T.C.
ISTANBUL UNIVERSITY-CERRAHPASA
INSTITUTE OF GRADUATE STUDIES

Ph.D. THESIS

CONTROL ALGORITHMS APPLIED TO UAV FOR SEARCH AND RESCUE OVER WATER

Fady M. ALALAMI

SUPERVISOR
Prof. Dr. Abdul Rahman HUSSIAN

Department of Electrical and Electronic Engineering

Electrical and Electronic Engineering Programme

ISTANBUL- July, 2020

This study was accepted on 28/7/2020 as a Ph. D. thesis in Department of Electrical and Electronic Engineering, Electrical and Electronic Engineering Programme by the following Committee.

Examining Committee Members

Prof. Dr. Abdul Rahman HUSSIAN(Supervisor)
İstanbul University-Cerrahpaşa
Faculty of Engineering

Assoc. Prof. Dr. Sedat BALLIKAYA
İstanbul University-Cerrahpaşa
Faculty of Engineering

Prof. Dr. Naim AJLOUNI
İstanbul Aydin University
Faculty of Engineering

Assist. Prof. Dr. Rana ORTAÇ KABAOĞLU
İstanbul University-Cerrahpaşa
Faculty of Engineering

Assoc. Prof. Dr. Moustapha ALHAJDIBO
Atılım University
Faculty of Engineering

As required by the 9/2 and 22/2 articles of the Graduate Education Regulation which was published in the Official Gazette on 20.04.2016, this graduate thesis is reported as in accordance with criteria determined by the Institute of Graduate Studies by using the plagiarism software to which Istanbul University-Cerrahpasa is a subscriber.

FOREWORD

In the beginning, I thank ALLAH for giving me the strength and health to let this work see light. I dedicate this thesis to my beloved father, Mamoun ALALAMI, my beloved mother, Wafaa ALALAMI, and my beloved aunt, Mona ALMUHTADI in recognition of their endless help, emotions, and support. I also dedicate this work to my lovely wife, Nada ALRAYYES and my lovely little daughter, TULIP for their love, patience, and care; I also dedicate this work to my beloved brother, Omar ALALAMI, and beloved sister, Reem ALALAMI for inspiring me and pushing me to achieve my goals. I am also very grateful to Dr. Sabbah ALALAMI to his scientific advice and suggestions. Words will not be enough to thank my family for their patience and encouragement during my thesis.

I thank my supervisor Prof. Dr. Abdul Rahman HUSSIAN for the time, consideration, suggestions, ideas and advice he gave me during this thesis. Special thanks go to my co-supervisor committee, Prof. Dr. Naim AJLOUNI and Prof. Dr. Sedat BALLIKAYA for their patience, guidance, and generous supports during this research.

July 2020 Fady M. ALALAMI

TABLE OF CONTENTS

Page

FOREWORD ..... **iv**
TABLE OF CONTENTS ..... **v**
LIST OF FIGURES ..... **viii**
LIST OF TABLES ..... **xiii**
LIST OF SYMBOLS AND ABBREVIATIONS ..... **xiv**
ÖZET ..... **xv**
SUMMARY ..... **xvii**
1. INTRODUCTION **1**
1.1. OVERVIEW 1
1.2. STATEMENT OF THE PROBLEM 3
1.3. OBJECTIVES 4
1.4. MOTIVATION 4
1.5. METHODOLOGY 4
1.6. CONTRIBUTION 5
1.7. OUTLINE OF THE THESIS 5
1.8. LITERATURE REVIEW 6
2. GENERAL SECTION **27**
2.1. OVERVIEW 27
2.2. CONTROL THEORY 27
2.2.1. Open Loop and Closed Loop Systems 27
2.2.2. Time Domain and Frequency Domain Representation 29
2.2.3. Stability Analysis Technique 29
2.2.4. Linear Control Systems 31
2.2.5. Non-Linear Control Systems 31
2.3. PID CONTROLLER 32
2.3.1. Proportional (P) Controller 33
2.3.2. Proportional-Integral (PI) Controller 34
2.3.3. Proportional-Derivative (PD) Controller 35
2.3.4. Proportional-Integral-Derivative (PID) Controller 36
2.4. GENETIC ALGORITHMS 38

2.4.1. Basic idea of GAs 39
2.4.2. Initial Population 40
2.4.2.1. Individual: *40*
2.4.2.2. Population *40*
2.4.2.3. Encoding *41*
2.4.3. Fitness Function 43
2.4.4. Selecting Parents 44
2.4.5. Crossover Operation 46
2.4.6. Mutation 48
2.4.7. Example for 8-queen problem 49
2.5. FUZZY LOGIC CONTROL 50
2.5.1. Historical Perspective 51
2.5.2. Fuzzy Logic and Fuzzy Sets 52
2.5.3. Fuzzy Set Operations 53
2.5.4. Membership Function 55
2.5.5. Fuzzy Logic Concept 56
2.5.5.1. Fuzzification *57*
2.5.5.2. Knowledge Base *57*
2.5.5.3. Fuzzy Inference Engine *59*
2.5.5.4. Defuzzification *62*
2.6. ARTIFICIAL NEURAL NETWORKS 62
2.6.1. ANN Learning Methods 63
2.6.2. Back-Propagation Algorithm 64
2.6.3. ANN Mathematical Model 65
3. MATERIALS AND METHODS 69
3.1. QUAD-ROTOR MODELING 69
3.1.1. Definitions of Quad-rotor's Variables 70
3.1.2. Transformation Between Frames 71
3.1.3. Translational and Rotational Kinematics 74
3.1.4. Translational and Rotational Dynamics 77
3.1.5. Quad-rotor Modeling Result 84
3.2. NON-INERTIAL FRAME OF REFERENCE 87
3.3. OPEN-LOOP SYSTEM STABILITY CHECKING 89
3.4. CONTROLLER DESIGN 90

