ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DIFFERENTIAL FLATNESS-BASED FUZZY CONTROLLER DESIGN FOR AGGRESSIVE MANEUVERING OF QUADCOPTERS

Ph.D. THESIS

Çağrı GÜZAY

Department of Control and Automation Engineering

Control and Automation Engineering Programme

MAY 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DIFFERENTIAL FLATNESS-BASED FUZZY CONTROLLER DESIGN FOR AGGRESSIVE MANEUVERING OF QUADCOPTERS

Ph.D. THESIS

Çağrı GÜZAY
(504142105)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Tufan KUMBASAR

MAY 2023

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÇOK ROTORLU HAVA ARAÇLARININ AGRESİF MANEVRA KONTROLÜ İÇİN DİFERANSİYEL DÜZLÜK TABANLI BULANIK KONTROLÖR TASARIMI

DOKTORA TEZİ

Çağrı GÜZAY
(504142105)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Tufan KUMBASAR

MAYIS 2023

Çağrı GÜZAY, a Ph.D. student of ITU Graduate School student ID 504142105 successfully defended the thesis entitled "DIFFERENTIAL FLATNESS-BASED FUZZY CONTROLLER DESIGN FOR AGGRESSIVE MANEUVERING OF QUADCOPTERS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Tufan KUMBASAR** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Müjde GÜZELKAYA** Istanbul Technical University	.....
	Assoc. Prof. Dr. Volkan SEZER Istanbul Technical University	.....
	Assoc. Prof. Dr. Yavuz EREN Yıldız Technical University	.....
	Assoc. Prof. Dr. Farzad HASHEMZADEH University of Tabriz	.....

Date of Submission : **10 February 2023**
Date of Defense : **4 May 2023**

To the uneven adventure called as existence,

FOREWORD

I would like to extend a heartfelt thank who stands by me for their unwavering support and guidance throughout this overlong story. This study would not have been possible without their encouragement and belief in me.

I would like to say "Thank you" to my advisor Assoc. Prof. Dr. Tufan KUMBASAR who is the most super successful person I have ever met. It would be impossible to include all of his contributions to this study in a two-word thank you, so this is more than that. I feel and will feel grateful for his guidance and mentorship. His wisdom and expertise have been instrumental in shaping this study and helping me to achieve my goals.

This study would not have been assembled without the encouragement and willingness of Res. Asst. Yusuf ŞAHİNKAYA to lend an ear which has been invaluable to me. He is currently on his own Ph.D. journey which will end soon happily.

My family, extending now, has always been with me with their love and support. I truly feel blessed for being a member of this kind of family. Their belief in me has been the foundation upon which I have built my confidence and determination. I would like to express my special thanks to Serenay BOZKURT, my better half, for being the supervisor of this foundation.

I feel that this thesis will be very incomplete if I do not thank my little editors; Cora, Saki, and Sütlaç. I would like to thank my cats who sit on the table day and night, watching line by line what I write, and even randomly making additions to this thesis sometimes.

I sincerely appreciate the contributions made by each and every member of my small but mighty team to this endeavor. Thanks to all for being a part of my long long journey.

May 2023

Çağrı GÜZAY
(M.Sc.)

TABLE OF CONTENTS

Page

ABBREVIATIONS

AI	**:** Artificial Intelligence
A-IT2-FM	**:** Aggressive IT2-FM
BLE	**:** Bluetooth Low Energy
CoM	**:** Center of Mass
CPU	**:** Central Processing Unit
CRTP	**:** Crazy Realtime Transfer Protocol
DFLC	**:** Double Input Fuzzy Logic Controller
DoF	**:** Degree of Freedom
EEPROM	**:** Electronically Erasable Programmable Read-Only Memory
FLC	**:** Fuzzy Logic Controller
FM	**:** Fuzzy Mapping
FOU	**:** Footprint of Uncertainty
FS	**:** Fuzzy Set
GCC	**:** GNU Compiler Collection
HLC	**:** High-Level Commander
IMU	**:** Inertial Measurement Unit
INDI	**:** Incremental Non-linear Dynamic Inversion
IT2	**:** Interval Type-2
LMF	**:** Lower Membership Function
LQR	**:** Linear Quadratic Regulator
LQT	**:** Linear Quadratic Tracking
MCU	**:** MicroController Unit
MF	**:** Membership Function
MPC	**:** Model Predictive Controller
PD	**:** Proportional-Derivative
PID	**:** Proportional-Integral-Derivative
PWM	**:** Pulse-Width Modulation
RTOS	**:** Real-Time Operating System
SD	**:** Standard Deviation
SFLC	**:** Single Input Fuzzy Logic Controller
SIT2-FLC	**:** Single Input IT2-FLC
SIT2-FLC-3R	**:** SIT2-FLC with 3 rules
SIT2-FLC-5R	**:** SIT2-FLC with 5 rules
S-IT2-FM	**:** Smooth IT2-FM
SO	**:** Special Orthogonal Group
ST1-FLC	**:** Single Input Type-1 Fuzzy Logic Controller
T1	**:** Type-1
T2	**:** Type-2
ToF	**:** Time of Flight
UAV	**:** Unmanned Aerial Vehicle
UM	**:** Unit Mapping
UMF	**:** Upper Membership Function

SYMBOLS

$\mathbb{X}$: Universe of discourse
σ : Input of fuzzy logic controller
μ_A : Membership function for a type-1 fuzzy set A
$\varsigma_{(.)}$: Left/right supports and centers for a membership function
ζ : Standard deviation
$\tilde{A}$: Type-2 fuzzy set
A : Type-1 fuzzy set
$\mathbb{U}$: Universe of discourse for secondary variable of type-2 fuzzy set
$\mathfrak{u}$: Secondary variable of a type-2 fuzzy set
$\bar{\mu}_{\tilde{A}}(\sigma)$: Upper membership function
$\underline{\mu}_{\tilde{A}}(\sigma)$: Lower membership function
K_u : Output scaling factor
φ_o : Output of the fuzzy logic controller
$\check{\sigma}^j$: Denormalized input
K_j : Input scaling factor
R^i : i^{th} rule
$A^{j,q}$: q^{th} antecedent type-1 membership function of j^{th} input
φ_i : Consequent membership function of the i^{th} rule
$\varphi_{o_{T1}}$: Output of the type-1 fuzzy logic controller
f_i : Firing strength of the i^{th} rule
$\varphi_{o_{IT2}}$: Output of the interval type-2 fuzzy logic controller
$\tilde{A}^{J,q}$: qt^{th} antecedent interval type-2 fuzzy logic controller of j^{th} input
$\underline{\varphi}_{o_{IT2}}$: Left endpoint of the type-reduced fuzzy set
$\bar{\varphi}_{o_{IT2}}$: Right endpoint of the type-reduced fuzzy set
L : Left switching point
R : Right switching point
$\underline{f}_i$: Lower firing strength for i^{th} rule
$\bar{f}_i$: Upper firing strength for i^{th} rule
$m_{j,q}$: Height of q^{th} antecedent interval type-2 fuzzy set for j^{th} input
u : Denormalized output of fuzzy logic controller
K_e : Fuzzy logic controller first input (error) scaling factor
K_{de} : Fuzzy logic controller second input (the derivative of error) scaling factor
Φ : Differential flatness function for flat outputs
Λ : Differential flatness function for system state
Γ : Differential flatness function for system input
$\mathbf{x}$: System state vector
ξ : Nonlinear system output, flat output vector
$\mathbb{R}$: Set of all real numbers
$\mathbf{u}$: System input vector
t_0 : Initial time
t_f : Final time

$\mathbf{x_0}$: Initial time state vector
$\mathbf{x_f}$: Final time state vector
$\rho_i(t)$: Basis function
ε_i : Coefficient of the basis function
$\mathscr{W}$: World frame
$\mathscr{B}$: Body frame
$\mathscr{C}$: Intermediate frame
ϕ : Roll angle
θ : Pitch angle
ψ : Yaw angle
R : Rotation matrix
${}^{W}R_B$: Rotation matrix from body frame to world frame
${}^{C}R_B$: Rotation matrix from body frame to intermediate frame
${}^{W}R_C$: Rotation matrix from intermediate frame to world frame
F_i : The force generated by ith rotor
M_i : The moment generated by ith rotor
k_F : Force coefficient
k_M : Moment coefficient
ω_i^{des} : Desired speed of ith motor
ω_i : Actual speed of ith motor
k_m : Motor gain
u_1 : Net thrust
u_2 : Roll moment
u_3 : Pitch moment
u_4 : Yaw moment
L : Distance between the center of the rotor and center of mass of the quadcopter
m : Mass of the quadcopter
g : Standard gravity
$\mathbf{r}$: Position vector of center of mass in world frame
x : x axis component of position
y : y axis component of position
z : z axis component of position, altitude
$\mathscr{I}$: Moment of inertia in body frame
$\omega_{\mathscr{B}\mathscr{W}}$: Angular velocity vector of the body frame in world frame
$\alpha_{\mathscr{B}\mathscr{W}}$: Angular acceleration vector of the body frame in world frame
$\omega_{\mathscr{B}\mathscr{C}}$: Angular velocity vector of the body frame in intermediate frame
$\omega_{\mathscr{C}\mathscr{W}}$: Angular velocity vector of the intermediate frame in world frame
p : Roll rate
q : Pitch rate
r : Yaw rate
$\mathbf{t}$: Acceleration vector with standard gravity (calculated from flat outputs)
$\mathbf{a}$: Acceleration vector of center of mass
$\mathbf{x}_C$: Intermediate frame x axis
$\mathbf{x_B}$: Body frame x axis
$\mathbf{y_B}$: Body frame y axis
$\mathbf{z_B}$: Body frame z axis
$\mathbf{x_W}$: World frame x axis
$\mathbf{y_W}$: World frame y axis
$\mathbf{z_W}$: World frame z axis

$\mathbf{h}_\omega$: Projection vector of angular velocity vector
$\mathbf{h}_\alpha$: Projection vector of angular acceleration vector
Υ : Angular acceleration characterization matrix
$\boldsymbol{\iota}$: Angular acceleration characterization vector
$\xi_T(t)$: Specified trajectory
$\mathbf{r}_T(t)$: Specified position
$\psi_T(t)$: Specified yaw
$\mathbf{e}_R$: Position error
$\mathbf{e}_v$: Velocity error
$\mathbf{F}_{des}$: Desired thrust vector
K_p : Position error gain matrix
K_v : Velocity error gain matrix
$\mathbf{e}_\omega$: Angular velocity error
$\mathbf{e}_R$: Orientation error
R_{des} : Desired rotation matrix
$\mathbf{x}_{C,des}$: Desired intermediate frame x axis
$\mathbf{x}_{B,des}$: Desired body frame x axis
$\mathbf{y}_{B,des}$: Desired body frame y axis
$\mathbf{z}_{B,des}$: Desired body frame z axis
${}^B[\omega_{\mathscr{B}\mathscr{W},T}]$: Desired angular velocity in body frame
$\boldsymbol{\sigma}$: Input vector of the single input fuzzy logic controllers
$\mathbf{K}_R$: Orientation error gain matrix
$\mathbf{K}_\omega$: Angular velocity error gain matrix
$\mathbf{K}_e$: Matrix of input scaling factors
$\boldsymbol{\varphi}$: Vector of single input fuzzy logic controllers outputs
$\boldsymbol{\varphi}_j$: j^{th} fuzzy mapping
$\mathbf{K}_u$: Matrix of output scaling factors
B_i : i^{th} crisp consequent function
c_i : Core of the i^{th} antecedent membership function
k_{T1}^i : Gain term for type-1 fuzzy mapping
η_{T1}^i : Offset term for type-1 fuzzy mapping
l : Radius
γ : Sensitivity angle
m_i : Height of the i^{th} lower membership function
σ' : Crisp input
$\varphi_{o_{IT2}}^r$: Right endpoint of the type-reduced set
$\varphi_{o_{IT2}}^l$: Left endpoint of the type-reduced set
k_{IT2}^i : Nonlinear gain term for interval type-2 fuzzy mapping
η_{IT2}^i : Nonlinear offset term for interval type-2 fuzzy mapping
α : Design parameter for single input interval type-2 fuzzy logic controllers
l_A : Radius for aggressive fuzzy mapping
l_S : Radius for smooth fuzzy mapping
γ_A : Sensitivity angle for aggressive fuzzy mapping
γ_S : Sensitivity angle for smooth fuzzy mapping
t_T : Total flight duration of a trajectory
$\mathbf{P}$: Matrix for the polynomial coefficients of the waypoints
$\kappa_{(.),i}^n$: n^{th} degree coefficient of i^{th} waypoint for specified position or yaw

x_T : Specified x position
y_T : Specified y position
z_T : Specified z position
t : Time
β : Dimensionless time constant, timescale
τ : Nondimensionalized time
$\tilde{\mathbf{r}}_T$: Nondimensional specified position
$\tilde{\psi}_T$: Nondimensional specified yaw
v_{max} : Planned maximum velocity
a_{max} : Planned maximum acceleration
h : Performance measure for a single flight
d : Two-dimensional Euclidean distance
h_E : Average performance for a set of flights
h_{min} : Minimum performance measure
h_{max} : Maximum performance measure

LIST OF TABLES

Page

LIST OF FIGURES

Page

DIFFERENTIAL FLATNESS-BASED FUZZY CONTROLLER DESIGN FOR AGGRESSIVE MANEUVERING OF QUADCOPTERS

SUMMARY

This study presents a new differential flatness-based single input fuzzy logic controller structure for aggressive maneuvering control alongside its real-world application on a nano quadcopter. We propose both type-1 and interval type-2 single input fuzzy logic controllers as the primary controllers in the flight control system, which are built on the concept of differential flatness.

Today, quadcopters are used for a wide variety of applications and purposes such as aerial photography, search and rescue operations, surveying and mapping, inspection, agriculture, and emergency response. Quadcopters are getting more and more well-liked in the commercial and consumer markets as a result of the rising demand for their usage areas. Additionally, the dimensions of quadcopters have significantly changed along with the rapid development in contemporary technology. As a result, we can discuss quadcopter types such as mini, micro, or nano. Nano quadcopters, the smallest ones, are lightweight, more portable, and easier to operate and maneuver with high agility since they are constructed with small-scale rotors and frames. Due to their greater maneuverability and agility, nano quadcopters are primarily used for aggressive flights. One of these nano quadcopters also used for the experimental part of this study is Crazyflie which is commercialized by Bitcraze AB.

In Chapter 1, we give detailed literature research on the modeling and control of Crazyflie. The control problem includes not only aggressive maneuvers but also hovering conditions. The mainstream controller design consists of utilizing the flight controllers for the model which is linearized around hover conditions. This approach is effective only for comparatively small linear and angular accelerations rather than aggressive maneuvering since agility requires bigger and faster changes in linear and angular accelerations. Although linear controllers still tend to be favored due to their ease of design and implementation, nonlinear controllers perform better in terms of robustness and disturbance rejection. Additionally, there are many studies based on employing intelligent control methods such as fuzzy logic controllers for attitude control and trajectory tracking of quadcopters. Even though conventional (type-1) fuzzy logic controllers are the most widely used type of fuzzy logic controllers, a significant amount of study on interval type-2 fuzzy logic controllers has been conducted recently. Researches show that interval type-2 fuzzy logic controllers perform better than those type-1 counterparts due to the additional degree of freedom provided by the footprint of uncertainty in their interval type-2 fuzzy sets. However fuzzy logic controllers with two or three inputs are considered to be used predominantly, studies present that single input fuzzy logic controllers can simplify the design without any side effects or performance loss. Moreover, single input interval type-2 fuzzy logic controllers can provide

high-performance tracking results, even in the presence of environmental disturbances. When quadcopters are required to perform aggressive maneuvers, the mainstream controllers become unsatisfactory since the flight control system cannot generate large linear and angular accelerations to control the quadcopter. Therefore, aggressive flight control is a challenging topic that several studies are focusing on for quadcopters. Alongside the capability of aggressive maneuvering, generating feasible trajectories is an essential point for agile flights. Different linear controller approaches are proposed for aggressive trajectory tracking control of quadcopters. Dissimilar to linear approaches, various controller strategies such as nonlinear tracking controllers, learning-based methods, and gain scheduling controllers, Lyapunov-based methods are also introduced for aggressive maneuver control of quadcopters. Aggressive maneuvering control is dependent on generating feasible trajectories since desired positions, velocities, and angular and linear accelerations are determined during trajectory generation. Differential flatness takes to the stage when it comes to trajectory tracking applications such as tracking control of quadcopters, robot manipulators, or ground vehicles. Differential flatness is a concept for the systems that shows state and inputs are functions of the designated flat outputs and their time derivatives. Since the trajectory can be planned for flat outputs, the differential flatness property is an appropriate concept for trajectory generation and tracking since the output space can be mapped to system inputs.

In Chapter 2, Crazyflie system analysis is performed comprehensively for both hardware and software. Software architecture is explained both visually and textually by giving references to firmware source code. Thus, major parts of the firmware such as the controller, estimator, and commander are examined alongside the software flow from power on to power off. Additionally, we work on controller implementations of Crazyflie in favor of deployment for the proposed controller structure.

We give mathematical descriptions for fuzzy set theory and fuzzy logic controllers in Chapter 3. This chapter covers preliminaries of fuzzy logic, type-1 and interval type-2 fuzzy sets, general structures of type-1/interval type-2 fuzzy logic controllers, and single input fuzzy logic controllers.

Chapter 4 is of primary importance in two respects: differential flatness and modeling of Crazyflie. First, we introduce the differential flatness concept, differentially flat systems, and the relationship between differential flatness and feasible trajectory generation. Then, the dynamic model is given by defining reference frames such as world, intermediate and body, and rotation matrix in the Z-X-Y Euler angles convention. Additionally, Newton-Euler equations are provided for modeling rotational and translational dynamics; and equations for the generated net thrust and generated roll, pitch, and yaw moments are given to be used in differential flatness property later. Afterward, the differential flatness-based characterization which is essential to show the quadcopter is a differentially flat system is performed for the position, orientation, angular velocity, angular acceleration, and control inputs (roll, pitch, yaw moments). Thus, it is proved that flat outputs can be mapped to the system state and control inputs which means the quadcopter presents differential flatness property.

In Chapter 5, we present novel differential flatness-based type-1 and interval type-2 fuzzy flight controllers for the agile trajectory tracking control problem of quadcopters. The proposed approach not only utilizes differential flatness property but also unveils

the capabilities of single input fuzzy logic controllers to handle nonlinear and uncertain dynamics. First, we develop and use a geometric approach to make interpretations for the fuzzy mapping characteristics which are determined by design parameters. This approach consists of a circle located at the origin with a radius l that provides valuable information about the region and level of sensitivity ($tan(\gamma)$), aggressiveness/smoothness, for fuzzy mappings of single input fuzzy logic controllers. We show that while a single input type-1 fuzzy logic controller with $N = 5$ rules generates piecewise linear fuzzy mapping, $N = 3$ counterpart has a unit mapping that is completely linear. However, unit mapping can also be generated by type-1 single input fuzzy logic controller with $N = 5$ rules; we focus to analyze how piecewise linear mapping can be shaped via design parameters to obtain an aggressive or smooth fuzzy mapping. We also employ single input interval type-2 fuzzy logic controllers with both $N = 3$ and $N = 5$ rules. Unlike type-1 counterparts, these interval type-2 fuzzy logic controllers have nonlinear fuzzy mappings. We present that interval type-2 fuzzy mapping with $N = 3$ rules cannot be tuned for radius l and sensitivity $tan(\gamma)$ at the same time since the design parameter of this configuration affects both region and level of sensitivity. Single input interval type-2 fuzzy logic controller with $N = 5$ rules is built up its type-1 baseline and exhibits piecewise nonlinear fuzzy mapping that can be shaped by design parameters for both region and level of sensitivity separately. Note that if the proposed method is tuned for unit mapping it reduces to the baseline crisp controller which is referenced in this study and employed in comparative experimental studies.

Chapter 6 demonstrates experimental studies which are conducted in an indoor environment. Based on the analysis in Chapter 5, we present simple tuning guidelines and then design fuzzy logic-based flight control systems, which were implemented as onboard real-time controllers that are differential flatness-based type-1 with $N = 5$ rules and interval type-2 with $N = 3$ and $N = 5$ rules fuzzy flight controllers. Also, we design four trajectories with different velocity, acceleration, and altitude characteristics; and determine two timescale values per trajectory. A single experiment consists of 10 flights for a trajectory with a selected controller and a timescale value which is a multiplier for trajectory duration. Then, we define a performance measure and evaluate the performances of each experiment by calculating the average, minimum, maximum values, and standard deviations.

For the aggressive trajectory tracking control problem, the experimental results present that proposed differential flatness-based type-1 and interval type-2 fuzzy flight controllers exhibit better flight performance than baseline crisp controller. Single input interval type-2 fuzzy logic controllers with $N = 5$ rules provide overall preferable results compared to type-1 counterpart with $N = 5$ rules and interval type-2 counterpart with $N = 3$ rules. Despite its simplicity of design, single input interval type-2 fuzzy logic controller with $N = 3$ rules give remarkable results. The tracking performance for a design of single input interval type-2 fuzzy logic controller with $N = 5$ rules is more consistent than any other across all trajectories as the trajectory becomes more aggressive with smaller timescale values.

ÇOK ROTORLU HAVA ARAÇLARININ AGRESİF MANEVRA KONTROLÜ İÇİN DİFERANSİYEL DÜZLÜK TABANLI BULANIK KONTROLÖR TASARIMI

ÖZET

Bu çalışma, agresif manevra kontrolü için yeni bir diferansiyel düzlük tabanlı tek girişli bulanık mantık kontrolör yapısı sunmasının yanı sıra bir nano çok rotorlu insansız hava aracı üzerindeki gerçekleme uygulamasını da içermektedir. Diferansiyel düzlük kavramı üzerine inşa edilen uçuş kontrol sisteminde birincil kontrolörler olarak hem tip-1 hem de aralık değerli tip-2 tek girişli bulanık mantık kontrolörleri önerilmektedir.

Günümüzde çok rotorlu insansız hava araçları hava fotoğrafçılığı, arama ve kurtarma operasyonları, ölçme ve haritalama, denetleme, tarım ve acil durum müdahalesi gibi çok çeşitli uygulamalar ve amaçlar için kullanılmaktadır. Bu araçlar, kullanım alanlarına olan talebin artması sonucunda ticari ve son tüketici pazarlarında giderek artan bir ilgi görmektedir. Ek olarak, günümüz teknolojisindeki hızlı gelişmeler ile birlikte çok rotorlu insansız hava araçlarının boyutları önemli ölçüde küçülmüştür. Sonuç olarak mini, mikro veya nano gibi türler üzerine konuşmak mümkündür. En küçükleri olan nano çok rotorlu insansız hava araçları hafif, kolay taşınabilir gövde materyalleri ve küçük ölçekli rotorlar ile yapıldıkları için bu tip araçların yüksek çeviklikle uçuş ve manevra yapması daha kolaydır. Daha yüksek manevra kabiliyetleri ve çeviklikleri nedeniyle, nano çok rotorlu insansız hava araçları öncelikle agresif uçuşlar için tercih edilmektedir. Bu çalışmanın deneysel kısmında da kullanılan bu nano araçlardan biri, Bitcraze firması tarafından ticari olarak üretilen Crazyflie ürünüdür.

Bölüm 1'de Crazyflie'ın modellenmesi ve kontrolü hakkında ayrıntılı literatür araştırması verilmektedir. Bu araçların kontrol problemi sadece agresif manevraları değil, havada asılı kalma koşullarını da içermektedir. Ana akım kontrolör tasarımı, havada asılı kalma koşulları etrafında doğrusallaştırılan model için uçuş kontrolörlerinin kullanılmasına dayanmaktadır. Ancak çeviklik, doğrusal ve açısal ivmelerde daha büyük ve daha hızlı değişiklikler gerektirdiğinden, bu yaklaşım agresif manevralardan ziyade yalnızca nispeten küçük doğrusal ve açısal ivmeler için etkilidir. Doğrusal kontrolörler, tasarım ve uygulama kolaylığı nedeniyle hâlâ tercih edilme eğiliminde olsalar da doğrusal olmayan kontrolörler, dayanıklılık ve bozucu bastırma açısından daha iyi performans göstermektedir. Ek olarak, çok rotorlu insansız hava araçlarının konum kontrolü ve yörünge takibi için bulanık mantık kontrolörleri gibi akıllı kontrol yöntemlerinin kullanılmasına dayalı birçok çalışma bulunmaktadır. Geleneksel ya da diğer adıyla tip-1 bulanık kontrolörler, bulanık mantık kontrolörlerinin en yaygın kullanılan tipi olmasına rağmen, son zamanlarda aralık değerli tip-2 bulanık kontrolörler üzerine önemli ölçüde çalışma yapılmıştır. Araştırmalar, aralık değerli tip-2 bulanık kontrolörlerinin, aralık değerli tip-2 bulanık kümelerindeki belirsizliğin ayak izi tarafından sağlanan ek serbestlik derecesi nedeniyle tip-1 muadillerinden daha iyi performans sergilediğini göstermektedir. İki veya üç girişli bulanık kontrolörler önceden

baskın olarak tercih edilse de çalışmalar, tek girişli bulanık kontrolörlerin herhangi bir istenmeyen etki veya performans kaybı yaratmadan tasarımı basitleştirebileceğini göstermektedir. Ayrıca, tek girişli aralık değerli tip-2 bulanık kontrolörler, çevresel bozucuların varlığında bile yüksek performanslı yörünge takip sonuçları sağlayabilir. Bir, çok rotorlu insansız hava aracının agresif manevra yapması gerektiğinde, uçuş kontrol sistemi bu aracı kontrol etmek için büyük doğrusal ve açısal ivmeler üretemediğinden ana akım kontrolörler yetersiz kalabilmektedir. Bu nedenle, çok rotorlu insansız hava araçları için agresif uçuş kontrolü birçok çalışmanın odaklandığı zorlu bir konudur. Agresif manevra kabiliyetinin yanı sıra uygun yörüngeler oluşturmak da çevik uçuşlar için önemli bir noktadır. Çok rotorlu insansız hava araçlarının agresif yörünge izleme kontrolü için farklı doğrusal kontrolör yaklaşımları önerilmiştir. Doğrusal yaklaşımlardan farklı olarak; doğrusal olmayan yörünge takip kontrolörleri, öğrenmeye dayalı yöntemler, Lyapunov tabanlı yöntemler ve kazanç programlama kontrolörleri gibi çeşitli kontrol stratejileri de çok rotorlu insansız hava araçlarının agresif manevra kontrolü için öne sürülmüştür. Agresif manevra kontrolü aynı zamanda; istenilen konumlar, hızlar, açısal ve doğrusal ivmeler yörünge oluşturma sırasında belirlendiğinden, uygun yörüngelerin oluşturulmasına bağlıdır. Çok rotorlu insansız hava araçlarının, robot manipülatörlerin veya kara araçlarının güzergâh kontrolü gibi yörünge izleme uygulamaları söz konusu olduğunda diferansiyel düzlük ön plana çıkmaktadır. Diferansiyel düzlük; sistem durum ve girişlerinin, belirlenen düz çıkışların ve bunların zaman türevlerinin fonksiyonları olduğunu gösteren sistemler için öne sürülen bir kavramdır. Yörünge, düz çıkışlar için planlandığı zaman çıkış uzayı sistem girdilerine eşlenebileceğinden, diferansiyel düzlük özelliği yörünge oluşturma ve izleme için uygun bir kavram haline gelmektedir.

Bölüm 2'de, Crazyflie sistem analizi hem donanımsal hem de yazılımsal olarak kapsamlı bir şekilde gerçekleştirilmektedir. Yazılım mimarisi, gömülü aygıt yazılımının kaynak koduna referanslar verilerek hem görsel hem de metinsel olarak açıklanmaktadır. Bu nedenle, aygıt yazılımının kontrolör, kestirici ve kumanda gibi ana parçaları, sistemin açılmasından kapanmasına kadar olan yazılım akışı ile birlikte incelenmektedir. Ek olarak, önerilen kontrolör yapısının entegrasyonu için Crazyflie'ın tümleşik kontrolör uygulamaları ve şablonları üzerine çalışmalar da yapılmaktadır.

Bulanık küme teorisi ve bulanık mantık kontrolörler için matematiksel tanımlamalar, Bölüm 3'te verilmektedir. Bu bölüm, bulanık mantığın temel bilgilerini, tip-1 ve aralık değerli tip-2 bulanık kümeleri, tip-1/aralık değerli tip-2 bulanık kontrolörlerinin ve tek girişli bulanık kontrolörlerin genel yapılarını kapsamaktadır.

Bölüm 4, iki açıdan büyük öneme sahiptir: diferansiyel düzlük ve Crazyflie'ın modellenmesi. İlk olarak, diferansiyel düzlük kavramı, diferansiyel düz sistemler ve diferansiyel düzlük ile uygulanabilir yörünge üretimi arasındaki ilişki gibi konulara değinilmektedir. Daha sonra yer, ara ve gövde gibi referans çerçeveleri ve Z-X-Y Euler açıları kuralında dönme matrisi tanımlanarak dinamik model gösterilmektedir. Ek olarak, dönme ve öteleme dinamiklerini modellemek için Newton-Euler denklemlerinden faydalanılmakla beraber; daha sonra diferansiyel düzlük özelliğinde kullanılmak üzere, üretilen yalpalama, yunuslama, sapma momentleri ve net itki kuvveti için denklemler verilmektedir. Ardından, çok rotorlu insansız hava aracının diferansiyel olarak düz bir sistem olduğunu göstermek için gerekli olan diferansiyel

düzlük tabanlı karakterizasyon; konum, oryantasyon, açısal hız, açısal ivme ve kontrol girişleri (yalpalama, yunuslama, sapma momentleri) için gösterilmektedir. Böylece, düz çıkışların sistem durumuna ve kontrol girişlerine eşlenebileceği kanıtlanmaktadır ve bu da çok rotorlu insansız hava aracının diferansiyel düzlük özelliği gösterdiği anlamına gelir.

Bölüm 5'te çok rotorlu insansız hava araçlarının çevik yörünge izleme kontrol problemi için yenilikçi bir yöntem olarak diferansiyel düzlük tabanlı tip-1 ve aralık değerli tip-2 bulanık uçuş kontrolörleri sunulmaktadır. Önerilen yaklaşım sadece diferansiyel düzlük özelliğini kullanmakla kalmayıp, aynı zamanda doğrusal olmayan ve belirsiz dinamikler karşısında tek girişli bulanık kontrolörlerin yeteneklerini de ortaya koymaktadır. İlk olarak, tasarım parametreleri tarafından belirlenen bulanık haritaların özelliklerini yorumlamak için geometrik bir yaklaşım geliştirilmiş ve kullanıma sokulmuştur. Bu yaklaşım; tek girişli bulanık kontrolörlerin bulanık haritaları için orijinde tanımlanmış l yarıçaplı bir dairenin içinde kalan bulanık haritanın etkin olduğu giriş aralığı bölgesi ve hassasiyet seviyesi ($tan(\gamma)$), bulanık haritanın agresif ya da düz olması olarak nitelendirilmesi için kullanılmaktadır. Bu bölümde, $N = 5$ kurallı tek girişli tip-1 bulanık kontrolörünün parçalı doğrusal bulanık haritalama oluştururken, $N = 3$ eşdeğerinin tamamen doğrusal olan bir birim haritalamasına sahip olduğunu gösterilmektedir. Bununla birlikte, birim eşleme aynı zamanda bir $N = 5$ kurallı tip-1 tek girişli bulanık kontrolör tarafından da üretilebilir; ancak bu tip kontrolörlerde, agresif veya düz bir bulanık haritalama elde etmek için tasarım parametreleri aracılığıyla parçalı doğrusal haritalamanın nasıl şekillendirilebileceğinin analizine odaklanılmıştır. Bu bölümde ek olarak, hem $N = 3$ hem de $N = 5$ kurallı tek girişli aralık değerli tip-2 bulanık kontrolörlerin tasarımı üzerine de çalışmalar yapılmaktadır. Tip-1 muadillerinden farklı olarak, bu aralık değerli tip-2 bulanık kontrolörlerin doğrusal olmayan bulanık haritalara sahip olduğu gösterilmiştir. $N = 3$ kurallı aralık değerli tip-2 bulanık haritalamanın, l yarıçapı ve $tan(\gamma)$ hassasiyeti için aynı anda ayarlanamayacağı, çünkü bu konfigürasyonun tasarım parametresinin hem bölgeyi hem de hassasiyet seviyesini etkilediği analiz edilmiştir. $N = 5$ kurallı tek girişli aralık değerli tip-2 bulanık kontrolörün, tip-1 muadilinin üzerine inşa edilmekle beraber hem l tarafından belirlenen bölge hem de hassasiyet seviyesi ($tan(\gamma)$) için tasarım parametreleri tarafından ayrı ayrı şekillendirilebilen parçalı ve doğrusal olmayan bulanık haritaya sahip olduğu gösterilmiştir. Önerilen yöntemin birim haritalama için ayarlanması durumunda, bu tezde başvurulan ve karşılaştırmalı deneysel çalışmalarda kullanılan temel keskin kontrolöre indirgenmesi de ayrıca sunulmaktadır.

