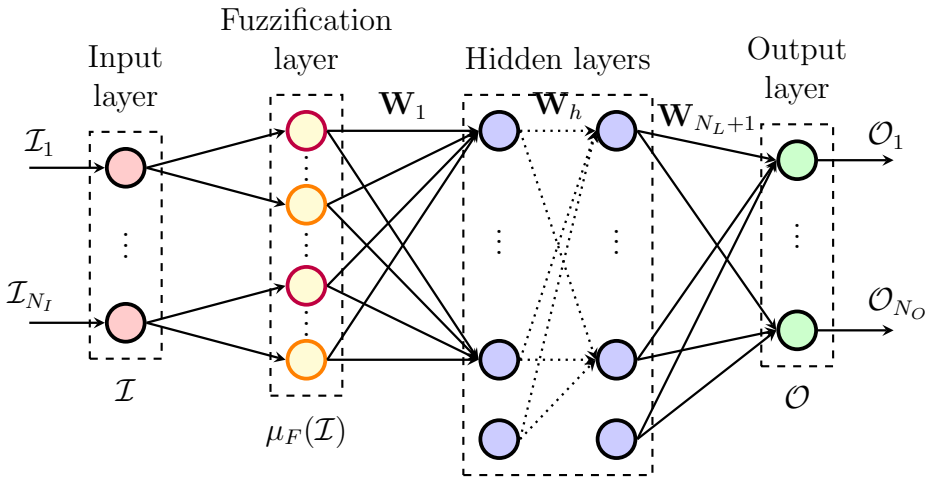


Figure 9.1: Structure of DFNN organised in input layer with N_I neurons, fuzzification layer with $(N_I \times N_F)$ neurons, N_L fully-connected hidden layers with $N_{H,h}$, $h \in \{1, \ldots, N_L\}$, neurons in each layer, and output layer with N_O neurons.

9.2 Network Training

The training is subdivided into two phases: offline pre-training and online training [151]. During the offline pre-training phase, a conventional controller performs a set of trajectories and the batch of training samples is collected. Then, DFNN-based controller, called DFNN_0, is pre-trained on the collected data samples, to approximate the inverse of the system's dynamics. However, DFNN_0 cannot adapt to the new conditions; therefore, the online training is required. During the online learning phase, DFNN controls the system and adapts the control input to improve performances. The expert knowledge encoded into the rule-base, thanks to the fuzzy mapping, provides the adaptation information to DFNN. The approximation of the inverse of the system dynamics is a typical regression problem; therefore, the cost function for both offline and online training is the mean squared error.

9.2.1 Offline Pre-Training

During the offline pre-training phase, a feed-forward DFNN_0 learns the approximate inverse dynamics of the system by adjusting the weights $\left\{\mathbf{W}_1^0, \ldots, \mathbf{W}_{N_L+1}^0\right\}$ in the network. In this control scheme, shown in Fig. 9.2, a conventional controller, e.g., PID controller, controls the system alone. Hence, it is utilized as an ordinary feedback controller to provide labelled training samples for DFNN_0. Each labelled training sample $\mathcal{D}_0(k)$ consists of input $\mathcal{I}_0(k)$ and expected output $\mathcal{O}_0(k)$ pair:

$$\begin{aligned} \mathcal{D}_0(k) &=< \mathcal{I}_0(k), \mathcal{O}_0(k) > \\ &=< \{\mathbf{x}(k-\bar{r}), \mathbf{y}(k)\}, \{\mathbf{u}(k-\bar{r})\} > . \end{aligned} \tag{9.1}$$

where $\bar{r} = \max_{\forall i \in [1, n_O]} \mathbf{r}_i$. The training of DFNN_0 involves back-propagation to minimize the loss over all training examples, and the network weights $\left\{\mathbf{W}_1^0, \ldots, \mathbf{W}_{N_L+1}^0\right\}$ are updated until the over-fitting appears. After the training, DFNN_0 can approximate the inverse dynamics of the nominal system. The pseudo-code of the offline pre-training is given in Algorithm 2.

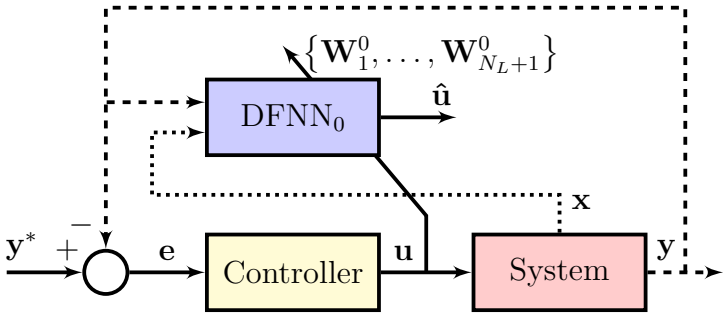


Figure 9.2: Block diagrams of the offline pre-training of DFNN by conventional controller (solid lines represent calculated quantities, dashed lines represent measured quantities, dotted lines represent estimated quantities).

9.2.2 Online Training

During the online training phase, DFNN controls the system and, at the same time, learns how to improve the control performances. Since DFNN training requires supervised learning, another process has to provide feedback about its performances. In the proposed approach, FLS is used to monitor the behaviour of the controlled system. By its nature, FLS incorporates the expert knowledge in the form of rules and uses this knowledge to provide useful advice [152]. The control scheme of the online training is shown in Fig. 9.3.

The objective is to learn the control of the system by only looking at its performance, i.e., the tracking error:

$$\mathbf{e}(k) = \mathbf{y}^*(k) - \mathbf{y}(k), \tag{9.2}$$

and its time derivative:

$$\dot{\mathbf{e}}(k) = \dot{\mathbf{y}}^*(k) - \dot{\mathbf{y}}(k), \tag{9.3}$$

Algorithm 2: Offline pre-training of DFNN.

Input: $\{N_{H,1}, \ldots, N_{H,N_L}\}$
Output: Pre-trained DFNN$_0$
begin
  while $k <$ *MaxSamples* **do**
    Get $\mathbf{x}(k-\bar{r})$, $\mathbf{y}(k)$, and $\mathbf{u}(k-\bar{r})$
    Collect $\mathcal{D}_0(k)$ in (9.1)
  end
  DFNN$_0 \leftarrow$ ConstructNetworkLayers$(N_{H,1}, \ldots, N_{H,N_L})$
  $\{\mathbf{W}^0_1, \ldots, \mathbf{W}^0_{N_L+1}\} \leftarrow$ InitializeWeights()
  Train DFNN$_0$ on $\mathcal{D}_0$
end

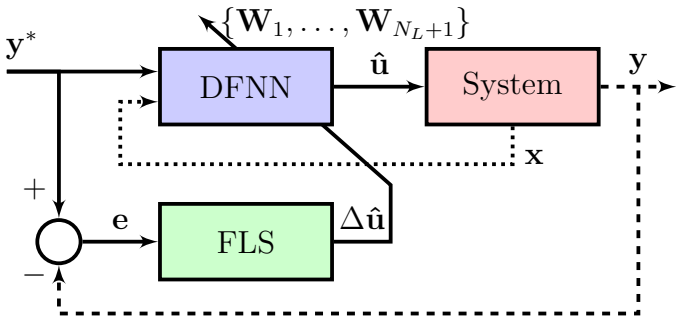


Figure 9.3: Block diagrams of the online training of DFNN by FLS (solid lines represent calculated quantities, dashed lines represent measured quantities, dotted lines represent estimated quantities).

where $\dot{\mathbf{y}} = \frac{\partial \mathbf{y}}{\partial k} \in \mathbb{R}^{N_O}$ is the time derivative of the output from the system, and $\dot{\mathbf{y}}^* = \frac{\partial \mathbf{y}^*}{\partial k} \in \mathbb{R}^{N_O}$ is the time derivative of the desired output.

In the proposed approach, FLS observes the behaviour of the system controlled by DFNN, and, depending on the situation, corrects the action from DFNN. The possible evolutions of the tracking error are depicted in Fig. 9.4. For example, if the error is positive, i.e., $e_j(k) > 0$, and its time derivative is also positive, i.e., $\dot{e}_j(k) > 0$, then the system diverges (top red curve in Fig. 9.4). In this case, FLS will force DFNN to decrease the control signal $u_j(k)$ significantly to stabilize the system, i.e., $\Delta u_j(k) \ll 0$. In another possible case, if the error is negative, i.e., $e_j(k) < 0$, and its time derivative is zero, i.e., $\dot{e}_j(k) = 0$, then the error is steady (bottom purple line in Fig. 9.4). In this case, DFNN falls down in a local minimum and FLS will give a small positive perturbation, i.e., $\Delta u_j(k) > 0$. Finally, if the error is zero, i.e., $e_j(k) = 0$, and its time derivative is also zero, i.e., $\dot{e}_j(k) = 0$, then, this is the optimal case (green line in Fig. 9.4) and no action has to be taken, i.e., $\Delta u_j(k) = 0$.

All these intuitive rules can be formally represented by a typical Mamdani FLS. For each case, one rule R_i, $i \in \{1, \ldots, N_R\}$, exists in the rule-base in Table 9.1, where the colour of the cell corresponds to the curve colour in Fig. 9.4. The inputs to FLS are selected to be the tracking error and its time derivative, i.e., $e_j(k)$ and $\dot{e}_j(k)$; while the output is the correction signal, i.e., $\Delta u_j(k)$. The inputs are represented by three FSs: negative, zero and positive; while the output can belong to five FSs: big decrease, small decrease, no changes, small increase and big increase.

The antecedent MFs are selected to be triangular MFs, as defined in (5.4) and illustrated in Fig. 5.1. On the other hand, the consequent FSs are selected to

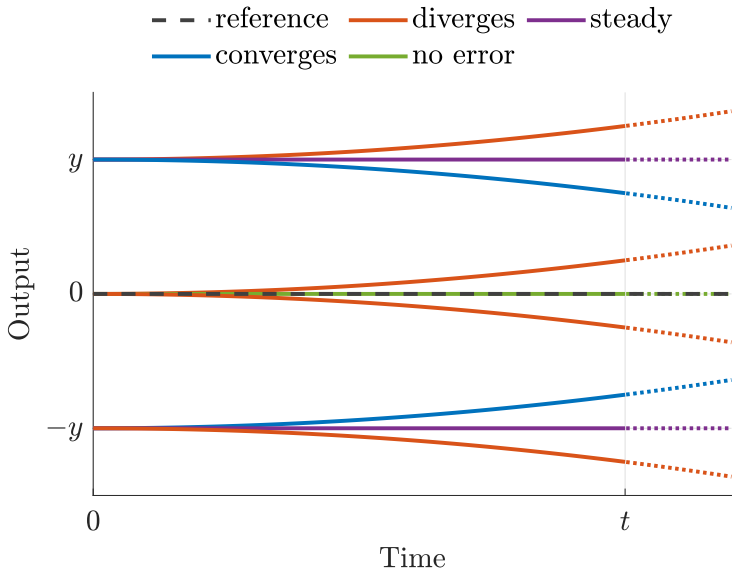


Figure 9.4: Possible evolution of the controlled dynamical system. The system can diverge (red curves), converge (blue curves), it can have a steady error (purple lines), or the error can be zero (green line).

be singleton MFs, as illustrated in Fig. 9.5. Hence, by using the approach in Section 5.2.1, FM which represents FLS described in Table 9.1 in an analytical form can be generated:

$$\varphi(e_j, \dot{e}_j) = |e_j| - \frac{|\dot{e}_j|}{2} - \frac{3}{4}|e_j \dot{e}_j| - \frac{3}{4} e_j \dot{e}_j. \tag{9.4}$$

Finally, (9.4) in its multidimensional form can be used to update the control signal:

$$\Delta \hat{\mathbf{u}}(k) = \alpha \varphi \left(\mathbf{e}(k), \dot{\mathbf{e}}(k) \right), \tag{9.5}$$

where $\alpha > 0$ is a scaling factor.

Table 9.1: Rule-base for the updates of $u_j(k)$.

$e_j(k)$	$\dot{e}_j(k)$		
	Negative	**Zero**	**Positive**
Negative	R_1 : Big decrease	R_2 : Big increase	R_3 : Small increase
Zero	R_4 : Small decrease	R_5 : No changes	R_6 : Small decrease
Positive	R_7 : Small increase	R_8 : Big increase	R_9 : Big decrease

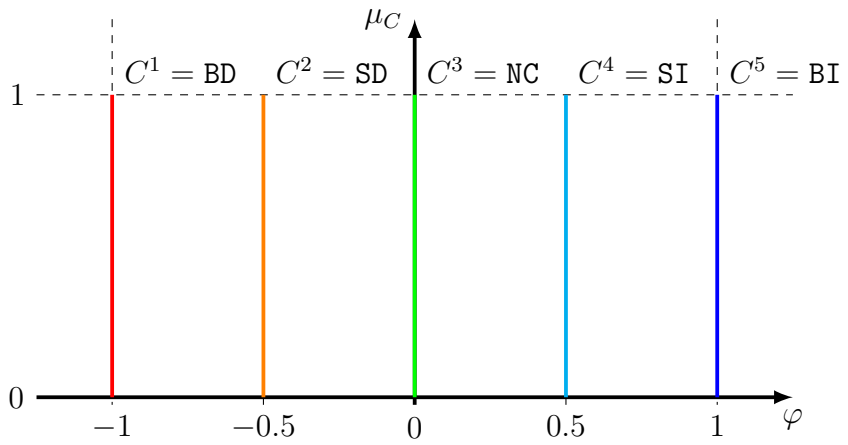


Figure 9.5: "Big decrease" (BD), "small decrease" (SD), "no change" (NC), "small increase" (SI) and "big increase" (BI) consequent FSs for the update of the control signal represented by five singleton MFs.