3.4.1. Multi-Stage PID Controller 90
3.4.2. First Stage Position Control 91
3.4.3. Second Stage attitude and altitude Control 93
3.4.4. Third Stage Rotors Controller 94
3.5. GA OPTIMIZATION PROCESS 96
3.5.1. Chromosome Structure 97
3.5.2. Initial Population 97
3.5.3. Fitness Function 98
3.5.4. Reproduction Process 99
3.5.5. Adaptive Genetic Algorithm 100
3.6. ADAPTIVE CONTROL DESIGN 102
3.7. FUZZY CONTROLLER DESIGN 104
3.8. NEURAL NETWORKS DESIGN 109
3.9. DISCRETE CONTROLLER DESIGN 112
4. RESULTS 114
4.1. ADAPTIVE GENETIC ALGORITHM OPTIMIZATION RESULTS 114
4.2. NON-INERTIAL OF REFERENCE REPRESENTATION 119
4.3. FLIGHT SIMULATION DURING LINEAR WIND DISTURBANCE 120
4.4. FLIGHT SIMULATION DURING ANGULAR WIND DISTURBANCE 123
4.4.1. Controller Response Before and After Angular Wind Disturbance 123
4.4.2. AGA Optimization Process for Obtaining Stability 125
4.5. COMPARISON BETWEEN OPTIMUM AND ROBUST PID PARAMETERS 128
4.6. ADAPTIVE FUZZY-PID VS ADAPTIVE NN-PID CONTROLLER RESPONSES 129
4.7. ADAPTIVE DISCRETE CONTROLLER RESPONSE 133
5. DISCUSSION 135
6. CONCLUSION AND RECOMMENDATIONS 139
REFERENCES 141
APPENDICES 146
APPX. 1. $\boldsymbol{Kp}$ VALUES VERSUS DISTURBANCE VALUES 146
APPX. 2. $\boldsymbol{Ki}$ VALUES VERSUS DISTURBANCE VALUES 147
APPX. 3. GA OPTIMIZATION ITERATIONS 148
APPX. 4. MATLAB AND SIMULINK CODES 150
CURRICULUM VITAE 151

LIST OF FIGURES

Page

Figure 1.1: Open loop response of the position vector....8

Figure 1.2: Close loop response with PD controller of the position vector....9

Figure 1.3: Flow chart of the quad-copter mathematical model....9

Figure 1.4: The result of cascaded PD controller for position vector....13

Figure 1.5: (a) displacement results of the open loop response, (b) displacement response with classical PID, (c) displacement of the system with GA 17

Figure 1.6: Block diagram of the system and controller 19

Figure 1.7: Compares ion between simulation and experimental results for the angels (a) Roll, (b) Pitch and (c) Yaw 20

Figure 1.8: (a) Step response for Roll angle in simulation, (b) Response of Roll angle in flight test....23

Figure 1.9: Inverse control block diagram 24

Figure 1.10: Comparison between PD, Inverse, Backstepping, and Sliding Mode control algorithms for controlling quad-copter....25

Figure 2.1: (a) Open Loop Systems, (b) Closed Loop Systems 28

Figure 2.2: Closed loop response to step without controller 32

Figure 2.3: Closed loop system with P controller block diagram 33

Figure 2.4: Step response of DC motor controlled by P controller 33

Figure 2.5: Closed loop system with PI controller block diagram 34

Figure 2.6: Step response of DC motor controlled by PI controller....35

Figure 2.7: Closed loop system with PD controller block diagram....35

Figure 2.8: Step response of DC motor controlled by PD controller 36

Figure 2.9: Closed loop system with PID controller block diagram 37

Figure 2.10: Step response of DC motor controlled by PID controller....37

Figure 2.11: Sequence of Genetic Algorithm 39
Figure 2.12: Chromosome representation 40
Figure 2.13: Population Representation 41
Figure 2.14: Two chromosomes represented in Decimal numbers 41
Figure 2.15: Roulette Wheel Selecting Method 45
Figure 2.16: Stochastic Universal Sampling Technique 46
Figure 2.17: Single-Point Crossover Operation 47
Figure 2.18: Two-Point Crossover operation 47
Figure 2.19: Uniform Crossover operation 48
Figure 2.20: (a) crisp set and (b) fuzzy set 52
Figure 2.21: Fuzzy sets A and B 54
Figure 2.22: Intersection Operation for two fuzzy sets 54
Figure 2.23: Union Operation for two fuzzy sets 55
Figure 2.24: Complement Operation for one fuzzy set 55
Figure 2.25: Different shapes of membership functions 56
Figure 2.26: Fuzzy Logic Process 56
Figure 2.27: Fuzzification methodology 57
Figure 2.28: Fuzzification process 60
Figure 2.29: Rules implementation results 62
Figure 2.30: ANN for autonomous vehicle project 64
Figure 2.31: General structure of artificial neural cell 65
Figure 2.32: General structure of Artificial Neural Network 66
Figure 2.33: Neural node evaluation process 66
Figure 2.34: Back-Propagation error evaluation 67
Figure 2.35: General structure of the Neural Network 68
Figure 3.1: Vector projections 72
Figure 3.2: Roll-Pitch-Yaw rotations 73