Bölüm 6'da, kapalı ortamda yürütülmüş deneysel çalışmalara dair detaylar ve deneysel sonuçlar verilmektedir. Bölüm 5'te yapılan analizler baz alınarak, bu bölümde bulanık mantık tabanlı uçuş kontrol sistemleri için sade ayarlama yönergeleri sunulmakla beraber, bu kontrol sistemleri ilgili yönergelere göre tasarlanmaktadır. Bahsi geçen bulanık mantık tabanlı uçuş kontrol sistemleri doğrudan hedef donanım üzerinde gerçek zamanlı olarak çalışan ve bu çalışmada önerilen $N = 5$ kurallı diferansiyel düzlük tabanlı tip-1 bulanık kontrolör ile $N = 3$ ve $N = 5$ kurallı aralık değerli tip-2 bulanık kontrolörleridir. Bu bölümde ayrıca; farklı hız, ivme ve irtifa özelliklerine sahip dört uçuş güzergâhı tasarlanıp her yörünge için iki farklı zaman ölçeği değeri belirlenmiştir. Zaman ölçeği, tasarlanan güzergâh uçuş süresi için bir çarpandır ve güzergâh boyunca uçuşun ne kadar hızlı ya da ne kadar yavaş olacağını gösterir. Bu yüzden belirlenen iki

zaman ölçeğinden biri nominal uçuş süresine, diğeri ise agresif uçuşu temsil eden daha kısa bir uçuş süresine tekabül etmektedir. Tek bir deney; seçilen bir kontrolör, güzergâh ve zaman ölçeği için yapılan 10 tekrarlı uçuşu içermektedir. Ardından bir performans ölçütü tanımlanıp, bu ölçütün ortalama, minimum, maksimum değerleri ve standart sapmaları hesaplanarak her deneyin performansı değerlendirilmiştir.

Deneysel sonuçlar, önerilen diferansiyel düzlük tabanlı tip-1 / aralık değerli tip-2 bulanık uçuş kontrolörlerinin agresif yörünge izleme kontrol problemi için temel keskin kontrolörden daha iyi uçuş performansı sergilediğini göstermektedir. $N = 5$ kurallı tek girişli aralık değerli tip-2 bulanık kontrolörleri, $N = 5$ kurallı tip-1 eşdeğerine ve $N = 3$ kurallı aralık değerli tip-2 eşdeğerine kıyasla tümüyle daha iyi sonuçlar sağlamaktadır. Basit tasarımına rağmen $N = 3$ kurallı tek girişli aralık değerli tip-2 bulanık kontrolör dikkate değer sonuçlar vermektedir. Bir diğer $N = 5$ kurallı tek girişli aralık değerli tip-2 bulanık kontrolör tasarımı için yörünge takip performansı, yörüngenin daha küçük zaman ölçeği değerleri ile daha agresif hale gelmesi koşulunda tüm yörüngelerde diğerlerinden daha tutarlı sonuçlar sergilemektedir.

1. INTRODUCTION

At the current state of science and technology, it is clear that humanity has made great advancements. Some of the hot topics from the past few years such as quantum computing, artificial intelligence, virtual reality, augmented reality, and genomics are quite sophisticated, and yet people are still creating new technologies on an ongoing basis in both a cumulative and collective way. The acquisition, processing, and maintenance of knowledge form the foundation of development. Although we have access to resources and tools that are cutting-edge for our age, our ancestors did not have it as simple. They had to collect basic information through observations to comprehend what occurs in nature. In this way, some events and actions could be created by drawing inspiration from or imitating nature. Although there are numerous examples of these, we would like to concentrate on one in particular since it is the focus of this study: flying.

The history of flying traces the development of mechanical flight, from the earliest attempts in the early 19th century to the first successful powered flights by the Wright brothers in 1903, and on to the development of larger and more powerful aerial vehicles throughout the 20th century. We can infer that this may be linked with the mechanical technologies that emerged with industrialization, the interest in the use of electricity and radio waves, and –unfortunately– factors like World War I.

Since the main subject of this study is quadcopters, we would like to briefly touch on the historical development of quadcopters and Unmanned Aerial Vehicles (UAVs). Although the terms such as UAVs and quadcopters may seem brand-new to us today, the development of these kinds of vehicles was happening at the same time as other air vehicles, and there are some exceptionally striking historical examples.

At the beginning of the 1900s, pioneers in the field of aeronautics began experimenting with the idea of using multiple rotors to lift and control aircraft. One of the first known references to a quadcopter was designed by Bréguet Aviation and appeared in 1907.

This Bréguet-Richet Gyroplane seen in Figure 1.1 was basically a four-rotor helicopter, and it achieved only vertical take-off to an altitude of a couple of meters. Due to unstability, this vehicle externally was held up during tests [3,4].

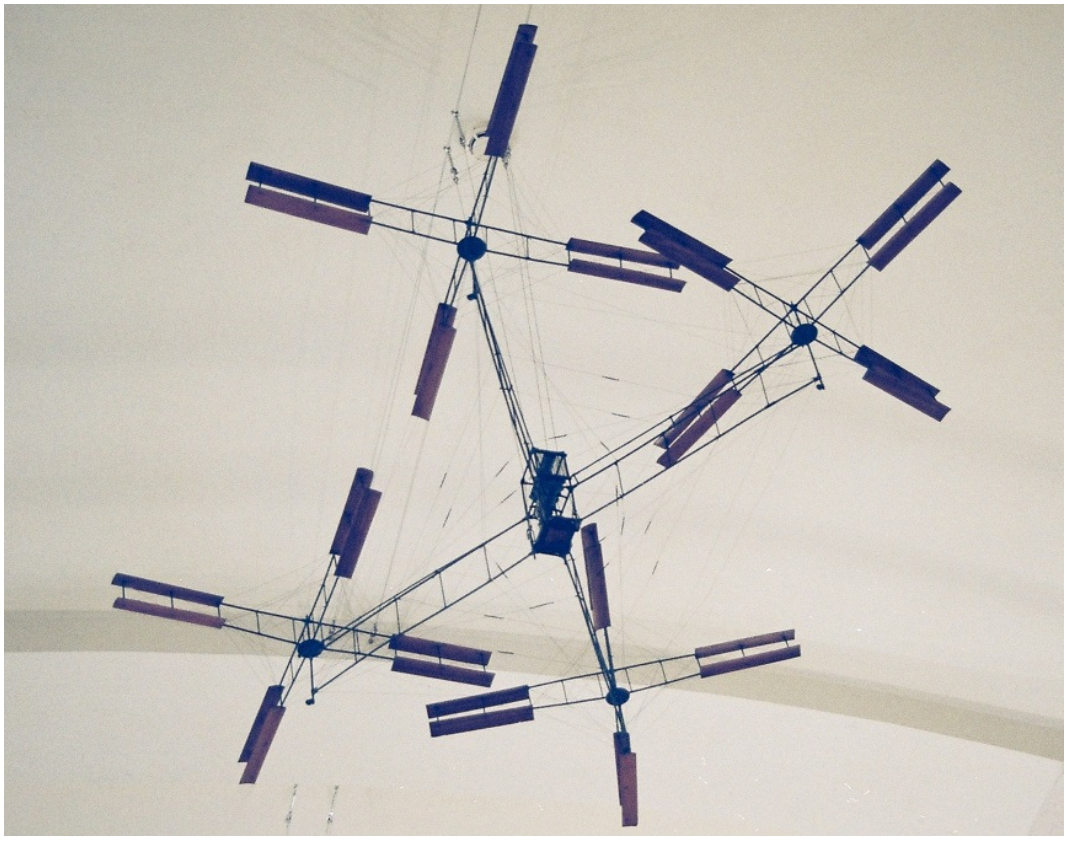


Figure 1.1 : Bréguet-Richet Gyroplane [1].

The radio-controlled aerial target, designed by Archibald Low and manufactured by the British aircraft research establishment for defensive and attacking purposes, was one of the first UAV prototypes to be used in the military during World War I [5]. In 1917, the first UAV torpedo named Kettering Bug which has a stabilization system based on pre-configured pneumatic and electrical controls to reach the target is developed. Due to the low accuracy testing results by 22%, this vehicle is found to be highly unreliable for military purposes [6]. In the early 1920s, an aircraft with 4 rotors, named as Bothezat Helicopter and seen in Figure 1.2, made 100 test flights with records of 2 min 45 seconds maximum flight time and almost 10m peak altitude [7].

In the years of World War II, UAVs with radio control and a simple autopilot that controls altitude and airspeed are mainly developed for aerial attack missions [8,9].

Figure 1.2 : De Bothezat Helicopter [2].

UAV development continued after World War II and throughout Cold War, with the main purposes of surveillance, missile target practice, and air defense [9,10]. In the second half of the 20th century, technological advancements and miniaturization made it possible for UAVs to be developed for both commercial and civilian purposes.

1.1 Literature Review

Quadcopters are utilized in an extensive application range with several purposes, such as remote sensing, delivery, airborne inspection, and surveillance [11]. For a particular mission, goals can be established such as trajectory tracking, completing side tasks that are either preset or appear as the mission progresses, avoiding collisions or obstructions, and reacting to environmental factors. System resources of the quadcopter should be employed effectively to carry out the mission due to the restricted electrical power supply. As a result, it becomes necessary to employ an automatic control structure that enables autonomous actions, optimized flying performance, trajectory tracking, and mission planning according to arbitrary criteria imposed by the electrical power and

capabilities of the CPU which is utilized for auto-pilot. Regardless of its application, the autonomous flight control system of the quadcopter is the fundamental requirement [12]–[14].

The control problem is challenging due to complex, dominant nonlinearities and highly coupled dynamics, especially in aggressive maneuvering. Aggressive maneuvering is critical in the autonomous indoor navigation of quadcopters due to the limited space of action and environment. Accordingly, small-sized ones such as nano quadcopters are predominantly used in interior environments with higher maneuverability and agile movement capability. One of these nano quadcopters is Crazyflie one from Bitcraze [15]. The aggressive maneuverability of nano quadcopters is perhaps one of the best examples of technology mimicking nature. It can be assumed that it resembles the agile flight styles of bats, although they do not have a good reputation in recent world history unluckily.

Many studies are focusing on the modeling and control problem of Crazyflie not only because it is capable of maneuvering aggressively, but it is also an open hardware product with open-source firmware [16]–[19]. A general perspective for Crazyflie 2.0 and its control is provided in [16]. Crazyflie 2.0 is modeled via using deep neural networks and controlled by two cascaded control structures as LQR on a host computer, PD on board [17]. Radiofrequency identification technology is utilized in Crazyflie 2.0 for remote sensing [18]. An ultra-low-power visual odometry concept is proposed to determine the position and orientation of nano quadcopters, and the method is integrated with Crazyflie 2.0 [20]. A swarm aggregation and control method based on the estimated positions of each Crazyflie 2.0 via OptiTrack, an offboard motion capture system, is implemented on a host computer to control a Crazyflie 2.0 swarm [21].

The mainstream controller design approach is to tune flight controllers based on the linearized model around hover conditions. Thus this approach is efficient only under relatively small linear and angular accelerations [22]–[25]. PID, LQR, and PD controllers are utilized for trajectory tracking of Crazyflie 2.0 which estimates its position via OptiTrack [25]. Similarly, PID and LQT controllers are implemented and compared for trajectory tracking of Crazyflie 2.0 by using position feedback from

two different sensing systems an ultra-wideband ranging system and VICON, another offboard motion capture system [18]. For instance, single loop and double loop PID controllers are used for attitude and position control [22]. LQR control scheme is utilized for trajectory tracking [23]. A hybrid PID-LQR control approach is proposed besides PID and LQR ones [24].

Although linear controllers are preferred for their simplicity in both design and implementation, nonlinear controllers exhibit better performances concerning robustness and disturbance rejection [26]–[28]. In [26,27], MPC with nonlinear H_∞ controller is employed for trajectory tracking under sustained disturbances. For attitude and position control, three different nonlinear control methods, backstepping, sliding mode control, and feedback linearization, are utilized in both simulations and experiments [29]. A recursive nonlinear control scheme, backstepping, is used for position control of under-actuated quadcopter [28].

On the other hand, intelligent control techniques are also widely used to control quadcopters alongside conventional controllers. Especially, conventional (Type-1) Fuzzy Logic Controllers (FLCs) are designed for attitude control and trajectory tracking [30]–[32]. Although Type-1 (T1) FLCs are the most commonly used FLCs, numerous research has also been done in recent years on Interval Type-2 (IT2) FLCs [33]–[35]. IT2-FLCs may provide better performance than those T1 counterparts due to having an additional degree of freedom contributed by the Footprint of Uncertainty (FOU) in their IT2 Fuzzy Sets (FSs) [36]. Besides their internal structure, FLCs can also be grouped concerning their input size. For years, the most commonly considered FLCs were the two or three inputs structures [33]. Nevertheless, it has been shown that Single Input FLCs (SFLCs) can simplify the design [37,38]. It has been shown in [39] that Single input IT2-FLC (SIT2-FLC) can result in high-performance tracking performance, even in the presence of wind disturbances.

The mainstream design approach becomes inefficient when there is a need for quadcopters capable of executing aggressive maneuvers since the flight control system must be capable of controlling/generating large linear and angular accelerations. In the literature, several studies are focusing on the aggressive flight control of quadcopters.

Different linear controller approaches are proposed for quadcopter's trajectory tracking control as well as generating feasible trajectories [14,40]. A survey which is providing different approaches for trajectory tracking control of quadcopters presents a modified PID controller that can be used for aggressive maneuvers of small-size quadcopters stating that insufficiency of mainstream design approaches [14]. Similarly, attitude and altitude PID controllers are used for aggressive maneuvering under aerodynamic effects which is observed on the further side of nominal hover conditions [41]. In [42], a nonlinear tracking controller is utilized for aggressive flight control of small-sized quadcopters using only an onboard monocular camera and IMU for onboard state estimation. A novel learning-based method is proposed for planning aggressive maneuvers by using motion primitives to increase navigation performance [43]. A cascade control strategy that is running off-board is used for aggressive maneuvering of Crazyflie 2.0 [44]. A control architecture based on [44] is implemented for aggressive maneuver control of Crazyflie 2.0 swarm that relies on orientation and position measurements done by OptiTrack [45]. Alternatively, a robust gain scheduling real-time control strategy is proposed [46], whereas Lyapunov-based controller synthesis is introduced for multi-flip maneuvers of Crazyflie [47]. Furthermore, differential flatness-based control techniques allow the designing of aggressive flight control systems that generate feasible trajectories and enable hierarchical control [40,48,49]. Attitude, hover and 3D trajectory controllers are implemented with feasible trajectory generation for aggressive maneuvers such as flying through narrow gaps and perching of quadcopters by using VICON for states estimation [40]. Linear controller structures with differential flatness are implemented on a quadcopter to provide highly agile trajectory tracking ability [50]. In [51], trajectory generation and differential flatness-based controller for aggressive maneuvering are proposed. This controller is one of the built-in onboard controllers of Crazyflie 2.1. The concept of a differentially flat system has been utilized to generate feasible trajectories to be tracked by nonlinear controllers [52,53]. Differential flatness-based MPC is proposed for trajectory tracking of quadcopters [52], and nonlinear geometric control with differential flatness is applied to a quadcopter with a cable-suspended load to provide tracking of quadcopter attitude, load attitude, and load position along aggressive trajectories [53]. A cascaded nonlinear feedback control

method with OptiTrack is used to control a differentially flat system consisting of a quadcopter with rotor drag effects on high-speed trajectories [48]. In [49], the aggressive trajectory tracking control problem of quadcopters is handled with incremental nonlinear dynamic inversion and differential flatness. A mixed integer programming algorithm with differential flatness property is introduced to perform aggressive maneuver control of Crazyflie in obstacle-dense environments by providing collision-free flights [54,55]. A nonlinear position control structure with the differentially flat model, based on [51], is proposed and deployed to a swarm of 49 Crazyflie 2.0 vehicles to handle with aggressive flight maneuvers of large swarm quadcopters [56].

1.2 Purpose of Thesis

This study presents novel differential flatness-based T1 and IT2 fuzzy flight controllers for aggressive maneuvering and demonstrates their capability with real-world applications conducted on the Crazyflie 2.1 nano quadcopter. The proposed structure exploits not only the differential flatness property of the quadcopter dynamics but also the capability of SFLCs in handling nonlinear and uncertain dynamics. We construct the proposed flight control system on the concept of differential flatness with SFLCs employing and processing T1 or IT2 FSs. We present a geometric interpretation within the study to analyze the Fuzzy Mappings (FMs) of the Single Input T1-FLCs (ST1-FLCs) and SIT2-FLCs. To analyze the aggressiveness/smoothness of the FM concerning the equilibrium point, we define slope and radius parameters. Based on this interpretation, we provide efficient yet straightforward tuning guidelines to shape the FMs and thus design ST1-FLCs and SIT2-FLCs. We deployed the proposed controllers into Crazyflie 2.1 firmware to evaluate the real-time aggressive maneuvering performances as onboard controllers. We conducted experiments by running indoor flights on four designed trajectories with different dynamics and characteristics. We compared the performances of the differential flatness-based T1 and IT2 fuzzy flight controllers alongside the crisp controller presented in [51]. The presented results clearly show that both the ST1 and SIT2 FLCs result in high performance even if the desired trajectory is highly aggressive. The real-time aggressive maneuvering performance of the proposed differential flatness-based SIT2-FLC is provided in the *Supplementary*

Material (see Appendix A). We conclude that the performance of SIT2-FLCs is better than its T1 and crisp counterparts, as they use and process IT2-FSs rather than T1-FSs.

The main contributions/results of this study are summarized as follows:

- A novel differential flatness-based SFLC structure for aggressive maneuvering, which exploits the differential flatness property and the capability of SFLCs in handling nonlinear and uncertain dynamics.
- Design and deployment of both T1 and IT2 SFLCs based on the concept of differential flatness.
- A geometric approach to analyze how the design parameters of SFLCs shape the characteristics of the FMs alongside tuning guidelines for T1 and IT2 FLCs.
- Extensive real-world experiments to show the superiority of the real-time aggressive maneuvering performance of SFLCs.

This study is organized as follows: in Chapter 2, detailed explanations for Crazyflie hardware and software architecture are given by both illustrative diagrams and reference of firmware source code. Preliminaries of FLCs are given in Chapter 3. FS Theory, T1-FLCs, IT2-FLCs, SFLCs, and double input FLCs are explained in this chapter, respectively. In Chapter 4, the differential flatness concept is given first and foremost. Then, a dynamical model is introduced for Crazyflie 2.1 as an algebraic framework. In this chapter, Crazyflie 2.1 is considered as a differentially flat system, and its differential flatness-based characterization is established by examining position, orientation, angular velocity, angular acceleration, and inputs. The design and analysis of ST1-FLC and SIT2-FLC structures via geometric approach are presented in Chapter 5. This chapter is directly linked to Chapter 4 since outcomes of the differential flatness property are used in FLC structures. Additionally, behaviors of ST1-FLC and SIT2-FLC are interpreted for variation in the values of design parameters. Chapter 6 provides all the details of the experimental studies alongside the comparative experimental results. The trajectory generation method, generated feasible trajectories, and performance measures are given also in this chapter. Finally, the driven conclusions and future works are presented in Chapter 7.

2. CRAZYFLIE 2.1 SYSTEM ANALYSIS

Crazyflie 2.1 is a member of the nano quadcopters family with 92mm motor-to-motor distance, 27g take-off weight, and a maximum 15g payload weight additionally. It has an onboard LiPo battery charger as well as 7 minutes of flight time and 40 minutes of charging time with its stock battery.

Crazyflie 2.1, illustrated in Figure 2.1, has the X formation for the body frame which means the propellers located on the same diagonal have the same rotation direction. While motor-1 and motor-3 have counter-clockwise propellers, motor-2 and motor-4 have clockwise ones.

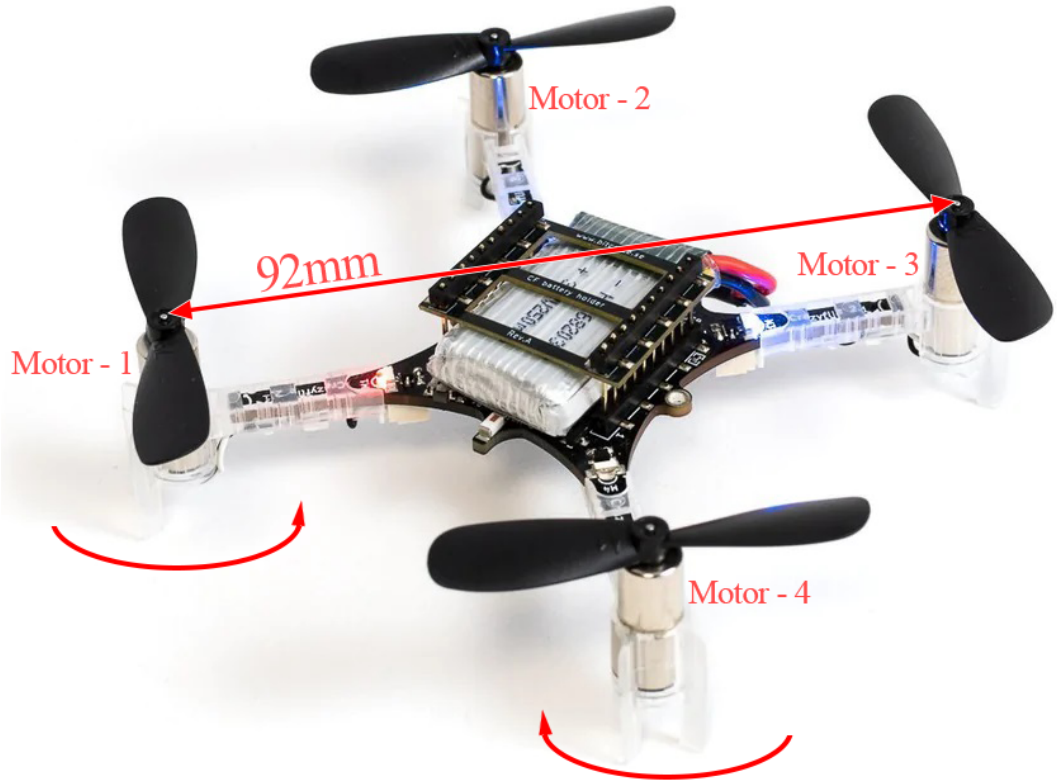


Figure 2.1 : Crazyflie 2.1 body frame.

2.1 System Architecture

Crazyflie 2.1 has two MicroController Units (MCUs) within the system architecture, as illustrated in Figure 2.2. The entire flight operation runs on the main MCU, while the communication and battery management operations run on the secondary MCU. Therefore main MCU is dedicated to flight control and the secondary MCU is only responsible for operations other than control. Hardware specifications for MCUs are that the main MCU is STM32F405 built on a 32-bit ARM® Cortex™-M4 core processor running at 168MHz with 192kB SRAM, 1MB flash, and the secondary MCU is nRF51822 built on 32-bit ARM® Cortex™-M0 core processor running at 32Mhz with 16kB SRAM, 128kB flash and 2.4 GHz radio supporting Bluetooth Low Energy (BLE) and proprietary protocols [57,58].

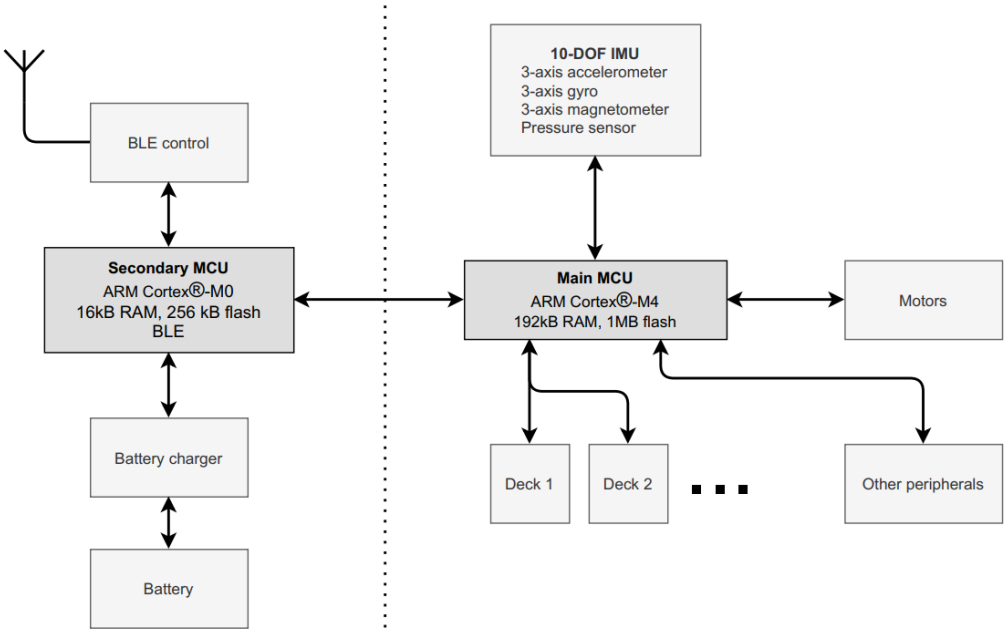


Figure 2.2 : Crazyflie 2.1 system architecture.

The nano quadcopter has a 6 DoF IMU (3-axis accelerometer, 3-axis gyro) and a high-precision pressure sensor out of the box. In addition to this, expansion boards called decks which are designed and manufactured for adding new features or extending existing features of the Crazyflie 2.1 can be mounted. For instance, active marker deck can be used for range tracking for external positioning systems, AI-deck adds on-board

computational capability for artificial intelligence applications, and flow deck extends the motion detection feature in any direction [59]–[61]. We added the flow deck module to Crazyflie 2.1 which has an optical flow sensor measuring the displacement of the quadcopter with respect to the ground.

2.2 Software Analysis

The main MCU runs the firmware built on FreeRTOS™, one of the Real-Time Operating Systems (RTOSes), which is used in embedded systems with time-critical applications. The firmware is written in mostly C language and is compiled by using GCC ARM Embedded toolchain on Ubuntu. Additionally, firmware communicates with the secondary MCU which is used for the BLE link. Thus, Crazyflie 2.1 can be commanded or monitored via radio protocol which is paired with a host computer running Crazyflie Python client or library [62,63].

Typical operating systems have wide features such as a graphical user interface, user accounts and management, file system abstraction, advanced memory handling, utilities, and system services. RTOSes are considered to have features other than typical operating systems. The main goal of an RTOS is execution speed with low system requirements.

Due to the memory, power, and computation restrictions of the embedded systems, FreeRTOS™ is designed to have a small footprint and simple integration. It was introduced around 2003 by Richard Barry. Today, it supports more than 40 MCU architectures and 15 toolchains. It is written in mostly C language and includes also assembly language, especially in device-specific configuration files. For multithreaded applications, FreeRTOS™ provides task scheduling, mutexes, semaphores, software timers, and thread ticks with millisecond-level intervals. Thus, it provides an environment consisting of shared resources, managed concurrency and lock mechanisms, and scheduling with priorities.

To understand the working principle of the firmware, we look closer at the source code provided. We follow firmware line by line from system power-on to system shutdown. First, all multithreaded operations are analyzed. These operations are simply tasks of which some are running concurrently that have mutexes and semaphores to access

shared resources and maintain synchronization. Crazyflie 2.1 firmware consists of predefined tasks of which each handles a different operation such as sensing, control, estimation, communication, and other processes, and some of them are given below with their descriptions:

- *PCA9685* is a 12-bit, 16-channel PWM driver for some peripherals such as motors and LEDs.
- *PWRMGNT* provides power-related operations such as monitoring and management.
- *SENSORS* is a driver for the IMU sensor which consists of an accelerometer and a gyro.
- *SYSLINK* handles communication between secondary and main MCUs and dispatches messages to each other.
- *APP* provides a layer for customized applications that are designed on top of the firmware.
- *CMDHL* is a task for the High-Level Commander (HLC) which computes smooth setpoints from high-level inputs such as take-off, landing, and polynomial trajectories. We set up HLC in the experiments due to the use of polynomial trajectories. The algorithm of this task will be mentioned later.
- *CRTP* is a stack (Crazy Realtime Transfer Protocol) for firmware data transfer.
- *KALMAN* is the task for Kalman filter implementation [64,65].
- *LOG* handles dynamic logging to monitor firmware states and warnings.
- *MEM* is the memory driver for handling one-wire, EEPROM, and other memory mappings over the CRTP link.
- *PARAM* is the task for managing and updating runtime parameters, i.e. a controller coefficient, in the firmware.
- *STABILIZER* runs all control operations after sensor readings, sensor fusion, and state estimation.

- *SYSTEM* handles top-level system-wide operations and runs other tasks.

FreeRTOS™ does the task management and scheduling in the given priority and stack size for Crazyflie 2.1. For illustrative purposes, some tasks of Crazyflie firmware are given with priorities in Figure 2.3.

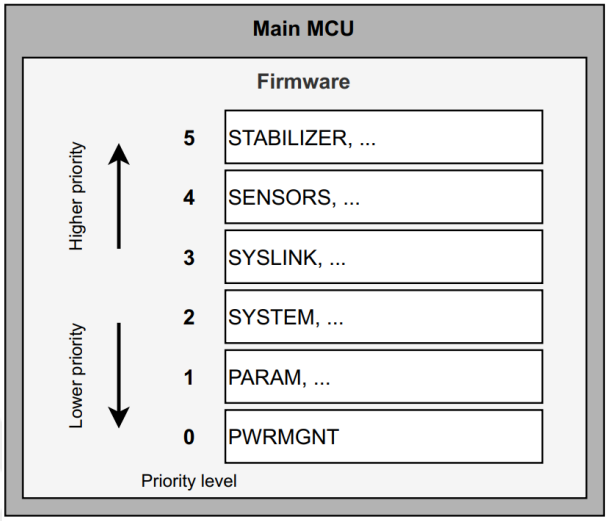


Figure 2.3 : Crazyflie task management.

The firmware entry point is `main()` which is implemented in `main.c`. This function does low-level platform initialization, launches *SYSTEM* task that invokes `systemTask` function found in `system.c`, and starts FreeRTOS™ task scheduler. *SYSTEM* task initializes high-level parts of the system such as top-level system modules, communication layer, commander block that includes HLC, and stabilizer by invoking `stabilizerInit()` function from `stabilizer.c`. This function does initializations of the sensors, state estimator, controller, power distribution, and so on. Additionally, *STABILIZER* task is created in this function, and it is bound to `stabilizerTask()` that is implemented in `stabilizer.c`. From system start to stabilization, source code flow is represented in Figure 2.4 by leaving out some parts of the code.

In one and/or more *STABILIZER* task cycle(s) following operations, illustrated in Figure 2.5, are done:

Figure 2.4 : Crazyflie 2.1 firmware from start up to stabilizer loop.

- The feedback information of the onboard sensors is processed by bundled estimators, complementary filter or Kalman filter, that are provided as default within the Crazyflie 2.1 firmware [32,66].
- Three state controllers are shipped with default firmware as PID, INDI and Mellinger, and one of these controllers can be selected dynamically via the host computer or by presetting it in the build phase of the firmware.
- Commander block handles with the desired state (setpoints) for position, velocity and attitude which can be set by CFlib (Crazyflie Python library, see [62]) or HLC.
- The controller calculates control signals consisting of roll, pitch, and yaw control commands which are transformed into PWM values in the power distribution block.
- Motors are commanded by calculated PWM values.

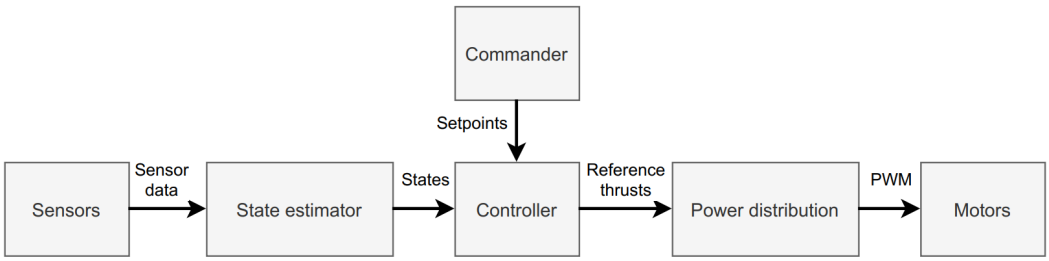


Figure 2.5 : Crazyflie stabilizer flow diagram.

2.2.1 State estimation and control

State estimation is a fundamental part of the quadcopters or robotic systems stabilization since the state estimator makes state estimation of the system from sensor readings, and the whole system relies on these estimations. The more accurate the estimation, the more proper control signals we get. As mentioned previously, Crazyflie 2.1 firmware comes with two kinds of estimators as the complementary filter and extended Kalman filter.

- *Complementary filter* is a lightweight filter that is considered for the general use case. It takes IMU sensor readings, gyroscope and accelerometer, and Time of Flight (ToF) distance measurement from the Z-ranger deck which is used for altitude sensing (if mounted). This filter, represented in Figure 2.6, gives estimated output for the attitude (roll, pitch, yaw) and altitude (z) of Crazyflie.

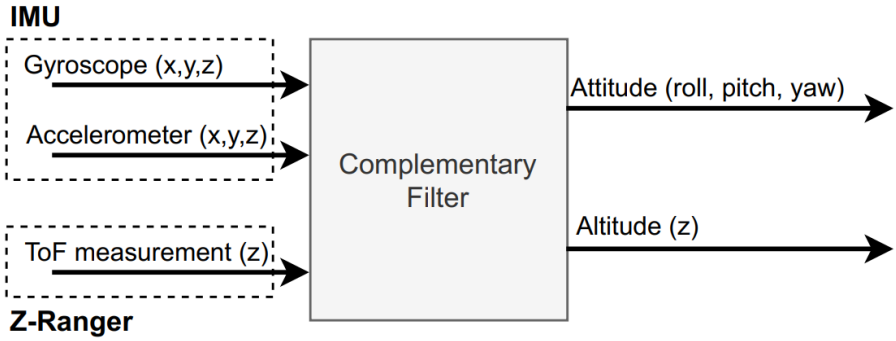


Figure 2.6 : Crazyflie complementary filter.

Implementation details of the complementary filter are seen in the firmware files `estimator_complementary.c` and `sensfusion6.c`. Additionally, this filter is the default choice for the firmware unless there is a mounted deck that requires extended Kalman filter [66].