Remark 9.1. A large α allows learning faster at the cost of possible divergence or oscillation of the controlled system. A smaller α may allow learning more safely and conservatively but may make the learning significantly longer.

Remark 9.2. If $\mathbf{e}(k)$ and $\mathbf{e}(k-1)$ are approximately 0, i.e., $\mathbf{e}(k) \approx \mathbf{0} \wedge \mathbf{e}(k-1) \approx \mathbf{0}$; then, $\dot{\mathbf{e}}(k)$ approaches 0, i.e., $\dot{\mathbf{e}}(k) \to \mathbf{0}$. Consequently, from (9.5) and (9.4), $\Delta\hat{\mathbf{u}}$ is asymptotically equivalent to 0, i.e., $\Delta\hat{\mathbf{u}}(k) \sim \mathbf{0}$. Therefore, the weights are not updated, and the convergence condition is reached.

Each labelled training sample $\mathcal{D}(k)$ consists of input $\mathcal{I}(k)$ and corrected output $\mathcal{O}(k)$ pair:

$$\begin{aligned} \mathcal{D}(k) &=< \mathcal{I}(k), \mathcal{O}(k) > \\ &=< \{\mathbf{x}(k), \mathbf{y}^*(k+\bar{r})\}, \{\hat{\mathbf{u}}(k) + \Delta\hat{\mathbf{u}}(k)\hat{\mathbf{u}}(k)\} > . \end{aligned} \tag{9.6}$$

At each iteration, DFNN is adapted with this training sample, and the new output from the network is computed:

$$\hat{\mathbf{u}}(k) = \text{DFNN}(\mathcal{I}(k)). \tag{9.7}$$

The pseudo-code of online training is provided in Algorithm 3.

For the system in (2.1), a necessary condition for the stability of the inverse dynamics, and hence for the effectiveness of the DFNN-based approach, is the stability of the zero dynamics of the system [153].

Algorithm 3: Online training of DFNN.

Input: Pre-trained DFNN_0
Output: Trained DFNN
begin
  $\text{DFNN} \leftarrow \text{DFNN}_0$
  repeat
    Get $\mathbf{y}(k)$, $\mathbf{y}^*(k)$, $\dot{\mathbf{y}}(k)$ and $\dot{\mathbf{y}}^*(k)$
    $\mathbf{e}(k) \leftarrow \mathbf{y}^*(k) - \mathbf{y}(k)$ by using (9.2)
    $\dot{\mathbf{e}}(k) \leftarrow \dot{\mathbf{y}}^*(k) - \dot{\mathbf{y}}(k)$ by using (9.3)
    $\Delta\hat{\mathbf{u}}(k) \leftarrow \alpha\text{FLS}(\mathbf{e}(k), \dot{\mathbf{e}}(k))$ by using (9.5)
    $\hat{\mathbf{u}}(k) \leftarrow \text{DFNN}(\mathcal{I}(k))$ by using (9.7)
    Adapt DFNN with $\mathcal{D}(k)$ by using (9.6)
    Send $\mathbf{u}(k)$ to the system
  until *Stop*
end

Theorem 9.2.1 (Stability of DNN)**.** Consider the system in (2.1) and the control signal $\hat{\mathbf{u}}(k)$ in (9.7). The overall closed-loop system is bounded-input bounded-output (BIBO) stable iff Assumptions 1, 2, 3, 4 and 7 are verified.

Proof. A dynamical system is BIBO stable, if for any bounded input corresponds a bounded output. In the proposed approach, the input is $\mathbf{y}^*$; while the output is $\mathbf{y}$.

i) If the controlled system is the nominal system on which DFNN_0 was trained, and if the output from DFNN_0, $\hat{\mathbf{u}}(k)$, accurately approximates the exact inverse of the system, $\mathbf{u}(k)$, i.e., $\hat{\mathbf{u}}(k) \approx \mathbf{u}(k)$; then, the inverse model update, $\Delta\hat{\mathbf{u}}(k) \sim \mathbf{0}$. Thus, the control input to the system is $\mathbf{u}(k)$. From (2.5), the system's output, $\mathbf{y}$, can be bounded by:

$$\|\mathbf{y}(k)\|_\infty \leq c_1\|\mathbf{x}(k)\|_\infty + c_2\|\mathbf{u}(k)\|_\infty + c_3 \quad \forall k, \tag{9.8}$$

where c_1, c_2 and c_3 are some positive constants. From Assumption 3, the system's state, $\mathbf{x}$, can be bounded by:

$$\|\mathbf{x}(k)\|_\infty \leq c_4\|\mathbf{u}(k)\|_\infty + c_5\|\mathbf{x}(0)\|_\infty + c_6 \quad \forall k, \tag{9.9}$$

where c_4, c_5 and c_6 are some positive constants. From (2.8), the system's input, $\mathbf{u}$, can be bounded by:

$$\|\mathbf{u}(k)\|_\infty \leq c_7\|\mathbf{x}(k)\|_\infty + c_8\|\mathbf{y}^*(k)\|_\infty + c_9 \quad \forall k, \tag{9.10}$$

where c_7, c_8 and c_9 are some positive constants. By using (9.9) and (9.10) in (9.8), the overall closed-loop system in Fig. 9.2 is BIBO stable.

ii) If the controlled system is different from the nominal system on which DFNN_0 was trained, or if the output from DFNN_0, $\hat{\mathbf{u}}(k)$, does not approximate accurately the exact inverse of the system, $\mathbf{u}(k)$, i.e., $\hat{\mathbf{u}}(k) \not\approx \mathbf{u}(k)$; then, the inverse model update, $\Delta\hat{\mathbf{u}}(k) \neq \mathbf{0}$. Therefore, the control input to the system is $\hat{\mathbf{u}}(k) + \Delta\hat{\mathbf{u}}(k)$. From (2.5), the system's output, $\mathbf{y}$, can be bounded by:

$$\begin{aligned}\|\mathbf{y}(k)\|_\infty &\leq c_1\|\mathbf{x}(k)\|_\infty + c_2\|\hat{\mathbf{u}}(k) + \Delta\hat{\mathbf{u}}(k)\|_\infty + c_3 \\ &\leq c_1\|\mathbf{x}(k)\|_\infty + c_2\|\hat{\mathbf{u}}(k)\|_\infty + c_2\|\Delta\hat{\mathbf{u}}(k)\|_\infty + c_3 \quad \forall k.\end{aligned} \tag{9.11}$$

From Assumption 3, the system' state, $\mathbf{x}$, can be bounded by:

$$\begin{aligned}\|\mathbf{x}(k)\|_\infty &\leq c_4\|\hat{\mathbf{u}}(k) + \Delta\hat{\mathbf{u}}(k)\|_\infty + c_5\|\mathbf{x}(0)\|_\infty + c_6 \\ &\leq c_4\|\hat{\mathbf{u}}(k)\|_\infty + c_4\|\Delta\hat{\mathbf{u}}(k)\|_\infty + c_5\|\mathbf{x}(0)\|_\infty + c_6 \quad \forall k.\end{aligned} \tag{9.12}$$

From (2.8), the output from the DFNN module, $\hat{\mathbf{u}}$, can be bounded by:

$$\|\hat{\mathbf{u}}(k)\|_\infty \leq \sum_{i=1}^{N_L} c_{\mathbf{W},i}\|\mathbf{W}_i(k)\|_\infty + c_{10} \quad \forall k, \tag{9.13}$$

where c_{10} is some positive constant. It has to be noted that the inputs to DFNN and FLS modules are bounded by the Gaussian and triangular MFs in (3.7) and (5.4), respectively. Simultaneously, the output from the FLS module is bounded by the singleton MFs in Fig. 9.5, i.e.:

$$\|\Delta\hat{\mathbf{u}}(k)\|_\infty \leq \alpha < \infty \quad \forall k. \tag{9.14}$$

By using (9.12), (9.13) and (9.14) in (9.11), the overall closed-loop system in Fig. 9.3 is BIBO stable.

Since both cases with and without online learning are BIBO stable, the overall closed-loop system is BIBO stable. □

In the universal approximation theorem [154], it has been proven that ANN with a single-hidden-layer containing a finite number of neurons can approximate any continuous function. In [155], the authors found the relation between the number of hidden neurons and the function complexity. Therefore more layers and neurons the neural network has, more complex mathematical relation it can approximate with higher accuracy.

On the other hand, from the asymptotic analysis, the runtime complexity for the forward-propagation is $\mathrm{O}(N_L \cdot N_H^3 + N_L \cdot N_H) \equiv \mathrm{O}(N_L \cdot N_H^3)$. While the runtime complexity for the back-propagation is $\mathrm{O}(N_{QN} \cdot N_L \cdot N_H^4 + N_L \cdot N_H^3) \equiv \mathrm{O}(N_{QN} \cdot N_L \cdot N_H^4)$, where N_{QN} is the number of iterations in the quasi-Newton method. Moreover, the runtime complexity for FM in (9.5) is constant w.r.t. the architecture of the network, i.e., $\mathrm{O}(1)$. The dominant operation in DFNN_0 is the forward-propagation; therefore, the runtime complexity of DFNN_0 is polynomial. However, DFNN with online learning involves both forward-propagation and back-propagation; therefore, the runtime complexity of DFNN is also polynomial but asymptotic to $\mathrm{O}(N_{QN} \cdot N_L \cdot N_H^4)$. At the same time, from the asymptotic analysis, the space complexity to store DFNN is $\mathrm{O}(N_I \cdot N_M \cdot N_H + (N_L - 1) \cdot N_H^2 + N_H \cdot N_O) \equiv \mathrm{O}(N_L \cdot N_H^2)$. Moreover, the space complexity for FM in (9.5) is constant w.r.t. the architecture of the network, i.e., $\mathrm{O}(1)$.

Table 9.2 summarises the runtime and space complexities for the considered controllers. It is possible to observe that DFNN has the highest runtime complexity, since it performs the online back-propagation which is computationally expensive. Besides, the space complexity of DFNN_0 and DFNN is asymptotically equivalent, since most of the space is occupied to store the weights of DFNN.

Remark 9.3. The architecture of DFNN in the proposed approach should be chosen as a compromise between the learning capability of the neural network and the update time through the back-propagation.

Table 9.2: Asymptotic analysis of different controllers.

Complexity	**PID**	**DFNN_0**	**DFNN**
Runtime	$\mathrm{O}(1)$	$\mathrm{O}(N_L \cdot N_H^3)$	$\mathrm{O}(N_{QN} \cdot N_L \cdot N_H^4)$
Space	$\mathrm{O}(1)$	$\mathrm{O}(N_L \cdot N_H^2)$	$\mathrm{O}(N_L \cdot N_H^2)$

9.3 Simulation Results

The proposed approach is tested on four single-input-single-output systems: nominal system, nominal system with internal uncertainties, nominal system with external disturbance, and nominal system with noisy measurements. The desired trajectory is:

$$y^*(k) = \frac{1}{3}\sin(3\pi k) + \frac{1}{2}\cos(2\pi k) - 1. \tag{9.15}$$

while its time derivative is:

$$\dot{y}^*(k) = \pi\cos(3\pi k) - \pi\sin(2\pi k). \tag{9.16}$$

The nominal nonlinear system is selected as:

$$\begin{cases} \mathbf{x}(k+1) &= \begin{bmatrix} x_2 \\ x_1 - x_1^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= -0.2x_1 + x_2. \end{cases} \tag{9.17}$$

The nominal system in (9.17) with internal uncertainties is:

$$\begin{cases} \mathbf{x}(k+1) &= \begin{bmatrix} 0.5x_2 \\ 0.5x_1 - 0.5x_1^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} u(k) \\ y(k) &= -0.1x_1 + 0.5x_2. \end{cases} \tag{9.18}$$

The nominal system in (9.17) with external variable disturbance is:

$$\begin{cases} \mathbf{x}(k+1) &= \begin{bmatrix} x_2 \\ x_1 - x_1^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ \cos(2k) \end{bmatrix} \\ y(k) &= -0.2x_1 + x_2. \end{cases} \tag{9.19}$$

The nominal system in (9.17) with noisy measurements is:

$$\begin{cases} \mathbf{x}(k+1) &= \begin{bmatrix} x_2 \\ x_1 - x_1^3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= -0.2x_1 + x_2 + \mathcal{N}(0, y(k-1)). \end{cases} \tag{9.20}$$

Throughout the simulations, the systems in (9.17)–(9.20) are assumed to be black-boxes. Therefore, only input-output data and some basic properties, such as the relative degree $\bar{r}$, are used. It is possible to verify with (2.2) that the relative degree $\bar{r} = 1$ for the systems in (9.17)–(9.20). A feed-forward DFNN with hyperbolic tangent (tanh) activation functions is designed. According to (9.1) and (9.6), DFNN has three inputs ($N_I = 3$) and one output ($N_O = 1$). In addition, after some heuristic analysis and experimental trials, the architecture of the network is chosen to consist of one fuzzification layer with three MFs ($N_F = 3$) and two fully connected hidden layers ($N_L = 2$) with 16 neurons in each layer ($N_{H,1} = N_{H,2} = 16$). The scaling factor $\alpha = 0.1$ in (9.5), for all cases.