Figure 3.3: Explanation of angular velocity notation 75
Figure 3.4: Rotors spins in opposite direction 82
Figure 3.5: Quad-rotor "×" and "+" structures 83
Figure 3.6: System Dynamics Block Inputs & Outputs 86
Figure 3.7: S-Function general structure 86
Figure 3.8: System Dynamics Block for NIFR referencing 89
Figure 3.9: System Open-Loop stability checking 90
Figure 3.10: The structure of the UAV System and controller 91
Figure 3.11: The structure of First Stage PID controller 92
Figure 3.12: The structure of first stage position controller 93
Figure 3.13: The structure of Second Stage, attitude and altitude controllers 94
Figure 3.14: General structure of motors controller 94
Figure 3.15: Motor/Angles relation 95
Figure 3.16: The structure of Motor's system and controller 96
Figure 3.17: PID random values constraints 98
Figure 3.18: Adaptive random gain value range 101
Figure 3.19: Best $\boldsymbol{Kp}$ gain values vs. disturbances 103
Figure 3.20: Best Ki gain values vs. disturbances 103
Figure 3.21: Membership functions of fuzzification process for $\boldsymbol{Kp}$ gain values 105
Figure 3.22: Membership functions of defuzzification process for $\boldsymbol{Kp}$ gain values 105
Figure 3.23: Best $\boldsymbol{Kp}$ gain value vs disturbance using Fuzzy controller 106
Figure 3.24: Membership functions of fuzzification process for $\boldsymbol{Ki}$ gain values 107
Figure 3.25: Membership functions of defuzzification process for $\boldsymbol{Ki}$ gain values 107
Figure 3.26: Best $\boldsymbol{Ki}$ gain value vs disturbance using Fuzzy controller 108
Figure 3.27: Adaptive Fuzzy-PID controller for altitude control 109
Figure 3.28: Neural Network controller internal structure 110

Figure 3.29: Neural Network training error vs number of iterations.....111
Figure 3.30: Neural Network vs Fuzzy results for $\boldsymbol{Kp}$ Gain values.....111
Figure 3.31: Neural Network vs Fuzzy results for $\boldsymbol{Ki}$ Gain values.....111
Figure 3.32: Adaptive NN-PID controller for altitude control.....112
Figure 3.33: Discretization of controller's inputs and outputs.....113
Figure 3.34: X-axis discrete PID controller.....113
Figure 4.1: Flight path represented in x-axis and y-axis.....115
Figure 4.2: Z-axis controller response.....116
Figure 4.3: Phi angle controllers' commands and responses.....117
Figure 4.4: Theta angle controllers' commands and responses.....118
Figure 4.5: Psi angle controllers' command and responses.....119
Figure 4.6: The result of relating quad-rotor with moving frame.....119
Figure 4.7: Linear wind disturbance effect on quad-rotor planned path.....121
Figure 4.8: Linear wind disturbance effect on x-axis controller.....122
Figure 4.9: Linear wind disturbance effect on Theta angle controller.....122
Figure 4.10: Angular wind disturbance effect on x-axis controller.....123
Figure 4.11: Angular wind disturbance effect on x-axis controller.....124
Figure 4.12: Angular wind disturbance effect on Psi angle controller.....124
Figure 4.13: AGA iterations to maintain stability for x-axis controller.....125
Figure 4.14: AGA iterations to maintain stability for y-axis controller.....125
Figure 4.15: AGA iterations to maintain stability for Theta angle controller.....126
Figure 4.16: AGA iterations to maintain stability for Psi angle controller.....127
Figure 4.17: AGA iterations to maintain stability for z-axis controller.....127
Figure 4.18: Z-controller response in different cases.....128
Figure 4.19: Adaptive Fuzzy-PID and NN-PID controllers for z-axis.....129
Figure 4.20: Quad-rotor planned path controlled by Adaptive controllers.....130
Figure 4.21: Adaptive controllers' response for Psi angle.....131

Figure 4.22: Adaptive controllers' response for Theta angle 131
Figure 4.23: Y-axis adaptive controller response for different disturbance values.....132
Figure 4.24: Z-axis adaptive controller response for different disturbance values 132
Figure 4.25: Comparison between different sample times for discrete controller 133
Figure 4.26: Controller response with load 134

LIST OF TABLES

Page

Table 2.1: Binary number representation. 42

Table 2.2: Binary number example. 42

Table 2.3: Chromosome consist of Octal numbers.....42

Table 2.4: Chromosome consist of Hexadecimal numbers. 43

Table 2.5: Sample Problem String and Fitness Value 45

Table 2.6: Sample of Initial Population for 8-Queen Problem.....49

Table 2.7: GA results for 8-Queen problem 50

Table 3.1: UAV modeling terms and symbols 71

Table 3.2: Vectors projections.....71

Table 3.3: GA Chromosome structure.....97

Table 3.4: Comparison between Conventional GA and Adaptive GA.....101

Table 3.5: Disturbance vs PID parameters 102

Table 3.6: Fuzzy control rules for $\boldsymbol{Kp}$ gain values.....106

Table 3.7: Fuzzy control rules for $\boldsymbol{Ki}$ gain values.....108

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Explanation
$\boldsymbol{\xi}$	: Absolute linear position of the quad-rotor defined in the inertial frame.
$\boldsymbol{\eta}$	: Angular position referenced to the inertial frame.
$\boldsymbol{V_b}$	: Linear velocities of the body frame.
$\dot{\boldsymbol{\xi}}, \boldsymbol{V_i}$	: Linear velocities of the body expressed in the inertial frame coordinate.
$\boldsymbol{\omega}$	: Angular velocities of the body frame.
$\dot{\boldsymbol{\eta}}$	: Angular velocities of the body frame expressed in the inertial frame.
$\ddot{\boldsymbol{\xi}}, \dot{\boldsymbol{V}}_{\boldsymbol{\iota}}$	: Linear acceleration of the body frame expressed in the inertial frame.
$\dot{\boldsymbol{\omega}}$	: Angular acceleration of the body frame.
$\ddot{\boldsymbol{\eta}}$	: Angular acceleration of the body frame expressed in the inertial frame.
$\boldsymbol{\varphi}, \boldsymbol{\theta}, \boldsymbol{\psi}$	: Phi, Theta, and Psi angles respectively.
$\dot{\boldsymbol{\varphi}}, \dot{\boldsymbol{\theta}}, \dot{\boldsymbol{\psi}}$	: Angualar velocities of Phi, Theta, and Psi angles respectively of inertial frame.
$\boldsymbol{\tau_{\varphi}}, \boldsymbol{\tau_{\theta}}, \boldsymbol{\tau_{\psi}}$	: Torque applied to Phi, Theta, and Psi angles respectively.
$\dot{\boldsymbol{p}}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{r}}$	:Angualar velocities of Phi, Theta, and Psi angles respectively of body frame.