- *Extended Kalman filter* is more sophisticated than the complementary filter, and it receives external sensor inputs in addition to internal ones as seen in Figure 2.7. Kalman filter is an iterative algorithm which is a special form of Bayesian filter and relies on a set of measurements that are sampled over time [67]. Kalman filter takes measurement noise (assumed to be Gaussian) into account as measurement error. Extended Kalman filter is an extension of Kalman filter which is used for non-linear dynamics.

  The feedback information of the 6 DoF IMU and the pressure sensor alongside the optical flow sensor measurements are filtered and fused through a bundled extended Kalman filter estimator that is provided as a built-in solution within the Crazyflie 2.1 firmware [32,66].

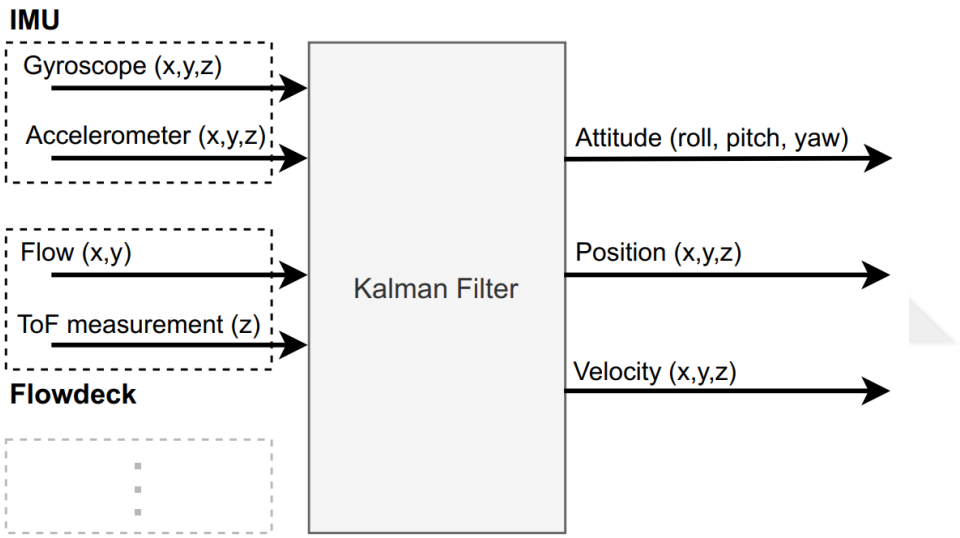


Figure 2.7 : Crazyflie extended Kalman filter.

  In firmware, extended Kalman filter is implemented in `estimator_kalman.c` and `kalman_core.c`, and managed by a supervisor, see `kalman_supervisor.c`, which resets the state estimation if it becomes out of bounds.

Despite the high performance of the complementary and Kalman filters, both measuring noise and measurement errors are unavoidable. We did not run any additional research in this work to enhance the sensing and estimation performances.

The controller block accepts estimated states for the position, velocity, attitude; and setpoints as inputs. Firmware has four layers to control as position, velocity, attitude,

and its rate. As mentioned before, the firmware comes with different control algorithms out of the box as PID controller, INDI controller, and Mellinger controller. Figure 2.8 shows which default controller setting has which control strategy and layer [66].

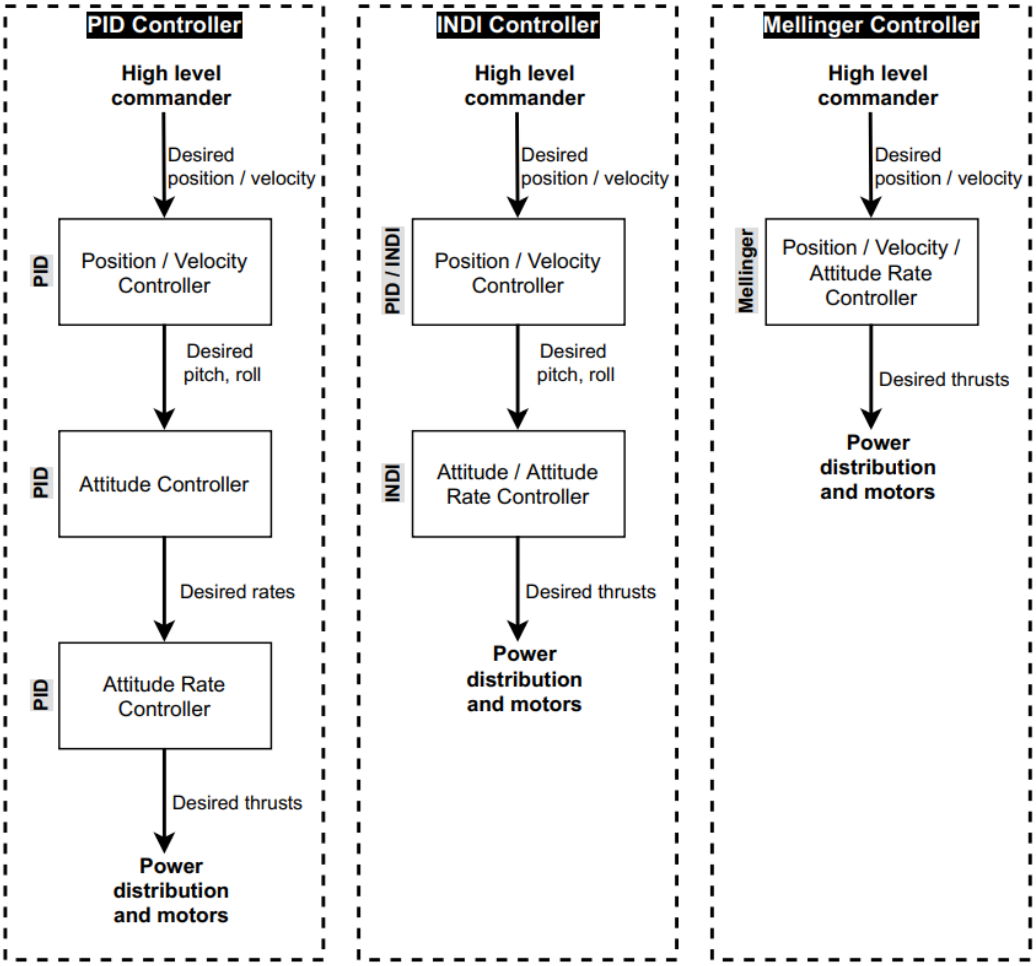


Figure 2.8 : Default controllers of Crazyflie 2.1 firmware.

- *PID controller* setting as seen in Figure 2.9 is the default controller option for Crazyflie 2.1 firmware. While PID controllers run at 100Hz/500Hz, each calculates desired inputs for the next one and gets feedback from the state estimator.

  PID controller is implemented in the following files `position_controller_pid.c` and `attitude_pid_controller.c`.

- *INDI controller* setting has an option for position controller as PID or INDI controller. The attitude and its rate are controlled natively by INDI controller. This contoller runs at 100Hz/500Hz, and is implemented in `controller_indi.c` and `pos`

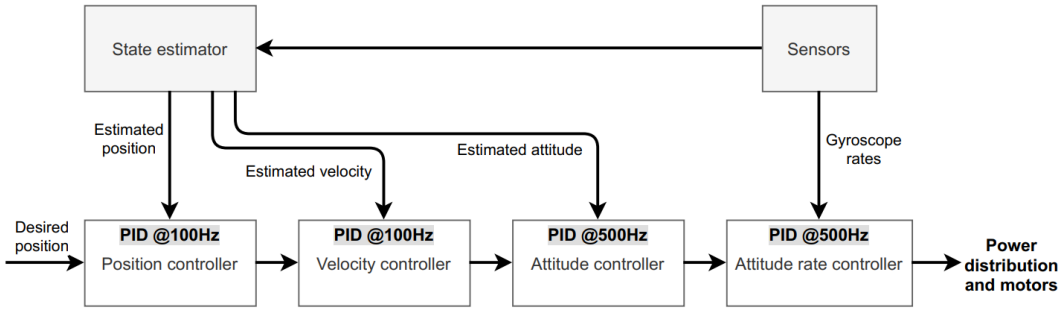


Figure 2.9 : PID controller of Crazyflie 2.1.

`ition_controller_indi.c`. We do not give any mathematical details and underlying theory for this controller to stay in the scope of this study. For further details, we would recommend the following study [68].

- *Mellinger controller* setting has a native position, velocity, and attitude rate controller which is provided in `controller_mellinger.c` and running at 500Hz. Mathematical details for this controller are given in Section 5.

Each controller implementation has its subroutines which have a standard code template to provide full firmware integration. Before going further with controller declarations and implementations, we will explain how firmware determines the current controller. As mentioned before, the current controller can be selected in two ways. While one is to set the controller via a host computer in runtime, the other is to assign it in `Makefile` before the build phase of the firmware as in Figure 2.10. The available controller

```
CONTROLLER ?= PID # one of Any, PID, Mellinger, INDI
```

Figure 2.10 : Default controller selection in Makefile.

options are the entries of an *typedef enum* (as seen in Figure 2.11) which is found in the header file `controller.h`.

Each controller is assigned to an element of an array with `ControllerFcns` type as in Figure 2.12.

Due to zero-based indexing for the entries of `ControllerType` *enum*, the current controller is an element of `controllerFunctions[]`. All controller implementations should provide a common structure to be compatible with the firmware.

```
typedef enum {
  ControllerTypeAny,
  ControllerTypePID,
  ControllerTypeMellinger,
  ControllerTypeINDI,
  ControllerType_COUNT,
} ControllerType;
```

Figure 2.11 : Controller type definitions in firmware.

```
static ControllerFcns controllerFunctions[] = {
  {.init = 0, .test = 0, .update = 0, .name = "None"},//Any
  {.init = controllerPidInit, .test = controllerPidTest,
    .update = controllerPid, .name = "PID"}, //PID
  {.init = controllerMellingerInit,
    .test = controllerMellingerTest,
    .update = controllerMellinger,
    .name = "Mellinger"}, //Mellinger
  {.init = controllerINDIInit, .test = controllerINDITest,
    .update = controllerINDI, .name = "INDI"}, //INDI
};
```

(a)

```
typedef struct {
  void (*init)(void);
  bool (*test)(void);
  void (*update)(control_t *control, const setpoint_t *setpoint,
      const sensorData_t *sensors, const state_t *state,
      const uint32_t tick);
  const char* name;
} ControllerFcns;
```

(b)

Figure 2.12 : Implementation of controller function (a) array (b) struct.

Thus, a controller is able to be initialized, tested, and invoked via functions named as `init`, `test`, `update`, respectively. Firmware has the following function shown in Figure 2.13 that is declared in `controller.h` to call the control signal calculation function of the current controller. Here, the selected controller is assigned to `currentController` variable in `controllerInit` function that is implemented in `controller.c`, and invoked in two locations which are `stabilizerInit` function for once and *STABILIZER* task for every tick – since current controller may be changed in runtime.

2.2.2 Commander and power distribution

While the controller block takes states and setpoints as inputs, the commander module provides setpoints that may be incoming from different sources. Setpoints can be sent

```
void controller(control_t *control, setpoint_t *setpoint,
    const sensorData_t *sensors, const state_t *state,
    const uint32_t tick)
{
    controllerFunctions[currentController].update(control,
            setpoint, sensors, state, tick);
}
```

Figure 2.13 : Current controller caller function.

directly from the *APP* task or a host computer running CFclient (Crazyflie desktop client) or CFlib [62,63]. Additionally, HLC can generate setpoints from commands which are sent by CFlib or *APP* task. The commander block and its relationships are illustrated in Figure 2.14 [66].

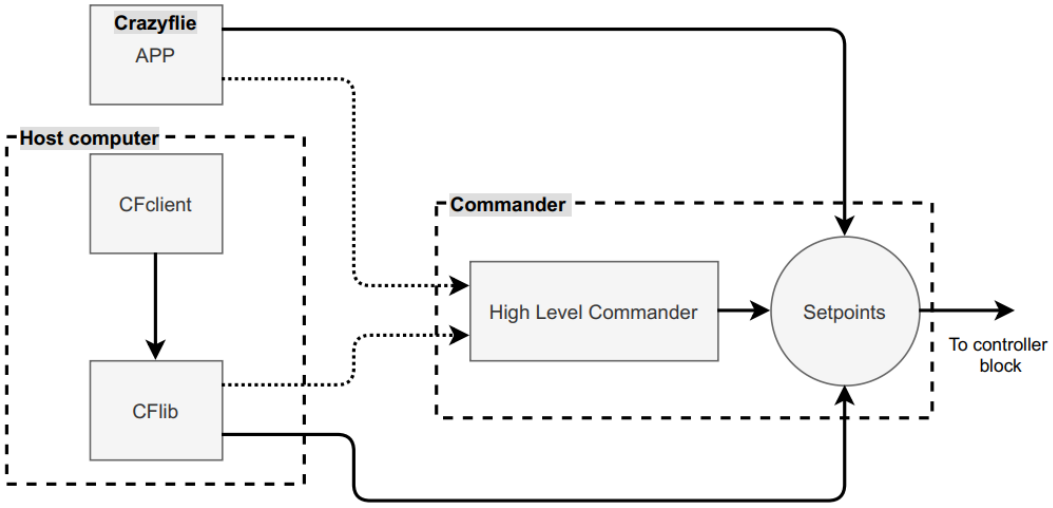


Figure 2.14 : Crazyflie commander block.

During Crazyflie startup, the commander is initialized via invoking `commanderInit()` alongside the stabilizer module in *SYSTEM* task (see Figure 2.4). Commander initializer function is implemented in `commander.c`, and this function starts *CMDHL* task that handles HLC-related operations. Code flow for the commander and HLC startup sequence is illustrated in Figure 2.15.

In *CMDHL* task, `handleCommand` function is called for every received packet. The first byte of the packet represents the command code which is defined in an *enum* as seen in Figure 2.16.

HLC simply handles the setpoints extracted from a predefined trajectory. Both action commands such as `COMMAND_TAKEOFF`, `COMMAND_LAND`, `COMMAND_STOP`, `CO`

Figure 2.15 : Crazyflie commander and HLC startup sequence.

MMAND_GO_TO and trajectory commands COMMAND_START_TRAJECTORY, COMMAND_DEFINE_TRAJECTORY call functions implemented in `planner.c`. These functions are based on 7th-order polynomial calculations. Thus planner generates setpoints, which will be handled by HLC later, by polynomial evaluation for every sampling. We would like to point out that further operations for trajectory planner which is based on piecewise evaluation of 7th-order polynomials are defined in `pptraj.c`.

We can approach the commander block from the stabilizer side also. Setpoint calculation and assignment are done by `crtpCommanderHighLevelGetSetpoint` function (briefly shown in Figure 2.17) for every loop of *STABILIZER* task which is represented in Figure 2.4.

```
enum TrajectoryCommand_e {
  COMMAND_SET_GROUP_MASK = 0,
  COMMAND_TAKEOFF = 1,
  COMMAND_LAND = 2,
  COMMAND_STOP = 3,
  COMMAND_GO_TO = 4,
  COMMAND_START_TRAJECTORY = 5,
  COMMAND_DEFINE_TRAJECTORY = 6,
  // ...
};
```

Figure 2.16 : Trajectory command code definition in firmware.

```
bool crtpCommanderHighLevelGetSetpoint(setpoint_t* setpoint,
        const state_t *state, uint32_t tick)
{
  // STABILIZER runs faster than HLC
  if (!RATE_DO_EXECUTE(RATE_HL_COMMANDER, tick)) {
    return false;
  }
  // System timestamp with micro-second resolution
  // Convert it to seconds as float
  float t = usecTimestamp() / 1e6;
  // Run planner for the specific time value
  // Get evaluated trajectory results
  struct traj_eval ev = plan_current_goal(&planner, t);
  if (plan_is_disabled(&planner) || plan_is_stopped(&planner)) {
    // ...
  }
  else if (is_traj_eval_valid(&ev)) {
    // Assign calculated setpoints to
    // current setpoints
    // ...
  }
  // ...
}
```

Figure 2.17 : crtpCommanderHighLevelGetSetpoint implementation.

The function `crtpCommanderHighLevelGetSetpoint` invokes `plan_current_goal` function from `planner.c` for a certain time in every call. If planner state is `TRAJECTORY_STATE_FLYING`, `plan_eval` function is called by `plan_current_goal`. Similarly, `plan_eval` invokes `piecewise_eval` function which is the core of the piecewise trajectory evaluations from 7th-order polynomials with time stretching and positional shifting. Thus, current setpoints are calculated for the given time and stored to the planner object in order to be used by the controller in the stabilizer loop.

Once control signals (net thrust and roll, pitch, and yaw moments) are calculated, there is a need for an additional operation that is called as power distribution. Power distribution is simply to assign body moments by adding or subtracting to each motor in a proper way according to the formation of the quadcopter body frame. Assigned thrusts are converted to PWM values subsequently. In the end, motors are commanded (see Figure 2.18) and the current stabilizer loop is completed.

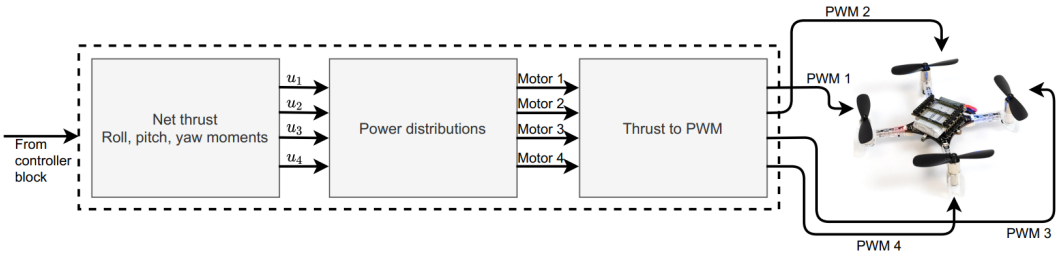


Figure 2.18 : Crazyflie commander block.

3. PRELIMINARIES OF FUZZY LOGIC CONTROLLERS

Human beings have experienced both concrete and abstract processes with their existence. The conclusions we draw from these experiences enable us to understand the world. However, it is often not possible for us to reach an inference only with one experience or with simultaneous experiences, in general terms we acquire knowledge through our inference mechanism which we built on our time-independent experiences [69].

According to Aristotle, the basic structural blocks of logic are simple propositions, and such propositions have only truthy or falsity [69]. With this approach, it can be said that a set, named as the classical set or crisp set, consists of distinct entities, and each entity is an element or member of this set. In classical sets, the universe of discourse which contains the all allowed values for a variable is only divided into two subsets, members and non-members.

Let A be a crisp set, namely a classical set, with a universe of discourse $\mathbb{X}$ defined by the elements $\sigma \subset A$. We can introduce a Membership Function (MF) given below $\mu_A(\sigma)$ that shows the membership grades for each $\sigma \in \mathbb{X}$ [70]:

$$A \Rightarrow \mu_A(\sigma) = \begin{cases} 1, & \text{if } \sigma \in A \\ 0, & \text{if } \sigma \notin A. \end{cases} \tag{3.1}$$

As illustrated in Figure 3.1, we can give a concrete example for a crisp set with a universe of discourse which is a subset of natural numbers. While some elements of this set are prime numbers, the remainings are not. A classical set has always precise divisions.

The fact that an entity definitely belongs to a set or not, or that a proposition is definitely true or false, raises new questions philosophically, but it can also be practically inefficient in the modern world due to uncertainty, ill-posed problems, and processes with incomplete information. For instance, on a road where the speed limit is 80 km/h, although the speed difference between a vehicle traveling at 79.6 km/h and another

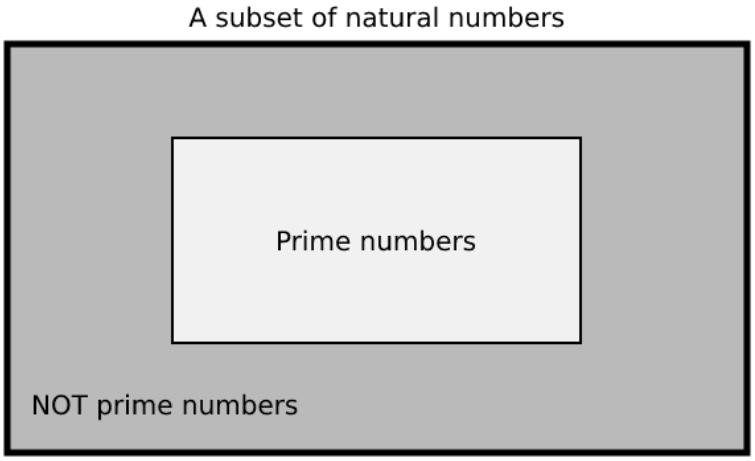


Figure 3.1 : A universe of discourse for an example classical set.

vehicle traveling at 80.4 km/h is quite small, one violates the speed limit while the other one does not. There is a definite line separating the speed limit here, and the speed of the vehicle can only fully belong to one side at a time. However, both vehicles are close to the speed limit at the same level.

In contrast to classical sets, the concept of the Fuzzy Sets (FS), which is an approach in which an entity belongs to a set to a certain degree, was first introduced by Zadeh in 1965 [71]. This first FS concept began to be known as Type-1 Fuzzy Set (T1-FS). Each set mathematically owns each element with a certain percentage, and this directly affects the result of the inference mechanism. Thus, the fuzzy logic decision mechanism produces an output according to the degree of membership rather than the exact values of the inputs, which creates a multi-dimensional output space instead of a one-dimensional output.

3.1 Fuzzy Sets

The concept of FS, which was first introduced by Zadeh, later became the center of controversies and debates about the certainty of membership degrees. Thereupon, the concept of Type-2 Fuzzy Set (T2-FS) was put forward by Zadeh in 1975 [72]. Thus, T2-FS have uncertain membership grades resulting in the Footprint of Uncertainty (FOU) while T1 counterparts have definite degrees of membership. Although it is possible to refer FSs that have higher levels, studies based on T1 and T2 FSs have much more wide coverage in the literature. We will not give any further information about

higher-level FSs to stay in the scope of this study. We will just presently mention T1-FS and T2-FS.

3.1.1 Type-1 fuzzy sets

In fuzzy set notation, let A be a T1-FS on universe of discourse $\mathbb{X}$, and it is defined as follows:

$$A = \{(\sigma, \mu_A(\sigma)) \mid \sigma \in \mathbb{X}\} \tag{3.2}$$

where membership grades $\mu_A(\sigma)$ are in $0 \leq \mu_A(\sigma) \leq 1$. For the sake of simplicity, $\mu_A(\sigma)$ is also called as μ rest of the study. Some of the commonly used MFs with various geometric shapes are defined as triangular, trapezoidal, or Gaussian.

A triangular MF that is shown in Figure 3.2 has a more simple shape and broader usage among all MFs. The definition of a triangular MF is given as follows:

$$\mu_A(\sigma) = \begin{cases} \frac{\sigma - \varsigma_0}{\varsigma_1 - \varsigma_0}, & \varsigma_0 < \sigma \leq \varsigma_1 \\ \frac{\varsigma_2 - \sigma}{\varsigma_2 - \varsigma_1}, & \varsigma_1 < \sigma \leq \varsigma_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where ς_1 is center, ς_0 and ς_2 are left and right supports respectively. It is obvious that left and right supports are not required to have an equal distance to the center. Thus, we can obtain various shapes of triangular MFs with different settings.

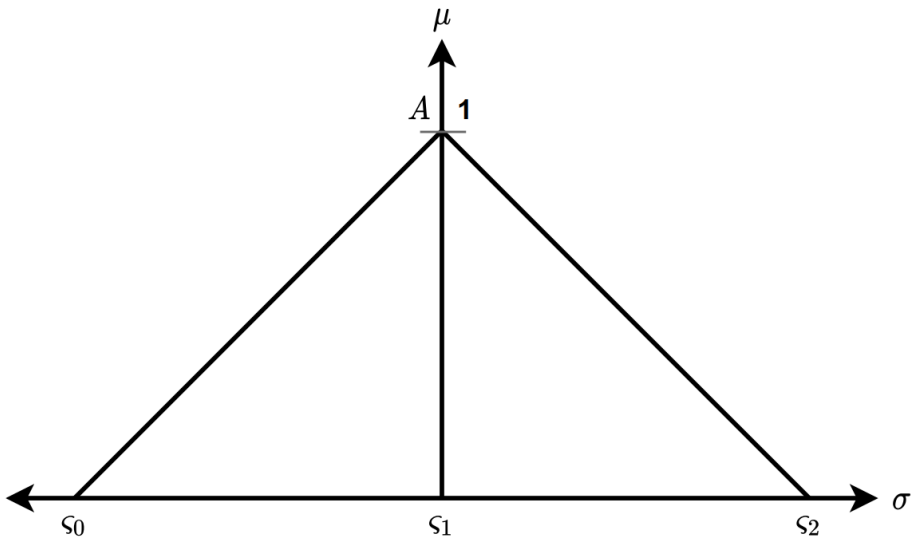


Figure 3.2 : Triangular T1-FS.

A trapezoidal MF, presented in Figure 3.3, has a slightly more complex structure than the triangular MF. Although the trapezoidal MF appears to be a special form of a

triangular MF, it is a completely separate MF. A trapezoidal MF is defined as:

$$\mu_A(\sigma) = \begin{cases} \frac{\sigma-\varsigma_0}{\varsigma_1-\varsigma_0}, & \varsigma_0 \leq \sigma < \varsigma_1 \\ 1, & \varsigma_1 \leq \sigma < \varsigma_2 \\ \frac{\varsigma_3-\sigma}{\varsigma_3-\varsigma_2}, & \varsigma_2 \leq \sigma < \varsigma_3 \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

where ς_0 and ς_3 are left and right supports; ς_1 and ς_2 are left and right centers, respectively. The triangular MF has special configurations as left aligned and right aligned for $\varsigma_0 = \varsigma_3 = -\infty$ and $\varsigma_1 = \varsigma_2 = \infty$, respectively.

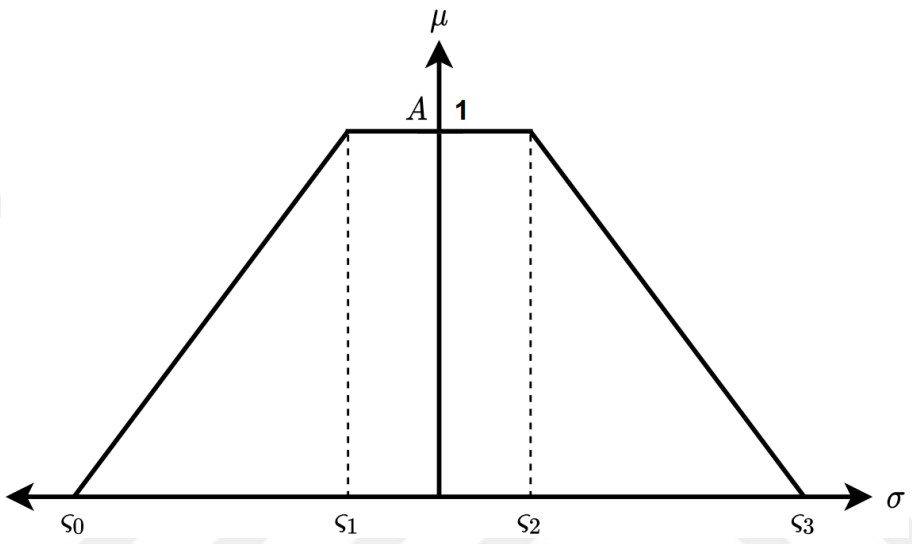


Figure 3.3 : Trapezoidal T1-FS.

A Gaussian MF, as the name suggests, is based on the Gaussian function, and is simply illustrated in Figure 3.4. The mathematical definition of the Gaussian MF is given as follows:

$$\mu_A(\sigma) = e^{-\frac{1}{2}(\frac{\sigma-\varsigma_1}{\zeta})^2} \tag{3.5}$$

where ς_1 and ζ are mean and standard deviation, respectively. Unlike triangular MF and trapezoidal MF, this MF is always symmetric regardless of the values of its parameters.

3.1.2 Type-2 fuzzy sets

In fuzzy set notation, a T2-FS, called as general T2-FS, $\tilde{A}$ is defined as follows [70]:

$$\tilde{A} = \{((\sigma, u), \mu_{\tilde{A}}(\sigma, u)) \mid \sigma \in \mathbb{X}, u \in \mathbb{U} \equiv [0,1]\} \tag{3.6}$$

where $\mathbb{X}$ and $\mathbb{U}$ are the universes for the primary variable σ and secondary variable u, respectively. $\mathbb{U}$ is assumed to be defined as [0, 1] and MF of set $\tilde{A}$, can have values

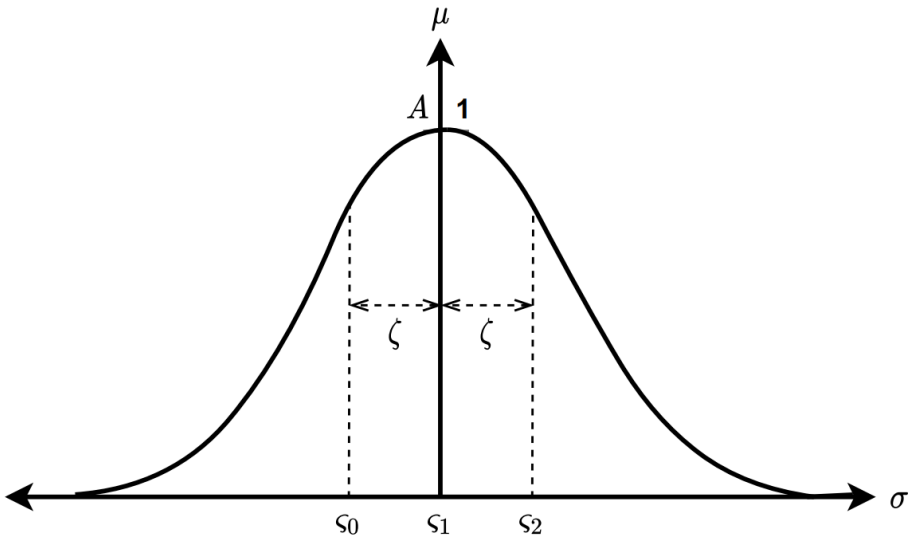


Figure 3.4 : Gaussian T1-FS.

$0 \leq \mu_{\tilde{A}}(\sigma,\mathfrak{u}) \leq 1$. In similar to T1-FS, $\mu_{\tilde{A}}(\sigma,\mathfrak{u})$ is also called as simply μ rest of the study. As mentioned above, T2-FSs have uncertainty in membership grades and MF of a T2-FS has a three-dimensional shape. An example MF of a T2-FS is illustrated in Figure 3.5 with its T1 counterpart [70].

3.1.2.1 Interval type-2 fuzzy sets

A typical general T2-FS when $\mathfrak{u} \in [0,1]$ and $\mu_{\tilde{A}}(\sigma,\mathfrak{u}) = 1$ for $\sigma \in \mathbb{X}$ turns into a special form called as Interval Type-2 Fuzzy Set, IT2-FS, and (3.6) becomes as follows [70]:

$$\tilde{A} = \{((\sigma,\mathfrak{u}), \mu_{\tilde{A}}(\sigma,\mathfrak{u}) = 1) \mid \sigma \in \mathbb{X}, \mathfrak{u} \in \mathbb{U} \equiv [0,1]\} \tag{3.7}$$

Due to its constant 3rd dimension, IT2-MF can be drawn as planar similar to a T1-MF – membership grade for secondary variable always equals to 1. As shown in Figure 3.6, an IT2-FS is defined with an Upper MF (UMF) and a Lower MF (LMF). The area between UMF ($\bar{\mu}_{\tilde{A}}(\sigma)$) and LMF ($\underline{\mu}_{\tilde{A}}(\sigma)$) defines the FOU, which provides an extra degree of freedom to the IT2-FS.

3.2 Type-1 and Interval Type-2 Fuzzy Logic Controllers

Years after the introduction of FS theory, it has started to give a good account of oneself in the control field as FLCs. They are being applied successfully to various processes in several domains with different complexity and linearity levels such as the

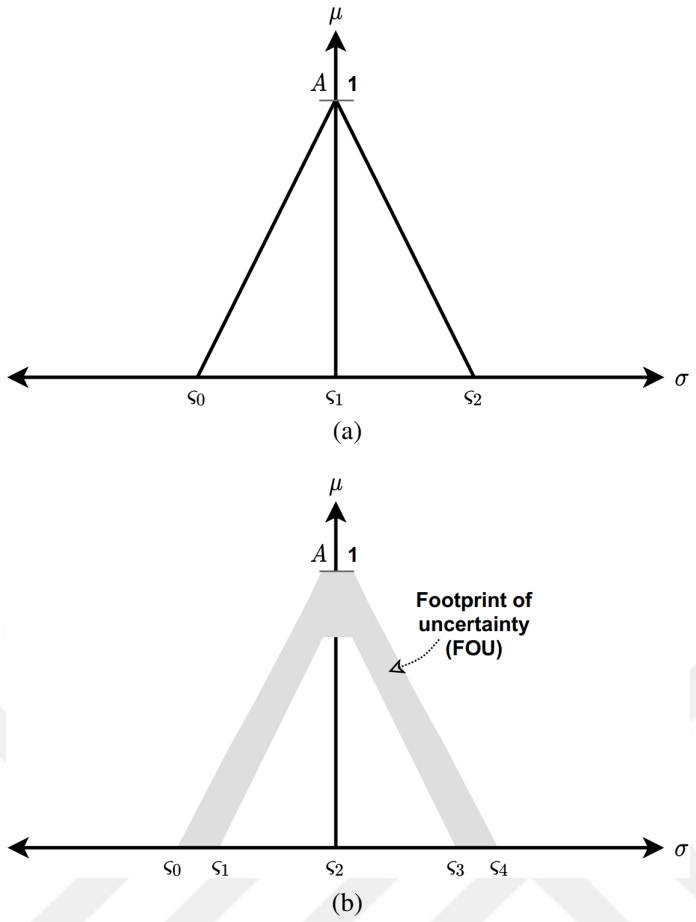
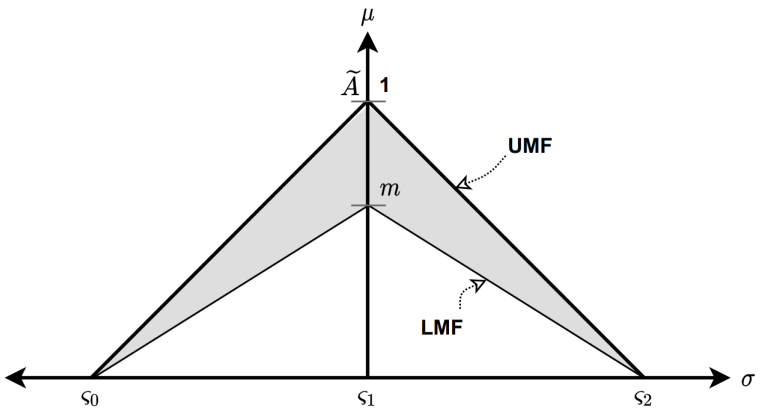


Figure 3.5 : (a) An example T1 MF (b) T2 MF with FOU.