In order to collect the training data for the offline pre-training, the nominal system is controlled by the PID controller which has been tuned by trial-and-error. Thus, 2′000 input-output training pairs $\mathcal{D}_0(k) =< \{x_1(k-1), x_2(k-1), y(k)\}, \{u(k-1)\} >$ are stored. Then, DFNN_0 is trained on $\mathcal{D}_0$ by using the LM algorithm. After that, the pre-trained DFNN controls the system online and, at each iteration, it is updated with $\mathcal{D}(k) =< \{x_1(k), x_2(k), y^*(k+1)\}, \{u(k) + \Delta u(k)\} >$.

9.3.1 Discussion

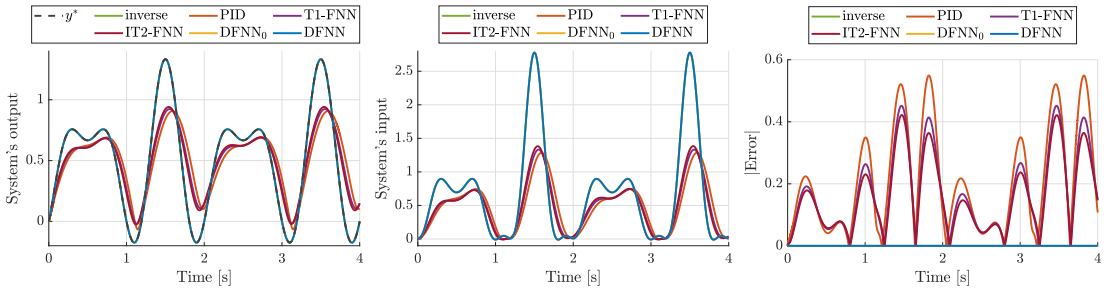
To show the efficiency of the proposed approach, the performances of the developed DFNN with online learning are compared with the performances of the exact analytical inverse of the system dynamics, PID controller (used for the collection of training samples), type-1 FNN (T1-FNN)-based controller with Gaussian antecedent MFs and LM theory-based update rules presented in Section 8.3, and interval type-2 FNN (IT2-FNN)-based controller with elliptic antecedent MFs and SMC theory-based adaptation laws from [6], and DFNN_0 controller without online training. As can be seen from Figs. 9.6–9.9, the exact analytical system inverse is able to track perfectly the reference trajectory. However, in real-world it is difficult, and sometimes even impossible, to calculate the exact inverse dynamics of the system. From Fig. 9.6, it is possible to observe that both DFNN_0 and DFNN with online

learning are able to track precisely the desired trajectory of the nominal system. Besides, DFNN_0 approximates very accurately the exact inverse of the system in (2.8). From Fig. 9.7, it is possible to observe that DFNN_0 is not able to track the desired trajectory on the system with modified dynamics, and its performances are worse than the ones of PID, T1-FNN and IT2-FNN controllers; while DFNN with online learning is able to learn the new system dynamics and obtain a good performance. From Fig. 9.8, it is possible to observe that PID, T1-FNN and DFNN_0 become unstable with time-varying disturbance; while DFNN with online learning is able to learn new conditions and obtain a good performance. From Fig. 9.9, it is possible to observe that PID controller is not able to deal with this level of noise; while all T1-FNN, IT2-FNN, DFNN_0 and DFNN are able to control the system.

As can be seen from Table 9.3, the exact analytical system inverse is able to track the reference trajectory with negligible error. On the other hand, the DFNN-based controller with online learning outperforms all PID, T1-FNN, IT2-FNN and DFNN_0 for all tested cases in terms of MAE. Only in the case with noisy measurements, DFNN with online learning tries to learn also the noise which makes its performances worst than DFNN_0.

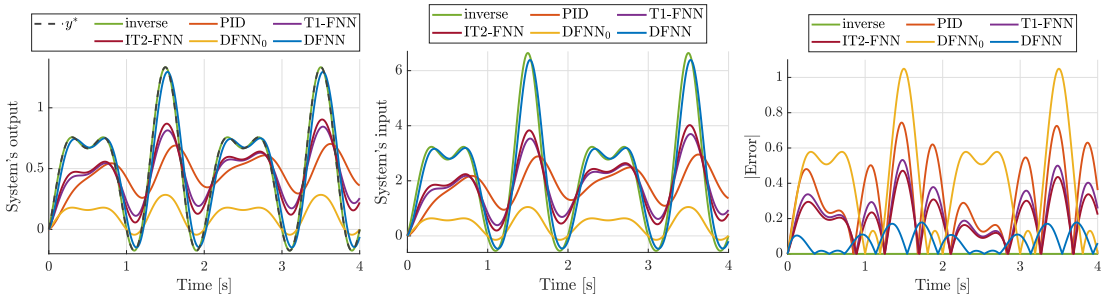
Table 9.3: Comparison of different controllers in terms of MAE [m].

Controller	**System in (9.17)**	**System in (9.18)**	**System in (9.19)**	**System in (9.20)**
Inverse	~ 0	~ 0	~ 0	~ 0
PID	0.265	0.386	∞	∞
T1-FNN	0.216	0.261	∞	0.225
IT2-FNN	0.196	0.225	0.223	0.204
DFNN_0	7.53×10^{-6}	0.505	∞	0.097
DFNN	1.01×10^{-6}	0.090	0.038	0.100



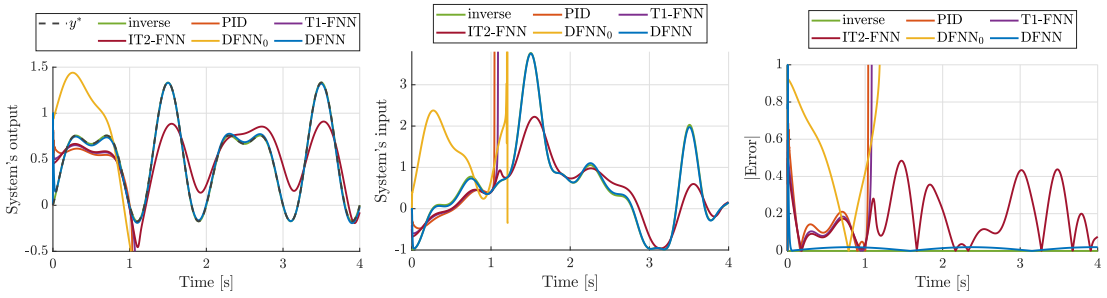
(a) Tracking. (b) Control signal. (c) Tracking error.

Figure 9.6: Performances on the nominal system in (9.17).



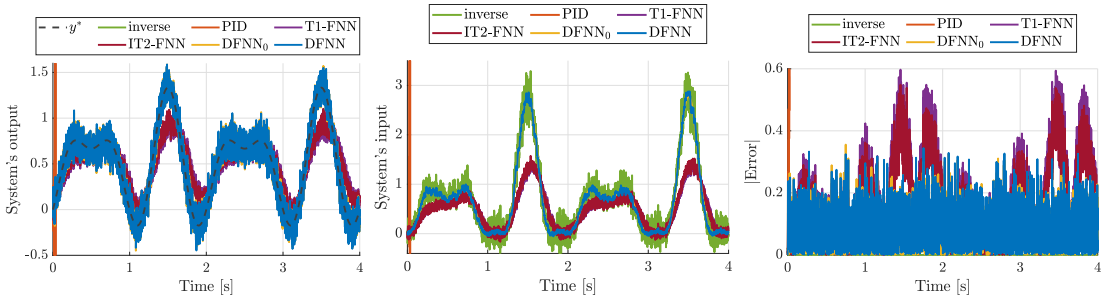
(a) Tracking. (b) Control signal. (c) Tracking error.

Figure 9.7: Performances on the system with internal uncertainties in (9.18).



(a) Tracking. (b) Control signal. (c) Tracking error.

Figure 9.8: Performances on the system with external disturbance in (9.19).



(a) Tracking. (b) Control signal. (c) Tracking error.

Figure 9.9: Performances on the system with noisy measurements in (9.20).

9.4 Experimental Results

To validate the capabilities of the proposed controller, the trajectory following problem of a quadcopter UAV is considered. The experimental platform used in this work is Parrot Bebop 2 quadcopter UAV, and ROS is used to communicate with UAV. The visual-inertial odometry algorithm is used to provide the UAV's real-time position at 24Hz. This information is fed into the ground station computer (CPU: 2.6GHz, 64bit, quad-core; GPU: 4GB; RAM: 16GB DDR4) where the controllers are executed. The computed control signal is sent to UAV at 100Hz.

The dynamical system of UAV is subdivided into three simpler subsystems to reduce the complexity and accelerate the learning process. Three feed-forward DFNNs are used to learn the control mapping for each controlled axis: x, y and z, as depicted in Fig. 2.5. From the dynamical model of UAV, it is possible to calculate that the relative degree $\bar{r} = 2$. According to (9.1) and (9.6), each DFNN has three inputs ($N_I = 3$) and one output ($N_O = 1$). In addition, after some heuristic analysis and experimental trials, the architecture of the network is chosen to consist of one fuzzification layer with three MFs ($N_F = 3$), and two fully connected hidden layers ($N_L = 2$) with 64 neurons in each layer ($N_{H,1} = N_{H,2} = 64$) and with hyperbolic tangent (tanh) activation functions. The inputs to DFNN for the x-axis are the state components relative to the x-axis, $\{x(k), u(k), x^*(k+2)\}$, and the output is the desired pitch angle, $\{\theta^*(k)\}$. Similarly, the inputs to DFNN for the y-axis are the state components relative to the y-axis, $\{y(k), v(k), y^*(k+2)\}$, and the output is the desired roll angle, $\{\phi^*(k)\}$. Finally, the inputs to DFNN for the z-axis are the errors and their time derivatives on the z-axis, $\{z(k), w(k), z^*(k+2)\}$, and the output is the desired vertical velocity, $\{w^*(k)\}$.

The error type is an important term in the loss index, and, in the proposed approach, it is chosen as the normalized squared error. The initialization algorithm is used to bring the neural network to a stable region of the loss function, and, in the proposed approach, it is selected as the random search. The training algorithm is the core part of the training, and, in the proposed approach, the quasi-Newton method is chosen for both offline and online training. The scaling factor $\alpha = 0.1$ in (9.5).

Remark 9.4. The DFNN controllers with and without online learning consist of three independent and parallel sub-networks for x, y and z axes to speed up the learning.

To prepare the training samples of the flight data, the system was controlled by a conventional stable position controller alone. The current and past states were saved as inputs, while the corresponding control signals was saved as the labelled output. By using the PID controller, 100′000 instances have been collected in the training dataset for each axis. This dataset is large enough for the trajectory tracking application, however, the proposed method does not have any limitations on the dataset size. The training data include circular and eight-shaped trajectories on xy-, xz- and yz-planes with the reference speed of 1m/s.

The tracking of four circular trajectories with different working conditions (slow, fast, near-ground and with payload) have been tested. The study cases are designed in a way to exploit different components of UAV dynamics. Furthermore, the visual-inertial algorithm for state estimation produces noisy output [117]. In order to show the efficiency and efficacy of the DFNN-based controller with online learning, it is compared with a well-tuned PID controller (used for the collection of training samples), T1-FNN-based controller with Gaussian antecedent MFs and LM theory-based update rules presented in Section 8.3, and IT2-FNN-based controller with elliptic antecedent MFs and SMC theory-based adaptation laws from [6], and DFNN_0 controller without online training.

Remark 9.5. For real-time experiments, the inverse dynamics of the system is not used to control the system, since, in real-world, it is difficult, and sometimes even impossible, to estimate the exact inverse dynamics of the system.

9.4.1 Discussion

The first study case is the tracking of the slow circular trajectory with a radius of 2m on the xy-plane at a velocity of 1m/s which has also been used during the pre-training phase. This case study is a reference example where UAV operates in its nominal conditions. The results for this study case are shown in Fig. 9.10.