Abbreviation	Explanation
UAV	: Unmanned Aerial Vehicle.
IFR	: Inertial Frame of Reference.
NIFR	: Non-Inertial Frame of Reference.
GA	: Genetic Algorithm.
AGA	: Adaptive Genetic Algorithm.
IMU	: Inertial Measurement Unit.
PID	: Proportional, Integral, Derivative.
ANN	: Artificial Neural Networks.
DOF	: Degree of Freedom.
rpm	: Revolution Per Minute.
MF	: Membership Function.
SAR	: Search and Rescue.

ÖZET

DOKTORA TEZİ

SU ÜZERINDE ARAMA VE KURTARMA ÇALIŞMALARI IÇIN UAV'YE UYGULANAN KONTROL ALGORITMALARI

Fady M. ALALAMI

İstanbul Üniversitesi-Cerrahpasa

Lisansüstü Eğitim Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Abdul Rahman HUSSIAN

Bu araştırma, özellikle su üstü arama ve kurtarma çalışmalarında kullanmak üzere çok aşamalı uyarlanabilir PID kontrolcü vasıtasıyla 4 pervaneli robot helikopter araçları zorlu hava koşulları altında kontrol etme metodunu anlatmaktadır. Çok aşamalı kontrolcü pozisyon, açısal pozisyon ve motor kontrolünü kapsamaktadır. Pozisyon ve rakım kontrolcüleri basit PID kontrol sürecine dayanmaktadır. PID parametrelerini optimize etmek için Genetik Algoritma'yı (GA) optimizasyon tekniği olarak düşündük. GA çevrimdışı planlanmış rotayı baz alarak çok aşamalı PID'nin parametlerini optimize etmeye çalışacak. Optimizasyon işlemi, kontrolcü parametrelerini planlanan yola göre ayarlamak için çevrimdışı olarak planlanan bir uçuş için gerçekleştirildi. Bütün PID kontrolcüleri diğer kontrolcüleri etkilediği için optimizasyon süreci bütün PID parametreler için eş zamanlı olarak yapıldı. Güçlü bir 4 pervaneli robot helikopter kontrol sistemi sunmak amacıyla alan ve oryantasyon için 6 adet PID kontrolcüsü uygulandı. GA'mızı uyarlanabilir bir GA olması için geliştirdik. Önerilen Uyarlanabilir GA(UGA) metodu daha küçük başlangıç popülasyonu ve daha az iterasyon sayısı kullanarak sonuçları yüksek oranda iyileştiriyor. Modifikasyon yeni kromozomların yani "yeni sonuçların" oluşma

mekanizmasını en iyi sonuca baglı olarak değiştiriyor. Adaptasyon süreci, sürec hızlandırıcısı olarak kullanılacak rasgele sayıların en iyi sonuç dolaylarından seçilmesiyle elde edilecek. Lokal minimum durumunu önlemek amacıyla ragele değerler belirli limitlerle sınırlandırıldı. Geleneksel GA metodlarına kıyasla, sunulan yeni kromozomlar üretme metodu çok daha hızlı sonuçlar veriyor. Optimize edilmiş bir 4 pervaneli robot helikopter sistemine dair simulasyon ve sunulan metodla geleneksel metodların performanslarını kıyaslayan bir karşılaştırma da hazır. Sonuçlar sunulan metodun geleneksel metodlara kıyasla çok daha iyi performans gösterdiğini ortaya çıkartıyor. Stabillikte, daha iyi geçici cevap konusunda ve güç tüketiminde iyileşme sağlıyor. Çevrimdığı optimizasyon süreci her kargaşa senaryosu için farklı PID parametreleri sağladı. Bu durum PID parametleri arasındaki ilişkinin lineer olmadığı çıkarımına varmamıza ve çok aşamalı bir PID kontrolcü üretmemize sebep oldu. PID parametrelerinin lineer olmama durumunu çözmek için bulanık kontrolcüler ve yapay sinir ağı kontrolcüleri PID kontrolcüleriyle etkileşime sokuldu. Bu kontrolcüler PID paremetrelerini motora etki eden harici kargaşaları baz alarak ayarlayacak. Her iki kontrolcü de stabilliğini korudu ve hedefe küçük farklarla ulaştı. 4 pervaneli robot helikopterin denizlerde kullanılanlar gibi bazı mevcut uygulamaları hareket halindeki araçlardan fırlatma şeklinde yapılıyor. Olduğunu, bu tür 4 pervaneli robot helikopterin sabitlenmemiş bir gemiye inişi özenle yapılmalıdır. En yaygın kullanılan eylemsizlik referans çerçeveleri. Dünya'nın yüzeyine ve Dünya Merkezli sabit kordinat sistemine bağımlı çalışan Geodetik Kordinat Sistemleri. Bu çalışma ivmelenme ve hareketli bir obje olan quad-rotorun pozisyonunu eylemsizlik referans çevrelerinin geçerli olmadığı bir düzlemde analiz etti. Sonuç olarak, kontrolcüleri ayırdık ki pratik operasyonlar için fiziksel programlanabilir cihazlarla programlanmaya hazır olsunlar.

Temmuz 2020, 151 sayfa.