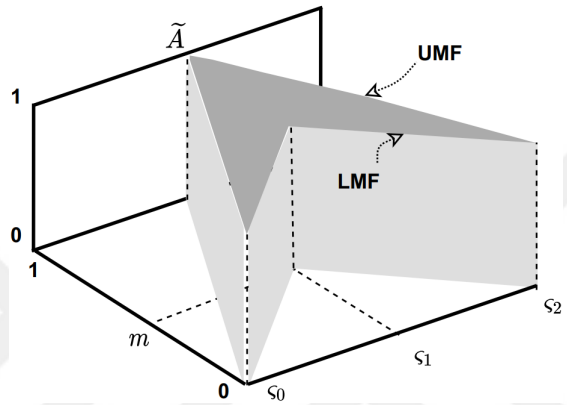
electrical industry, consumer electronics, mechanical and robotic systems, power plants and systems, telecommunications, transportation and automotive systems, chemical processes, and nuclear reactors [73,74].

An FLC simply maps its inputs to outputs by a designated rule table which consists of IF-THEN rules with linguistic variables. For both T1 and IT2 fuzzy systems, the general structure of an FLC is built by following common blocks:

- *Fuzzifier* takes crisp inputs from the real world or simulation environment and computes T1/IT2 input sets.
- *Fuzzy rulebase*, as mentioned above, includes fuzzy rules having fuzzy input-output mappings.
- *Inference mechanism* interprets and combines values of the T1/IT2 fuzzy input sets with fuzzy rules of the rule base, and produces T1/IT2 fuzzy output sets.



(a)



(b)

Figure 3.6 : Triangular IT2 MF with UMF, LMF and FOU (a) 2D view (b) 3D view.

- *Defuzzifier*, as its name very signifies, takes T1/IT2 fuzzy output sets as input and calculates crisp output to real-world or simulation environment

In the real world, input/output signals can have different upper and lower bounds from each other. For FLCs, this can usually lead to some design difficulties. FLCs are designed for normalized domains which are defined for [-1, 1] to keep input(s) and output consistent with each other. To achieve this, input and output scaling factors are utilized for FLCs. While input scaling factors are downscaling input(s) to [-1, 1] interval, output scaling factor (K_u) is upscaling output (φ_o) from [-1, 1] interval as:

$$\begin{aligned} \sigma^j &= \check{\sigma}^j K_j \\ u &= K_u \varphi_o \end{aligned} \tag{3.8}$$

where $j = 1,2,\ldots,J$; J is the total number of inputs, $\breve{\sigma}^j$, σ^j, K_j are non-normalized input, normalized input, and input scaling factor for j^{th} input, respectively, and u shows the denormalized output.

3.2.1 Type-1 fuzzy logic controllers

A typical T1-FLC, illustrated in Figure 3.7, consists of a fuzzifier, rule base, inference mechanism, and defuzzifier as mentioned previously.

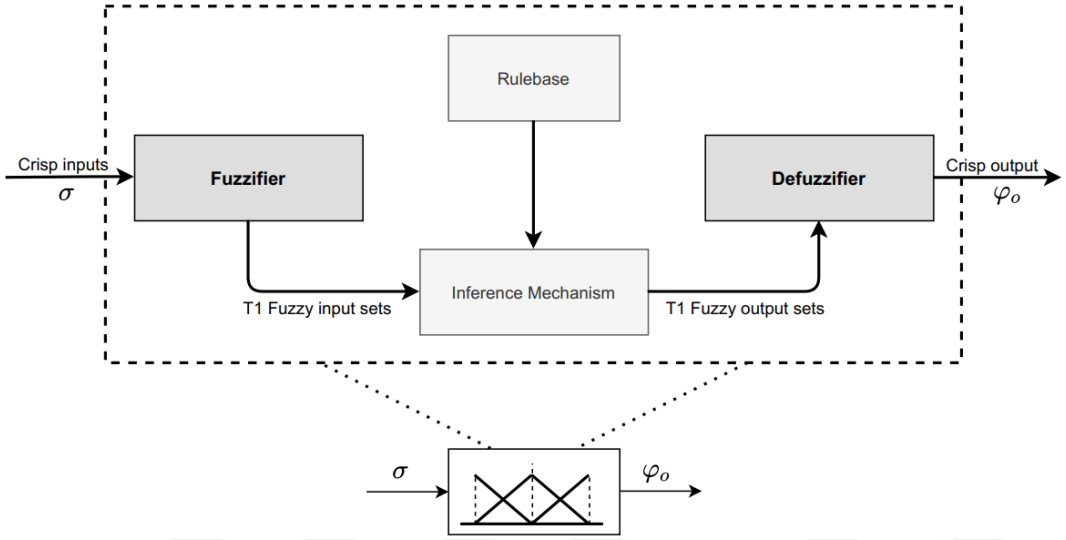


Figure 3.7 : T1 Fuzzy system block diagram.

The generic rule structure of the T1-FLC with N rules is defined as follows:

$$R^i : \text{IF } \sigma^1 \text{ is } A^{1,q} \text{ and } \cdots \text{ and } \sigma^J \text{ is } A^{J,q} \text{ THEN } \varphi_{o_{T1}} \text{ is } \varphi_i \tag{3.9}$$

where $i = 1,2,...,N$, σ^j $(j \in 1,2,\ldots,J)$ is the input variable, $A^{j,q}$ $(q \in 1,2,\ldots,Q)$ is an antecedent MF, Q shows total number of antecedent MFs, $\varphi_{o_{T1}}$ is the output of the T1-FLC and φ_i is the singleton consequent MF of the rule R^i. The final output is calculated by a suitable defuzzification method for the given input(s). If center-of-sets defuzzification method is used, output $\varphi_{o_{T1}}$ is calculated as follows:

$$\varphi_{o_{T1}} = \frac{\sum_{i=1}^{N} f_i \varphi_i}{\sum_{i=1}^{N} f_i} \tag{3.10}$$

where f_i the firing strength of the rule R^i as:

$$f_i = \prod_{j=1}^{J} \mu_{A_{j,q}}(\sigma^j) = \mu_{A_{1,q}}(\sigma^1) \times \cdots \times \mu_{A_{J,q}}(\sigma^J) \tag{3.11}$$

where $\times$ is product t-norm operator, and $\mu_{A_{j,q}}(\sigma^j)$ is the membership degree for input σ^j.

3.2.2 Interval type-2 fuzzy logic controllers

As distinct from their T1 counterparts, IT-2 FLCs shown in Figure 3.8 have also the type reducer block which is required to translate IT2-fuzzy output sets to T1-fuzzy output sets (type-reduced sets) for defuzzifier block since IT2-FSs cannot be mapped to crisp output directly while T1 counterparts can be. As the main defect of the T2-FLC, the type-reducing operation causes computational cost, although type reducer is a requirement for T2-FLC due to its nature.

In literature, there are several studies on type reduction of T2-FSs [70]. One of these strategies is commonly used and well known as Karnik-Mendel algorithms [70,75]–[77]. This method is based on the concept that although the exact defuzzified value of an IT2-FS cannot be calculated directly, it can be reduced to a T1-FS, whose left and right endpoints can be determined. This method iteratively calculates the switch points which are between upper and lower membership grades and is essential for computing type-reduced FSs [75,78].

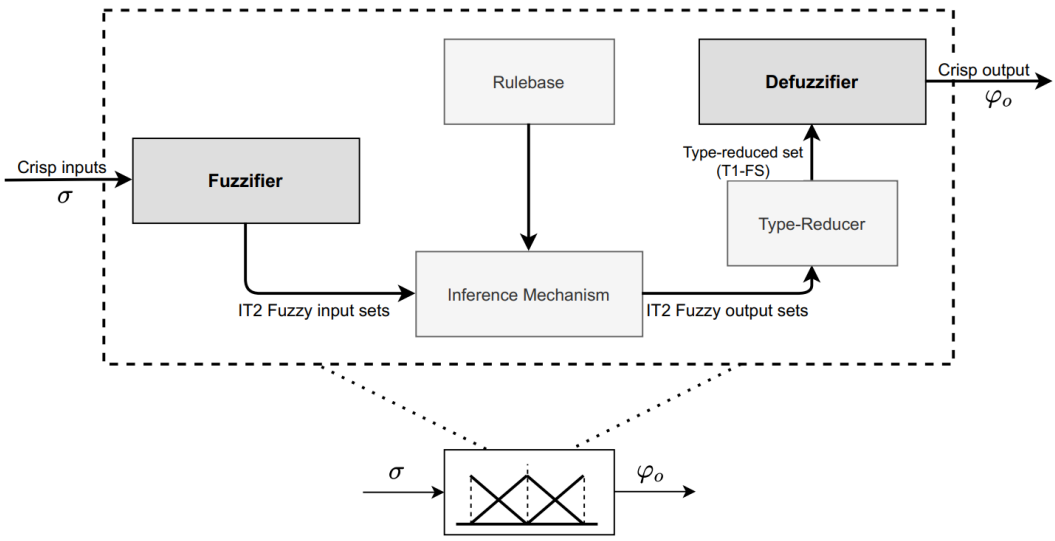


Figure 3.8 : IT2 Fuzzy system block diagram.

The rule structure of the IT2-FLC with N rules, similar to its T1 counterpart, is as:

$$R^i : \text{IF } \sigma^1 \text{ is } \tilde{A}^{1,q} \text{ and } \cdots \text{ and } \sigma^J \text{ is } \tilde{A}^{J,q} \text{ THEN } \varphi_{o_{IT2}} \text{ is } \varphi_i \tag{3.12}$$

where $i = 1,2,...,N$, σ^j ($j \in 1,2,\ldots,J$) is the input variable, $\tilde{A}^{j,q}$ ($q \in 1,2,\ldots,Q$) is an antecedent MF, Q shows total number of antecedent MFs, $\varphi_{o_{IT2}}$ is the output of the IT2-FLC and φ_i is the singleton consequent MF of the rule R^i. The final output is calculated by a suitable defuzzification method for the given input(s) after completion of the type-reducing operation on IT2 Fuzzy output sets. If the Karnik-Mendel algorithm is used for type reduction and center-of-sets defuzzification method is applied, output $\varphi_{o_{IT2}}$ is calculated as follows [70,76]:

$$\varphi_{o_{IT2}} = \frac{\underline{\varphi}_{o_{IT2}} + \bar{\varphi}_{o_{IT2}}}{2} \tag{3.13}$$

where $\underline{\varphi}_{o_{IT2}}$ and $\bar{\varphi}_{o_{IT2}}$ are left and right endpoints of the type-reduced FS, respectively, and calculated as:

$$\underline{\varphi}_{o_{IT2}} = \frac{\sum_{i=1}^{\Omega_L} \bar{f}_i \varphi_i + \sum_{i=\Omega_L+1}^{N} \underline{f}_i \varphi_i}{\sum_{i=1}^{\Omega_L} \bar{f}_i + \sum_{i=\Omega_L+1}^{N} \underline{f}_i} \tag{3.14}$$

$$\bar{\varphi}_{o_{IT2}} = \frac{\sum_{i=1}^{\Omega_R} \underline{f}_i \varphi_i + \sum_{i=\Omega_R+1}^{N} \bar{f}_i \varphi_i}{\sum_{i=1}^{\Omega_R} \underline{f}_i + \sum_{i=\Omega_R+1}^{N} \bar{f}_i} \tag{3.15}$$

where Ω_L and Ω_R are left and right switching points, respectively. For the i^{th} rule R^i, lower and upper firing strengths, $\underline{f}_i$ and $\bar{f}_i$ respectively, are calculated as follows:

$$\underline{f}_i = \prod_{j=1}^{J} \underline{\mu}_{\tilde{A}_{j,q}}(\sigma^j) = \underline{\mu}_{\tilde{A}_{1,q}}(\sigma^1) \times \cdots \times \underline{\mu}_{\tilde{A}_{J,q}}(\sigma^J) \tag{3.16}$$

$$\bar{f}_i = \prod_{j=1}^{J} \bar{\mu}_{\tilde{A}_{j,q}}(\sigma^j) = \bar{\mu}_{\tilde{A}_{1,q}}(\sigma^1) \times \cdots \times \bar{\mu}_{\tilde{A}_{J,q}}(\sigma^J) \tag{3.17}$$

where corresponding LMF and UMF have the membership degrees $\underline{\mu}_{\tilde{A}_{j,q}}(\sigma^j)$ and $\bar{\mu}_{\tilde{A}_{j,q}}(\sigma^j)$, respectively. As a design approach of IT2-FLCs, UMFs, and LMFs can be defined with respect to their T1 baseline [79]. Thus, an IT2-FLC can be designed with the design parameter $m_{j,q}$ and a T1 baseline as:

$$\bar{\mu}_{\tilde{A}_{j,q}}(\sigma^j) = \mu_{A_{j,q}}(\sigma^j) \tag{3.18}$$

$$\begin{aligned} \underline{\mu}_{\tilde{A}_{j,q}}(\sigma^j) &= \bar{\mu}_{\tilde{A}_{j,q}}(\sigma^j) m_{j,q} \\ &= \mu_{A_{j,q}}(\sigma^j) m_{j,q} \end{aligned} \tag{3.19}$$

where $m_{j,q}$ is the height of the LMF of IT2-FS $\tilde{A}_{j,q}$ as shown in Figure 3.6. The parameter $m_{j,q}$ allows defining directly the size of FOU, and so it is known also as FOU design parameter [37]. The value of $m_{j,q}$ inversely affects the size of FOU.

3.3 Single Input and Double Input Fuzzy Logic Controllers

FLCs can be classified as their number of inputs as Single Input FLCs (SFLCs) and Double Input FLCs (DFLCs). While error signal (e) is employed as input for SFLCs, e and its derivative ($\dot{e}$) are utilized for DFLCs inputs, and control signal (u) is output for both. Additionally, there are studies focusing to transform DFLCs to SFLCs by combining their inputs with various methods, e.g. weighting, to use SFLC structures [80].

For SFLC structure, normalized input (σ) and control signal (u) are calculated as follows:

$$\begin{aligned} \sigma &= eK_e \\ u &= K_u \varphi_o \end{aligned} \tag{3.20}$$

where K_e is input scaling factor, K_u is output scaling factor and φ_o is SFLC normalized output. The scaling factors are determined via the lower and upper bounds of the signals. An application of SFLC in a closed-loop control system is illustrated in Figure 3.9.

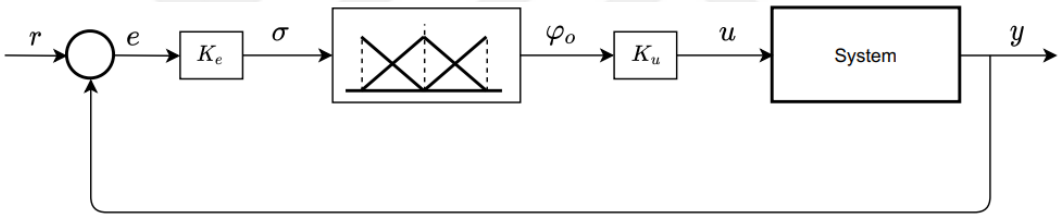


Figure 3.9 : SFLC control system.

For DFLC structure, normalized inputs (σ^1, σ^2) and control signal (u) are calculated as follows:

$$\begin{aligned} \sigma^1 &= eK_e \\ \sigma^2 &= \dot{e}K_{de} \\ u &= K_u \varphi_o \end{aligned} \tag{3.21}$$

where K_{de} is input scaling factor for $\dot{e}$. An application of DFLC in a closed-loop control system is illustrated in Figure 3.10.

Relations defined in (3.20) and (3.21) are priorly given in (3.8) with general notation. For the sake of simplicity, we drop some superscripts, e.g. j, and substitute some variables, e.g. $\breve{\sigma}^1$ with e and $\breve{\sigma}^2$ with $\dot{e}$, when required. In this study, the simple form is preferred to provide readers with a light notation to make it easy to follow.

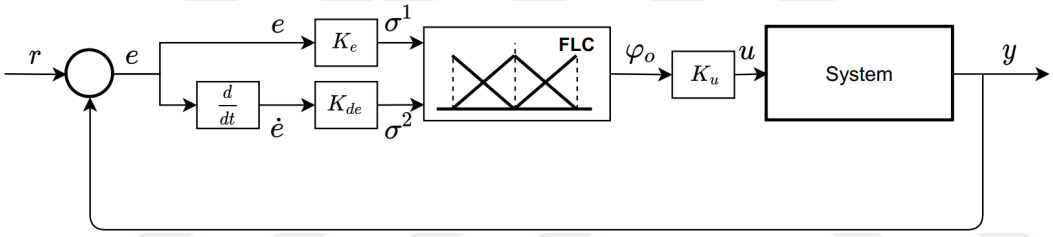


Figure 3.10 : DFLC control system.

4. DIFFERENTIAL FLATNESS FOR THE CRAZYFLIE 2.1

In this section, we define differential flatness and provide the dynamical model of the nano quadcopter Crazyflie 2.1 with its differential flat representation.

4.1 Differential Flatness

The differential flatness paradigm was first introduced by [81] in the context of linear algebra [82]. Differentially flat systems have the controllability property, and they are equivalent to those linearized systems which have endogenous feedback [81,82].

Let us define a nonlinear system with an output $\xi(t) \in \mathbb{R}^m$ as below:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{4.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is state and $\mathbf{u}(t) \in \mathbb{R}^m$ $(m \leq n)$ is input. This system is called as differentially flat if there exists a flat system output vector ξ such that [81,83]:

- The system output ξ is a function of state $\mathbf{x}$, input $\mathbf{u}$ and the finite number (δ) of time derivatives of $\mathbf{u}$:

$$\xi = \Phi(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(\delta)}). \tag{4.2}$$

- The state $\mathbf{x}$ and input $\mathbf{u}$ can be written as functions of the output ξ and its finite number (w) of time derivatives:

$$\begin{aligned} \mathbf{x} &= \Lambda(\xi, \dot{\xi}, \ldots, \xi^{(w)}) \\ \mathbf{u} &= \Gamma(\xi, \dot{\xi}, \ldots, \xi^{(w)}). \end{aligned} \tag{4.3}$$

- The system output ξ and its derivatives with respect to time $\xi, \dot{\xi}, \ldots, \xi^{(w)}$ are independent.

The flat output ξ actually is a fictitious term that is used to express system input $\mathbf{u}(t)$ and state $\mathbf{x}(t)$ with its derivatives. If a nonlinear system given in (4.1) is linearizable via endogenous feedback is said to be differentially flat since flat output ξ expressed in (4.2)

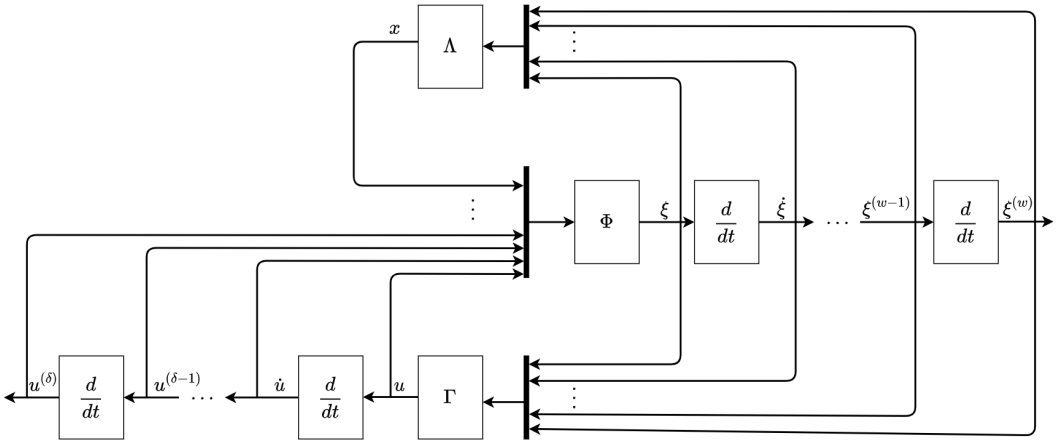


Figure 4.1 : Differential flatness block diagram.

exists only with such special type of feedback [81]. The block diagram representation of a differential flat system is illustrated in Figure 4.1 [84].

The idea of expressing $\mathbf{u}(t)$ and $\mathbf{x}(t)$ directly without needing any integration by using (4.2) and (4.3) originates from studies [85,86] on under-determined systems which the number of system equations is less than system unknowns [81]. Therefore this idea makes the differential flatness property a suitable concept for trajectory generation and tracking as the trajectory can be planned for flat outputs, and thus the output space can be mapped to system inputs [82].

To explain the trajectory tracking application of a differentially flat system, the state equation is written below with boundary conditions:

$$\begin{aligned} \mathbf{x}(t_0) &= \Lambda\big(\xi(0), \dot{\xi}(0), \ldots, \xi^{(w)}(0)\big) = \mathbf{x_0} \\ \mathbf{x}(t_f) &= \Lambda\big(\xi(t_f), \dot{\xi}(t_f), \ldots, \xi^{(w)}(t_f)\big) = \mathbf{x_f} \end{aligned} \tag{4.4}$$

where $t_0 = 0$ and t_f are initial and final times, and $\mathbf{x_0}$ and $\mathbf{x_f}$ are state vectors at t_0 and t_f, respectively. A trajectory, designed for ξ, that satisfies these conditions is said to be a feasible trajectory for the system given in (4.1). Flat output ξ can be written as:

$$\xi(t) = \sum_{i=1}^{N} \varepsilon_i \rho_i(t) \tag{4.5}$$

where $i = 1, \ldots, N$, $\rho_i(t)$ are basis functions and $\varepsilon_i \in \mathbb{R}$ are their coefficients. Thus, the flat output is projected to space which is spanned by its $\rho_i(t)$ [87,88]. Similar to (4.5),

the time derivatives of the flat output can be written as a set of basis functions:

$$\begin{aligned} \dot{\xi}(t) &= \sum_{i=1}^{N} \varepsilon_i \dot{\rho}_i(t) \\ &\vdots \\ \xi^{(w)}(t) &= \sum_{i=1}^{N} \varepsilon_i \rho_i^{(w)}(t). \end{aligned} \tag{4.6}$$

By using (4.4), (4.5) and (4.6), the equation set can be written as matrix form:

$$\begin{bmatrix} \rho_1(0) & \rho_2(0) & \cdots & \rho_N(0) \\ \dot{\rho}_1(0) & \dot{\rho}_2(0) & \cdots & \dot{\rho}_N(0) \\ \vdots & \vdots & & \vdots \\ \rho_1^{(w)}(0) & \rho_2^{(w)}(0) & \cdots & \rho_N^{(w)}(0) \\ \rho_1(t_f) & \rho_2(t_f) & \cdots & \rho_N(t_f) \\ \dot{\rho}_1(t_f) & \dot{\rho}_2(t_f) & \cdots & \dot{\rho}_N(t_f) \\ \vdots & \vdots & & \vdots \\ \rho_1^{(w)}(t_f) & \rho_2^{(w)}(t_f) & \cdots & \rho_N^{(w)}(t_f) \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{bmatrix} = \begin{bmatrix} \xi(0) \\ \dot{\xi}(0) \\ \vdots \\ \xi^{(w)}(0) \\ \xi(t_f) \\ \dot{\xi}(t_f) \\ \vdots \\ \xi^{(w)}(t_f) \end{bmatrix} \tag{4.7}$$

where if the matrix of basis functions is full rank, there is a solution for ε_i coefficients for the trajectory generation.

4.2 Dynamic Model

Let us define two main coordinate systems, the world frame ($\mathscr{W}$) and the body frame ($\mathscr{B}$) with an additional intermediate frame ($\mathscr{C}$), as shown in Figure 4.2. The frame $\mathscr{B}$ is fixed to the Center of Mass (CoM) of Crazyflie 2.1 while the frame $\mathscr{C}$ is used to define the yaw angle (ψ) explicitly between two frames. The rotation matrix (R) describing the orientation from $\mathscr{B}$ to $\mathscr{W}$, by using Z-X-Y Euler angle convention, is defined as:

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{4.8}$$

where $c(.)$ and $s(.)$ are cosine and sine functions, respectively. The rotation matrix from $\mathscr{B}$ to $\mathscr{W}$ (${}^{W}R_B$) consists of two main parts, namely the rotation from $\mathscr{B}$ to $\mathscr{C}$ (${}^{C}R_B$), to transform the roll (φ) and pitch (θ), and the rotation from $\mathscr{C}$ to $\mathscr{W}$ (${}^{W}R_C$), to transform ψ. Therefore, we can define ${}^{W}R_B$ as follows:

$${}^{W}R_B = {}^{W}R_C \, {}^{C}R_B \tag{4.9}$$

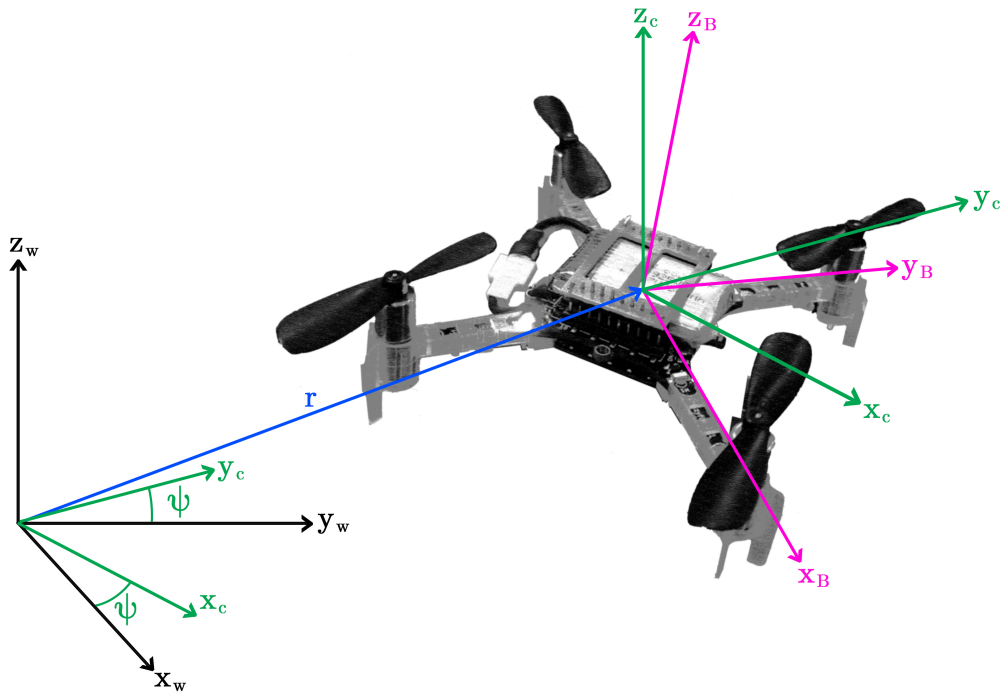


Figure 4.2 : Illustration of the reference frames of Crazyflie 2.1.

Each rotor of the quadcopter generates a force (F_i) and a moment (M_i) by rotating with a specific angular speed (ω_i), for $i = 1, ..., 4$, as follows:

$$F_i = k_F \omega_i^2, \quad M_i = k_M \omega_i^2 \tag{4.10}$$

where k_F is the force coefficient and k_M is the moment coefficient. A complex interaction between the motor controller, the physics of the propeller, and the motor determines the precise relationship between the desired and measured motor speed, ω_i^{des} and ω_i respectively. Not only the rotor speed but also the increment or decrement of it determine the actual performance. For the sake of simplicity, a first-order differential equation can show the relationship between actual and desired rotor speeds as below [89]:

$$\dot{\omega}_i = k_m(\omega_i^{des} - \omega_i) \tag{4.11}$$

where k_m is motor gain. The net thrust resulting in Crazyflie 2.1 (u_1) and the generated roll, pitch, and yaw moments (u_2, u_3, and u_4) are defined as:

$$\mathbf{u} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{4.12}$$

where $\mathbf{u} = [u_1, u_2, u_3, u_4]^\top$ and L is the distance between the center of the rotor and CoM of the Crazyflie 2.1.

The forces acting on the system are the gravity with the direction of $-z_W$ and the produced thrust u_1 with the direction of z_B. Thus, the net force results in the acceleration of the CoM is:

$$m\ddot{\mathbf{r}} = -mg\mathbf{z_W} + u_1\mathbf{z_B}, \tag{4.13}$$

where $\mathbf{r}$ is the position vector of CoM in $\mathscr{W}$ while $\ddot{\mathbf{r}}$ is the corresponding acceleration vector. The rotors also produce body moments (u_2, u_3, and u_4) resulting in the angular acceleration of the body that is defined via Coriolis' equation. The corresponding motion equations are:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \mathscr{I}\,\dot{\boldsymbol{\omega}}_{\mathscr{B}\mathscr{W}} + \boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times \mathscr{I}\,\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}}, \tag{4.14}$$

where $\mathscr{I}$ represents the moment of inertia matrix in frame $\mathscr{B}$. $\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}}$ is the angular velocity vector of the frame $\mathscr{B}$ in frame $\mathscr{W}$ with components of p, q, r (roll, pitch and yaw rates respectively):

$$\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} = p\mathbf{x_B} + q\mathbf{y_B} + r\mathbf{z_B}. \tag{4.15}$$

Body frame angular velocities, p, q and r correlate with derivatives of roll, pitch, and yaw angles, $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ respectively:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{4.16}$$

The final system state of Crazyflie 2.1 ($\mathbf{x}$) is defined as follows:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^\top. \tag{4.17}$$

In this representation, the orientation is defined with the Euler angles parameterization (ϕ, θ, ψ) with angular velocities (p, q, r). It is also possible to represent it via the rotation matrix (${}^W R_B$) and the angular velocity ($\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}}$) without the parameterization [51]. Differential flatness-based characterization can be performed regardless of which representation format is selected.

4.3 Differential Flatness Based Characterization

Here, to develop the proposed differential flatness-based SFLC, we show that the quadcopter dynamics exhibit the differential flatness property. We follow the same approach as in [51] and thus assume that the system can track the generated smooth trajectories in flat output space with appropriate limits of derivatives. In this context, we define the position and yaw angle of Crazyflie 2.1 as the four flat outputs as follows [51]:

$$\xi = \left[x, y, z, \psi\right] \tag{4.18}$$

Now, we are going to redefine the position (x, y, z), orientation $({}^{W}R_{B})$, angular velocity $(\omega_{\mathscr{B}\mathscr{W}})$ and angular acceleration $(\alpha_{\mathscr{B}\mathscr{W}})$ in terms of the flat outputs (ξ) and those derivatives $(\dot{\xi}, \ddot{\xi}, ...)$ to show a relationship between $\xi, \dot{\xi}, \ddot{\xi}, ...$ and $\mathbf{u}$.

4.3.1 Characterizing the position and orientation of the quadcopter

The position can be straightforwardly characterized via the flat output vector ξ as its first three terms define the position ($\xi_1 = x$, $\xi_2 = y$, $\xi_3 = z$). As a consequence, the first derivative of ξ is used to define velocity ($\dot{\xi}_1$, $\dot{\xi}_2$, $\dot{\xi}_3$), while the acceleration definition by the flat outputs is as follows [51]:

$$\mathbf{t} = \left[\ddot{\xi}_1, \ddot{\xi}_2, \ddot{\xi}_3 + g\right]^{\top} \tag{4.19}$$

where $\mathbf{t}$ is the acceleration vector, including the gravity for the z axis.