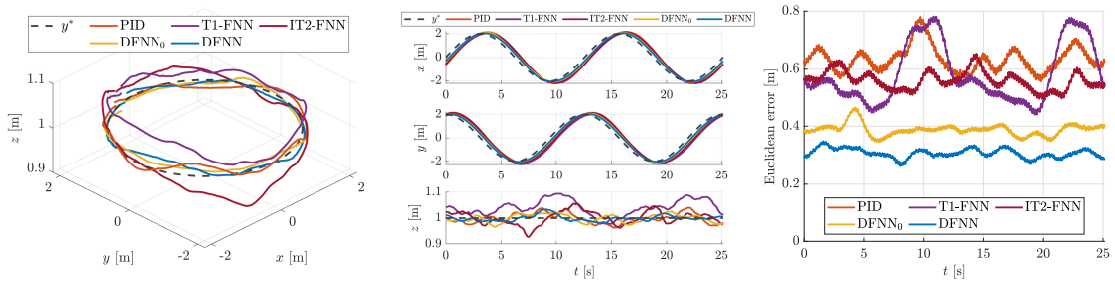
The second study case is the tracking of the fast circular trajectory with a radius of 2m on the xy-plane at a velocity of 2m/s. In this study case, the fast responses of the controllers and the robustness of the visual-inertial state estimator to the motion blur are verified. The results for this study case are shown in Fig. 9.11.

The third study case is the tracking of the circular trajectory while flying at a height of 0.2m. In this study case, the ground effect generates an external disturbance on UAV. The results for this study case are shown in Fig. 9.12.

The fourth study case is the tracking of the circular trajectory while flying with a payload (Odroid-C2 onboard computer on top and office scissors attached to the front left arms) of 209g. In this study case, the parameters of the dynamical model of UAV (mass and moments of inertia) are altered by the payload. The results for this study case are shown in Fig. 9.13.

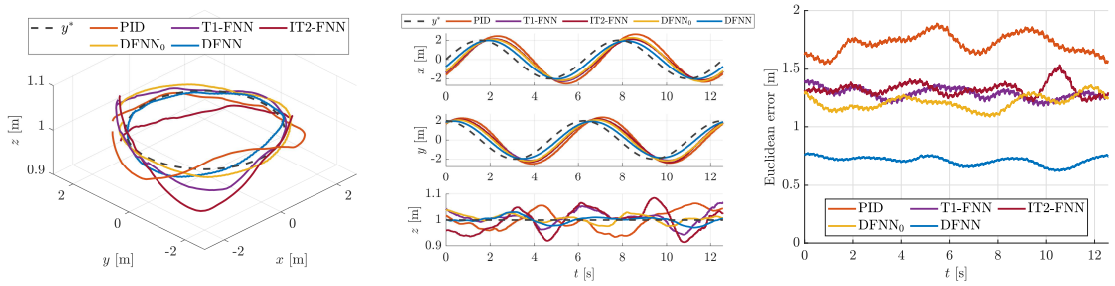
Experimental results for five controllers (PID, T1-FNN, IT2-FNN, DFNN_0 and DFNN) on four different circular trajectories (slow, fast, near-ground and with payload) are illustrated in Figs. 9.10–9.13, respectively. It is possible to observe from Figs. 9.10a–9.13a that DFNN-based controller with online training is able to track more closely the reference trajectory. As visualized from Figs. 9.10b–9.13b, DFNN-base controller has faster responses since it is able to estimate the evolution of the system dynamics and compensate it. From Figs. 9.10c–9.13c, it is possible to observe that DFNN-based controller with online training is able to learn the system dynamics and decrease the tracking error on all tested trajectories.

For the statistical analysis of control performances, the experiments are repeated five times for each controller-case combination for a total of 100 experiments under quasi-same conditions. To compare the trajectory tracking performances, a box-plot is presented in Fig. 9.14. It is possible to observe that on average the DFNN-based controller with online learning outperforms other controllers on the tested trajectories. It has to be emphasised that DFNN evolves online from the pre-trained DFNN_0 during the learning process. Moreover, as expected, DFNN_0 has poor performances in the cases which have not been used for its training. It is also interesting to observe that the performances of the PID controller do not get worse in case of near-ground and with payload trajectories, because derivative and integral components can compensate for these disturbances. Furthermore, the FNN-based controllers (T1-FNN and IT2-FNN) have similar performances for the slow, near-ground and with payload trajectories because their fast learning capabilities can compensate the disturbances coming from the ground effects and increased mass. The maximum absolute error is lower for the online DFNN-based controller, even for the cases unseen during the pre-training. Finally, DFNN-based controller with online learning has the lowest variance of the error.



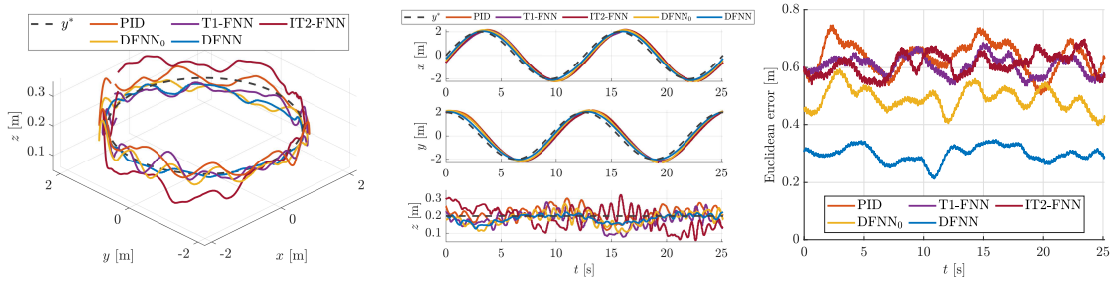
(a) 3D view. (b) x, y and z axes. (c) Euclidean error.

Figure 9.10: Results for the slow circular trajectory at velocity of 1m/s.



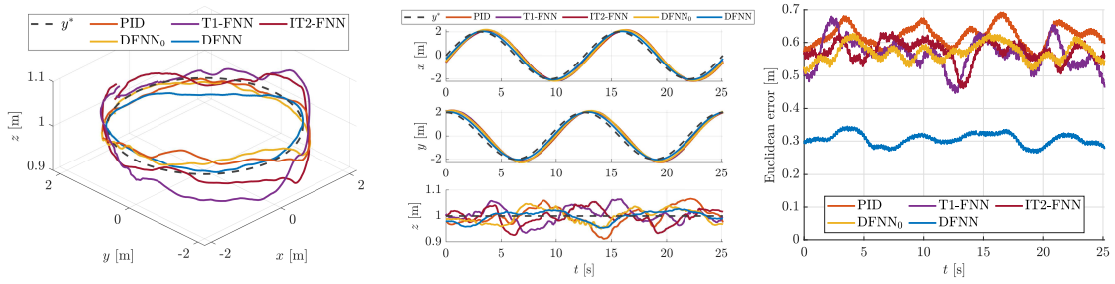
(a) 3D view. (b) x, y and z axes. (c) Euclidean error.

Figure 9.11: Results for the fast circular trajectory at velocity of 2m/s.



(a) 3D view. (b) x, y and z axes. (c) Euclidean error.

Figure 9.12: Results for the near-ground circular trajectory at height of 0.2m.



(a) 3D view. (b) x, y and z axes. (c) Euclidean error.

Figure 9.13: Results for the circular trajectory with payload of 209g.

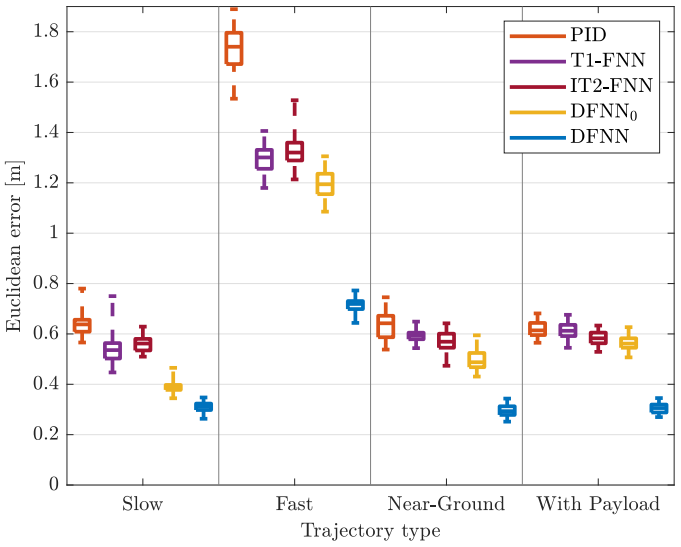


Figure 9.14: Tracking performances of five controllers in four scenarios.

As can be seen from Table 9.4, the DFNN-based controller with online learning outperforms all tested controllers in all tested study cases in terms of MAE. Averaged results from numerous experiments depict that the overall improvement of 51%, 59%, 53% and 51% in terms of MAE is achieved as compared to a well-tuned PID controller for slow, fast, near-ground and with payload cases, respectively.

Nevertheless, the online DFNN-based controller can learn promptly the system dynamics, the computing time is still the main drawback of this controller because of the online back-propagation. The computing time is polynomially proportional to the number of hidden layers and the number of neurons in each hidden layer, i.e., $O(N_L \cdot N_H^4)$. Therefore, deeper is the network, more complex functions it can learn, but more computational power it requires. The average experimental computation time for DFNN with online back-propagation is around 9.4ms, while for PID, T1-FNN, IT2-FNN and DFNN_0 without online learning this time is only 8μs, 11μs, 13μs and 32μs, respectively. However, 9.4ms is still an acceptable time for real-time applications, which allows the controller to run at 100Hz.

Table 9.4: Comparison of different controllers in terms of MAE [m].

Trajectory	**PID**	**T1-FNN**	**IT2-FNN**	**DFNN_0**	**DFNN**
Slow	0.640	0.593	0.568	0.387	0.307
Fast	1.710	1.182	1.265	1.204	0.704
Near-Ground	0.638	0.609	0.601	0.497	0.299
With Payload	0.620	0.555	0.546	0.570	0.306

9.5 Conclusion

In this chapter, a novel approach is presented for the control of dynamical systems that improves the system's control performance online by combining deep learning and fuzzy logic. The learning is subdivided into two phases: offline and online training. During the offline training phase, a conventional controller performs a set of trajectories and a batch of training samples is collected. Then, a DFNN-based controller, DFNN_0, is pre-trained on the collected data samples. However, DFNN_0 cannot adapt to new operating conditions different from the pre-training cases; therefore, online training is required. During the online training phase, DFNN-based controller takes control of the system and adapts to improve the tracking performance. The expert knowledge encoded into the rule-base, thanks to the derived fuzzy mapping, provides the adaptation information to DFNN allowing the online learning. Once DFNNs are trained, the experimental results show that the proposed approach improves the performance by more than 50% when compared to a conventional controller. The results of this study might open doors to wider use of DFNN-based controllers in real-world control applications.

Part V

Final Remarks

Chapter 10

Conclusion

In this thesis, a possible solution for an accurate trajectory following of UAVs in uncertain and noisy environments is presented.

The investigation has started with T1-FLCs with different inference engines, namely SFLC, Sta-NSFLC, Cen-NSFLC and Sim-NSFLC. Extensive simulation and experimental tests have shown that non-singleton FLCs are able to obtain better control performances when compared to singleton FLCs, especially at higher flight speeds which induce higher uncertainty and noise levels. Moreover, different input fuzzification levels can achieve various capabilities for capturing input uncertainties. In other words, the higher input fuzzification has more capability to handle higher-level input noise. However, T1-FLCs can effectively handle only bounded levels of uncertainty and noise, while real-world applications frequently have to deal with high levels of uncertainty and multiple sources of noise.

There exist neither a systematic way to choose MFs to achieve better uncertainty modelling capability nor an objective criterion to check its performance. A comparative analysis is made in which IT2-FLCs are compared and contrasted to T1-FLCs for modelling uncertainty. Elliptic MFs are unique amongst existing type-2 fuzzy MFs because of the decoupled parameters for its support and width. The findings say that IT2-FLCs with elliptic MFs have better performances when compared to conventional PID controller and type-1 counterpart. However, the developed IT2-FLCs are computationally slower than the traditional PID controller.

Therefore, an alternative systematic approach to explicitly derive the mathematical input-output relationships of T1-FLCs and IT2-FLCs has been presented. These nonlinear closed-form relationships allowed to verify some important characteristics of both T1-FLC and IT2-FLC, like symmetry, continuity and monotonicity. The presented design method for IT2-FLC has only one parameter of FOU to be selected, i.e., aggressiveness parameter. By only modifying this parameter, IT2-FLCs can be designed in an easy manner to have more aggressive or smoother behaviour. To prove these theoretical claims, the developed controllers have been tested in simulation and experimental case studies for the way-points tracking control of UAV. It has been shown that the theoretical claims and expectations match the results in the case studies. However, one weakness of all FLCs is that the parameters of their MFs have to be tuned to deal with uncertainties.