Anahtar kelimeler: Quad-rotor, Genetic Algorithm, Fuzzy Control, Neural Networks

SUMMARY

Ph.D. THESIS

CONTROL ALGORITHMS APPLIED TO UAV FOR SEARCH AND RESCUE OVER WATER

Fady M. ALALAMI

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Electrical and Electronic Engineering

Supervisor: Prof. Dr. Abdul Rahman HUSSIAN

This research addresses the method of controlling quad-rotor under harsh weather conditions using Multi-Stage Adaptive PID controller; particularly for search and rescue operations overwater. Multi-stage controller includes position control, attitude control, and motor control. Controllers for position and attitude were based on the basic PID control process. In order to optimize PID parameters, we considered Genetic Algorithm (GA) as an optimization technique. GA will attempt to optimize the parameters of multi-stage PID controllers, depending on an offline-planned path. The optimization process is performed for an offline-planned flight to adjust the parameters of the controller according to the planned path. The optimization process is carried out for all PID parameters simultaneously as each PID controller affects the other controllers. Six PID controllers for the location and orientation implemented to provide a robust quad-rotor control system. We have developed GA to be an Adaptive GA, the proposed AGA method produces highly improved results using smaller initial population and a smaller number of iterations. The modification affects the mechanism of generating new chromosomes “new

solutions” depending on best solutions. The process of adaptation is achieved by creating new random values around the best solution which will accelerate the process. To prevent a local minima solution, random values are constrained by limits. Compared with conventional GA methods, the proposed method of generating new chromosomes provides much faster results. A simulation of the optimized controller for a quad-rotor system is presented and a performance comparison between proposed and conventional methods is presented as well. The results show that the proposed method has outperformed the conventional method. It provides improvement in stability, better transient response and less power consumption. The offline optimization process provided different PID parameters for each disturbance case. Hence, the relationship between PID parameters and disturbance values is non-linear, which has led us to develop multi-stage PID controller to multi-stage Adaptive PID controller. To solve the non-linearity of PID parameters, Fuzzy and Neural Networks controllers were used to interact with the PID controller. Those controllers will adjust PID parameters based on the strength of external disturbances affecting the quad-rotor. Both controllers retained stability and achieved their target with small differences in performance. Some of current applications of quadrotors, such as those used in sea search and rescue operations, will be launched from a moving vessel. That is, landing of such quad-rotors would have to take into account the vessel's non-inertial location to land on. The most widely used inertial frames are Geodetic Coordinate System that depends on earth’s surface and Earth-Centered Fixed Coordinate System. This work analyzed the velocity, and position of quad-rotor that is based on a moving object as a non-inertial frame of reference. Finally, we discretized the controllers so that they would be ready to be programmed in physical programmable devices for practical operations.

July 2020, 151 pages.

Keywords: Quad-rotor, Genetic Algorithm, Fuzzy Control, Neural Networks

1. INTRODUCTION

1.1. OVERVIEW

Nowadays, multi-copters “Drones” applications are increasing; some of these applications play an important role in our life such as fire-fighting, precision agriculture, weather forecast, shipping and delivery, aerial photography, search and rescue and others (Kumar et al., 2015) (Dallal Bashi et al., 2017). For these applications, different multi-copters and controllers can be used to achieve the desired task. In some cases, special controllers should be designed to control multi-copter flights under severe weather conditions. It requires a special path to achieve the task, or it can be related to non-inertial frame of reference (NIFR) during its operation. In all of the previously mentioned, the multi-copter is modeled carefully for the purpose of designing appropriate controllers. The multi-copter structure should be studied well to construct the model of the system; some previous studies provide a survey of multi-copter modeling and system identification (Zhang et al., 2014) (Chovancová et al., 2014) (Musa 2017). They all agreed that the system is non-linear, but from one study to another, they vary in a way that helps in knowing how to linearize the system. Thus, in some cases, they neglect a lot of terms to convert the non-linear system to a linear system. Also, a linear system can be represented in state-space model (Khuwaja et al., 2018) (Bouabdallah et al., 2004).

In this study, effective forces and torques will be considered so that multi-copter can overcome adverse weather conditions without losing stability. Gravitational Force, Thrust, Aerodynamic Drag Force, centripetal and gyroscopic forces have been considered in this study. In order to complete the drone modeling, we must understand some terms and definitions i.e. Kinematics, Dynamics, Newton-Euler equations and Euler-Lagrange equations of motion (Mahmoud et al., 2014) (Stevens et al., 2015). The kinematics of the system was described using Euler angle representation (Luukkonen 2011); in some cases, Euler angle fails because it is susceptible to gimbal lock (Hemingway and O’Reilly 2018). This problem can be solved by quaternion mathematical representation (Abaunza et al., 2018), but the gimbal lock can happen only in rare cases such as in acrobatic quad-rotors; thus, to simplify mathematical formulation, we’ve considered Euler-angles for angle representation.

Most of the studies consider the inertial frame of reference (IFR) as a reference for the transformations between body frames and fixed frames. Thus, all the transformations of position, velocity and acceleration depend on the inertial frame of reference. In some applications, the drone can be mounted over a moving object such as in search and rescue (SAR) operations over water. In SAR, operations over water, the drone could be referenced to a moving vessel; in that case, the drone should be related to a NIFR. For IFR, there are some known frames such as Geodetic coordinate system (Latitude, Longitude and altitude), earth – Centered coordinate system and Local North-East- Down or Up (NED /NEU) coordinate systems (Cai et al., 2011). IFR was considered in modeling the quad-rotor system so that it can be used for determining the kinematics and dynamics of the system. Also, NIFR was added in modeling the system to give a relative reference for the position of the quad-rotor body frame. This idea of relating the drone to a NIFR can help us to do a lot of applications that can be very useful. One of the most important applications is the search and rescue operations, shipping and delivery, fire-fighting and weather forecast.