To determine the orientation of the quadcopter via the world frame rotation matrix $({}^{W}R_{B})$, we define the body frame z axis by normalizing it via $\mathbf{t}$ as follows [51]:

$$\mathbf{z}_B = \frac{\mathbf{t}}{\|\mathbf{t}\|} \tag{4.20}$$

Now, we define the intermediate frame x axis $(\mathbf{x}_C)$ via $\xi_4 = \psi$ as:

$$\mathbf{x}_C = \left[cos\xi_4, sin\xi_4, 0\right]^{\top} \tag{4.21}$$

and then, $\mathbf{x}_B$ and $\mathbf{y}_B$ are defined by using body frame z axis $(\mathbf{z}_B)$ and $\mathbf{x}_C$ as [51]:

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{\|\mathbf{z}_B \times \mathbf{x}_C\|}, \quad \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \tag{4.22}$$

We can now characterize the world frame rotation matrix $({}^{W}R_{B})$ via the flat outputs and their derivatives as follows:

$${}^{W}R_{B} = \begin{bmatrix} \mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B \end{bmatrix} \tag{4.23}$$

4.3.2 Characterizing the angular velocity of the quadcopter

As we have done in the preceding part, we characterize the angular velocity vector also with respect to $\xi, \dot{\xi}, \ddot{\xi}, ...$ Before taking the derivative of (4.13), the net thrust and its derivative are written as follows via (4.19) and (4.20):

$$u_1 = m\|\mathbf{t}\|, \quad \dot{u}_1 = \mathbf{z}_B \cdot m\dot{\mathbf{a}}. \tag{4.24}$$

Here, $\dot{\mathbf{a}}$ is the third derivative of $\mathbf{r}$. The derivative of (4.13) is then as:

$$m\dot{\mathbf{a}} = \dot{u}_1\mathbf{z}_B + \omega_{\mathscr{B}\mathscr{W}} \times u_1\mathbf{z}_B \tag{4.25}$$

Now, by using (4.24) and (4.25), $\omega_{\mathscr{B}\mathscr{W}} \times \mathbf{z}_B$ is defined with m, u_1, $\mathbf{z}_B$ and $\dot{\mathbf{a}}$:

$$\mathbf{h}_\omega = \omega_{\mathscr{B}\mathscr{W}} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\mathbf{a}} - (\mathbf{z}_B \cdot \dot{\mathbf{a}})\mathbf{z}_B). \tag{4.26}$$

Here, we can define $\mathbf{h}_\omega$ as a composition of p and q in the body frame as follows:

$$\mathbf{h}_\omega = -p\mathbf{y}_B + q\mathbf{x}_B. \tag{4.27}$$

Then, via (4.26) and (4.27), we define p and q as:

$$p = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \quad q = \mathbf{h}_\omega \cdot \mathbf{x}_B, \tag{4.28}$$

while r is obtained via a simple transformation. In this context, we define

$$\omega_{\mathscr{B}\mathscr{W}} = \omega_{\mathscr{B}\mathscr{C}} + \omega_{\mathscr{C}\mathscr{W}} \tag{4.29}$$

and extract r by taking account that no $\mathbf{z}_B$ component is present in $\omega_{\mathscr{B}\mathscr{C}}$, as:

$$r = \omega_{\mathscr{C}\mathscr{W}} \cdot \mathbf{z}_B = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B \tag{4.30}$$

Thus, we show that the angular velocity of the quadcopter can be written in terms of flat output and its time derivatives by resulting (4.28) and (4.30).

4.3.3 Characterizing the angular acceleration of the quadcopter

Demonstrating the angular acceleration vector as a function of flat outputs can be done similarly via the angular velocities. As the initial step, the first derivative of (4.25) is taken as follows:

$$\begin{aligned} m\ddot{\mathbf{a}} &= \ddot{u}_1\mathbf{z}_B + 2\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times \dot{u}_1\mathbf{z}_B \\ &+ \boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times \boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times u_1\mathbf{z}_B \\ &+ \boldsymbol{\alpha}_{\mathscr{B}\mathscr{W}} \times u_1\mathbf{z}_B, \end{aligned} \tag{4.31}$$

and can be projected along $\mathbf{z}_B$ axis as:

$$\ddot{u}_1 = \mathbf{z}_B \cdot m\ddot{\mathbf{a}} - \mathbf{z}_B \cdot (\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times \boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times u_1\mathbf{z}_B). \tag{4.32}$$

Let us define an orthogonal vector $\mathbf{h}_\alpha$ and compute the components of $\boldsymbol{\alpha}_{\mathscr{B}\mathscr{W}}$ along $\mathbf{x}_B$ and $\mathbf{y}_B$ axes as given below:

$$\begin{aligned} \mathbf{h}_\alpha &= \boldsymbol{\alpha}_{\mathscr{B}\mathscr{W}} \times \mathbf{z}_B \\ \dot{p} &= -\mathbf{h}_\alpha \cdot \mathbf{y}_B \\ \dot{q} &= \mathbf{h}_\alpha \cdot \mathbf{x}_B. \end{aligned} \tag{4.33}$$

Before finding $\mathbf{z}_B$ component of the $\boldsymbol{\alpha}_{\mathscr{B}\mathscr{W}}$, we can write (4.29) in the form given below:

$$\boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} = \begin{bmatrix} \mathbf{x}_C & \mathbf{y}_B & \mathbf{z}_W \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{4.34}$$

then deriving this equation, we get the following form:

$$\begin{aligned} {}^{W}R_B\left(\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) &= \boldsymbol{\omega}_{\mathscr{C}\mathscr{W}} \times \dot{\phi}\mathbf{x}_C \\ &+ \boldsymbol{\omega}_{\mathscr{B}\mathscr{W}} \times \dot{\theta}\mathbf{y}_B \\ &+ \begin{bmatrix} \mathbf{x}_C & \mathbf{y}_B & \mathbf{z}_W \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}. \end{aligned} \tag{4.35}$$

We can write the resulting equation as follows:

$$\Upsilon \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \boldsymbol{\iota} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \tag{4.36}$$

where Υ and $\boldsymbol{\iota}$ are a matrix and a vector with known values, respectively. For chosen value of the $\ddot{\psi}$, we are able to calculate $\dot{r}$ from (4.36) with the values of $\dot{p}$ and $\dot{q}$ which are calculated via (4.33).

4.3.4 Characterizing the inputs of the quadcopter

It is in clear view that the net thrust u_1 is the function of flat output ξ and its derivatives in (4.19) and (4.24). For the body moments u_2, u_3 and u_4, (4.14) can be written as following form to show body moments are also the functions of flat output and its derivatives [51]:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \mathscr{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \mathscr{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.37}$$

where angular velocities p, q, r and accelerations $\dot{p}$, $\dot{q}$, $\dot{r}$ have been characterized for differential flatness priorly.

5. DIFFERENTIAL FLATNESS-BASED T1 AND IT2 FUZZY CONTROL SYSTEM

Here, we present the proposed differential flatness-based SFLC structure illustrated in Figure 5.1 [90]. The control inputs u_2, u_3, and u_4 are generated by three SFLCs while u_1 is generated via a crisp controller for the sake of simplicity.

Let us firstly define a specified trajectory $\xi_T(t) = [\mathbf{r}_T(t), \psi_T(t)]^\top$ where $\mathbf{r}_T(t)$ is the specified position vector and $\psi_T(t)$ is the specified yaw angle. The resulting position error ($\mathbf{e}_R$) and velocity error ($\mathbf{e}_v$) are defined as:

$$\mathbf{e}_p = \mathbf{r} - \mathbf{r}_T, \quad \mathbf{e}_v = \dot{\mathbf{r}} - \dot{\mathbf{r}}_T \tag{5.1}$$

The desired thrust vector ($\mathbf{F}_{des}$) is calculated as follows [51]:

$$\mathbf{F}_{des} = -K_p\mathbf{e}_p - K_v\mathbf{e}_v + mg\mathbf{z}_W + m\ddot{\mathbf{r}}_T \tag{5.2}$$

where K_p and K_v are positive definite gain matrices. As the net body force occurs only in the direction of the body frame z axis, the magnitude of the desired thrust can be defined via the control input u_1 as follows:

$$u_1 = \mathbf{F}_{des} \cdot \mathbf{z}_B \tag{5.3}$$

The control inputs u_2, u_3 and u_4 are generated by three SFLCs as shown in Figure 5.1. To compute u_2, u_3 and u_4, we use the differential flatness characterization to define the angular velocity error $\mathbf{e}_\omega = [e_{\omega_1}, e_{\omega_2}, e_{\omega_3}]^\top$ and the orientation error $\mathbf{e}_R = [e_{R_1}, e_{R_2}, e_{R_3}]^\top$. To accomplish this goal, we first need to define the desired rotation matrix R_{des}. The desired $\mathbf{z}_B$ ($\mathbf{z}_{B,des}$) in the body frame is straightforwardly written as follows [51]:

$$\mathbf{z}_{B,des} = \frac{\mathbf{F}_{des}}{\|\mathbf{F}_{des}\|}. \tag{5.4}$$

Then, $\mathbf{z}_{B,des}$ is used to define the third column of R_{des}:

$$R_{des}\mathbf{e}_3 = \mathbf{z}_{B,des} \tag{5.5}$$

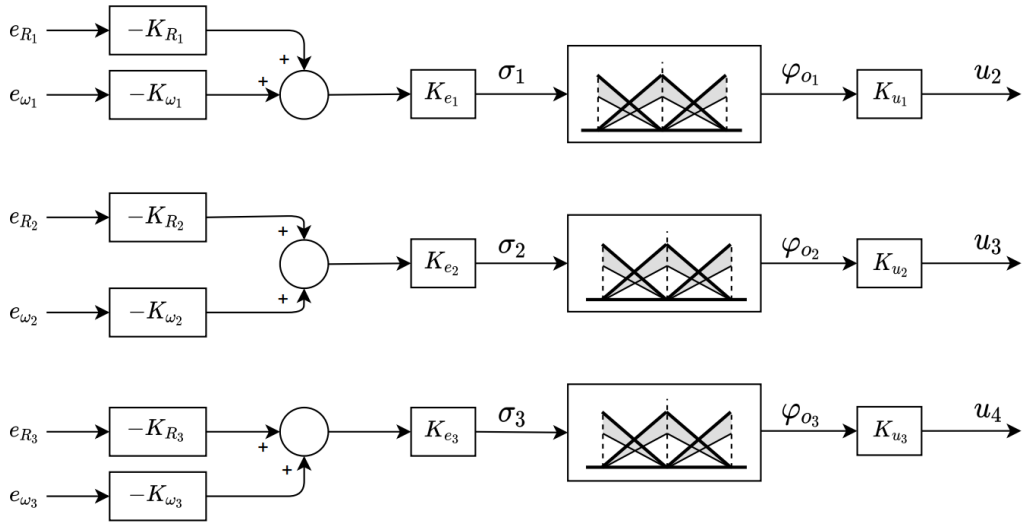


Figure 5.1 : Differential flatness-based fuzzy control system.

where $\mathbf{e}_3 = [0,0,1]^\top$. Then, as it has been done in (4.21) and (4.22), the desired $\mathbf{x}_B$ ($\mathbf{x}_{B,des}$) and desired $\mathbf{y}_B$ ($\mathbf{y}_{B,des}$) are defined via the following relationships [51]:

$$\begin{aligned} \mathbf{x}_{C,des} &= [cos\psi_T, sin\psi_T, 0]^\top \\ \mathbf{y}_{B,des} &= \frac{\mathbf{z}_{B,des} \times \mathbf{x}_{C,des}}{\|\mathbf{z}_{B,des} \times \mathbf{x}_{C,des}\|} \\ \mathbf{x}_{B,des} &= \mathbf{y}_{B,des} \times \mathbf{z}_{B,des} \end{aligned} \tag{5.6}$$

By combining (5.5) and (5.6), we define R_{des} as follows:

$$R_{des} = \begin{bmatrix} \mathbf{x}_{B,des} & \mathbf{y}_{B,des} & \mathbf{z}_{B,des} \end{bmatrix}. \tag{5.7}$$

The orientation error $\mathbf{e}_R$ and angular velocity error $\mathbf{e}_\omega$ are then defined as rotation errors via R_{des} [51]. $\mathbf{e}_R$ is computed as follows:

$$\mathbf{e}_R = \frac{1}{2}(R_{des}^\top {}^W R_B - {}^W R_B^\top R_{des})^\vee \tag{5.8}$$

where $\vee$ shows *vee operator*, $\vee : \mathsf{SO(3)} \to \mathbb{R}^3$ [91]. Similarly, $\mathbf{e}_\omega$ is defined as:

$$\mathbf{e}_\omega = {}^B[\omega_{\mathscr{B}\mathscr{W}}] - {}^B[\omega_{\mathscr{B}\mathscr{W},T}]. \tag{5.9}$$

The angular velocity error $\mathbf{e}_\omega = [e_{\omega_1}, e_{\omega_2}, e_{\omega_3}]^\top$ and the orientation error $\mathbf{e}_R = [e_{R_1}, e_{R_2}, e_{R_3}]^\top$ are then processed by a dynamic prefiltering operation which can also be seen equivalently to PD compensation as shown in Figure 5.1. The input vector of the SFLCs $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3]$ is defined as follows:

$$\boldsymbol{\sigma} = \mathbf{K}_e(-\mathbf{K}_R\mathbf{e}_R - \mathbf{K}_\omega\mathbf{e}_\omega). \tag{5.10}$$

Here $\mathbf{K}_R = \text{diag}(\{K_{R_1}, K_{R_2}, K_{R_3}\})$ and $\mathbf{K}_\omega = \text{diag}(\{K_{\omega_1}, K_{\omega_2}, K_{\omega_3}\})$ are gain matrices for orientation and angular velocity errors ($\mathbf{e}_R$, $\mathbf{e}_\omega$), respectively. $\mathbf{K}_e = \text{diag}(\{K_{e_1}, K_{e_2}, K_{e_3}\})$ is an input scaling matrix for normalization purposes.

The input vector $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3]$ is then processed by the SFLCs to calculate the output vector $\boldsymbol{\varphi} = [\varphi_{o_1}, \varphi_{o_2}, \varphi_{o_3}]$ as follows:

$$\varphi_{o_j} = \boldsymbol{\varphi}_j(\sigma_j) \tag{5.11}$$

where $\boldsymbol{\varphi}_j(.), j = 1,2,3$ represent the FMs. The outputs of SFLCs are then denormalized with an output scaling matrix $\mathbf{K}_u = \text{diag}(\{K_{u_1}, K_{u_2}, K_{u_3}\})$ to calculate the actual control inputs u_2, u_3 and u_4 [90]:

$$\begin{bmatrix} u_2 & u_3 & u_4 \end{bmatrix}^\top = \mathbf{K}_u \begin{bmatrix} \varphi_{o_1} & \varphi_{o_2} & \varphi_{o_3} \end{bmatrix}^\top \tag{5.12}$$

with

$$\mathbf{K}_u = \mathbf{K}_e^{-1} \tag{5.13}$$

Here, all entries on the main diagonal of $\mathbf{K}_e$ are nonzero.

It is worth underlining that if the FMs of the SFLCs are Unit Mappings (UMs) (i.e., $\boldsymbol{\varphi}_{o_j} = \sigma_j$), then the control law presented in (5.12) reduces to

$$\begin{bmatrix} u_2 & u_3 & u_4 \end{bmatrix}^\top = -\mathbf{K}_R \mathbf{e}_R - \mathbf{K}_\omega \mathbf{e}_\omega \tag{5.14}$$

which is the differential flatness-based nonlinear controller presented in [51] and is named the Mellinger controller in the rest of the study.

In the design of the proposed differential flatness-based T1 and IT2 FLCs [90], we firstly suggest designing the crisp controller [51] as the baseline and then shaping the FMs $\boldsymbol{\varphi}_j(.)$, $j = 1,2,3$. In the rest of the section, we assume a baseline controller exists and thus only focus on the design of T1 and IT2 FMs.

5.1 Analyzing and Shaping the FM of the ST1-FLC

The generic rule structure of the ST1-FLC is defined as follows:

$$R^i : \text{IF } \sigma \text{ is } A_i \text{ THEN } \varphi^i_{o_{T1}} \text{ is } \varphi_i = B_i \tag{5.15}$$

where $i = -n, ..., -1, 0, 1, ..., n$ and B_i are crisp consequent MFs satisfying $B_{-p} = -B_p$, $p = 1, ..., n$ and $B_0 = 0 < B_1 < ... < B_n = 1$. The antecedents are defined with 50% overlapping triangular T1-FSs as shown in Figure 5.2. The cores of T1-FSs (c_i) satisfy $c_{-p} = -c_p$, $p = 1, ..., n$ and $c_0 = 0 < c_1 < ... < c_n = 1$ as in Figure 5.2. The total number of rules is $N = 2n + 1$. The ST1-FLC uses the product implication and the center of sets defuzzification method.

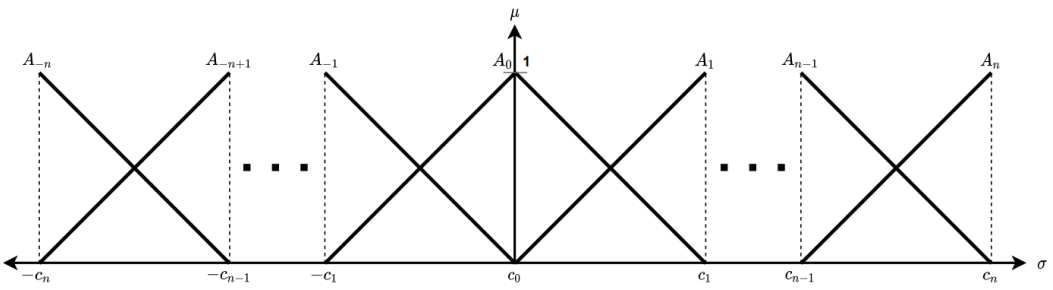


Figure 5.2 : Illustration of the antecedent MFs of the ST1-FLC.

The FM of the ST1-FLCs for an input $\sigma \in [c_i, c_{i+1}]$ is defined as follows [33]:

$$\varphi^i_{o_{T1}}(\sigma) = k^i_{T1}\sigma + \eta^i_{T1} \tag{5.16}$$

where k^i_{T1} is the gain term and η^i_{T1} is the offset term that are defined as:

$$k^i_{T1} = \frac{B_{i+1} - B_i}{c_{i+1} - c_i} \quad \eta^i_{T1} = \frac{B_i c_{i+1} - B_{i+1} c_i}{c_{i+1} - c_i}. \tag{5.17}$$

The generalized T1-FM representation is given for N rule ST1-FLC [33]:

$$\varphi_{o_{T1}}(\sigma) = \begin{cases} k^i_{T1}\sigma + \eta^i_{T1} & \text{if } \sigma \in [c_i, c_{i+1}] \\ k^0_{T1}\sigma & \text{if } \sigma \in [-c_1, c_1] \\ -(k^i_{T1}\sigma + \eta^i_{T1}) & \text{if } \sigma \in [-c_{i+1}, -c_i] \end{cases} \tag{5.18}$$

where k^0_{T1} is the gain term for $\sigma \in [-c_1, c_1]$. We observe that the resulting T1-FM is a piecewise linear mapping.

In this study, we prefer to design a ST1-FLC composed of $N = 5$ rules, and thus we need to tune only the design parameters B_1 and c_1 ($B_2 = 1$ and $c_2 = 1$). To examine how they shape the characteristics of FMs, we define a circle centered at the origin ($\sigma = 0$) with a radius l as shown in Figure 5.3. Here, the slope $tan(\gamma_{c_{p-1}})$ indicates the sensitivity between input σ and output $\varphi_{o_{T1}}$ with respect to $\{c_{p-1}, B_{p-1}\}$. Radius l and angle $\gamma_{c_{p-1}}$ are defined as follows [90]:

$$l = \sqrt{B_1^2 + c_1^2} \quad \gamma_{c_{p-1}} = tan^{-1}(B_p/c_p) \tag{5.19}$$

where $p = 1, ..., n$. In this study, we calculate the sensitivity according to $\{c_0, B_0\}$, i.e., around $\sigma = 0$. Thus, we drop the subscript of γ_{c_0} and use γ in the rest of the paper to keep it simple and easy to follow.

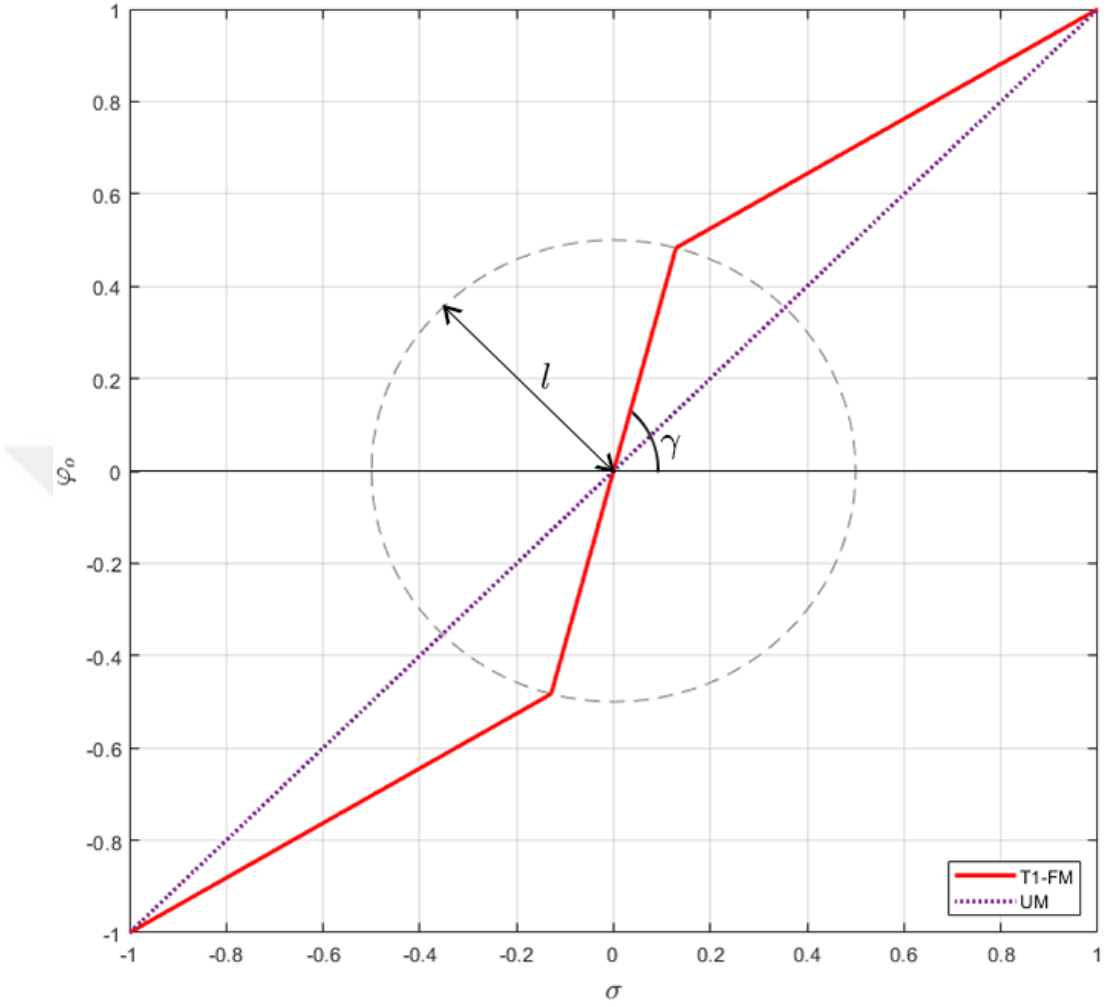


Figure 5.3 : Geometric interpretation of the FM generated with ST1-FLC-5R.

The characteristic of the T1-FM is aggressive for $\gamma > 45°$ while smooth for $\gamma < 45°$. Note that if $\gamma = 45°$, the FM reduces to UM, and thus we end up with the baseline crisp controller (i.e., Mellinger controller). The radius (l) defines the region of aggressiveness/smoothness around the origin. Thus, by defining the radius l and angle value γ, we can easily analyze and tune the region and level of aggressiveness/smoothness of the T1-FMs. For illustrative purposes, we considered two smooth ($\gamma \in \{15°, 30°\}$) and two aggressive ($\gamma \in \{60°, 75°\}$) settings with varying radius $l \in \{0.25, 0.5, 0.75, 1.0\}$ to show the how the design parameters c_1 and B_1 affect the FM. In Figure 5.4, the FMs defined as presented in Table 5.1 are shown. It can be

seen that it is easily possible to generate Aggressive (A) and Smooth (S) T1-FMs with different characteristics.

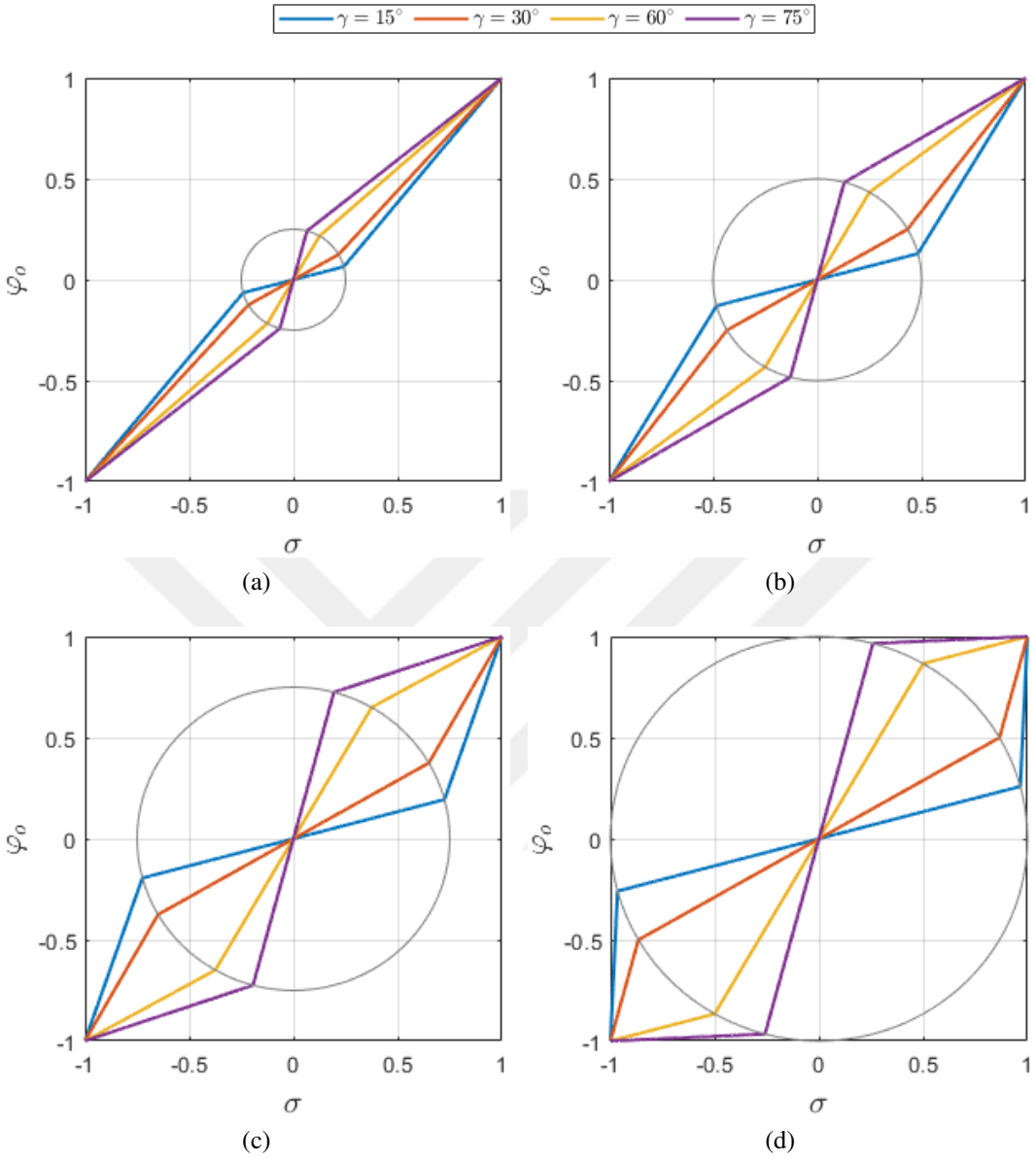


Figure 5.4 : Illustration of T1-FMs for (a) $l = 0.25$ (b) $l = 0.5$ (c) $l = 0.75$ (d) $l = 1.0$.

We want to state that we have not analyzed nor designed an ST1-FLC composed of $N = 3$ rules since the T1-FM reduces to:

$$\varphi^0_{o_{T1}}(\sigma) = k^0_{T1}\sigma \quad \text{with} \quad k^0_{T1} = B_1/c_1 \tag{5.20}$$

The resulting T1-FM has two significant properties as given below [33]:

Table 5.1 : $\{c_1, B_1\}$ pairs for generation of T1-FMs with different characteristics.

	$\gamma = 15^\circ$	$\gamma = 30^\circ$	$\gamma = 60^\circ$	$\gamma = 75^\circ$
$l = 0.25$	{0.241, 0.065}	{0.217, 0.125}	{0.125, 0.217}	{0.065, 0.241}
$l = 0.5$	{0.483, 0.129}	{0.433, 0.250}	{0.250, 0.433}	{0.129, 0.483}
$l = 0.75$	{0.724, 0.194}	{0.650, 0.375}	{0.375, 0.650}	{0.194, 0.724}
$l = 1.0$	{0.966, 0.259}	{0.866, 0.500}	{0.500, 0.866}	{0.259, 0.966}

i $\varphi_{o_{T1}}(\sigma)$ is symmetric with respect to its input variable, σ, $\varphi_{o_{T1}}(\sigma) = -\varphi_{o_{T1}}(-\sigma)$ for $\forall\sigma \neq 0$, and $\varphi_{o_{T1}}(0) = 0$.

ii $\varphi_{o_{T1}}(\sigma)$ is a continuous FM for the interval of input $\sigma \in [-c_n, c_n]$.

It can be seen that the resulting T1-FM is always a linear mapping which means that ST1-FLC is always a linear proportional controller. We obtain $l = \sqrt{2}$ and $\gamma = 45^\circ$ for $B_1 = 1$ and $c_1 = 1$, and thus the T1-FLC reduces to its baseline counterpart (that is, the Mellinger controller) since it has a UM.

5.2 Analyzing and Shaping the FM of the SIT2-FLC

The rule structure of the SIT2-FLC, similar to its T1 counterpart, is as:

$$R^i : \text{IF } \sigma \text{ is } \tilde{A}_i \text{ THEN } \varphi^i_{o_{IT2}} \text{ is } \varphi_i = B_i \tag{5.21}$$

where B_i are crisp consequent MFs and $\tilde{A}_i$ are triangular type IT2-FSs, which are built on top of their T1 counterparts. As shown in Figure 5.5, IT2-FSs are defined with Upper MFs (UMFs) and Lower MFs (LMFs). The area between UMF and LMF defines the FOU, which provides an extra degree of freedom to the SIT2-FLC. The heights of the LMFs m_i are the main parameters that create the FOU and satisfy $m_{-p} = m_p$, $p = 1, ..., n$.