Another branch of artificial intelligence (AI)-based controllers is ANN-based controller which enables learning for the control of UAV in various challenging conditions. A fast flight manoeuvre at speeds of 18m/s is performed to show the superior performance of the proposed controller. While performing a pre-defined task, UAV experiences a single motor failure, and the controller handles the failure ensuring the safety of the mission as well as UAV. The model-free nature of the controller helps in accurate trajectory tracking even for high-speed and agile manoeuvres. The advantage of the proposed controller is that it does not need a well-tuned set of PD gains as it learns online and improves the performance metrics while following the trajectory. Moreover, the proposed controller is computationally light to be implemented on the onboard computer of UAV. The real-time experiments are carried out in the outdoor environment with the use of RTK-GPS for localization. For all the phases of the considered scenario, the proposed controller outperforms the conventional PID controllers. The average improvement of the ANN-based controller is above 40%. However, common single-hidden-layer ANNs are not able to learn the complete inverse dynamics of the system.

Given the ability of DNNs to generalise knowledge, a DNN-enhanced control architecture has been proposed to improve the tracking performance of traditional feedback controllers for any given desired trajectory. For the problem of transfer learning, an online learning approach has been proposed to efficiently transfer a DNN module trained on a source robot system to control a target robot system. An expression of the online module for achieving exact tracking has been derived.

Then, based on a linear system formulation, an approach for characterising system similarity has been proposed. This approach was verified experimentally by applying the proposed online learning approach to transfer a DNN inverse dynamics module across two similar but different quadrotor platforms. On 10 arbitrary hand-drawn trajectories, the DNN module of the source system reduces the tracking error of the target system by an average of 46%. The incorporation of the online module further reduces the tracking error and leads to an average of 74% error reduction. These experimental results show that the proposed online learning and knowledge transfer approach can efficaciously circumvent data recollection on the target robot.

Additionally, in this thesis, SMC and LM theory-based learning algorithms for intelligent FNN controllers are proposed for the control and stabilising of a quadrotor UAV along a predefined trajectory in the presence of periodic wind gust conditions. It was also demonstrated that proposed methods are capable to significantly reduce the steady-state errors and overcome the periodic disturbances and existed uncertainties which are generated by the lack of modelling. Simulations and experimental results show that the combination of PD and FNN which is tuned by SMC and LM algorithms gives a significantly lower tracking error than the conventional PD controller when it works alone.

In the end, in this thesis, a novel approach has been proposed for the control of dynamical systems to improve system's control performances online by combining deep learning and fuzzy logic. The learning is subdivided into two phases: offline and online training. During the offline training phase, a conventional controller performs a set of trajectories and a batch of training samples is collected. Then, a DFNN-based controller, DFNN_0, is pre-trained on the collected data samples. However, DFNN_0 cannot adapt to new operating conditions different from the pre-training cases; therefore, online training is required. During the online training phase, DFNN-based controller takes control of the system and adapts to improve the tracking performance. The expert knowledge encoded into the rule-base, thanks to the derived fuzzy mapping, provides the adaptation information to DFNN allowing the online learning. Once DFNNs are trained, the experimental results show that the proposed approach improves the performance by more than 50% when compared to a conventional controller. This approach might open new doors to a wider use of DFNN-based controllers in real-world control applications.

10.1 Future Work

In the future, the proposed adaptive controllers can be applied to solve some new real-world problems in which they can obtain better results when compared to classical approaches. A possible class of applications is the one where the system dynamics varies drastically, e.g., cooperative aerial transportation with multirotor UAVs. In the cooperative aerial transportation, several issues, such as reciprocal interaction caused by multiple UAVs, should be considered. Besides, the physical properties of the carried item may be either unknown, e.g., mass or the moments of inertia, or variable, e.g., the COM of an item containing a fluid. Also, during the transportation, aerial robots might be affected by unknown external disturbances, e.g., wind gust which is exceptionally strong and unpredictable on high altitudes.

At the same time, autonomous drone racing is an exciting case study that aims to develop innovative ways of solving complex problems. In autonomous drone racing, the goal is to pass with UAV through a sequence of gates in a minimum amount of time while avoiding collisions in an unknown environment by relying only on onboard sensors and onboard computation. Thus, what makes drone racing such an interesting challenge is the cumulative complexity of each sub-problem to be solved, such as perception, localisation, path planning and, of course, control.

On the other hand, an adaptive controller can be designed to consider the characteristics of the controlled system and working conditions by adjusting the aggressiveness parameter of fuzzy mapping. For example, based on the uncertainty and noise levels, FLC can be configured to operate in a smoother or more aggressive mode.

In addition, it has been demonstrated that generalised T2-FLCs have enhanced capabilities of noise rejection. However, they are computationally complex and expensive for real-time applications. One of the possible future directions can focus on the management of this complexity in order to decrease it.

In the end, deep reinforcement learning is becoming one of the most popular AI-based techniques since it resembles the human way of learning. A mobile robot, e.g., multirotor UAV, can be considered as an agent which performs actions and receives rewards. This paradigm can be employed in many possible control applications, e.g., learning control in an unknown environment.

Appendix A

Attitude of Rigid Body

The rotation of a rigid body in space can be parametrized by using three Euler angles: roll (ϕ), pitch (θ) and yaw (ψ). By considering right-hand oriented coordinate frame, the three single rotations are described by:

- $\mathbf{R}(x, \phi)$ is the rotation around x-axis by ϕ;
- $\mathbf{R}(y, \theta)$ is the rotation around y-axis by θ;
- $\mathbf{R}(z, \psi)$ is the rotation around z-axis by ψ.

They are represented by:

$$\mathbf{R}(x, \phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \tag{A.1}$$

$$\mathbf{R}(y, \theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{A.2}$$

and

$$\mathbf{R}(z, \psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.3}$$

The complete rotation matrix is the product of three successive rotations around the world fixed z, y and x axes defined by (A.1)–(A.3):

$$\begin{aligned}\mathbf{R}(\phi,\theta,\psi) &= \mathbf{R}(z,\psi)\mathbf{R}(y,\theta)\mathbf{R}(x,\phi)\\ &= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix}.\end{aligned} \tag{A.4}$$

A.1 Transformation of Angular Velocities

The idea is to consider small changes in each Euler angle, and determine the effects on the rotation vector. To get the angular rates in the proper frames, the x-axis must be rotated into the inertial frame, the y-axis must be rotated by $\mathbf{R}^T(x,\phi)$ into the first frame, and the z-axis must be rotated by $\mathbf{R}^T(x,\psi)\mathbf{R}^T(y,\theta)$ into the second frame. The relationship between the body-fixed angular velocity vector $\boldsymbol{\omega}_B$ and the rate of change of the Euler angles $\boldsymbol{\omega}$ can be determined by resolving the Euler rates into the body-fixed coordinate frame:

$$\begin{aligned}\boldsymbol{\omega}_B &= \begin{bmatrix}\dot\phi\\0\\0\end{bmatrix} + \mathbf{R}^T(x,\phi)\begin{bmatrix}0\\\dot\theta\\0\end{bmatrix} + \mathbf{R}^T(x,\psi)\mathbf{R}^T(y,\theta)\begin{bmatrix}0\\0\\\dot\phi\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & -\sin\theta\\ 0 & \cos\phi & \sin\phi\cos\theta\\ 0 & -\sin\phi & \cos\phi\cos\theta\end{bmatrix}\boldsymbol{\omega}.\end{aligned} \tag{A.5}$$

Taking the inverse of (A.5) gives

$$\boldsymbol{\omega} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}\boldsymbol{\omega}_B. \tag{A.6}$$

Appendix B

Hat and Vee Mapping

Let $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$ be a vector in $\mathbb{R}^3$. Then, the hat map $[\mathbf{v}]^\wedge$ can be defined as a mathematical operator $\mathbb{R}^3 \to \mathsf{SO}(3)$:

$$[\mathbf{v}]^\wedge = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{B.1}$$

The hat map is equivalent to the cross product skew-symmetric matrix $[\mathbf{v}]_\times$, i.e.:

$$[\mathbf{v}]_\times \equiv [\mathbf{v}]^\wedge. \tag{B.2}$$

Let $\mathbf{M} = \begin{bmatrix} 0 & -m_{21} & m_{13} \\ m_{21} & 0 & -m_{32} \\ -m_{13} & m_{32} & 0 \end{bmatrix}$ be a skew-symmetric matrix in $\mathsf{SO}(3)$. Then, the vee map $[\mathbf{M}]^\vee$ can be defined as a mathematical operator $\mathsf{SO}(3) \to \mathbb{R}^3$:

$$[\mathbf{M}]^\vee = \begin{bmatrix} m_{32} & m_{13} & m_{21} \end{bmatrix}^T. \tag{B.3}$$

The vee map is the inverse of the hat map, i.e., $\left[[\mathbf{v}]^\wedge\right]^\vee = \mathbf{v}$. On the other side, the hat map is the inverse of the vee map, i.e., $\left[[\mathbf{M}]^\vee\right]^\wedge = \mathbf{M}$.

List of Author's Publications

Journal Articles

[1] **A. Sarabakha**, N. Imanberdiyev, E. Kayacan, M. A. Khanesar, and H. Hagras, "Novel Levenberg–Marquardt Based Learning Algorithm for Unmanned Aerial Vehicles," *Information Sciences*, vol. 417, pp. 361–380, Nov. 2017. doi:10.1016/j.ins.2017.07.020

[2] C. Fu, **A. Sarabakha**, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi, "Input Uncertainty Sensitivity Enhanced Nonsingleton Fuzzy Logic Controllers for Long-Term Navigation of Quadrotor UAVs," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 725–734, Apr. 2018. doi:10.1109/TMECH.2018.2810947

[3] E. Kayacan, **A. Sarabakha**, S. Coupland, R. John, M. A. Khanesar, "Type-2 Fuzzy Elliptic Membership Functions for Modeling Uncertainty," *Engineering Applications of Artificial Intelligence*, vol. 70, pp. 170–183, Apr. 2018. doi:10.1016/j.engappai.2018.02.004

[4] **A. Sarabakha**, C. Fu, E. Kayacan, and T. Kumbasar, "Type-2 Fuzzy Logic Controllers Made Even Simpler: From Design to Deployment for UAVs," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 5069–5077, June 2018. doi:10.1109/TIE.2017.2767546

[5] S. Patel, **A. Sarabakha**, D. Kircali, and E. Kayacan, "An Intelligent Hybrid Artificial Neural Network-Based Approach for Control of Aerial Robots," *Journal of Intelligent & Robotic Systems*, pp. 1–12, May 2019. doi:10.1007/s10846-019-01031-z

[6] **A. Sarabakha**, C. Fu, and E. Kayacan, "Intuit Before Tuning: Type-1 and Type-2 Fuzzy Logic Controllers," *Applied Soft Computing*, vol. 81, pp. 105495, Aug. 2019. doi:10.1016/j.asoc.2019.105495

[7] **A. Sarabakha**, and E. Kayacan, "Online Deep Fuzzy Learning for Control of Nonlinear Systems Using Expert Knowledge," *IEEE Transactions on Fuzzy Systems*. doi:10.1109/TFUZZ.2019.2936787

Conference Proceedings

[1] C. Fu, **A. Sarabakha**, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi, "A Comparative Study on the Control of Quadcopter UAVs by Using Singleton and Non-Singleton Fuzzy Logic Controllers," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Vancouver, Canada, 2016, pp. 1023–1030. doi:10.1109/FUZZ-IEEE.2016.7737800

[2] **A. Sarabakha** and E. Kayacan, "Y6 Tricopter Autonomous Evacuation in an Indoor Environment Using Q-Learning Algorithm," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, USA, 2016, pp. 5992–5997. doi:10.1109/CDC.2016.7799189

[3] **A. Sarabakha**, C. Fu, and E. Kayacan, "Double-Input Interval Type-2 Fuzzy Logic Controllers: Analysis and Design," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Naples, Italy, 2017, pp. 1–6. doi:10.1109/FUZZ-IEEE.2017.8015485

[4] C. Fu, **A. Sarabakha**, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi, "Novel, Similarity-Based Non-Singleton Fuzzy Logic Control for Improved Uncertainty Handling in Quadrotor UAVs," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Naples, Italy, 2017, pp. 1–6. doi:10.1109/FUZZ-IEEE.2017.8015440

[5] **A. Sarabakha**, and E. Kayacan, "Online Deep Learning for Improved Trajectory Tracking of Unmanned Aerial Vehicles Using Expert Knowledge," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019, pp. 7727–7733. doi:10.1109/ICRA.2019.8794314

[6] S. Zhou, **A. Sarabakha**, E. Kayacan, M. K. Helwa, and A. P. Schoellig, "Knowledge Transfer Between Robots with Similar Dynamics for High-Accuracy Impromptu Trajectory Tracking," in *2019 European Control Conference (ECC)*, Naples, Italy, 2019, pp. 1–8. doi:10.23919/ECC.2019.8796140