To design the controller under the mentioned circumstances; a PID controller has been chosen. PID controller is one of the most used controllers for controlling multi-copters (Kotarski et al., 2016) (Akhil et al., 2012). One of the most important advantages of this controller is simplicity, but, sometimes, PID controller can fail, especially when controlling nonlinear systems such as multi-copters. To avoid these disadvantages, PID parameters should be chosen carefully by considering different conditions. A multi-stage PID controller presented here to control the position (x, y, z) and the angular position (φ, θ, ψ) of the quad-rotor; thus, in total 6 PID controllers are used to optimize the performance of multi-copter. Adaptive Genetic Algorithm (AGA) was used to optimize PID controller's parameter. Some studies used GA to optimize PID controller's parameter (Khuwaja et al., 2018) (Salam and Ibrahim 2019), but these studies didn't make the optimization for all six controllers together at the same time. This issue can affect the optimization process because controllers depend on one another. Also, they didn't consider asymmetric paths. GA was modified to be AGA, the modification was done for generating a new population of chromosomes, instead of just making a crossover between parent chromosomes; but also, we can generate a new random solution around the current success solutions to accelerate the optimization process.

Most of the scoped applications in this research depend generally on a pre-defined path. The importance of dealing with a predefined path is that the optimization of PID parameters can vary from one path to another in abnormal conditions. Many studies concentrated on optimizing the controller parameter relying on perfect weather conditions such as, in indoor flights. Sometimes in bad weather conditions optimal PID parameters can't give satisfactory results to the extent that even the flight stability can fail. Instead, modest controllers can be used to get satisfactory results and maintain stability. However, if we consider wind disturbances, the multi-copters will always try to resist the wind from a specific direction. And if we study the overall error for that flight, we come to the conclusion that the error in that wind direction axis is higher than others, which is what's expected. However, if it is intended to optimize the flight for specific disturbances, as in offline path optimization and NIFR, the optimization of PID controller should be done simultaneously and that is what we have done in this study.

The negative impact on the stability and overall performance of wind disturbances pushed us to develop our multi-stage PID controller. Therefore, the effects of wind disturbances can be classified into two categories: linear wind disturbances and angular wind disturbances. The effect of the disturbances on the quad-rotor was non-linear, so we were needed to use artificial intelligent techniques to handle the issue.

Some previous studies used Fuzzy Control and Neural Networks to control the quad-rotor, but the heavy operation of these controllers slowed down the processing of the physical controllers, which in some cases resulted in failure. In this regard, we suggested that these intelligent controllers be used as additional controllers with the main controller. These controllers will tune the main controller's PID parameters, so that we can separate the intelligent controller and program it to external programmable devices.

1.2. STATEMENT OF THE PROBLEM

The problems can be described as:

- The challenge of modeling a multi-copter in an optimized form without ignoring essential physical terms.
- Difficulty in operating multi-copters under extreme weather conditions.
- Optimization of controller parameters due to many external disturbances.

- The challenge of adapting the controllers to different weather conditions.
- The complexity of linking the multi-copters to the non-inertial frame of reference.
- Difficulty with the use of artificial intelligence controllers to control multi-copters.

1.3. OBJECTIVES

The main objectives of this thesis are to provide a proper controller for the quad-rotor system to:

- Maintain Stability under various weather conditions.
- Optimize the flight response
- Decrease power consumption.
- Decrease the processing operations for quad-rotor main controller device.
- Relate quad-rotor to a moving object.

1.4. MOTIVATION

The main stimulus for selecting this study is the need for robust multi-copters to assist in search and rescue operations under severe weather. The general idea is that any ship in the sea, particularly Coast Guard rescue ships and SAR ships, should have an Unmanned Aerial Vehicle (UAV) over it, so it can be used to save human life or search for lost people, lost ships or sunken aircraft at lower cost than specialist search and rescue technologies. Searching by Unmanned Aerial Vehicle can be the quickest way to reach its target, easy to lunch, small body and efficient enough. Within this planet the most important thing is human life. The most honorable act is to work hard to save human life. There are also projects and operations that are seeking to save the lives of people who need help; as if they are in imminent danger. So, if we've designed a powerful multi-copter controller that can work in bad weather to save human lives; it's going to be a great success.

1.5. METHODOLOGY

First, we'll construct the quad-rotor model in a way that will decrease the mathematical model’s complexity without ignoring important terms. Make a relation between the quad-rotor body frame and a moving object's base frame. Design the quad-rotor controller based on the basic

PID controller, and then optimize controller parameters. Develop the controller to be capable of dealing with extreme weather conditions, then prepare the controller for implementation in programmable physical devices.

1.6. CONTRIBUTION

- An optimized model of a quad-rotor was obtained.
- Genetic Algorithm was improved to adaptive GA to operate faster than the conventional method and to reach the solution with minimal number of iterations.
- Parallel optimization strategy was developed to optimize the parameters of the controllers.
- We designed an Adaptive Fuzzy-PID controller which adapts the external disturbances applied to quad-rotor based on learning from example method.
- We designed an Adaptive NN-PID controller which adapts the external disturbances applied to quad-rotor based on learning from example method.

1.7. OUTLINE OF THE THESIS

The structure of this thesis is as follows: in Chapter 1 a general introduction about our application, system and controller, as well as the strategy that we have provided to solve the proposed problems, the previous studies have been included in Chapter 1. Chapter 2 contains the theories we used, such as PID Control, Genetic Algorithm, Fuzzy Control, and Neural Networks. Chapter 3 includes the mathematical modeling of 6-DOF quad-rotor system including the transformations between IFR, NIFR and body frame, also the design of multi-stage PID controllers for controlling the position and orientation of the quad-rotor system, the design of Adaptive Fuzzy-PID and Adaptive NN-PID controllers, GA modification and PID parameters optimization process are also presented in chapter 3. Chapter 4 illustrates the numerical simulations and optimization processes that have been carried out. Chapter 5 discusses the results, advantages and disadvantages, and the future work. Chapter 6 concludes the work done in this thesis.