Consider a crisp input denoted by σ', the corresponding SIT2-FLC output is calculated as below:

$$\varphi_{o_{IT2}}(\sigma') = \frac{1}{2}(\varphi^r_{o_{IT2}}(\sigma') + \varphi^l_{o_{IT2}}(\sigma')) \tag{5.22}$$

where $\varphi^r_{o_{IT2}}$ and $\varphi^l_{o_{IT2}}$ which are the endpoints of the type-reduced set are given as below:

$$\varphi^r_{o_{IT2}} = \frac{\sum_{i=1}^{\Omega_R} \underline{\mu}_{\tilde{A}_i}(\sigma')B_i + \sum_{i=\Omega_R+1}^{N} \bar{\mu}_{\tilde{A}_i}(\sigma')B_i}{\sum_{i=1}^{\Omega_R} \underline{\mu}_{\tilde{A}_i}(\sigma') + \sum_{i=\Omega_R+1}^{N} \bar{\mu}_{\tilde{A}_i}(\sigma')} \tag{5.23}$$

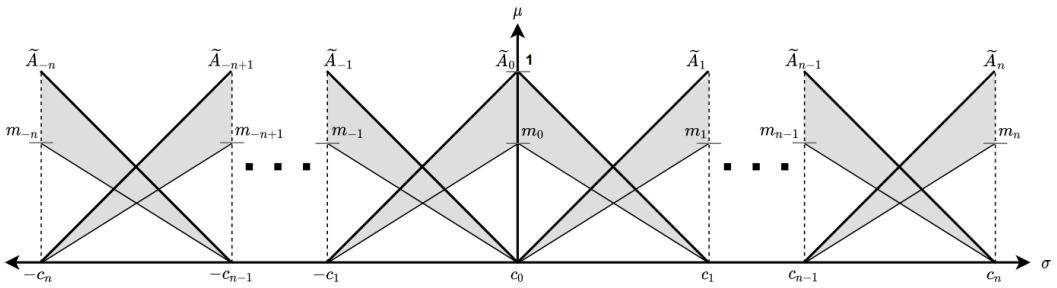


Figure 5.5 : Illustration of the antecedent MFs of the SIT2-FLC.

$$\varphi^l_{o_{IT2}} = \frac{\sum_{i=1}^{\Omega_L} \bar{\mu}_{\tilde{A}_i}(\sigma')B_i + \sum_{i=\Omega_L+1}^{N} \underline{\mu}_{\tilde{A}_i}(\sigma')B_i}{\sum_{i=1}^{\Omega_L} \bar{\mu}_{\tilde{A}_i}(\sigma') + \sum_{i=\Omega_L+1}^{N} \underline{\mu}_{\tilde{A}_i}(\sigma')} \tag{5.24}$$

where Ω_R and Ω_L are right and left switching points; $\bar{\mu}_{\tilde{A}_i}(\sigma')$ and $\underline{\mu}_{\tilde{A}_i}(\sigma')$ are UMFs and LMFs which are defined as below, respectively:

$$\bar{\mu}_{\tilde{A}_i}(\sigma') = \begin{cases} \frac{\sigma'-c_{i+1}}{c_i-c_{i+1}} & \text{if } \sigma' \in [c_i, c_{i+1}] \\ \frac{c_{i-1}-\sigma'}{c_{i-1}-c_i} & \text{if } \sigma' \in [c_{i-1}, c_i] \end{cases} \tag{5.25}$$

$$\underline{\mu}_{\tilde{A}_i}(\sigma') = m_i \, \bar{\mu}_{\tilde{A}_i}(\sigma'). \tag{5.26}$$

As seen in Figure 5.5, the crisp input value σ' will always belong to two subsequent IT2-FSs as $\tilde{A}_i$ and $\tilde{A}_{i+1}$ since antecedents are defined with 50% overlapping triangular IT2-FSs. Thus, the IT2-FM is defined for an input $\sigma \in [c_i, c_{i+1}]$ as follows:

$$\varphi^i_{o_{IT2}}(\sigma) = \frac{1}{2}\left(\frac{\bar{\mu}_{\tilde{A}_i}(\sigma)B_i + \underline{\mu}_{\tilde{A}_{i+1}}(\sigma)B_{i+1}}{\bar{\mu}_{\tilde{A}_i}(\sigma) + \underline{\mu}_{\tilde{A}_{i+1}}(\sigma)} + \frac{\underline{\mu}_{\tilde{A}_i}(\sigma)B_i + \bar{\mu}_{\tilde{A}_{i+1}}(\sigma)B_{i+1}}{\underline{\mu}_{\tilde{A}_i}(\sigma) + \bar{\mu}_{\tilde{A}_{i+1}}(\sigma)} \right) \tag{5.27}$$

Similarly to its T1 counterpart given in (5.16), FM of the SIT2-FLC is formulated by using (5.25), (5.26) and (5.27) for an input $\sigma \in [c_i, c_{i+1}]$ as:

$$\varphi^i_{o_{IT2}}(\sigma) = k^i_{IT2}(\sigma)\sigma + \eta^i_{IT2}(\sigma) \tag{5.28}$$

with the main difference that now $k^i_{IT2}(\sigma)$ is a nonlinear gain and $\eta^i_{IT2}(\sigma)$ is a nonlinear offset term which are defined as follows [33]:

$$k^i_{IT2}(\sigma) = \frac{1}{2}\left(\frac{B_{i+1} - B_i m_i}{c_{i+1}m_i - c_i + \sigma(-m_i+1)} + \frac{B_i - B_{i+1}m_{i+1}}{c_i m_{i+1} - c_{i+1} + \sigma(m_{i+1}+1)} \right) \tag{5.29}$$

$$\eta^i_{IT2}(\sigma) = \frac{1}{2}\left(\frac{B_{i+1}c_i - B_i c_{i+1} m_i}{-c_{i+1}m_i + c_i + \sigma(m_i - 1)} + \frac{B_i c_{i+1} - B_{i+1} c_i m_{i+1}}{-c_i m_{i+1} + c_{i+1} + \sigma(m_{i+1} - 1)} \right). \quad (5.30)$$

In this study, we handle and design SIT2-FLC composed of $N = 3$ (SIT2-FLC-3R) and $N = 5$ rules (SIT2-FLC-5R). Both structures are constructed on their T1 counterparts, and thus their cores (c_i) and crisp consequent MFs (B_i) are identical to their T1 counterparts. In Section 5.1, we have shown that an ST1-FLC composed of 3 rules reduces to a linear mapping. However, the FM of SIT2-FLC-3R does not reduce to a linear mapping for an input $\sigma \in [c_0, c_1]$ [33]. The corresponding IT2-FM is as follows:

$$\varphi^0_{o_{IT2}}(\sigma) = k^0_{IT2}(\sigma)\sigma \quad (5.31)$$

where k^0_{IT2} is defined as follows:

$$k^0_{IT2}(\sigma) = \frac{1}{2}\left(\frac{B_1}{\sigma - \sigma m_0 + c_1 m_0} - \frac{B_1 m_1}{\sigma - c_1 - \sigma m_1} \right). \quad (5.32)$$

We can conclude that the SIT2-FLC has a nonlinear FM even for $N = 3$ rules, unlike its T1 counterpart.

In the latter subsections, we present the design parameters of SIT2-FLCs for $N = 3$ and $N = 5$ rules and analyze their effect on IT2-FM generation.

5.2.1 The FM of the SIT2-FLC-3R

We construct the SIT2-FLC-3R from its T1 fuzzy counterpart by setting $c_1 = 1$ and $B_1 = 1$. Note that the T1-FM with these settings is identical to a UM, and therefore it reduces to the Mellinger controller [51]. In the design of the IT2-FLC-3R, there is only a need to tune the design parameters m_p, $p = 0, 1$. To simplify the design process, we define a single design parameter, $\forall \alpha \in (0, 1)$ [33] and characterize m_p as:

$$m_0 = \alpha, \quad m_1 = 1 - \alpha. \quad (5.33)$$

Then the nonlinear gain k^0_{IT2} given in (5.32) reduces to:

$$k^0_{IT2}(\sigma) = \frac{1}{2}\left(\frac{1}{\alpha + \sigma - \alpha\sigma} + \frac{\alpha - 1}{\alpha\sigma - 1} \right). \quad (5.34)$$

As expected, the FM of the IT2-FLC-3R becomes a function of the primary design parameter α.

In Figure 5.6, we presented the resulting IT2-FMs for $\alpha = \{0.05, 0.95\}$ in comparison with its T1 counterpart (i.e., a UM). We observe that it is possible to shape the FMs with different regions and levels of aggressiveness/smoothness (i.e., l and $tan(\gamma)$), which is impossible with its T1 fuzzy counterpart composed of $N = 3$ rules. As illustrated in Figure 5.6, it is possible to generate an Aggressive IT2-FM (A-IT2-FM) for $\alpha = 0.05$ while a Smooth IT2-FM (S-IT2-FM) by $\alpha = 0.95$. It can be seen that if α is increased from 0.05 (A-IT2-FM) to 0.95 (S-IT2-FM), both the slope and region are affected such that $tan(\gamma_A) > tan(\gamma_S)$ with the radius of the region $l_A < l_S$ [90]. Therefore, unlike the resulting FM of the ST1-FLC-5R, it is not easy to tune the slope and the radius of IT2-FM independently since α affects both the region and level of aggressiveness/smoothness. On the other hand, it is possible to generate more sophisticated FMs by simply tuning α. The characteristic of the IT2-FM varies for $\forall\alpha \in (0,1)$ as follows [33]:

- An A-IT2-FM for $\alpha \in (0, 0.382]$.
- A moderate IT2-FM for $\alpha \in (0.382, 0.618)$.
- An S-IT2-FM for $\alpha \in [0.618, 1)$.

5.2.2 The FM of the SIT2-FLC-5R

We construct the IT2-FLC-5R based on its T1 counterpart. The design parameters of SIT2-FLC-5R are m_0, m_1, m_2 alongside c_1 and B_1 which are the design parameters of its baseline T1 counterpart. For the sake of simplicity, we represent m_0, m_1 and m_2 via a single parameter α, $\forall\alpha \in (0,1)$ as follows [33]:

$$m_0 = m_2 = \alpha, \quad m_1 = 1 - \alpha. \tag{5.35}$$

The resulting FM of the SIT2-FLC-5R for an input $\sigma \in [0, c_1]$ is as

$$\varphi^0_{o_{IT2}}(\sigma) = k^0_{IT2}(\sigma)\sigma \tag{5.36}$$

with

$$k^0_{IT2}(\sigma) = \frac{1}{2}\left(\frac{B_1}{\sigma - \sigma\alpha + c_1\alpha} - \frac{B_1(1-\alpha)}{\sigma\alpha - c_1}\right) \tag{5.37}$$

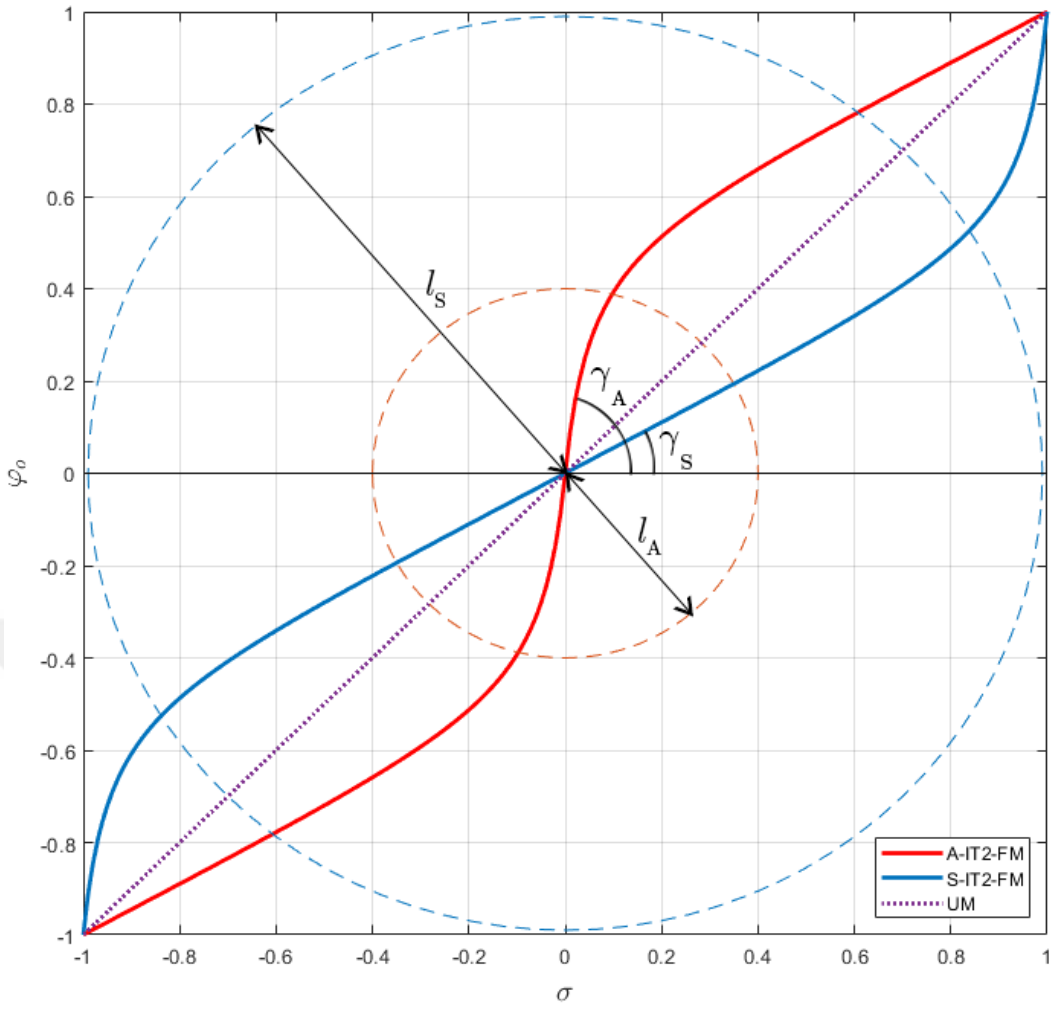


Figure 5.6 : Geometric interpretation of IT2-FMs generated via SIT2-FLC-3R.

while SIT2-FM-5R is as follows:

$$\varphi^1_{o_{IT2}}(\sigma) = k^1_{IT2}(\sigma)\sigma + \eta^1_{IT2}(\sigma) \tag{5.38}$$

Here, $k^1_{o_{IT2}}(\sigma)$ and $\eta^1_{IT2}(\sigma)$ are defined as:

$$k^1_{IT2}(\sigma) = \frac{1}{2}\left(\frac{1-B_1(1-\alpha)}{1-\alpha-c_1+\sigma\alpha} + \frac{B_1-\alpha}{c_1\alpha-1+\sigma(\alpha+1)}\right) \tag{5.39}$$

$$\eta^1_{IT2}(\sigma) = \frac{1}{2}\left(\frac{c_1-B_1(1-\alpha)}{\alpha-1+c_1-\sigma\alpha} + \frac{B_1-c_1\alpha}{-c_1\alpha+1+\sigma(\alpha-1)}\right). \tag{5.40}$$

It can be concluded from (5.36) and (5.38), unlike its T1 fuzzy counterpart given in (5.16) and (5.17), that the IT2-FM is a piecewise nonlinear mapping.

As we have done in Section 5.2.1, we will analyze the effect of α compared to its T1 fuzzy counterpart via a geometric interpretation. For illustrative purposes, let us set

the design parameters of the baseline T1-FM as $c_1 = 0.5$ and $B_1 = 0.866$. Thus, we end up with an aggressive T1-FM in comparison with UM since $\gamma_{T1} = 60^\circ$ as shown in Figure 5.7. The corresponding radius value is $l = 1$. We have also illustrated the resulting IT2-FMs for $\alpha = 0.05$ (A-IT2-FM) and $\alpha = 0.95$ (S-IT2-FM). It can be concluded that similar to its $N = 3$ rule counterpart, it is possible to increase/decrease the sensitivity of the FM of the IT2-FLC-5R through α. In other words, the tuning of α influences the departure angle from the origin. Thus it is possible to generate more aggressive IT2-FM for $\alpha = 0.05$ ($\gamma_A > 60^\circ$) while a smoother IT2-FM for $\alpha = 0.95$ ($\gamma_S < 60^\circ$) in comparison with its baseline T1-FM within the radius of $l = 1$. It is also worth mentioning that the characteristic is vice versa outside the radius of $l = 1$. To sum up, through α, it is possible to increase/decrease the level of sensitivity (aggressiveness/smoothness) around the steady-state point ($\sigma = 0$) of the SIT2-FLC to increase the performance of the control system [90].

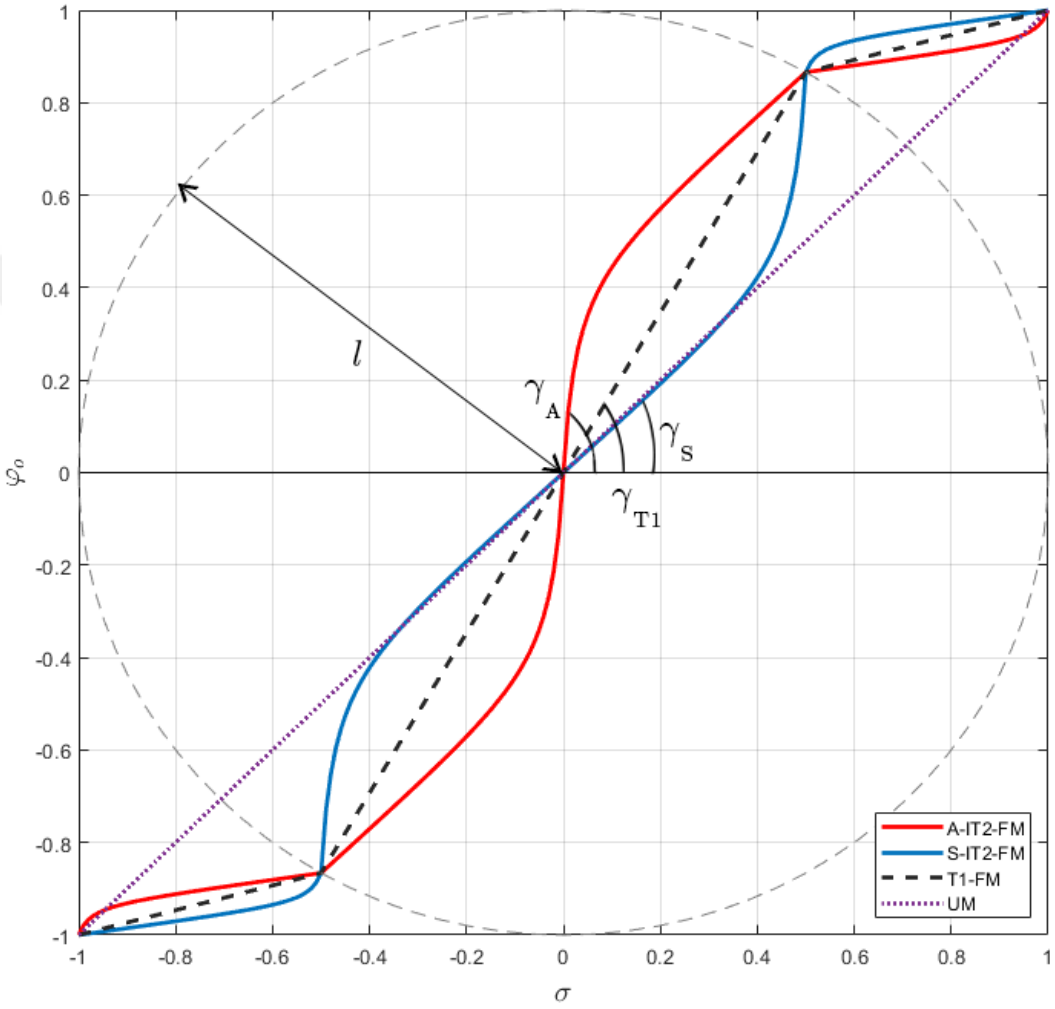


Figure 5.7 : Geometric interpretation of the IT2-FMs generated via SIT2-FLC-5R.

6. EXPERIMENTAL STUDIES

This section explains all the details from trajectory generation to the deployed controllers. We provide comparative results for experimental studies to evaluate the performances of differential flatness-based SFLCs compared to their baseline counterpart, the Mellinger controller. The experiments were conducted in an indoor environment, as shown in Figure 6.1. The real-time performance of the SIT2-FLC-3R and SIT2-FLC-5R-2 structures can be observed via the video file provided in the *Supplementary Material* (see Appendix A).

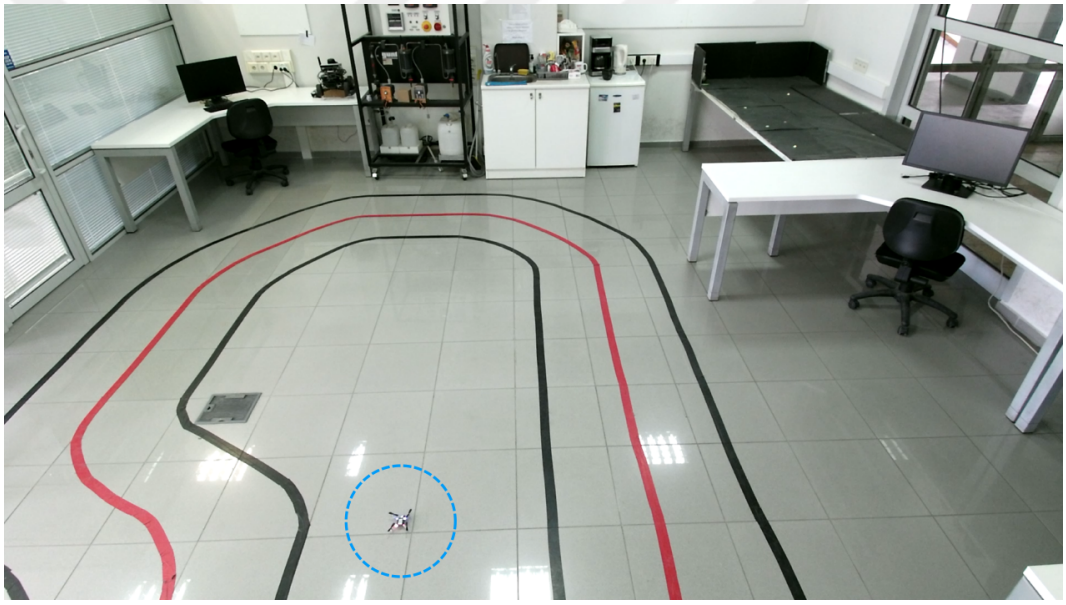


Figure 6.1 : Experimental indoor environment.

As stated in Section 2, we also mounted the module flow deck [61] to Crazyflie 2.1 for experimental studies. Thus the feedback information of the 6 DoF IMU alongside the flow deck displacement measurements are filtered and fused through bundled Kalman filter estimator that is provided as a built-in solution within the Crazyflie 2.1 firmware. Although the performances of the built-in filters are high, both measuring noise and measurement errors are inevitable. In this thesis, we did not make any further study on improving the sensing performance as the main contribution of the study is the

design and development of the differential flatness-based T1 and IT2 SFLC structure for aggressive maneuvering. Then, we implemented the proposed control scheme of ST1-FLC/SIT2-FLC-3R/SIT2-FLC-5R as a custom controller to the firmware, see Appendix B. The pseudo-code of the differential flatness-based SFLC implementation is given in Figure 6.2. We also wrote an additional Python script to be run on the host computer to collect flight data from Crazyflie 2.1, send trajectory commands, and set controller parameters.

Algorithm

Get r_T, v_T, ψ_T, $\omega_{BW,T}$ and their derivatives
Measure/estimate r, v, ψ, ω_{BW} and their derivatives
Calculate e_p and e_v via (5.1) to obtain the desired thrust F_{des} in (5.2)
Transform F_{des} into thrust control signal u_1 by using (5.3)
Compute R_{des} in (5.7) by using (5.4)-(5.6)
Compute the rotation error e_R via (5.8) and the angular velocity error e_ω via (5.9)
Calculate the input vector of the SFLCs $\boldsymbol{\sigma}$ as given in (5.10)
if T1-FM is defined **then**
    use (5.16) to compute SFLCs outputs φ_{o_j}
else if IT2-FM is defined **then**
    use (5.28) to compute SFLCs outputs φ_{o_j}
end if
Calculate moment control signals u_2, u_3, u_4 via (5.12)

Figure 6.2 : Implementation of the differential flatness-based SFLC.

6.1 Trajectory Generation

We use the Python script in [62] to generate the whole trajectory as a set of 7th-degree Bézier curves. For any trajectory, P waypoints are defined with a specified position, velocity, acceleration, and yaw angle for $t_i \in [0, t_T]$ where $i = 1, 2, ..., P$, and t_T is the total duration of the trajectory. Then, a set of Bézier curves is generated with these waypoints that are transformed into a matrix $\mathbf{P}$ of coefficients of the polynomials, which are compatible with Crazyflie 2.1 firmware. Each row of this matrix consists of 7th-degree polynomial coefficients, each of which is generated by the specified position and yaw angle ($x_T(t_i)$, $y_T(t_i)$, $z_T(t_i)$, $\psi_T(t_i)$, $i = 1, 2, ..., P$) for a defined waypoint as

below:

$$\mathbf{P} = \begin{bmatrix} t_0 & \kappa^0_{x_T,0} & \cdots & \kappa^7_{x_T,0} & \kappa^0_{y_T,0} & \cdots & \kappa^7_{y_T,0} & \kappa^0_{z_T,0} & \cdots & \kappa^7_{z_T,0} & \kappa^0_{\psi_T,0} & \cdots & \kappa^7_{\psi_T,0} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ t_P & \kappa^0_{x_T,P} & \cdots & \kappa^7_{x_T,P} & \kappa^0_{y_T,P} & \cdots & \kappa^7_{y_T,P} & \kappa^0_{z_T,P} & \cdots & \kappa^7_{z_T,P} & \kappa^0_{\psi_T,P} & \cdots & \kappa^7_{\psi_T,P} \end{bmatrix} \tag{6.1}$$

where $\kappa^n_{(.),i}$ represents the n^{th} ($n = 0, 1, ..., 7$) coefficient of the 7th-degree polynomial for specified position and yaw (x_T, y_T, z_T, ψ_T) of i^{th} ($i = 1, 2, ..., P$) waypoint. For each flight, generated trajectories are uploaded to Crazyflie 2.1 with the form given by (6.1).

To define slow and aggressive flights, $\mathbf{r}_T(t) = [x_T(t), y_T(t), z_T(t)]^\top$ and $\psi_T(t)$ is stretched or compressed in time. To accomplish such a goal, we define a dimensionless time constant β that transforms the dimensional time t into a nondimensionalized time τ as $t = \beta\tau$. Thus, we define a nondimensional specified position, $\tilde{\mathbf{r}}_T = [\tilde{x}_T, \tilde{y}_T, \tilde{z}_T]^\top$ and nondimensional yaw angle $\tilde{\psi}_T$ for the trajectory. Then, via $t_i = \beta\tau_i$ ($i = 1, 2, ..., P$), the trajectory becomes the time-scaled version of the nondimensional trajectory [51]:

$$\begin{aligned} \mathbf{r}_T(t) &= \tilde{\mathbf{r}}_T(t/\beta) \\ \psi_T(t) &= \tilde{\psi}_T(t/\beta). \end{aligned} \tag{6.2}$$

If β is increased, the trajectory has a longer execution time and provides a slower and safer flight. In contrast, with decreased β, the trajectory takes less time to execute and becomes more aggressive due to the increase of derivatives of position [51].

As shown in Figure 6.3 and Figure 6.4, we generated four trajectories with different characteristics to compare the performances of the implemented controllers for different timescales, β. The generated trajectories are given in Appendix C with the matrix $\mathbf{P}$ form which is defined in (6.1). The trajectories do not include take-off and landing processes. The first one is Trajectory-1, which is shown in Figure 6.3a, with an execution time of $t_T = 7.3s$ for $\beta = 1.0$. The planned maximum velocity (v_{max}) and acceleration (a_{max}) are $v_{max} = 1.22m/s$ and $a_{max} = 3.07m/s^2$ for $\beta = 1.0$. Trajectory-2 is illustrated in Figure 6.3b with $v_{max} = 2.17m/s$ and $a_{max} = 3.9m/s^2$, $t_T = 8s$ for $\beta = 1.0$. As shown in Figure 6.3c, Trajectory-3 is defined as a cardioid-like shape with $v_{max} = 1.82m/s$ and $a_{max} = 2.69m/s^2$, and execution time is $t_T = 8.8s$ for $\beta = 1.0$. The values of v_{max} and a_{max} are between Trajectory 1 and Trajectory 2. For the first three trajectories, the altitude is constant as $z = 1.0m$ along the path. Unlike the first three trajectories, Trajectory-4 has variable altitude as $z \in [0.97, 1.52]m$ as shown in Figure 6.4a (x-y

plot) and Figure 6.4b (x-y-z plot). For $\beta = 1.0$, the its characteristics are $t_T = 8s$, $v_{max} = 2.25m/s$ and $a_{max} = 2.98m/s^2$. It is worth underlining that the calculated t_T, v_{max} and a_{max} values naturally depend on the value of β.

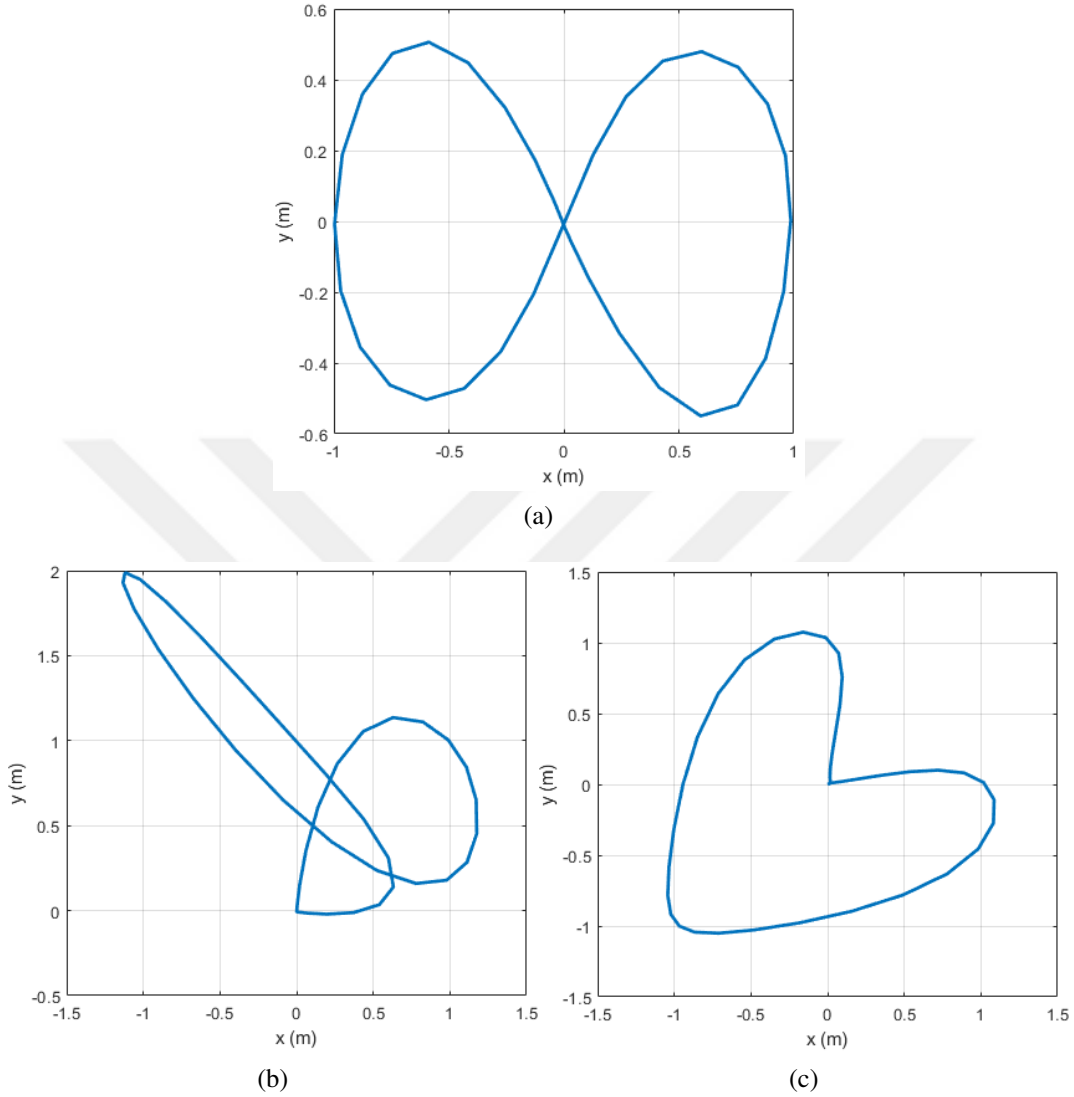
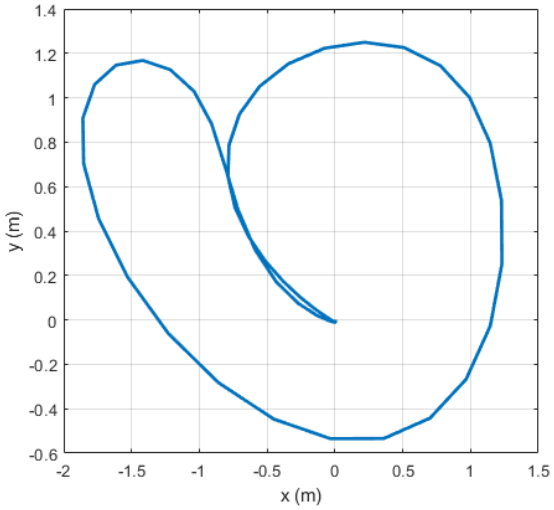


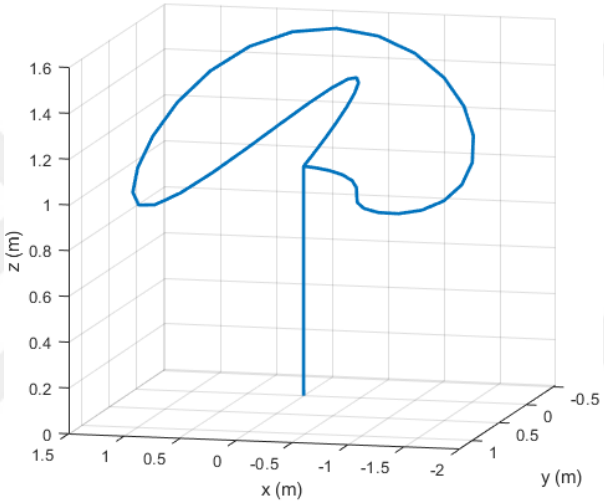
Figure 6.3 : Illustration of (a) Trajectory-1 (b) Trajectory-2 (c) Trajectory-3.

6.2 Design of the Flight Controllers

In the experiments, we directly employed the built-in Mellinger controller without further tuning and used it directly as the baseline controller to design the proposed SFLCs. In the tuning of the proposed differential flatness-based FLCs shown in Figure 5.1, we have configured the FM $\boldsymbol{\varphi}_3$ that generates the control signal u_4 with a UM and only shaped the FMs $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ that generate u_2 and u_3. Note that, due to the



(a)



(b)

Figure 6.4 : Illustration of Trajectory-4 (a) 2D view (b) 3D view.

symmetric structure of the quadcopter [89], we employed the same FMs for $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$. The designed FMs are shown in Figure 6.5 while $\gamma - l$ is given in Figure 6.6 to compare the aggressiveness/smoothness of the designed FMs. The following SFLCs are designed and employed:

- ST1-FLC-5R: The design parameters are set as $c_1 = 0.25$ and $B_1 = 0.433$ to end up with a more aggressive FM in comparison with the UM since $l = 0.5$ and $\gamma = 60^\circ$. Thus, the ST1-FLC-5R is a more aggressive controller when compared to its baseline (Mellinger) controller.