[7] S. Patel, **A. Sarabakha**, D. Kircali, G. Loianno, and E. Kayacan, "Artificial Neural Network-Assisted Controller for Fast and Agile UAV Flight: Onboard Implementation and Experimental Results," in *2019 International Workshop on Research, Education and Development on Unmanned Aerial Systems (RED-UAS)*, Cranfield, UK, 2019, pp. 37–43. doi:10.1109/REDUAS47371.2019.8999677

Bibliography

[1] Chowdhary Vinayak G., Frazzoli E., How P. J., and Liu H. *Nonlinear flight control techniques for unmanned aerial vehicles*, pages 577–612. Springer Netherlands, January 2015. ISBN 2014944662. doi: 10.1007/978-90-481-9707-1_87. 3

[2] P. Cintula, C. G. Fermüller, and C. Noguera. Fuzzy Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2017 edition, 2017. 4, 33

[3] D. Wu. On the Fundamental Differences Between Interval Type-2 and Type-1 Fuzzy Logic Controllers. *IEEE Transactions on Fuzzy Systems*, 20(5):832–848, Oct 2012. ISSN 1063-6706. doi: 10.1109/TFUZZ.2012.2186818. 4, 8, 33

[4] A. K. Ravandi, E. Khanmirza, and K. Daneshjou. Hybrid force/position control of robotic arms manipulating in uncertain environments based on adaptive fuzzy sliding mode control. *Applied Soft Computing*, 70:864 – 874, 2018. ISSN 1568-4946. doi: 10.1016/j.asoc.2018.05.048. 4, 49

[5] E. Ontiveros, P. Melin, and O. Castillo. High order α-planes integration: A new approach to computational cost reduction of General Type-2 Fuzzy Systems. *Engineering Applications of Artificial Intelligence*, 74:186 – 197, 2018. ISSN 0952-1976. doi: 10.1016/j.engappai.2018.06.013. 4

[6] E. Kayacan, A. Sarabakha, S. Coupland, R. John, and M. A. Khanesar. Type-2 fuzzy elliptic membership functions for modeling uncertainty. *Engineering Applications of Artificial Intelligence*, 70:170 – 183, 2018. ISSN 0952-1976. doi: 10.1016/j.engappai.2018.02.004. 4, 49, 146, 150

[7] D. Wu and J. M. Mendel. Uncertainty measures for interval type-2 fuzzy sets. *Information Sciences*, 177(23):5378 – 5393, 2007. ISSN 0020-0255. doi: 10.1016/j.ins.2007.07.012. Including: Mathematics of Uncertainty. 4, 49

[8] D. K. Jana and R. Ghosh. Novel interval type-2 fuzzy logic controller for improving risk assessment model of cyber security. *Journal of Information Security and Applications*, 40:173 – 182, 2018. ISSN 2214-2126. doi: 10.1016/j.jisa.2018.04.002. 4, 49

[9] S. Pramanik, D. K. Jana, S. K. Mondal, and M. Maiti. A fixed-charge transportation problem in two-stage supply chain network in gaussian type-2 fuzzy environments. *Information Sciences*, 325:190 – 214, 2015. ISSN 0020-0255. doi: 10.1016/j.ins.2015.07.012. 4, 49

[10] O. Castillo and P. Melin. A review on interval type-2 fuzzy logic applications in intelligent control. *Information Sciences*, 279:615 – 631, 2014. ISSN 0020-0255. doi: 10.1016/j.ins.2014.04.015. 4, 49

[11] O. Linda and M. Manic. Uncertainty-Robust Design of Interval Type-2 Fuzzy Logic Controller for Delta Parallel Robot. *IEEE Transactions on Industrial Informatics*, 7(4):661–670, Nov 2011. ISSN 1551-3203. doi: 10.1109/TII.2011.2166786. 4, 49

[12] A. Sarabakha, C. Fu, E. Kayacan, and T. Kumbasar. Type-2 Fuzzy Logic Controllers Made Even Simpler: From Design to Deployment for UAVs. *IEEE Transactions on Industrial Electronics*, 65(6):5069–5077, June 2018. ISSN 0278-0046. doi: 10.1109/TIE.2017.2767546. 4, 59

[13] S. Jothilakshmi and V.N. Gudivada. Chapter 10 - Large Scale Data Enabled Evolution of Spoken Language Research and Applications. In V. N. Gudivada, V. V. Raghavan, V. Govindaraju, and C. R. Rao, editors, *Cognitive Computing: Theory and Applications*, volume 35 of *Handbook of Statistics*, pages 301 – 340. Elsevier, 2016. doi: 10.1016/bs.host.2016.07.005. 4, 83

[14] W. Mao and F.-Y. Wang. Chapter 8 - Cultural Modeling for Behavior Analysis and Prediction. In W. Mao and F.-Y. Wang, editors, *New Advances in Intelligence and Security Informatics*, pages 91 – 102. Academic Press, Boston, 2012. ISBN 978-0-12-397200-2. doi: 10.1016/B978-0-12-397200-2.00008-7. 4, 83

[15] U. Porwal, Z. Shi, and S. Setlur. Chapter 18 - Machine Learning in Handwritten Arabic Text Recognition. In C. R. Rao and V. Govindaraju, editors,

Handbook of Statistics, volume 31 of *Handbook of Statistics*, pages 443 – 469. Elsevier, 2013. doi: 10.1016/B978-0-444-53859-8.00018-7. 5, 83

[16] Y. Roh, G. Heo, and S. E. Whang. A Survey on Data Collection for Machine Learning: a Big Data – AI Integration Perspective, 2018. 5, 83

[17] E. Conrad, S. Misenar, and J. Feldman. Chapter 9 - Domain 8: Software Development Security (Understanding, Applying, and Enforcing Software Security). In E. Conrad, S. Misenar, and J. Feldman, editors, *CISSP Study Guide (Third Edition)*, pages 429 – 477. Syngress, Boston, third edition edition, 2016. ISBN 978-0-12-802437-9. doi: 10.1016/B978-0-12-802437-9.00009-6. 5, 83, 119

[18] E. Kayacan, E. Kayacan, H. Ramon, and W. Saeys. Adaptive Neuro-Fuzzy Control of a Spherical Rolling Robot Using Sliding-Mode-Control-Theory-Based Online Learning Algorithm. *IEEE Transactions on Cybernetics*, 43(1): 170–179, Feb 2013. ISSN 2168-2267. doi: 10.1109/TSMCB.2012.2202900. 5, 83

[19] F. Ornelas-Tellez, J. J. Rico-Melgoza, A. E. Villafuerte, and F. J. Zavala-Mendoza. Chapter 3 - Neural Networks: A Methodology for Modeling and Control Design of Dynamical Systems. In A. Y. Alanis, N. Arana-Daniel, and C. Lopez-Franco, editors, *Artificial Neural Networks for Engineering Applications*, pages 21 – 38. Academic Press, 2019. ISBN 978-0-12-818247-5. doi: 10.1016/B978-0-12-818247-5.00012-5. 5

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, May 2015. 5, 101

[21] X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, and A. Ozcan. All-optical machine learning using diffractive deep neural networks. *Science*, 361(6406):1004–1008, 2018. ISSN 0036-8075. doi: 10.1126/science.aat8084. 5

[22] M. Bianchini and F. Scarselli. On the Complexity of Neural Network Classifiers: A Comparison Between Shallow and Deep Architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, Aug 2014. doi: 10.1109/TNNLS.2013.2293637. 5, 101

[23] S. Zhou, M. K. Helwa, and A. P. Schoellig. An Inversion-Based Learning Approach for Improving Impromptu Trajectory Tracking of Robots With Non-Minimum Phase Dynamics. *IEEE Robotics and Automation Letters*, 3 (3):1663–1670, July 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2801471. 5, 101

[24] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza. DroNet: Learning to Fly by Driving. *IEEE Robotics and Automation Letters*, 3(2): 1088–1095, April 2018. 5, 101

[25] A. Celikyilmaz and I. B. Turksen. *Modeling Uncertainty with Fuzzy Logic: With Recent Theory and Applications*. Springer-Verlag Berlin Heidelberg, 1st edition, 2009. ISBN 3540899235, 9783540899235. doi: 10.1007/978-3-540-89924-2. 5, 6, 119

[26] F. Behrooz, N. Mariun, M. Marhaban, M. A. Mohd Radzi, and A. Ramli. Review of Control Techniques for HVAC Systems—Nonlinearity Approaches Based on Fuzzy Cognitive Maps. *Energies*, 11:495, 02 2018. doi: 10.3390/en11030495. 5, 119, 135

[27] T. Roxlo and M. Reece. Opening the black box of neural nets: case studies in stop/top discrimination, 2018. 5, 119

[28] F. Gaxiola, P. Melin, F. Valdez, and O. Castillo. Generalized type-2 fuzzy weight adjustment for backpropagation neural networks in time series prediction. *Information Sciences*, 325:159 – 174, 2015. ISSN 0020-0255. doi: 10.1016/j.ins.2015.07.020. 5, 119

[29] N. Wang, M. J. Er, and M. Han. Dynamic Tanker Steering Control Using Generalized Ellipsoidal-Basis-Function-Based Fuzzy Neural Networks. *IEEE Transactions on Fuzzy Systems*, 23(5):1414–1427, Oct 2015. doi: 10.1109/TFUZZ.2014.2362144. 5, 119

[30] D. Jirak and S. Wermter. Potentials and Limitations of Deep Neural Networks for Cognitive Robots, 2018. 5, 135

[31] T. Zhou, F. Chung, and S. Wang. Deep TSK Fuzzy Classifier With Stacked Generalization and Triplely Concise Interpretability Guarantee for Large Data. *IEEE Transactions on Fuzzy Systems*, 25(5):1207–1221, Oct 2017. ISSN 1063-6706. 5, 135

[32] A. Sarabakha and E. Kayacan. Y6 Tricopter Autonomous Evacuation in an Indoor Environment Using Q-Learning Algorithm. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5992–5997, Dec 2016. doi: 10.1109/CDC.2016.7799189. 6, 28

[33] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi. Efficient Road Detection and Tracking for Unmanned Aerial Vehicle. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):297–309, 2015. 6

[34] C. Fu, A. Carrio, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy. Robust Real-Time Vision-Based Aircraft Tracking from Unmanned Aerial Vehicles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5441–5446, May 2014. doi: 10.1109/ICRA.2014.6907659. 6

[35] J. Valente, D. Sanz, A. Barrientos, J. Del Cerro, A. Ribeiro, and C. Rossi. An Air-Ground Wireless Sensor Network for Crop Monitoring. *Sensors*, 11 (6):6088–6108, 2011. 6

[36] C. Fu, A. Carrio, and P. Campoy. Efficient visual odometry and mapping for Unmanned Aerial Vehicle using ARM-based stereo vision pre-processing system. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 957–962, June 2015. doi: 10.1109/ICUAS.2015.7152384. 6

[37] H. Alzu'bi, B. H. Sababha, and B. Alkhatib. *Model-Based Control of a Fully Autonomous Quadrotor UAV*. 08 2013. doi: 10.2514/6.2013-5136. 6

[38] Y. A. Younes, A. Drak, H. Noura, A. Rabhi, and A. E. Hajjaji. Robust Model-Free Control Applied to a Quadrotor UAV. *Journal of Intelligent & Robotic Systems*, 84(1):37–52, Dec 2016. ISSN 1573-0409. doi: 10.1007/s10846-016-0351-2. 6

[39] A. Eresen, N. Imamoglu, and M. O. Efe. Autonomous Quadrotor Flight with Vision-Based Obstacle Avoidance in Virtual Environment. *Expert Systems with Applications*, 39(1):894 – 905, 2012. ISSN 0957-4174. 6

[40] M. D. Hua, T. Hamel, P. Morin, and C. Samson. Introduction to feedback control of underactuated VTOLvehicles: A review of basic control design ideas and principles. *IEEE Control Systems*, 33(1):61–75, Feb 2013. ISSN 1066-033X. doi: 10.1109/MCS.2012.2225931. 6

[41] Y. Dong, F. Jun, Y. Bin, Z. Youmin, and A. Jianliang. Position and Heading Angle Control of an Unmanned Quadrotor Helicopter Using LQR Method. In *Control Conference (CCC), 2015 34th Chinese*, pages 5566–5571, July 2015. doi: 10.1109/ChiCC.2015.7260508. 6

[42] M. Hofer, M. Muehlebach, and R. D'Andrea. Application of an approximate model predictive control scheme on an unmanned aerial vehicle. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2952–2957, 2016. 6

[43] E. Kayacan and R. Maslim. Type-2 Fuzzy Logic Trajectory Tracking Control of Quadrotor VTOL Aircraft With Elliptic Membership Functions. *IEEE/ASME Transactions on Mechatronics*, (99):1, 2016. 6

[44] M. Mehndiratta, E. Kayacan, and T. Kumbasar. Design and experimental validation of single input type-2 fuzzy PID controllers as applied to 3 DOF helicopter testbed. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1584–1591, 2016.