1.8. LITERATURE REVIEW

Nowadays, drones are considered (among) the active study materials due to the variation of usage. The usage of drones added constraints to the production process of these drones. So, different usages of drones can result in various shapes, power, payloads, number of motors, speeds, shape of the propellers and much more variables. These differences lead to a huge research material. So, to conduct research on the previous studies related to the quad-copter; the exact application and exact problems should be specified. In our thesis, as mentioned in the introduction, the study will concentrate on the (SAR) operations over sea in bad weather conditions. Some studies related to this field have been summarized in this section of the study.

One of the most useful research studies was conducted by Luukkonen (2011). In this research, Luukkonen summarizes the basic fundamentals of modeling and controlling of the Quad-copter drone. In this study the author focused on the modeling of a quad-copter using two methods; the first method was Newton-Euler while the second method was Euler-Lagrange. To evaluate and test the model of the quad-copter, a flight simulation was done to test the model stability and to compare between the results of the control algorithms.

First step is building the mathematical model. The mathematical model of the quad-copter considered the body of the drone as a rigid body; the assumption of the rigidity can be acceptable in most of the cases because generally most of the applications that use the quad-copter depend on a small body structure, and with this small body structure, we can neglect the small effect of the flexibility. To model the system; the research started to define the main terms of this system, such as forces and torques produced by the motors, the position and rotation of the drone on the body frame and the inertial frame (Luukkonen 2011, pp. 3-4).

Each rotating motor creates force f_i in the direction of its axis, $f_i = kw_i^2$. Where, k is the lifting constant, w is the angular velocity. Also, rotating motor produces angular velocity and acceleration torque $\tau_{Mi} = bw_i^2 + I_M\dot{w}_i$. For model simplification, the author depends on a plus (+) quad-copter structure. In this case, the forces generated by the rotors will be perpendicular over the x-axis and the y-axis.

With the plus structure, we avoid making calculation for the distance between the forces generated by the rotors and the axes of the drone. The torque generated by the rotors will be:

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-w_2^2 + w_4^2) \\ lk(-w_1^2 + w_3^2) \\ \sum_{i=1}^{4} \tau_{Mi} \end{bmatrix} \quad (1.1)$$

where *l* is the length between the center of the body frame and the rotors.

In this paper, two methods were used to describe the dynamical model of the quad-copter; the first method is Newton-Euler method; the second one is the Euler-Lagrange method. In Newton-Euler method, the coordinate system and all the forces and torques that affect the system should be considered, but for the Euler-Lagrange method, it is based on defining the kinetic and potential energies of the system. For the quad-copter modeling, we can choose even Newton-Euler method or Euler-Lagrange method or both; depending on the complexity of the mathematical calculations. Mathematical calculations are important due to microcontroller constraints such as memory size and speed.

There are two main equations for modeling the quad-copter that clarify the relation between the forces that affect the motion of the system, the linear acceleration and the angular acceleration. The linear acceleration can be represented as: ($m\ddot{\xi} = mG + RT_B$)

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\phi S_\theta S_\psi - C_\psi S_\phi \\ C_\theta C_\phi \end{bmatrix} \quad (1.2)$$

The angular acceleration can be represented as:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})qr/I_{xx} \\ (I_{zz} - I_{xx})pr/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} \end{bmatrix} - I_r \begin{bmatrix} q/I_{xx} \\ -p/I_{yy} \\ 0 \end{bmatrix} \omega_\Gamma + \begin{bmatrix} \tau_\phi/I_{xx} \\ \tau_\theta/I_{yy} \\ \tau_\psi/I_{zz} \end{bmatrix} \quad (1.3)$$

The author added the Aerodynamic effects to the forces that affect the motion of the system (Luukkonen, p. 6); thus, the drag force is one of the Aerodynamic effects; it includes the resistance of the air faced by the quad-copter, but he didn't consider the other effects, such as blade flapping, angle of attack and airflow disruptions.

Before using the controllers, the author tried to give a direct signal to the inputs of the system and examine stability of the system. The result of the angle vector was marginally stable, but the result of the position vector was unstable as shown in Figure (1.1).

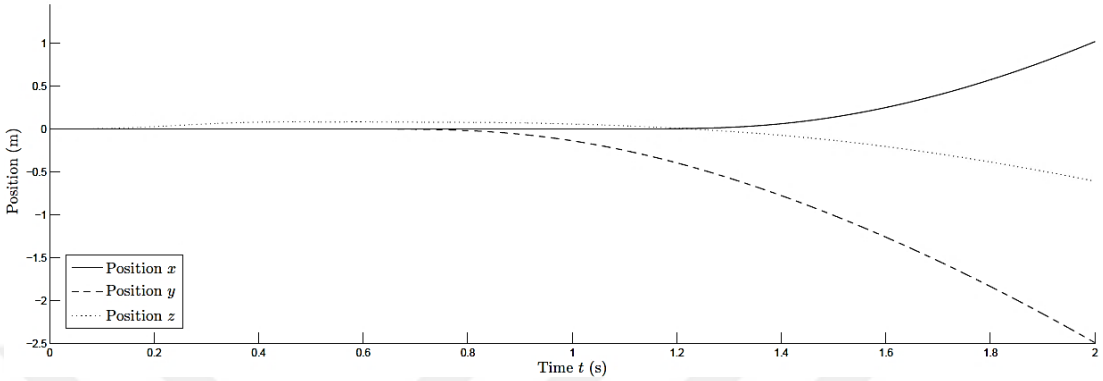


Figure 1.1: Open loop response of the position vector

First of all, the stability of the angle vector was not acceptable, so a PD controller was used to control the inputs of the system depending on the desired angle vector. The input of the system is the angular velocities of the four rotors. From equations (1.2) and (1.3), a relation between the angular velocities of the motor and the torques can be found. From these relations, we can find the angle and position vector of the system as follows (Luukkonen 2011, p. 10), (Dikmen, Arisoy, & Temeltas 2009, p. 723),