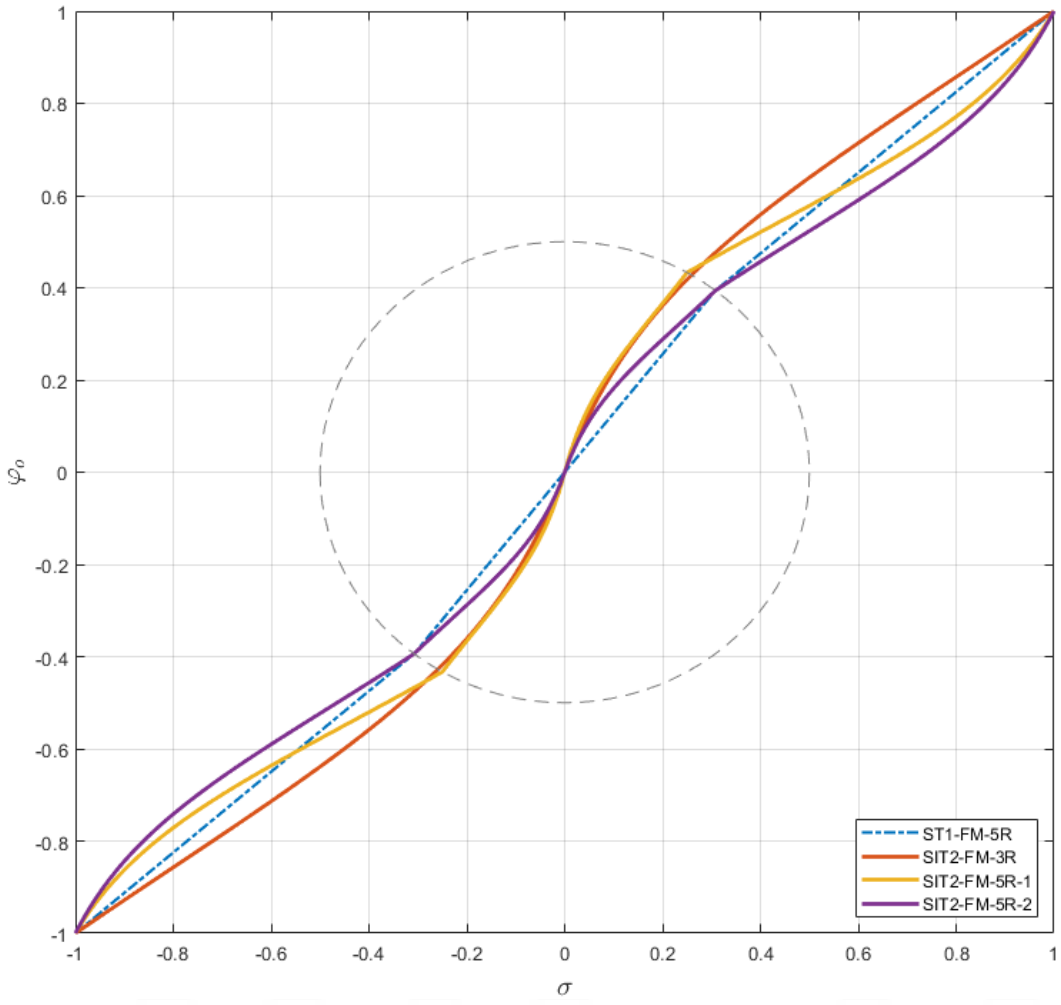


Figure 6.5 : Designed FMs for the experimental studies.

- SIT2-FLC-3R: The single design parameter is set as $\alpha = 0.2$ to define an A-IT2-FM. This IT2-FM is more aggressive than T1-FM since $\gamma > 60^\circ$ within the region of $l < 0.4525$ according to Figure 6.6.
- SIT2-FLC-5R: For this structure, we have designed two IT2-FLCs:
  - SIT2-FLC-5R-1 is designed based on ST1-FLC-5R as suggested in Section 5.2.2. Thus, we set the parameters as $c_1 = 0.25$, $B_1 = 0.433$. The design parameter α is set as $\alpha = 0.25$ to increase $\gamma > 60^\circ$, and thus a more aggressive FM is shaped in comparison to its T1 fuzzy counterpart within the region of $l < 0.5$. Note that SIT2-FLC-5R-1 is the most aggressive design among all controllers for $l < 0.5$.

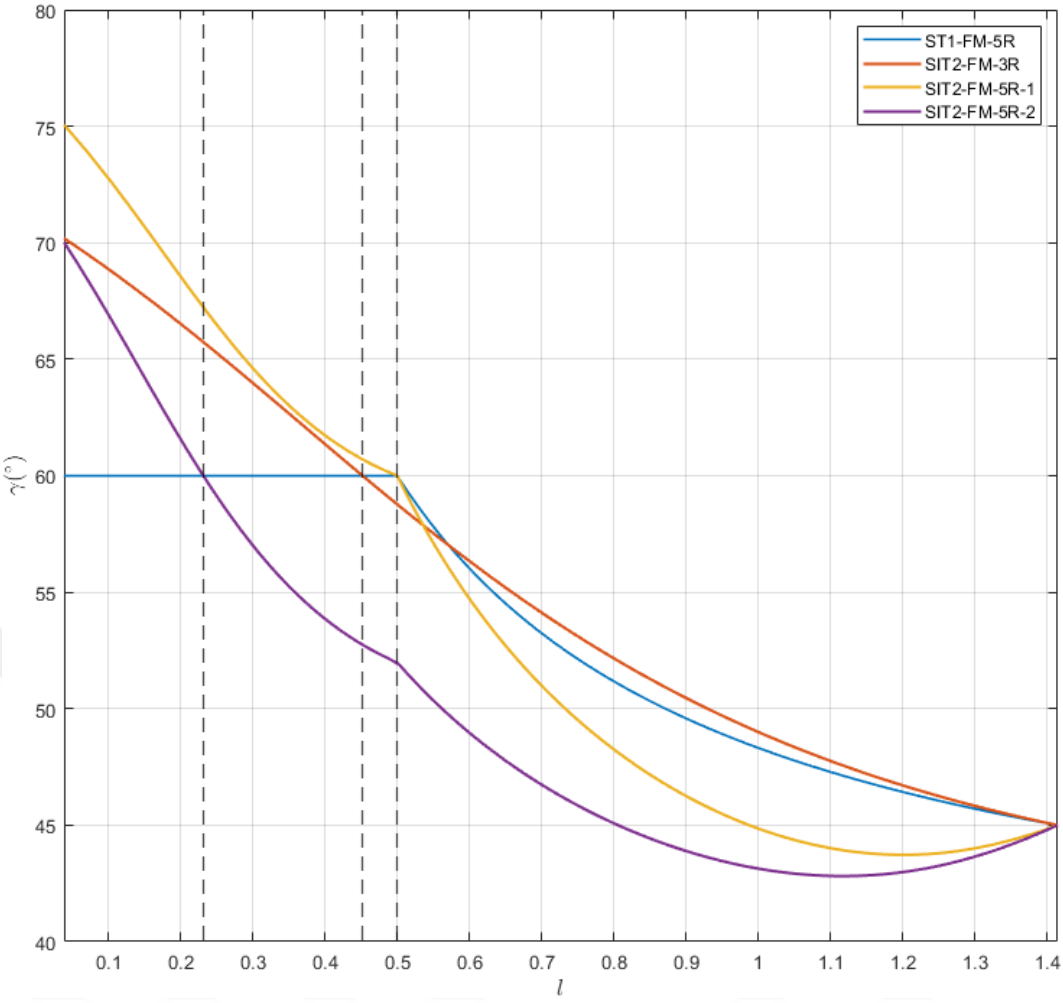


Figure 6.6 : γ - l relation for the designed FMs.

– SIT2-FLC-5R-2 is designed by tuning all design parameters c_1, B_1 and α from scratch. We first set $c_1 = 0.308$, $B_1 = 0.394$ to end up with a smoother T1-FM ($\gamma = 52°$) than the T1 baseline of ST1-FLC-5R-1 ($\gamma = 60°$). Then, we defined the FOU parameter as $\alpha = 0.25$ and increased the level of smoothness (i.e., $\gamma > 60°$) for $l < 0.2313$ when compared to other SIT2-FLC designs. SIT2-FLC-5R-2 exhibits the smoothest FM among all SFLCs within the region of $l > 0.2313$.

6.3 Performance Measures

We evaluated the performances of the control systems over 10 flights for each presented trajectory. We define the flight performance of a single flight with:

$$h = \sum_{n=1}^{M} d[n] \tag{6.3}$$

where M is the number of samples, and d is the two-dimensional Euclidean distance that is defined as:

$$d[n] = \sqrt{(\mathbf{e}_{p_1}[n])^2 + (\mathbf{e}_{p_2}[n])^2} \tag{6.4}$$

where $\mathbf{e}_{p_1}$ and $\mathbf{e}_{p_2}$ are x axis position error and y axis position error, respectively. Note that, since we have not designed SFLC for the z axis and employed a crisp controller, we evaluated the performances only in the $x-y$ axis coordinate system.

To compare flight performances, we calculated the average h value (h_E) and the corresponding standard deviation value (SD), and also reported the minimum and maximum h values (h_{min} and h_{max}) over 10 flights. Note that we calculated the performance measures for the flight time t_T with an extra of 2s defined for hovering upon the end of the trajectory. Neither take-off nor landing is taken into account for calculations.

6.4 Comparative Experimental Results

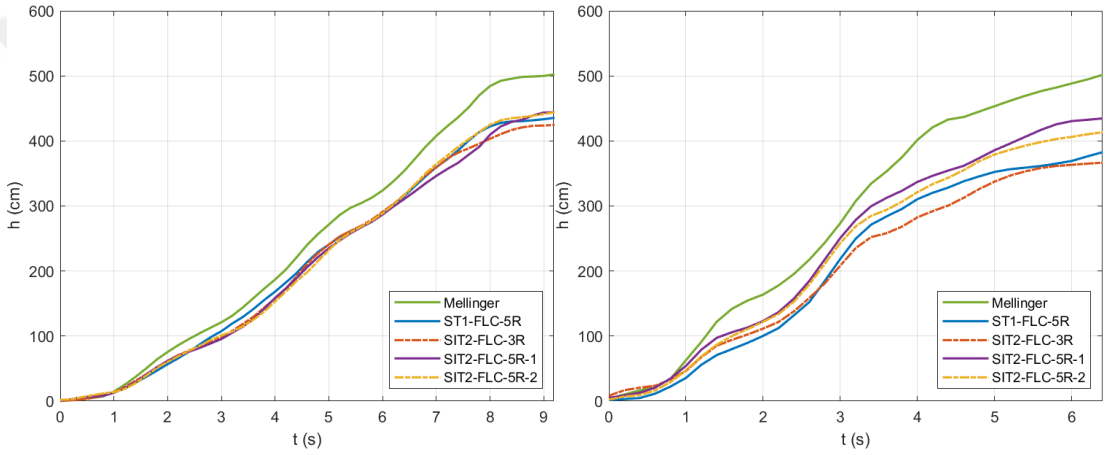
We have evaluated the performances of the controllers for $\beta = 1$ and for $\beta < 1$ to further increase the aggressiveness of the trajectory. Thus, we examine how the deployed controller can handle a high level of nonlinearity as the coupled dynamics of the quadcopter becomes more dominant in aggressive maneuvering.

For Trajectory-1, the resulting performance measures for $\beta = 1.0$ and $\beta = 0.6$ are tabulated in Table 6.1. The variations of h values of all controllers are shown in Figure 6.7 for a single flight, while the tracking responses are shown in Figure 6.8. It is observed that the mean performance values, h_E, of all FLCs are smaller than h_{min} of Mellinger controller for $\beta = 1.0$. SIT2-3R-FLC (which has a single design parameter) exhibits the best h_E for both $\beta = 1.0$ and $\beta = 0.6$. On the other hand, the SD values

of SIT2-FLC-5Rs are significantly smaller than the other ones. Thus, their aggressive maneuvering performance can be seen as more consistent when compared to other flight control systems.

Table 6.1 : Comparative performance measures: Trajectory-1.

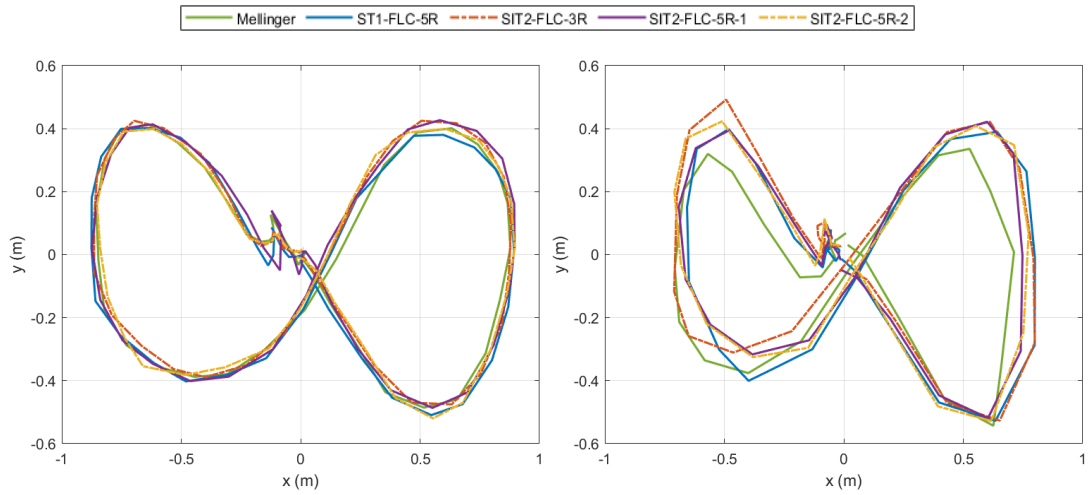
	$\beta = 1.0$				$\beta = 0.6$			
Controller	h_E	h_{min}	h_{max}	SD	h_E	h_{min}	h_{max}	SD
Mellinger	487.16	478.31	502.32	10.46	462.22	423.00	501.61	34.49
ST1-FLC-5R	457.93	435.56	472.30	15.43	429.08	382.68	460.55	31.03
SIT2-FLC-3R	444.28	424.67	462.55	17.02	406.47	366.54	446.22	29.41
SIT2-FLC-5R-1	466.75	444.04	490.78	19.07	445.47	434.57	454.33	7.67
SIT2-FLC-5R-2	464.30	443.96	504.62	23.52	421.21	413.34	430.27	7.59



(a) (b)

Figure 6.7 : Variation of h for a single flight on Trajectory-1 (a) $\beta = 1.0$ (b) $\beta = 0.6$.

For Trajectory-2, the flight performances and the change of h for all controllers are shown in Figure 6.9 and Figure 6.10, respectively. The resulting comparative performance measures are tabulated in Table 6.2 for $\beta = 1.0$ and $\beta = 0.65$. For this case, although the SD value of the Mellinger controller is the smallest, the ST1-FLC-5R has the best h_E for $\beta = 1.0$. However, for $\beta = 0.65$, the SD value of the Mellinger controller dramatically increases, and the performance of the ST1-FLC-5R deteriorates compared to the SIT2-FLC-5R structures. On the other hand, it can be seen that for higher trajectory speeds ($\beta = 0.65$), the resulting performances of the SIT2-FLC-5R structures are superior concerning all performance measures.

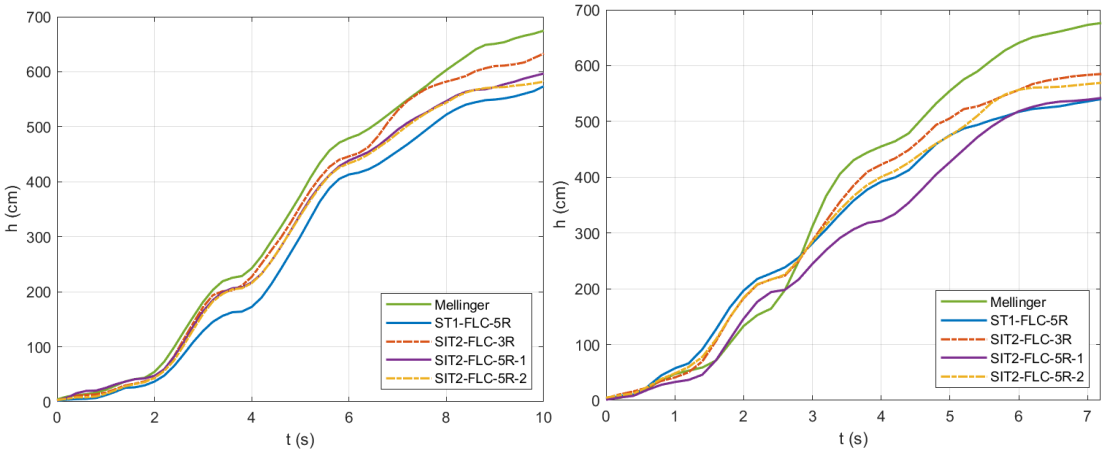


(a) (b)

Figure 6.8 : Trajectory-1 tracking responses for (a) $\beta = 1.0$ (b) $\beta = 0.6$.

Table 6.2 : Comparative performance measures: Trajectory-2.

	$\beta = 1.0$				$\beta = 0.65$			
Controller	h_E	h_{min}	h_{max}	SD	h_E	h_{min}	h_{max}	SD
Mellinger	659.35	623.57	674.69	21.22	636.08	534.35	676.36	58.15
ST1-FLC-5R	610.42	573.52	635.15	29.14	609.66	540.04	655.45	46.90
SIT2-FLC-3R	655.78	632.96	688.97	22.52	626.85	584.99	682.37	37.53
SIT2-FLC-5R-1	643.02	596.71	689.20	39.82	590.18	541.93	615.93	30.62
SIT2-FLC-5R-2	632.07	581.92	663.13	31.44	574.96	548.06	594.04	20.88



(a) (b)

Figure 6.9 : Variation of h for a single flight on Trajectory-2 (a) $\beta = 1.0$ (b) $\beta = 0.65$.

For Trajectory-3, the obtained performance measures for $\beta = 1.0$ and $\beta = 0.65$ are presented in Table 6.3. The corresponding trajectory tracking performances are given in Figure 6.11 and Figure 6.12. For both β values, SIT2-FLC-5R-1 has the smallest h_E

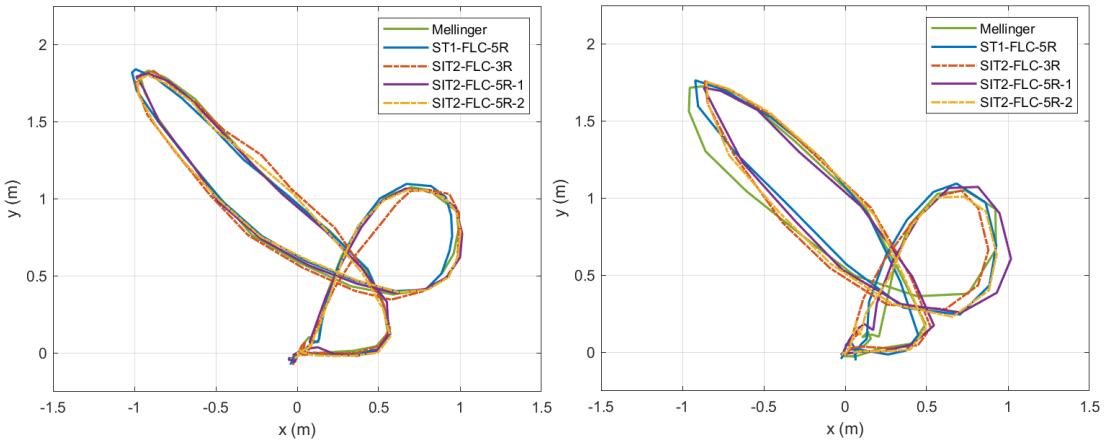


Figure 6.10 : Trajectory-2 tracking responses for (a) $\beta = 1.0$ (b) $\beta = 0.65$.

values and proposed FLCs exhibit better performances than the Mellinger controller. ST1-FLC-5R has a significantly small SD value compared to the others for $\beta = 1.0$. At $\beta = 0.65$, SD values either stay almost the same or increase except for SIT2-FLC-5R-2.

Table 6.3 : Comparative performance measures: Trajectory-3.

	$\beta = 1.0$				$\beta = 0.65$			
Controller	h_E	h_{min}	h_{max}	SD	h_E	h_{min}	h_{max}	SD
Mellinger	680.28	587.12	753.49	67.38	559.10	464.11	631.15	68.76
ST1-FLC-5R	639.68	632.06	659.09	11.11	524.12	495.46	563.75	27.76
SIT2-FLC-3R	632.49	597.32	656.60	27.93	525.25	501.88	560.68	24.69
SIT2-FLC-5R-1	625.42	590.95	675.37	38.40	459.42	403.52	500.23	36.32
SIT2-FLC-5R-2	648.17	588.53	679.06	36.20	491.72	462.77	509.85	18.99

For Trajectory-4, the experimental results are given in Table 6.4. In this experiment, h values of all controllers for a single flight are shown in Figure 6.13, while the tracking responses are given in Figure 6.14. ST1-FLC-5R and SIT2-FLC-5R-2 have the two smallest SD values at $\beta = 1.0$. SIT2-FLC-5R-2 and SIT2-FLC-5R-1 show the best performances for $\beta = 1.0$ and $\beta = 0.7$, respectively. For smaller β, the SD value of SIT2-FLC-5R-2 decreases while the other ones increase. Even the SD value of the Mellinger controller increases drastically by the higher speed of trajectory.

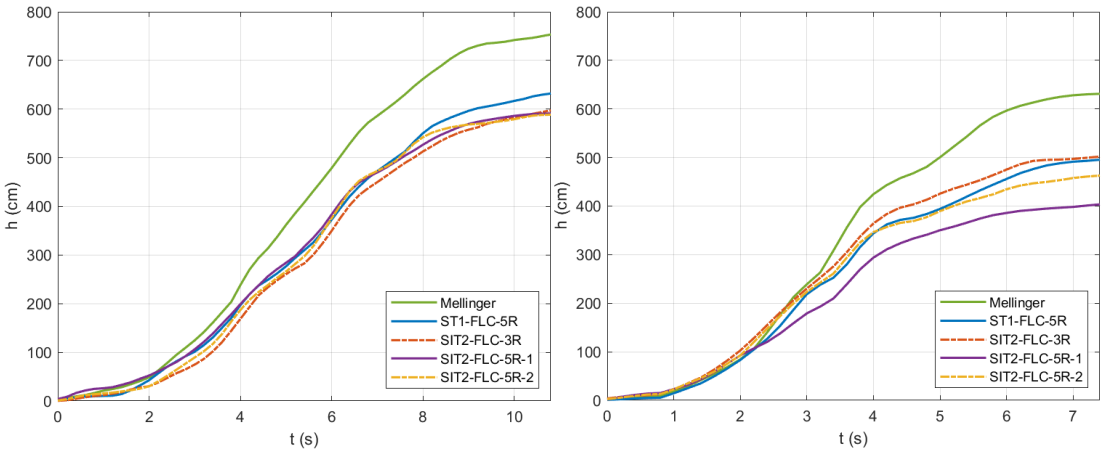


Figure 6.11 : Variation of h for a single flight on Trajectory-3 (a) $\beta = 1.0$ (b) $\beta = 0.65$.

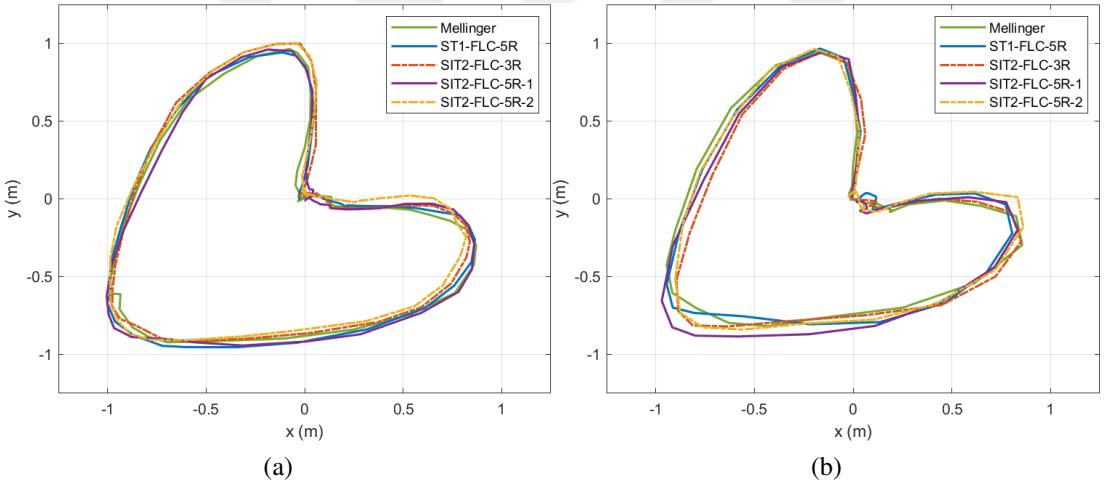
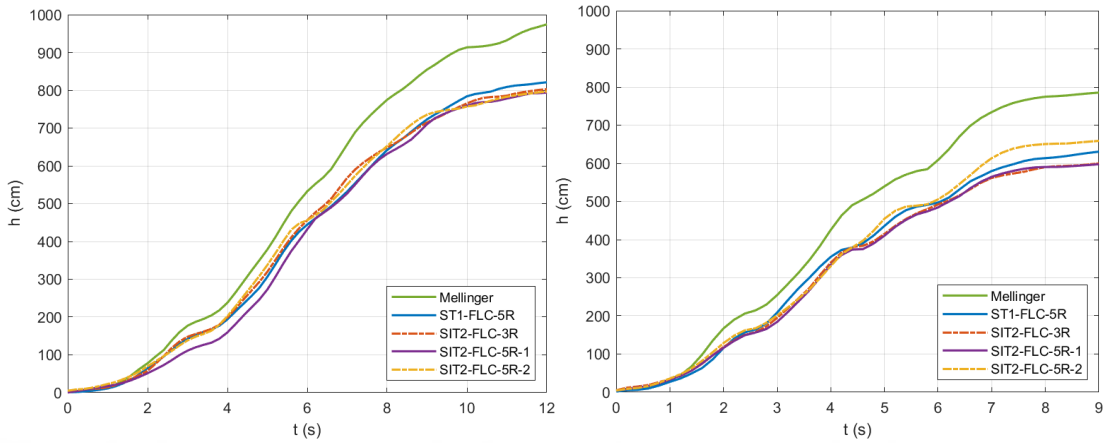


Figure 6.12 : Trajectory-3 tracking responses for (a) $\beta = 1.0$ (b) $\beta = 0.65$.

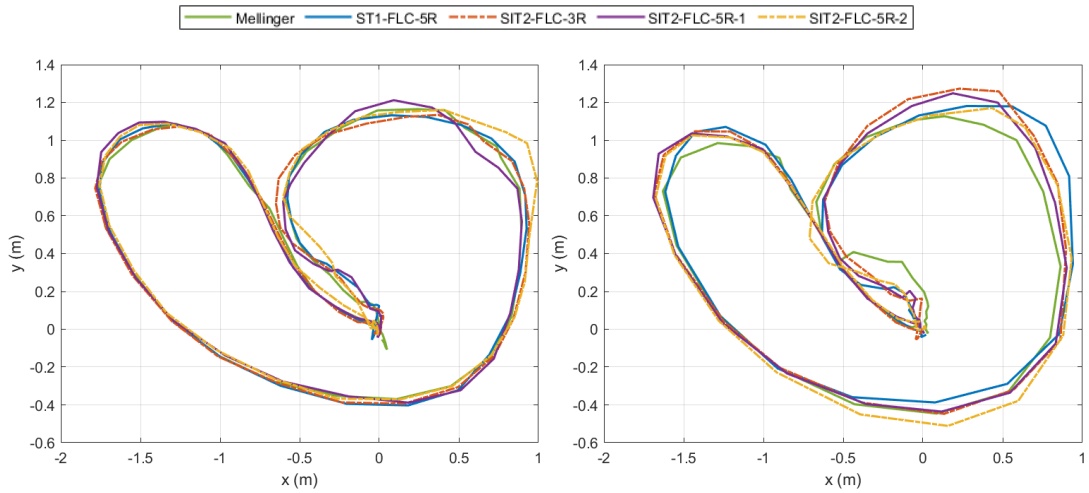
Table 6.4 : Comparative performance measures: Trajectory-4.

	$\beta = 1.0$				$\beta = 0.7$			
Controller	h_E	h_{min}	h_{max}	SD	h_E	h_{min}	h_{max}	SD
Mellinger	882.37	848.85	974.47	51.90	673.14	596.49	785.45	81.86
ST1-FLC-5R	850.93	821.11	886.69	27.87	669.13	630.61	724.45	39.58
SIT2-FLC-3R	884.47	802.77	937.51	50.59	650.55	599.80	726.49	53.66
SIT2-FLC-5R-1	850.75	793.69	897.33	37.06	626.64	597.69	697.76	41.65
SIT2-FLC-5R-2	821.73	796.79	868.52	29.91	685.85	658.79	722.90	25.97



(a) (b)

Figure 6.13 : Variation of h for a single flight on Trajectory-4 (a) $\beta = 1.0$ (b) $\beta = 0.7$.



(a) (b)

Figure 6.14 : Trajectory-4 tracking responses for (a) $\beta = 1.0$ (b) $\beta = 0.7$.

7. CONCLUSIONS AND FUTURE WORKS

We presented a novel differential flatness-based SFLC that employs T1 or IT2 FM to result in an efficient aggressive maneuvering performance. The effects of the design parameters on SFLC behaviors are examined via a geometric interpretation to measure the level and region of aggressiveness/smoothness. In this thesis, we also show the implementation of the proposed controller structure into the real-time system alongside the theoretical introduction of differential flatness-based SFLC.

As the first step, we define the differential flatness property for the quadcopter to model the system in terms of flat outputs, states, and inputs. We give the details of the dynamic modeling with the Z-X-Y Euler angle convention and Newton-Euler equations. Then, differential flatness-based characterization is applied to the position, orientation, angular velocity, angular acceleration, and inputs of the quadcopter to show the system is a differentially flat one. While net thrust is calculated via a crisp controller that is based on position and velocity errors, the other body moments (control inputs for roll, pitch, and yaw) are generated via a SFLC structure that uses orientation and angular velocity errors computed by differential flatness characterization.

The proposed SFLC structure includes two controller schemes as ST1-FLC and SIT2-FLC. We give details for the generic rule structure, antecedent and consequent settings, FM calculations, and design parameters for both controllers. We use an authentic method based on a geometric approach to make interpretations for the effects of the design parameters. Thus, we can decide whether the FM is aggressive or smooth by looking at the gain contribution of the partial FM that remains in a certain circle located at the origin.

In this study, we prefer to use the ST1-FLC design with $N = 5$ rules (ST1-FLC-5R) which provide piecewise linear T1-FM. Design parameters of the ST1-FLC-5R, B_1 and c_1, directly shape the resulting T1-FM. To inspect how they shape the characteristics of FMs, we define a circle centered at the origin of T1-FM with a radius l, and the

slope, $tan(\gamma)$, indicates the sensitivity between input and output. We give some $\{c_1$, $B_1\}$ pairs for the generation of T1-FMs with different characteristics to analyze and illustrate aggressive/smooth T1-FMs. We also show that if B_1 and c_1 are selected to obtain T1-FM with $\gamma = 45°$, T1-FM reduces to completely linear unit mapping which is referenced as baseline (Mellinger) controller in this study [51]. Additionally, the same resulting T1-FM is observed via ST1-FLC with $N = 3$ rules which is not preferred in this study due to having always a linear mapping.

We employ SIT2-FLC with $N = 3$ rules (SIT2-FLC-3R) alongside its $N = 5$ design (SIT2-FLC-5R) similar to ST1-FLC. The same geometric approach with ST1-FLC is used to analyze the effect of the design parameters of SIT2-FLCs on IT2-FMs. SIT2-FLC-3R has a simple design process with a single design parameter, $\forall\alpha \in (0,1)$. We observed that if α is increased both the slope and region are affected such that decrement in γ and increment in l. Since α affects both the region and level of aggressiveness/smoothness, it is difficult to tune the slope and the radius of IT2-FM for SIT2-FLC-3R independently unlike the resulting FM of the ST1-FLC-5R. On the flip side, we can generate more sophisticated FMs for SIT2-FLC-3R by simply tuning α.

SIT2-FLC-5R is based on its T1 counterpart with the design parameters α ($\forall\alpha \in (0,1)$) and c_1, B_1 which are the design parameters of its baseline T1 counterpart. While its T1 counterpart has a piecewise linear mapping for the T1-FM, FM of the SIT2-FLC-5R is a piecewise nonlinear mapping. For SIT2-FLC, we analyze the effect of α compared to its T1 fuzzy counterpart. We make a design for the baseline T1-FM, and construct IT2-FMs on the baseline via different α values. The sensitivity of the FM of the IT2-FLC-5R can be increased or decreased through α, similar to its $N = 3$ rule counterpart. Additionally, it is possible to generate a more aggressive or smoother IT2-FM in comparison with its baseline T1-FM within the radius of l. Thus, we can increase/decrease the level of sensitivity (aggressiveness/ smoothness) around the steady-state point ($\sigma = 0$) of the SIT2-FLC to increase the performance of the control system.

We designed SFLCs based on our analysis and implemented them into Crazyflie 2.1 firmware to run as embedded real-time controllers for aggressive maneuvering. For experimental studies, we generated four trajectories with different characteristics to

compare the performances of the implemented controllers for different timescales, β. The experiments were conducted in an indoor environment by Crazyflie 2.1 with onboard position estimation. We evaluated the performances of the control systems over 10 flights for each designed trajectory. The presented experimental results show that the proposed SFLC provides better performance for aggressive maneuvering than its baseline controller [51]. We conclude that the SIT2-FLC-5R structures give overall preferable results compared to ST1-FLC and SIT2-FLC-3R. Significantly, the tracking performance of SIT2-FLC-5R-2 is more consistent than any other as the trajectory becomes more aggressive with smaller β values. On the other hand, the performance of the SIT2-FLC-3R is also significant due to its simplicity of design.