[45] C. J. Kim and D. Chwa. Obstacle Avoidance Method for Wheeled Mobile Robots Using Interval Type-2 Fuzzy Neural Network. *IEEE Transactions on Fuzzy Systems*, 23(3):677–687, June 2015. ISSN 1063-6706. doi: 10.1109/TFUZZ.2014.2321771. 6

[46] F. Cuevas, O. Castillo, and P. Cortes-Antonio. Towards an Adaptive Control Strategy Based on Type-2 Fuzzy Logic for Autonomous Mobile Robots. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, June 2019. doi: 10.1109/FUZZ-IEEE.2019.8858801. 6

[47] H. Hagras. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12(4):524–539, Aug 2004. 6

[48] F. Fakurian, M. B. Menhaj, and A. Mohammadi. Design of a fuzzy controller by minimum controlling inputs for a quadrotor. In *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 619–624, 2014. 6

[49] G. K. I. Mann, Bao-Gang Hu, and R. G. Gosine. Analysis of direct action fuzzy PID controller structures. *IEEE Transactions on Systems, Man, and*

Cybernetics, Part B (Cybernetics), 29(3):371–388, June 1999. ISSN 1083-4419. doi: 10.1109/3477.764871. 6

[50] H. X. Li and H. B. Gatland. Conventional fuzzy control and its enhancement. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(5):791–797, Oct 1996. ISSN 1083-4419. doi: 10.1109/3477.537321. 6

[51] H. X. Li, H. B. Gatland, and A. W. Green. Fuzzy variable structure control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):306–312, April 1997. ISSN 1083-4419. doi: 10.1109/3477.558824. 6

[52] J. Carvajal, G. Chen, and H. Ogmen. Fuzzy PID controller: Design, performance evaluation, and stability analysis. *Information Sciences*, 123(3):249 – 270, 2000. ISSN 0020-0255. doi: 10.1016/S0020-0255(99)00127-9. 6

[53] B.-G. Hu, G. K. I. Mann, and R. G. Gosine. A systematic study of fuzzy PID controllers-function-based evaluation approach. *IEEE Transactions on Fuzzy Systems*, 9(5):699–712, Oct 2001. ISSN 1063-6706. doi: 10.1109/91.963756. 7

[54] H. Ying. Deriving Analytical Input–Output Relationship for Fuzzy Controllers Using Arbitrary Input Fuzzy Sets and Zadeh Fuzzy AND Operator. *IEEE Transactions on Fuzzy Systems*, 14(5):654–662, Oct 2006. ISSN 1063-6706. doi: 10.1109/TFUZZ.2006.877355. 7

[55] B. M. Mohan and A. Sinha. Analytical structure and stability analysis of a fuzzy PID controller. *Applied Soft Computing*, 8(1):749 – 758, 2008. ISSN 1568-4946. doi: 10.1016/j.asoc.2007.06.003. 7

[56] P. S. Londhe, B. M. Patre, and A. P. Tiwari. Design of Single-Input Fuzzy Logic Controller for Spatial Control of Advanced Heavy Water Reactor. *IEEE Transactions on Nuclear Science*, 61(2):901–911, April 2014. 7

[57] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy. Monocular Visual-Inertial SLAM-Based Collision Avoidance Strategy for Fail-Safe UAV Using Fuzzy Logic Controllers. *Journal of Intelligent & Robotic Systems*, 73(1-4):513–533, 2014.

[58] M. A. Olivares-Mendez, C. Fu, S. Kannan, H. Voos, and P. Campoy. Using the Cross-Entropy method for control optimization: A case study of see-and-avoid

on unmanned aerial vehicles. In *22nd Mediterranean Conference on Control and Automation*, pages 1183–1189, 2014. 7

[59] A.B. Cara, I. Rojas, H. Pomares, C. Wagner, and H. Hagras. On comparing non-singleton type-1 and singleton type-2 fuzzy controllers for a nonlinear servo system. In *Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), 2011 IEEE Symposium on*, pages 126–133, 2011. 7

[60] J. Bharali and M. Buragohain. A comparative analysis of PID, LQR and Fuzzy logic controller for active suspension system using 3 Degree of Freedom quarter car model. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pages 1–5, July 2016. doi: 10.1109/ICPEICES.2016.7853152. 7

[61] C. Fu, A. Sarabakha, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi. A comparative study on the control of quadcopter UAVs by using singleton and non-singleton fuzzy logic controllers. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1023–1030, July 2016. 7

[62] A. Pourabdollah, C. Wagner, J. H. Aladi, and J. M. Garibaldi. Improved Uncertainty Capture for Nonsingleton Fuzzy Systems. *IEEE Transactions on Fuzzy Systems*, 24(6):1513–1524, Dec 2016. ISSN 1063-6706. doi: 10.1109/TFUZZ.2016.2540065. 7, 39

[63] C. Wagner, A. Pourabdollah, J. McCulloch, R. John, and J. M. Garibaldi. A similarity-based inference engine for non-singleton fuzzy logic systems. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 316–323, July 2016. 39, 41

[64] A. Pourabdollah, C. Wagner, and J. Aladi. Changes under the hood - a new type of non-singleton fuzzy logic system. In *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*, pages 1–8, 2015. 7

[65] P. Melin and O. Castillo. A review on type-2 fuzzy logic applications in clustering, classification and pattern recognition. *Applied Soft Computing*, 21: 568 – 577, 2014. ISSN 1568-4946. doi: 10.1016/j.asoc.2014.04.017. 7, 49

[66] S. Greenfield, F. Chiclana, R. John, and S. Coupland. The sampling method of defuzzification for type-2 fuzzy sets: Experimental evaluation. *Information Sciences*, 189:77 – 92, 2012. ISSN 0020-0255. doi: 10.1016/j.ins.2011.11.042.

[67] O. Castillo and P. Melin. *3 Type-2 Fuzzy Logic*, pages 29–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 7

[68] O. Castillo, L. Amador-Angulo, J. R. Castro, and M. Garcia-Valdez. A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Information Sciences*, 354:257 – 274, 2016. ISSN 0020-0255. doi: 10.1016/j.ins.2016.03.026. 7

[69] J. M. Mendel and R. I. B. John. Type-2 fuzzy sets made simple. *IEEE Transactions on Fuzzy Systems*, 10(2):117–127, Apr 2002. ISSN 1063-6706. doi: 10.1109/91.995115. 7

[70] A. K. Das, S. Sundaram, and N. Sundararajan. A self-regulated interval type-2 neuro-fuzzy inference system for handling non-stationarities in EEG signals for BCI. *IEEE Transactions on Fuzzy Systems*, PP(99):1–1, 2016. ISSN 1063-6706. doi: 10.1109/TFUZZ.2016.2540072. 7

[71] F. Baghbani, M.-R. Akbarzadeh-T., Alireza Akbarzadeh, and M. Ghaemi. Robust adaptive mixed H_2/H_∞ interval type-2 fuzzy control of nonlinear uncertain systems with minimal control effort. *Engineering Applications of Artificial Intelligence*, 49:88–102, 2016.

[72] Saugat Bhattacharyya, Debabrota Basu, Amit Konar, and D.N. Tibarewala. Interval type-2 fuzzy logic based multiclass ANFIS algorithm for real-time EEG based movement control of a robot arm. *Robotics and Autonomous Systems*, 68:104 – 115, 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2015.01.007.

[73] A. Mohammadzadeh, O. Kaynak, and M. Teshnehlab. Two-mode Indirect Adaptive Control Approach for the Synchronization of Uncertain Chaotic Systems by the Use of a Hierarchical Interval Type-2 Fuzzy Neural Network. *IEEE Transactions on Fuzzy Systems*, 22(5):1301–1312, Oct 2014. ISSN 1063-6706. doi: 10.1109/TFUZZ.2013.2291568. 7

[74] J. M. Mendel and X. Liu. Simplified Interval Type-2 Fuzzy Logic Systems. *IEEE Transactions on Fuzzy Systems*, 21(6):1056–1069, Dec 2013. ISSN 1063-6706. doi: 10.1109/TFUZZ.2013.2241771. 7, 49

[75] Tufan Kumbasar. A simple design method for interval type-2 fuzzy PID controllers. *Soft Computing*, 18(7):1293–1304, 2014. ISSN 1433-7479. doi: 10.1007/s00500-013-1144-1. 7

[76] D. Wu and J. M. Mendel. On the Continuity of Type-1 and Interval Type-2 Fuzzy Logic Systems. *IEEE Transactions on Fuzzy Systems*, 19(1):179–192, Feb 2011. ISSN 1063-6706. doi: 10.1109/TFUZZ.2010.2091962. 7, 8

[77] M. Nie and W. W. Tan. Analytical Structure and Characteristics of Symmetric Karnik–Mendel Type-Reduced Interval Type-2 Fuzzy PI and PD Controllers. *IEEE Transactions on Fuzzy Systems*, 20(3):416–430, June 2012. ISSN 1063-6706. doi: 10.1109/TFUZZ.2011.2174061. 7

[78] X. Du and H. Ying. Derivation and Analysis of the Analytical Structures of the Interval Type-2 Fuzzy-PI and PD Controllers. *IEEE Transactions on Fuzzy Systems*, 18(4):802–814, Aug 2010. ISSN 1063-6706. doi: 10.1109/TFUZZ.2010.2049022. 7, 8

[79] H. Zhou and H. Ying. A Method for Deriving the Analytical Structure of a Broad Class of Typical Interval Type-2 Mamdani Fuzzy Controllers. *IEEE Transactions on Fuzzy Systems*, 21(3):447–458, June 2013. ISSN 1063-6706. doi: 10.1109/TFUZZ.2012.2226891. 8

[80] M. F. Dodurka, T. Kumbasar, A. Sakalli, and E. Yesil. Boundary function based Karnik-Mendel type reduction method for Interval Type-2 Fuzzy PID controllers. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 619–625, July 2014. doi: 10.1109/FUZZ-IEEE.2014.6891832. 8

[81] A. Sakalli, T. Kumbasar, M. F. Dodurka, and E. Yesil. The simplest interval type-2 fuzzy PID controller: Structural analysis. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 626–633, July 2014. doi: 10.1109/FUZZ-IEEE.2014.6891830. 8

[82] H. Zhou and H. Ying. Deriving and Analyzing Analytical Structures of a Class of Typical Interval Type-2 TS Fuzzy Controllers. *IEEE Transactions on Cybernetics*, 47(9):2492–2503, Sept 2017. ISSN 2168-2267. doi: 10.1109/TCYB.2016.2570239. 8

[83] C. M. T. Yip, W. W. Tan, and M. Nie. On the difference in control performance of interval type-2 fuzzy PI control system with different FOU shapes. *Applied Soft Computing*, 76:517 – 532, 2019. ISSN 1568-4946. doi: 10.1016/j.asoc.2018.12.039. 8

[84] A. Sarabakha, C. Fu, and E. Kayacan. Double-Input Interval Type-2 Fuzzy Logic Controllers: Analysis and Design. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, July 2017. doi: 10.1109/FUZZ-IEEE.2017.8015485. 8, 59

[85] B. J. Emran and H. Najjaran. Adaptive neural network control of quadrotor system under the presence of actuator constraints. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2619–2624, Oct 2017. 8

[86] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin. Learning quadrotor dynamics using neural network for flight control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4653–4660, Dec 2016. 8

[87] S. A. Nivison and P. P. Khargonekar. Development of a robust deep recurrent neural network controller for flight applications. In *2017 American Control Conference (ACC)*, pages 5336–5342, May 2017. 8

[88] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230, May 2015. 8

[89] N. Mohajerin and S. L. Waslander. Modular Deep Recurrent Neural Network: Application to Quadrotors. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1374–1379, Oct 2014. 8

[90] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig. Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5183–5189, May 2017. 8

[91] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and M. J. Er. An Incremental Learning of Concept Drifts Using Evolving Type-2 Recurrent Fuzzy Neural

Networks. *IEEE Transactions on Fuzzy Systems*, 25(5):1175–1192, Oct 2017. ISSN 1063-6706. 8, 135

[92] S. Zhou, Q. Chen, and X. Wang. Fuzzy deep belief networks for semi-supervised sentiment classification. *Neurocomputing*, 131:312–322, 2014. ISSN 0925-2312. 8

[93] A. Sarabakha, N. Imanberdiyev, E. Kayacan, M. A. Khanesar, and H. Hagras. Novel Levenberg–Marquardt Based Learning Algorithm for Unmanned Aerial Vehicles. *Information Sciences*, 417:361 – 380, 2017. ISSN 0020-0255. doi: 10.1016/j.ins.2017.07.020. 8, 86

[94] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai. A Hierarchical Fused Fuzzy Deep Neural Network for Data Classification. *IEEE Transactions on Fuzzy Systems*, 25(4):1006–1012, Aug 2017. ISSN 1063-6706. 8, 135