As future works, different parts of this study can be handled. One of these topics is the stability analysis of the proposed differential flatness-based SFLC. As mentioned earlier; if the design parameters are selected to generate a UM, our proposed controller structure reduces to its baseline crisp controller [51]. In [51,89]; stability and convergence are proved under some conditions such as the addition of feedforward and feedback terms, and an assumption for gain matrices. As a future study, detailed stability analysis can be built on the analysis of the baseline crisp controller to present the differential flatness-based SFLC structure that is stable under the same conditions. The other two topics are related to the control perspective. In Chapter 5 and Chapter 6, we presented how the design parameters shape the FM, we conducted experimental studies, and we made interpretations with these results. This process can be extended with a learning-based controller design approach. Thus, tuning of the design parameters for the ST1-FLC and SIT2-FLC can be achieved with this learning-based method as forthcoming studies. On the other hand, the design parameters for SFLC structure can be tuned according to the aggressiveness of the trajectory in our next works. In this study; we designed the trajectories just by considering their velocity, acceleration and altitude bounds, and we used a timescale parameter β to show how much the trajectory is relatively aggressive/smooth. If we take velocity profile and curvature distributions into account rather than velocity and acceleration limits, we can define a parameter for the trajectory to calculate how much it is aggressive/smooth absolutely. Thus, we

can propose a design parameter tuning mechanism for differential flatness-based SFLC according to the absolute aggressiveness level of the trajectory in our future works.

REFERENCES

[1] **Jean-Noël Passieux** (2023). *Breguet Gyroplane 1*, `http://jnpassieux.fr/www/html/Gyroplane1.php`, accessed: 2023-01-12.

[2] **Thomas Edison NHP Historical Photograph Collection** (2023). *Helicopter designed by George de Bothezat, in flight*, `https://npgallery.nps.gov/AssetDetail/5b9c341f-6af2-43e2-9e37-7f6ea24ffc55`, accessed: 2023-01-14.

[3] **Leishman, J.G.** (2004). Development of the Autogiro: a technical perspective, *Journal of Aircraft*, *41*(4), 765–781.

[4] **Yu, F.**, **Chen, G.**, **Fan, N.**, **Song, Y. and Zhu, L.** (2017). Autonomous flight control law for an indoor UAV quadrotor, *2017 29th Chinese Control And Decision Conference (CCDC)*, pp.6767–6771.

[5] **Pensieri, M.G.**, **Garau, M. and Barone, P.M.** (2020). Drones as an integral part of remote sensing technologies to help missing people, *Drones*, *4*(2), 15.

[6] **Hamad, A.M. and Alaiwi, Y.** (2022). Localization and identification of UAVs system using image processing, *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*.

[7] **Mohammad Zain, Z.**, **Mohamad, N. and Mohamad Ali, Z.** (2002). Redesign of de Bothezat helicopter : the way forward for the rotorcraft industry, *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*.

[8] **Miličević, Z. and Bojković, Z.** (2021). From the early days of unmanned aerial vehicles (UAVs) to their integration into wireless networks, *Vojnotehnički glasnik*, *69*(4), 941–962.

[9] **Garcia Carrillo, L.R.**, **Dzul, A.**, **Lozano, R. and Pégard, C.** (2012). *Quad rotorcraft control. Vision-based hovering and navigation*, Springer London.

[10] **Palik, M. and Nagy, M.** (2019). Brief history of UAV development, *Repüléstudományi Közlemények*, *31*(1), 155–166.

[11] **Hassanalian, M. and Abdelkefi, A.** (2017). Classifications, applications, and design challenges of drones: a review, *Progress in Aerospace Sciences*, *91*, 99–131.

[12] **Al-Younes, Y.M.**, **Al-Jarrah, M.A. and Jhemi, A.A.** (2010). Linear vs. nonlinear control techniques for a quadrotor vehicle, *7th International Symposium on Mechatronics and its Applications*, pp.1–10.

[13] **Dikmen, I.C.**, **Arisoy, A. and Temeltas, H.** (2009). Attitude control of a quadrotor, *4th International Conference on Recent Advances in Space Technologies*, pp.722–727.

[14] **Lee, H. and Kim, H.J.** (2016). Trajectory tracking control of multirotors from modelling to experiments: a survey, *International Journal of Control, Automation and Systems*, *15*(1), 281–292.

[15] **Bitcraze AB** (2021). *Crazyflie 2.1 | Bitcraze*, `https://www.bitcraze.io/products/crazyflie-2-1/`, accessed: 2021-09-21.

[16] **Giernacki, W.**, **Skwierczyński, M.**, **Witwicki, W.**, **Wroński, P. and Kozierski, P.** (2017). Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering, *22nd International Conference on Methods and Models in Automation and Robotics*, pp.37–42.

[17] **Bansal, S.**, **Akametalu, A.K.**, **Jiang, F.J.**, **Laine, F. and Tomlin, C.J.** (2016). Learning quadrotor dynamics using neural network for flight control, *IEEE 55th Conference on Decision and Control*, pp.4653–4660.

[18] **Luis, C. and Ny, J.L.** (2016). Design of a trajectory tracking controller for a nanoquadcopter, *arXiv:1608.05786 [cs]*, `http://arxiv.org/abs/1608.05786`, arXiv: 1608.05786.

[19] **Kim, W.** (2022). *Crazyflie quadcopter simulation using Simmechanics*, `https://www.mathworks.com/matlabcentral/fileexchange/65469-crazyflie-quadcopter-simulation-using-simmechanics`, accessed: 2022-04-17.

[20] **Palossi, D.**, **Marongiu, A. and Benini, L.** (2017). Ultra low-power visual odometry for nano-scale unmanned aerial vehicles, *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp.1647–1650.

[21] **Faelden, G.E.**, **Maningo, J.M.**, **Nakano, R.C.**, **Bandala, A.**, **Vicerra, R.R. and Dadios, E.** (2016). Implementation of swarm aggregation in quadrotor swarms using an artificial potential function model, *2016 IEEE Region 10 Conference (TENCON)*, pp.2021–2026.

[22] **Nguyen Duc, M.**, **Trong, T.N. and Xuan, Y.S.** (2015). The quadrotor MAV system using PID control, *IEEE International Conference on Mechatronics and Automation*, pp.506–510.

[23] **Cowling, I.D.**, **Yakimenko, O.A.**, **Whidborne, J.F. and Cooke, A.K.** (2007). A prototype of an autonomous controller for a quadrotor UAV, *European Control Conference*, pp.4001–4008.

[24] **Argentim, L.M.**, **Rezende, W.C.**, **Santos, P.E. and Aguiar, R.A.** (2013). PID, LQR and LQR-PID on a quadcopter platform, *International Conference on Informatics, Electronics and Vision*, pp.1–6.

[25] **Garcia, G.A.**, **Kim, A.R.**, **Jackson, E.**, **Keshmiri, S.S. and Shukla, D.** (2017). Modeling and flight control of a commercial nano quadrotor, *International Conference on Unmanned Aircraft Systems*, pp.524–532.

[26] **Raffo, G.V.**, **Ortega, M.G. and Rubio, F.R.** (2008). MPC with nonlinear H_∞ control for path tracking of a quad-rotor helicopter, *IFAC Proceedings Volumes*, *41*(2), 8564–8569.

[27] **Raffo, G.V.**, **Ortega, M.G. and Rubio, F.R.** (2010). An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter, *Automatica*, *46*(1), 29–39.

[28] **Madani, T. and Benallegue, A.** (2006). Backstepping control for a quadrotor helicopter, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.3255–3260.

[29] **Joukhadar, A.**, **Alchehabi, M. and Jejeh, A.** (2020). Advanced UAVs nonlinear control systems and applications, *Unmanned Robotic Systems and Applications*.

[30] **Gautam, D. and Ha, C.** (2013). Control of a quadrotor using a smart self-tuning fuzzy PID controller, *International Journal of Advanced Robotic Systems*, *10*, 380.

[31] **Zhang, C.**, **Zhou, X.**, **Zhao, H.**, **Dai, A. and Zhou, H.** (2016). Three-dimensional fuzzy control of mini quadrotor UAV trajectory tracking under impact of wind disturbance, *International Conference on Advanced Mechatronic Systems*, pp.372–377.

[32] **Candan, F.**, **Beke, A. and Kumbasar, T.** (2018). Design and deployment of fuzzy PID controllers to the nano quadcopter Crazyflie 2.0, *Innovations in Intelligent Systems and Applications*, pp.1–6.

[33] **Kumbasar, T.** (2016). Robust stability analysis and systematic design of single-input interval type-2 fuzzy logic controllers, *IEEE Transactions on Fuzzy Systems*, *24*(3), 675–694.

[34] **Sarabakha, A.**, **Fu, C. and Kayacan, E.** (2019). Intuit before tuning: type-1 and type-2 fuzzy logic controllers, *Applied Soft Computing*, *81*, 105495.

[35] **Khooban, M.H. and Gheisarnejad, M.** (2021). A novel deep reinforcement learning controller based type-II fuzzy system: frequency regulation in microgrids, *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(4), 689–699.

[36] **Raj, R. and Mohan, B.** (2020). General structure of interval type-2 fuzzy PI/PD controller of Takagi–Sugeno type, *Engineering Applications of Artificial Intelligence*, *87*, 103273.

[37] **Kumbasar, T.** (2013). A simple design method for interval type-2 fuzzy PID controllers, *Soft Computing*, *18*(7), 1293–1304.

[38] **Beke, A. and Kumbasar, T.** (2018). Single vs. double input interval type-2 fuzzy PID controllers: which one is better?, *IEEE International Conference on Fuzzy Systems*, pp.1–7.

[39] **Sarabakha, A.**, **Fu, C.**, **Kayacan, E. and Kumbasar, T.** (2018). Type-2 fuzzy logic controllers made even simpler: from design to deployment for UAVs, *IEEE Transactions on Industrial Electronics*, *65*(6), 5069–5077.

[40] **Mellinger, D.**, **Michael, N. and Kumar, V.** (2014). Trajectory generation and control for precise aggressive maneuvers with quadrotors, Springer Berlin Heidelberg, pp.361–373.

[41] **Huang, H.**, **Hoffmann, G.**, **Waslander, S. and Tomlin, C.** (2009). Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering, *2009 IEEE International Conference on Robotics and Automation*.

[42] **Loianno, G.**, **Brunner, C.**, **McGrath, G. and Kumar, V.** (2017). Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU, *IEEE Robotics and Automation Letters*, *2*(2), 404–411.

[43] **Camci, E. and Kayacan, E.** (2019). Learning motion primitives for planning swift maneuvers of quadrotor, *Autonomous Robots*, *43*(7), 1733–1745.

[44] **Naldi, R.**, **Furci, M.**, **Sanfelice, R.G. and Marconi, L.** (2013). Global trajectory tracking for underactuated VTOL aerial vehicles using a cascade control paradigm, *52nd IEEE Conference on Decision and Control*, pp.4212–4217.

[45] **Furci, M.**, **Casadei, G.**, **Naldi, R.**, **Sanfelice, R. and Marconi, L.** (2015). An open-source architecture for control and coordination of a swarm of micro-quadrotors, *International Conference on Unmanned Aircraft Systems*, pp.139–146.

[46] **Chen, Y. and Pérez-Arancibia, N.O.** (2016). Generation and real-time implementation of high-speed controlled maneuvers using an autonomous 19-gram quadrotor, *IEEE International Conference on Robotics and Automation*, pp.3204–3211.

[47] **Chen, Y. and Pérez-Arancibia, N.O.** (2017). Lyapunov-based controller synthesis and stability analysis for the execution of high-speed multi-flip quadrotor maneuvers, *American Control Conference*, pp.3599–3606.

[48] **Faessler, M.**, **Franchi, A. and Scaramuzza, D.** (2018). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories, *IEEE Robotics and Automation Letters*, *3*(2), 620–626.

[49] **Tal, E. and Karaman, S.** (2021). Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness, *IEEE Transactions on Control Systems Technology*, *29*(3), 1203–1218.

[50] **Ferrin, J.**, **Leishman, R.**, **Beard, R. and McLain, T.** (2011). Differential flatness based control of a rotorcraft for aggressive maneuvers, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.2688–2693.

[51] **Mellinger, D. and Kumar, V.** (2011). Minimum snap trajectory generation and control for quadrotors, *IEEE International Conference on Robotics and Automation*, pp.2520–2525.

[52] **Greeff, M. and Schoellig, A.P.** (2018). Flatness-based model predictive control for quadrotor trajectory tracking, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.6740–6745.

[53] **Zeng, J.**, **Kotaru, P. and Sreenath, K.** (2019). Geometric control and differential flatness of a quadrotor UAV with load suspended from a pulley, *American Control Conference*, pp.2420–2427.

[54] **Landry, B.**, **Deits, R.**, **Florence, P.R. and Tedrake, R.** (2016). Aggressive quadrotor flight through cluttered environments using mixed integer programming, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp.1469–1475.

[55] **Deits, R. and Tedrake, R.** (2015). Efficient mixed-integer planning for UAVs in cluttered environments, *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[56] **Preiss, J.A.**, **Honig, W.**, **Sukhatme, G.S. and Ayanian, N.** (2017). Crazyswarm: a large nano-quadcopter swarm, *2017 IEEE International Conference on Robotics and Automation (ICRA)*.

[57] **STMicroelectronics** (2023). *STM32F405/415 - STMicroelectronics*, `https://www.st.com/en/microcontrollers-microprocessors/stm32f405-415.html`, accessed: 2023-01-12.

[58] **Nordic Semiconductor** (2023). *nRF51822 - Bluetooth low energy, ANT and 2.4 GHz SoC - nordicsemi.com*, `https://www.nordicsemi.com/products/nrf51822`, accessed: 2023-01-12.

[59] **Bitcraze AB** (2023). *Active marker deck - Bitcraze store*, `https://store.bitcraze.io/collections/decks/products/active-marker-deck`, accessed: 2023-01-12.

[60] **Bitcraze AB** (2023). *AI-deck 1.1 - Bitcraze store*, `https://store.bitcraze.io/collections/decks/products/ai-deck-1-1`, accessed: 2023-01-12.

[61] **Bitcraze AB** (2023). *Flow deck v2 - Bitcraze store*, https://store.bitcraze.io/collections/decks/products/flow-deck-v2, accessed: 2023-01-12.

[62] **Bitcraze AB** (2021). *GitHub - bitcraze/crazyflie-lib-python: Python library to communicate with Crazyflie*, https://github.com/bitcraze/crazyflie-lib-python, accessed: 2021-09-21.

[63] **Bitcraze AB** (2023). *GitHub - bitcraze/crazyflie-clients-python: host applications and library for Crazyflie written in Python*, https://github.com/bitcraze/crazyflie-clients-python, accessed: 2023-01-20.

[64] **Mueller, M.W.**, **Hamer, M. and D'Andrea, R.** (2015). Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp.1730–1736.

[65] **Mueller, M.W.**, **Hehn, M. and D'Andrea, R.** (2016). Covariance correction step for Kalman filtering with an attitude, *Journal of Guidance, Control, and Dynamics*, 1–7.

[66] **Bitcraze AB** (2022). *GitHub - bitcraze/crazyflie-firmware: The main firmware for the Crazyflie nano quadcopter, Crazyflie Bolt quadcopter and Roadrunner positioning tag.*, https://github.com/bitcraze/crazyflie-firmware, accessed: 2022-04-17.

[67] **Mochnac, J.**, **Marchevsky, S. and Kocan, P.** (2009). Bayesian filtering techniques: Kalman and extended Kalman filter basics, *2009 19th International Conference Radioelektronika*, pp.119–122.

[68] **Smeur, E.J.J.**, **Chu, Q. and de Croon, G.C.H.E.** (2016). Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles, *Journal of Guidance, Control, and Dynamics*, *39*(3), 450–461.

[69] **Adams, M.J.**, (1984). Aristotle's logic, volume 18 of *Psychology of Learning and Motivation*, Academic Press, pp.255–311.

[70] **Mendel, J.M.** (2017). *Uncertain rule-based fuzzy systems: introduction and new directions, 2nd edition*, Springer.

[71] **Zadeh, L.** (1965). Fuzzy sets, *Information and Control*, *8*(3), 338–353.

[72] **Zadeh, L.** (1975). The concept of a linguistic variable and its application to approximate reasoning—I, *Information Sciences*, *8*(3), 199–249.

[73] **Precup, R.E. and Hellendoorn, H.** (2011). Survey paper: a survey on industrial applications of fuzzy control, *Comput. Ind.*, *62*(3), 213–226.

[74] **Yesil, E. and Guzay, C.** (2014). A self-tuning fuzzy PID controller design using gamma aggregation operator, *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp.2032–2038.

[75] **Torshizi, A.D.**, **Zarandi, M.H. and Zakeri, H.** (2015). On type-reduction of type-2 fuzzy sets: a review, *Applied Soft Computing*, *27*, 614–627.

[76] **Karnik, N.N. and Mendel, J.M.** (2001). Centroid of a type-2 fuzzy set, *Information Sciences*, *132*(1-4), 195–220.

[77] **Karnik, N. and Mendel, J.** (1998). Type-2 fuzzy logic systems: type-reduction, *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, volume 2, pp.2046–2051.

[78] **Kumbasar, T.** (2015). Revisiting KM algorithms: a linear programming approach, *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*.

[79] **Sakallı, A.** (2020). *Analysis and design of general type-2 fuzzy logic controllers*, Istanbul Technical University.

[80] **Var, A.**, **Kumbasar, T. and Yesil, E.** (2015). An internal model control based design method for single input fuzzy PID controllers, *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp.1–7.

[81] **Fliess, M.**, **Lévine, J.**, **Martin, P. and Rouchon, P.** (1995). Flatness and defect of non-linear systems: introductory theory and examples, *International Journal of Control*, *61*(6), 1327–1361.

[82] **Murray, R.M.**, **Rathinam, M. and Sluis, W.** (1995). Differential flatness of mechanical control systems: a catalog of prototype systems, *Proceedings of the 1995 ASME International Congress and Exposition*.

[83] **Greiff, M.** (2017). *Modelling and control of the Crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation*, Department of Automatic Control, Lund University.

[84] **Kumar, R.**, **Nemati, A.**, **Kumar, M.**, **Sharma, R.**, **Cohen, K. and Cazaurang, F.** (2017). Tilting-rotor quadcopter for aggressive flight maneuvers using differential flatness based flight controller, *ASME Dynamic Systems and Control Conference*, volume 3.

[85] **Hilbert, D.** (1912). Über den begriff der klasse von differentialgleichungen, *Mathematische Annalen*, *73*(1), 95–108.

[86] **Cartan, M.** (1915). Sur l'intégration de certains systèmes indéterminés d'équations différentielles, *1915*(145), 86–91.

[87] **Rigatos, G.G.** (2015). Differential flatness theory and flatness-based control, Springer International Publishing, pp.47–101.

[88] **python-control** (2023). *Differentially flat systems*, `https://python-control.readthedocs.io/en/latest/flatsys.html`, accessed: 2023-01-20.

[89] **Mellinger, D.** (2012). *Trajectory generation and control for quadrotors*, Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania.

[90] **Guzay, C. and Kumbasar, T.** (2022). Aggressive maneuvering of a quadcopter via differential flatness-based fuzzy controllers: from tuning to experiments, *Applied Soft Computing*, *126*, 109223.

[91] **Blanco-Claraco, J.L.** (2021). A tutorial on SE(3) transformation parameterizations and on-manifold optimization, *arXiv:2103.15980 [cs]*, `http://arxiv.org/abs/2103.15980`, arXiv: 2103.15980.

APPENDICES

APPENDIX A : Supplementary Data
APPENDIX B : Source Code of the Proposed Method
APPENDIX C : Waypoint Matrices of Generated Trajectories

APPENDIX A : Supplementary Data

Supplementary material related to this study can be found online at following links:

1. https://doi.org/10.1016/j.asoc.2022.109223
2. https://www.youtube.com/watch?v=sePZLpZq0xY.

APPENDIX B : Source Code of the Proposed Method

```
/*File: controller_proposed.h*/
#ifndef __CONTROLLER_PROPOSED_H__
#define __CONTROLLER_PROPOSED_H__
#include "stabilizer_types.h"
void controllerProposedInit(void);
bool controllerProposedTest(void);
void controllerProposed(control_t *control, setpoint_t *setpoint,
    const sensorData_t *sensors,
    const state_t *state,
    const uint32_t tick);
#endif //__CONTROLLER_PROPOSED_H__
```

Figure B.1 : Proposed controller declaration.

```
/*File: controller_proposed.c*/
// ............
// mx and my are inputs for roll and pitch SFLCs respectively
// mx and my are calculated previously
mx = mx / 1000.0f; // additional input scaling factor is applied
my = my / 1000.0f;
// SFLC type select
// (0 => ST1-FLC-5R, 1 => SIT2-FLC-3R, 2 => SIT2-FLC-5R)
if (sflc_select == 0)
{
  mx = runST1(mx, 0.0f, 1.0f, 0.0f, 0.03125f,
        1.0f, u_c2xy, u_b2xy);
  my = runST1(my, 0.0f, 1.0f, 0.0f, 0.03125f,
        1.0f, u_c2xy, u_b2xy);
}
else if (sflc_select == 1)
{
  mx = runSIT2_3R(mx, 0.0f, 1.0f, 0.0f, 0.03125f,
        1.0f, u_alpha);
  my = runSIT2_3R(my, 0.0f, 1.0f, 0.0f, 0.03125f,
        1.0f, u_alpha);
}
else if (sflc_select == 2)
{
  mx = runSIT2_5R(mx, 0.0f, 1.0f, 0.0f, 0.03125f, 1.0f,
      sit2_5RuB1, sit2_5Ruc1, sit2_5RuB2, sit2_5Rum0,
      sit2_5Rum1, sit2_5Rum2);
  my = runSIT2_5R(my, 0.0f, 1.0f, 0.0f, 0.03125f, 1.0f,
      sit2_5RuB1, sit2_5Ruc1, sit2_5RuB2, sit2_5Rum0,
      sit2_5Rum1, sit2_5Rum2);
}
else
{
  mx = 0;
  my = 0;
}
control->roll = mx * 1000; // additional output scaling factor
control->pitch = my * 1000;
// .... to the power distribution block
```

Figure B.2 : Proposed controller implementation.

```
/*File: sflc.h*/
#include <math.h>
#include "math3d.h"
#ifndef __SFLC_H__
#define __SFLC_H__
float runST1(float error, float derror, float e_gain,
        float ed_gain, float kds, float kc, float c, float b);
float runSIT2_3R(float error, float derror, float e_gain,
        float de_gain, float kds, float kc, float alpha);
float runSIT2_5R(float error, float derror, float e_gain,
        float de_gain, float kds, float kc,float B1,
        float c1, float B2, float m0, float m1, float m2);
float doSIFLCinputScaling(float error, float derror,
        float e_gain, float de_gain, float kds);
float doSIFLCoutputScaling(float u, float kds, float kc);
#endif //__SFLC_H__
```

Figure B.3 : Proposed controller helper declaration.

```
/*File: sflc.c*/
float runST1(float error, float derror, float e_gain,
        float de_gain, float kds, float kc, float c, float b)
{
  // Input & output mf. parameters wrt.
  float c1, c2, c3, c4, c5;
  float b1, b2, b4, b5; // b3
  float Kf, Tof;
  float ds;
  float u;
  float output;
  Kf = 0;
  Tof = 0;
  ds = doSIFLCinputScaling(error, derror, e_gain, de_gain, kds);
  c1 = -1;
  c2 = c;
  c3 = 0;
  c4 = fabsf(c2);
  c5 = 1;
  b1 = -1;
  b2 = b;
  // b3 = 0;
  b4 = fabsf(b2);
  b5 = 1;
  // Saturate input for [c1, c5] interval
  ds = clamp(ds, -1, 1);
  // Calculate output according to input region
  if(ds >= c1 && ds <= c2)
  {
    Kf = (b2 - b1) / (c2 - c1);
    Tof = (b2*c1 - b1*c2) / (c2 - c1);
  }
  else if(ds > c2 && ds <= c3)
  {
    Kf = (b2 / c2);
    Tof = 0;
  }
  else if(ds > c3 && ds <= c4)
  {
    Kf = (b4 / c4);
    Tof = 0;
  }
  else if(ds > c4 && ds <= c5)
  {
    Kf = (b5 - b4) / (c5 - c4);
    Tof = (b5*c4 - b4*c5) / (c5 - c4);
  }
  u = Kf * ds - Tof;
  output = doSIFLCoutputScaling(u, kds, kc);
  return output;
}
```

Figure B.4 : ST1-FLC implementation.

```
/*File: sflc.c*/
float runSIT2_3R(float error, float derror, float e_gain,
        float de_gain, float kds, float kc, float alpha)
{
  float ds;
  float output;
  float k;
  float phi;
  float ds_abs;
  ds = doSIFLCinputScaling(error, derror, e_gain, de_gain, kds);
  // Saturate input if it remains out of bounds
  ds = clamp(ds, -1, 1);
  ds_abs = fabsf(ds);
  k = 1 / (alpha + ds_abs - alpha * ds_abs);
  k += (alpha - 1) / (alpha * ds_abs - 1);
  k = k * 0.5f;
  phi = ds_abs * k;
  if(ds < 0) {
    phi = -phi;
  }
  output = doSIFLCoutputScaling(phi, kds, kc);
  return output;
}
```

Figure B.5 : SIT2-FLC-3R implementation.

```
/*File: sflc.c*/
float runSIT2_5R(float error, float derror, float e_gain,
    float de_gain, float kds, float kc, float B1,
    float c1, float B2, float m0, float m1, float m2)
{
  float c2 = 1;
  float phi = 0;
  float eta = 0;
  short sigma_sign = 1;
  float ds;
  float output;
  float k;
  ds = doSIFLCinputScaling(error, derror, e_gain, de_gain, kds);
  // Saturate input if it remains out of bounds
  ds = clamp(ds, -1, 1); //-c2, c2
  if(ds < 0) {
    sigma_sign = -1;
    ds = fabsf(ds);
  }
  if (ds < c1) {
    k = 0.5f * (B1/(ds - ds * m0 + c1 * m0)
    - B1*m1/(ds - c1 - ds * m1));
    phi = k * ds * sigma_sign;
  } else if(ds >= c1 && ds <= c2) {
    k = 0.5f * ((B2 - B1 * m1)/(c2 * m1 - c1 + ds*(-m1 + 1))
    + (B1 - B2 * m2)/(c1 * m2 - c2 + ds*(-m2 + 1)));
    eta = 0.5f * (
    (B2 * c1 - B1 * c2 * m1)/(-c2 * m1 + c1 + ds*(m1 - 1))
    + (B1 * c2 - B2 * c1 * m2)/(-c1 * m2 + c2 + ds*(m2 - 1)));
    phi = (k * ds + eta) * sigma_sign;
  }
  output = doSIFLCoutputScaling(phi, kds, kc);
  return output;
}
```

Figure B.6 : SIT2-FLC-5R implementation.

```
/*File: sflc.c*/
float doSIFLCinputScaling(float error, float derror,
        float e_gain, float de_gain, float kds)
{
  float ds;
  ds = error * e_gain + derror * de_gain;
  ds = ds * kds;
  return ds;
}

float doSIFLCoutputScaling(float u, float kds, float kc)
{
  float output;
  u = u * 1.0f / kds;
  output = u * kc;
  return output;
}
```

Figure B.7 : SFLC input and output scaling implementations.

APPENDIX C : Waypoint Matrices of Generated Trajectories

```
trajectory_1 = [
    [
        1.050000, 0, 0, 0, 0, 0.830443, -0.276140,
        -0.384219, 0.180493, 0, 0, 0, 0, -1.356107,
        0.688430, 0.587426, -0.329106, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    [
        0.710000, 0.396058, 0.918033, 0.128965,
        -0.773546, 0.339704, 0.034310, -0.026417,
        -0.030049, -0.445604, -0.684403, 0.888433,
        1.493630, -1.361618, -0.139316, 0.158875,
        0.095799, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0
    ],
    [
        0.620000, 0.922409, 0.405715, -0.582968,
        -0.092188, -0.114670, 0.101046, 0.075834,
        -0.037926, -0.291165, 0.967514, 0.421451,
        -1.086348, 0.545211, 0.030109, -0.050046,
        -0.068177, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0
    ],
    [
        0.700000, 0.923174, -0.431533, -0.682975,
        0.177173, 0.319468, -0.043852, -0.111269,
        0.023166, 0.289869, 0.724722, -0.512011,
        -0.209623, -0.218710, 0.108797, 0.128756,
        -0.055461, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0
    ],
    [
        0.560000, 0.405364, -0.834716, 0.158939,
        0.288175, -0.373738, -0.054995, 0.036090,
        0.078627, 0.450742, -0.385534, -0.954089,
        0.128288, 0.442620, 0.055630, -0.060142,
        -0.076163, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0
    ],
    [
        0.560000, 0.001062, -0.646270, -0.012560,
```

```
        -0.324065, 0.125327, 0.119738, 0.034567,
        -0.063130, 0.001593, -1.031457, 0.015159,
        0.820816, -0.152665, -0.130729, -0.045679,
        0.080444, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0
    ],
    [
        0.700000, -0.402804, -0.820508, -0.132914,
        0.236278, 0.235164, -0.053551, -0.088687,
        0.031253, -0.449354, -0.411507, 0.902946,
        0.185335, -0.239125, -0.041696, 0.016857,
        0.016709, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0
    ],
    [
        0.620000, -0.921641, -0.464596, 0.661875,
        0.286582, -0.228921, -0.051987, 0.004669,
        0.038463, -0.292459, 0.777682, 0.565788,
        -0.432472, -0.060568, -0.082048, -0.009439,
        0.041158, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0
    ],
    [
        0.710000, -0.923935, 0.447832, 0.627381,
        -0.259808, -0.042325, -0.032258, 0.001420,
        0.005294, 0.288570, 0.873350, -0.515586,
        -0.730207, -0.026023, 0.288755, 0.215678,
        -0.148061, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0
    ],
    [
        1.053185, -0.398611, 0.850510, -0.144007,
        -0.485368, -0.079781, 0.176330, 0.234482,
        -0.153567, 0.447039, -0.532729, -0.855023,
        0.878509, 0.775168, -0.391051, -0.713519,
        0.391628, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0
    ]
]

trajectory_2 = [
    [
        2, 0, 0, 0, 0, 2.625, -3.4125, 1.4875,
        -0.21875, 0, 0, 0, 0, -0.4375, 0.7875,
        -0.39375, 0.0625, 1.0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
```

```
    [
        4, 0, -1.4, 0, 0, 0.54140, -0.296953,
        0.057558, -0.003886, 1.0, 1.4, 0, 0,
        -0.798437, 0.490546, -0.104453, 0.007587,
        1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0
    ],
    [
        2, 1.0, -0.7, -0.63, 0, 1.1375, -0.91875,
        0.284375, -0.031875, 1.0, 0.7, -0.63, 0,
        -2.3625, 3.01875, -1.290624, 0.186875,
        1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0
    ]
]

trajectory_3 = [
    [
        2, 0, 0, 0, 0, -0.18375, 0.349125, -0.196,
        0.0336875, 0, 0, 0, 0, 0.16625, 0.112874,
        -0.1435, 0.030562, 1.0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    [
        2.2, 0, -0.49, -0.882, 0, 1.297887, -1.075583,
        0.3501, -0.041371, 1.0, 0.49, -0.882, 0,
        -1.050824, 1.2715, -0.490049, 0.063505, 1.0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    [
        2.2, -1.0, 0.35, 0.525, 0, 1.177045, -1.45079,
        0.569508, -0.074914, -1.0, -0.35, 0.525, 0,
        -0.38278, 0.465993, -0.205443, 0.029962, 1.0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    [
        2, 1.0, -0.49, -0.735, 0, 0.875, -0.590625,
        0.146562, -0.0121875, 0, 0.49, -0.735, 0,
        0.6125, -0.459375, 0.137812, -0.0153125,
        1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0
    ]
]

trajectory_4 = [
    [
        2, 0, 0, 0, 0, -1.6625, 2.02125, -0.861875,
```

```
        0.125625, 0, 0, 0, 0, 0.0875, 0.18375,
        −0.161875, 0.031875, 1.0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    [
        4, −1.0, −0.7, −0.63, 0, 0.843281, −0.44346,
        0.084328, −0.005633, 1.0, 0.7, −0.63, 0,
        −0.294218, 0.24396, −0.0595, 0.004668, 1.0,
        0, 0, 0, 0.19277, −0.135351, 0.031479,
        −0.002424, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    [
        4, 1.0, −0.91, −0.819, 0, 0.659531, −0.333867,
        0.06289, −0.0042, 1.0, 0.91, −0.82, 0, 0.090781,
        −0.013945, −0.00109375, 0.000243, 0.97, 0, 0.63,
        0, −0.389648, 0.194414, −0.036401, 0.002424, 0,
        0, 0, 0, 0, 0, 0, 0
    ]
]
```

CURRICULUM VITAE

Name Surname: Çağrı GÜZAY

EDUCATION:

- **B.Sc.:** 2012, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Control Engineering Department.
- **M.Sc.:** 2014, Istanbul Technical University, Graduate School of Science Engineering and Technology, Department of Control and Automation Engineering.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Guzay, C.**, and Kumbasar, T. (2022). Aggressive maneuvering of a quadcopter via differential flatness-based fuzzy controllers: from tuning to experiments. *Applied Soft Computing*.

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Yesil, E., and **Guzay, C.** (2014). A self-tuning fuzzy PID controller design using gamma aggregation operator. *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*.
- Yesil, E., Savran, A.I., and **Guzay, C.** (2014). Load frequency controller design using new big bang - big crunch 2 algorithm. *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*.
- Yesil, E, Dodurka, M.F., Sakalli, A., Ozturk, C., and **Guzay, C.** (2013). Self-tuning PI controllers via fuzzy cognitive maps. *IFIP Advances in Information and Communication Technology*.
- Dodurka, M.F., Yesil, E., Ozturk, C., Sakalli, A., and **Guzay, C.** (2013). Concept by concept learning of fuzzy cognitive maps. *IFIP Advances in Information and Communication Technology*.
- Yesil, E., Sakalli, A., and **Guzay, C.** (2012). Modeling and control of fan and plate system. *Otomatik Kontrol Ulusal Kongresi*.