[95] J. Fernández de Cañete, C. Galindo, and I. G. Moral. *Introduction to Systems*, pages 1–11. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-20230-8. doi: 10.1007/978-3-642-20230-8_1. 13

[96] J. Fernández de Cañete, C. Galindo, and I. G. Moral. *System Description*, pages 43–83. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-20230-8. doi: 10.1007/978-3-642-20230-8_3. 13

[97] G. Boeing. Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction. *Systems*, 4(4), 2016. ISSN 2079-8954. doi: 10.3390/systems4040037. 13

[98] A. Isidori. *Elementary Theory of Nonlinear Feedback for Multi-Input Multi-Output Systems*, pages 219–291. Springer London, London, 1995. 14

[99] A. Isidori. The zero dynamics of a nonlinear system: From the origin to the latest progresses of a long successful story. *European Journal of Control*, 19 (5):369–378, 2013. ISSN 0947-3580. The Path of Control. 14

[100] D. Liberzon, A. S. Morse, and E. D. Sontag. Output-input stability and minimum-phase nonlinear systems. *IEEE Transactions on Automatic Control*, 47(3):422–436, March 2002. ISSN 0018-9286. doi: 10.1109/9.989070. 14

[101] J. P. Hespanha, D. Liberzon, D. Angeli, and E. D. Sontag. Nonlinear norm-observability notions and stability of switched systems. *IEEE Transactions on Automatic Control*, 50(2):154–168, Feb 2005. ISSN 0018-9286. 14

[102] S. Zhou, M. K. Helwa, and A. P. Schoellig. Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5201–5207, Dec 2017. doi: 10.1109/CDC.2017.8264430. 15, 103, 105

[103] S. Zhou, A. Sarabakha, E. Kayacan, M. K. Helwa, and A. P. Schoellig. Knowledge Transfer Between Robots with Similar Dynamics for High-Accuracy Impromptu Trajectory Tracking. In *2019 European Control Conference (ECC)*, pages 1–8, June 2019. 15

[104] A. Kehlenbeck. *Quaternion-based control for aggressive trajectory tracking with a micro-quadrotor UAV*. PhD thesis, University of Maryland, College Park, 2014. 17

[105] G. Loianno, V. Spurny, J. Thomas, T. Baca, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar. Localization, Grasping, and Transportation of Magnetic Objects by a Team of MAVs in Challenging Desert-Like Environments. *IEEE Robotics and Automation Letters*, 3(3): 1576–1583, July 2018. 17

[106] S. Siebert and J. Teizer. Mobile 3D mapping for surveying earthwork projects using an Unmanned Aerial Vehicle (UAV) system. *Automation in Construction*, 41:1 – 14, 2014. ISSN 0926-5805. 17

[107] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue. *IEEE Robotics Automation Magazine*, 19(3):46–56, Sept 2012. ISSN 1070-9932. 17

[108] S. Bouabdallah. *Design and control of quadrotors with application to autonomous flying*. PhD thesis, EPFL, Dec 2007. 18

[109] V. Mistler, A. Benallegue, and N. K. M'Sirdi. Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *Proceedings 10th IEEE International Workshop on Robot and Human Interactive Communication. ROMAN 2001 (Cat. No.01TH8591)*, pages 586–593, 2001. 19

[110] T. Lee, M. Leok, and N. H. McClamroch. Nonlinear Robust Tracking Control of a Quadrotor UAV on SE(3). *Asian Journal of Control*, 15(2):391–408, 2013. ISSN 1934-6093. doi: 10.1002/asjc.567. 21

[111] R. Mahony, V. Kumar, and P. Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32, Sept 2012. ISSN 1070-9932. doi: 10.1109/MRA.2012.2206474. 22

[112] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Dec 2010. doi: 10.1109/CDC.2010.5717652. 29

[113] A. Sarabakha, C. Fu, and E. Kayacan. Intuit before tuning: Type-1 and type-2 fuzzy logic controllers. *Applied Soft Computing*, 81:105495, 2019. ISSN 1568-4946. doi: 10.1016/j.asoc.2019.105495. 36

[114] C. Fu, A. Sarabakha, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi. Similarity-Based Non-Singleton Fuzzy Logic Control for Improved Performance in UAVs. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, July 2017. doi: 10.1109/FUZZ-IEEE.2017.8015440. 36

[115] J.M. Mendel. *Uncertain rule-based fuzzy logic system: introduction and new directions*. Upper Saddle River, NJ, USA, Prentice-Hall, 2001. 39, 60

[116] D. Mellinger and V. Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525, May 2011. doi: 10.1109/ICRA.2011.5980409. 42

[117] C. Fu, A. Sarabakha, E. Kayacan, C. Wagner, R. John, and J. M. Garibaldi. Input Uncertainty Sensitivity Enhanced Nonsingleton Fuzzy Logic Controllers for Long-Term Navigation of Quadrotor UAVs. *IEEE/ASME Transactions on Mechatronics*, 23(2):725–734, April 2018. ISSN 1083-4435. doi: 10.1109/TMECH.2018.2810947. 45, 150

[118] E. Ontiveros-Robles, P. Melin, and O. Castillo. Comparative analysis of noise robustness of type 2 fuzzy logic controllers. *Kybernetika*, 54(1):175–201, 2018. doi: 10.14736/kyb-2018-1-0175. 49

[119] M. A. Khanesar, E. Kayacan, M. Teshnehlab, and O. Kaynak. Analysis of the Noise Reduction Property of Type-2 Fuzzy Logic Systems Using a Novel Type-2 Membership Function. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(5):1395–1406, Oct 2011. ISSN 1083-4419. doi: 10.1109/TSMCB.2011.2148173. 49

[120] T. Kumbasar. Robust Stability Analysis and Systematic Design of Single-Input Interval Type-2 Fuzzy Logic Controllers. *IEEE Transactions on Fuzzy Systems*, 24(3):675–694, June 2016. ISSN 1063-6706. doi: 10.1109/TFUZZ.2015.2471805. 51, 59

[121] G. Ruiz, H. Hagras, H. Pomares, I. Rojas, and H. Bustince. Join and Meet Operations for Type-2 Fuzzy Sets With Nonconvex Secondary Memberships. *IEEE Transactions on Fuzzy Systems*, 24(4):1000–1008, Aug 2016. doi: 10.1109/TFUZZ.2015.2489242. 52

[122] J. Mendel, H. Hagras, W.-W. Tan, W. W. Melek, and H. Ying. *Introduction to Type-2 Fuzzy Logic Control: Theory and Applications*. Wiley-IEEE Press, 1st edition, 2014. ISBN 1118278399, 9781118278390. doi: 10.1002/9781118886540. 52

[123] N. N. Karnik and J. M. Mendel. Centroid of a type-2 fuzzy set. *Information Sciences*, 132(1):195 – 220, 2001. ISSN 0020-0255. doi: 10.1016/S0020-0255(01)00069-X. 52

[124] O. Castillo. *Design of Stable Type-2 Fuzzy Logic Controllers*, chapter 4, pages 49–61. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-24663-0. doi: 10.1007/978-3-642-24663-0_4. 59

[125] W. Pedrycz and F. Gomide. *Notions and Concepts of Fuzzy Sets*, chapter 2, pages 27–44. Wiley-Blackwell, 2007. ISBN 9780470168967. doi: 10.1002/9780470168967.ch2. 61

[126] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. 84

[127] E. Johnson and S. Kannan. Adaptive flight control for an autonomous unmanned helicopter. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4439, 2002. 84

[128] G. Buskey, G. Wyeth, and J. Roberts. Autonomous helicopter hover using an artificial neural network. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1635–1640 vol.2, 2001. 84

[129] Y. Yildiz, A. Sabanovic, and K. Abidi. Sliding-Mode Neuro-Controller for Uncertain Systems. *IEEE Transactions on Industrial Electronics*, 54(3): 1676–1685, June 2007. ISSN 0278-0046. 85

[130] S. Patel, A. Sarabakha, D. Kircali, G. Loianno, and E. Kayacan. Artificial Neural Network-Assisted Controller for Fast and Agile UAV Flight: Onboard Implementation and Experimental Results. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 37–43, 2019. doi: 10.1109/REDUAS47371.2019.8999677. 86

[131] S. Patel, A. Sarabakha, D. Kircali, and E. Kayacan. An Intelligent Hybrid Artificial Neural Network-Based Approach for Control of Aerial Robots. *Journal of Intelligent & Robotic Systems*, May 2019. ISSN 1573-0409. doi: 10.1007/s10846-019-01031-z. 86

[132] E. Kayacan and M. A. Khanesar. Chapter 7 - Sliding Mode Control Theory-Based Parameter Adaptation Rules for Fuzzy Neural Networks. In E. Kayacan and M. A. Khanesar, editors, *Fuzzy Neural Networks for Real Time Control Applications*, pages 85 – 131. Butterworth-Heinemann, 2016. ISBN 978-0-12-802687-8. 87

[133] N. Imanberdiyev and E. Kayacan. A fast learning control strategy for unmanned aerial manipulators. *Journal of Intelligent & Robotic Systems*, Jun 2018. ISSN 1573-0409. 87

[134] M. Önder Efe. *Sliding Mode Control for Unmanned Aerial Vehicles Research*, pages 239–255. Springer International Publishing, Cham, 2015. ISBN 978-3-319-18290-2. 87

[135] E. Kayacan and O. Kaynak. Sliding mode control theory-based algorithm for online learning in type-2 fuzzy neural networks: application to velocity control of an electro hydraulic servo system. *International Journal of Adaptive Control and Signal Processing*, 26(7):645–659, 2012. doi: 10.1002/acs.1292. 87

[136] L. Meier, D. Honegger, and M. Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, May 2015. doi: 10.1109/ICRA.2015.7140074. 91

[137] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1):21–39, Aug 2012. ISSN 1573-7527. 91

[138] B. Wilburn, M. Perhinschi, H. Moncayo, O. Karas, and J. Wilburn. Unmanned aerial vehicle trajectory tracking algorithm comparison. *International Journal of Intelligent Unmanned Systems*, 1:276–302, 07 2013. doi: 10.1108/IJIUS-05-2013-0018. 94

[139] M. Mehndiratta and E. Kayacan. Reconfigurable Fault-tolerant NMPC for Y6 Coaxial Tricopter with Complete Loss of One Rotor. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 774–780, 2018. 96

[140] B. K. Wilburn, M. G. Perhinschi, H. Moncayo, O. Karas, and J. N. Wilburn. Unmanned aerial vehicle trajectory tracking algorithm comparison. *International Journal of Intelligent Unmanned Systems*, 1(3):276–302, 2013. 97

[141] J. Schoukens and L. Ljung. Nonlinear System Identification: A User-Oriented Roadmap, 2019. 101

[142] M. E. Taylor and P. Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009. ISSN 1532-4435. 103

[143] M. Whorton, L. Yang, and R. Hall. *Similarity Metrics for Closed Loop Dynamic Systems*. doi: 10.2514/6.2008-6624. 106

[144] C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2006. 110

[145] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig. Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 5183–5189, 2017. 112

[146] O. Castillo, L. Amador-Angulo, J. R. Castro, and M. Garcia-Valdez. A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Information Sciences*, 354:257 – 274, 2016. ISSN 0020-0255. doi: 10.1016/j.ins.2016.03.026. 119

[147] X. Gao and L. Liao. A New One-Layer Neural Network for Linear and Quadratic Programming. *IEEE Transactions on Neural Networks*, 21(6): 918–929, June 2010. doi: 10.1109/TNN.2010.2045129. 119

[148] Y. Jin. *Advanced Fuzzy Systems Design and Applications*. Physica-Verlag, 1st edition, 2012. ISBN 3790825204, 9783790825206. 120

[149] V. I. Utkin. *Sliding modes in control optimization*. Springer-Verlag, 1992. 122

[150] A. Sarabakha and E. Kayacan. Online Deep Learning for Improved Trajectory Tracking of Unmanned Aerial Vehicles Using Expert Knowledge. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, May 2019. 135

[151] A. Sarabakha and E. Kayacan. Online Deep Fuzzy Learning for Control of Nonlinear Systems Using Expert Knowledge. *IEEE Transactions on Fuzzy Systems*, pages 1–1, 2019. ISSN 1941-0034. doi: 10.1109/TFUZZ.2019.2936787. 137

[152] J. M. Mendel. *Type-1 Fuzzy Systems*, chapter 3, pages 101–159. Springer International Publishing, Cham, 2017. ISBN 978-3-319-51370-6. doi: 10.1007/978-3-319-51370-6_3. 138

[153] F. J. Doyle and M. A. Henson. Nonlinear Systems Theory. In M. A. Henson and D. E. Seborg, editors, *Nonlinear Process Control*, pages 111–147. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. ISBN 0-13-625179-X. 141

[154] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. 144

[155] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning Polynomials with Neural Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1908–II–1916. JMLR.org, 2014. 144