$$\begin{bmatrix} T \\ \tau_{\emptyset} \\ \tau_{\theta} \\ \tau_{\varphi} \end{bmatrix} = \begin{bmatrix} \left(g + K_{z,D}(\dot{z}_d - \dot{z}) + K_{z,P}(z_d - z)\right)\dfrac{m}{C_{\emptyset}C_{\theta}} \\ \left(K_{\emptyset,D}(\dot{\emptyset}_d - \dot{\emptyset}) + K_{\emptyset,P}(\emptyset_d - \emptyset)\right)I_{xx} \\ \left(K_{\theta,D}(\dot{\theta}_d - \dot{\theta}) + K_{\theta,P}(\theta_d - \theta)\right)I_{yy} \\ \left(K_{\varphi,D}(\dot{\varphi}_d - \dot{\varphi}) + K_{\varphi,P}(\varphi_d - \varphi)\right)I_{zz} \end{bmatrix} \tag{1.4}$$

From the calculated torques, the squared of the angular velocities of the motor can be found,

$$\begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} = \begin{bmatrix} \dfrac{T}{4K} - \dfrac{\tau_{\theta}}{2kl} - \dfrac{\tau_{\psi}}{4b} \\ \dfrac{T}{4K} - \dfrac{\tau_{\phi}}{2kl} + \dfrac{\tau_{\psi}}{4b} \\ \dfrac{T}{4K} + \dfrac{\tau_{\theta}}{2kl} - \dfrac{\tau_{\psi}}{4b} \\ \dfrac{T}{4K} + \dfrac{\tau_{\phi}}{2kl} + \dfrac{\tau_{\psi}}{4b} \end{bmatrix} \tag{1.5}$$

The second simulation was to apply a PD controller over a desired angle and a desired altitude. So, to complete solving the system model equations, we need to know the angular velocities of the rotors (the inputs of the system). In that case, we can return to equation (1.4) to get the $T, \tau_{\emptyset}, \tau_{\theta}, \tau_{\varphi}$, so we get the torque vector and the thrust T. From these values, we can substitute the torque vector and thrust in equation (1.5) to get the squared angular velocities for each rotor. The output results of the PD controller stabilized the altitude and the attitude as shown in figure (1.2); the error was eliminated and the time response was improved. The difference can be seen between the open loop result in figure (1.1) and the PD close loop system in figure (1.2). The position deviation has been decreased, but it's still not the desired position because there was no trajectory to follow up (Luukkonen 2011, pp. 8-12).

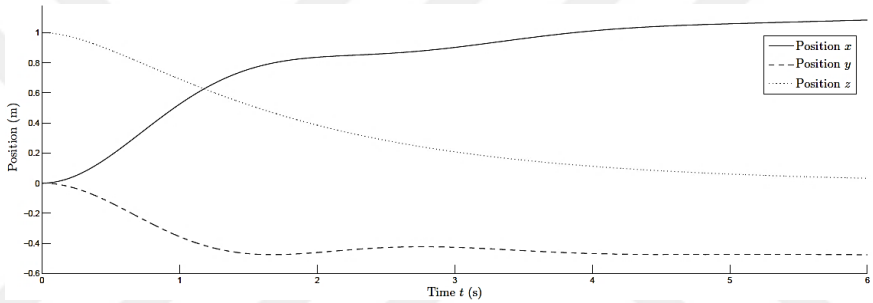


Figure 1.2: Close loop response with PD controller of the position vector

The quad-copter system has six degrees of freedom, such as 6 outputs and 4 inputs. The inputs are the angular velocities of the 4 rotors; the outputs are three for the positions vector ξ, and three for the angles vector η. For the first simulation process, it was simple to make the mathematical calculations because inputs are given, so it is easy to solve the kinematics and dynamics equations, but if the input is a trajectory, it will be much difficult. Also, to find the desired rotor velocities, we need to make reverse calculations from position and orientation to rotors' velocities. The flow chart of the system model from inputs to outputs can be seen in figure 1.3.

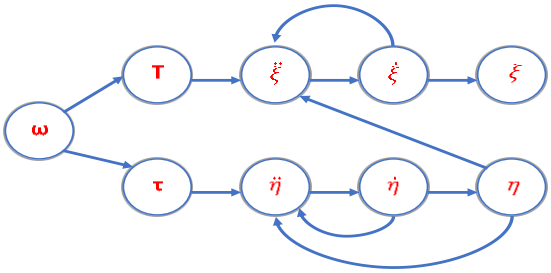


Figure 1.3: Flow chart of the quad-copter mathematical model

The third simulation was the trajectory control; the idea is to follow the desired position by controlling the angular velocity of the 4 rotors. From the state diagram in figure (1.3), we can figure out that the position depends on the acceleration, and by solving the acceleration equations, we can get the desired position. So, the operation will be reverse steps. Trajectory planning is a detailed subject, and to make a trajectory planning we need to make intensive mathematical calculations. By summarizing the most important terms, we come to the conclusion that one method for trajectory planning is to get the values of the linear acceleration $\ddot{\xi}$, linear velocity $\dot{\xi}$ and the Yaw angle ψ. These three variables are important to start our reverse calculations and to get the 4 rotor angular velocities. We can start the reverse calculations and solve equation (1.7) to get φ, θ and T as explained by Zuo (2010, pp. 4 - 5)

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = -g\begin{bmatrix}0\\0\\1\end{bmatrix} + \frac{T}{m}\begin{bmatrix}S_\phi S_\psi + C_\phi C_\psi S_\theta\\ C_\phi S_\theta S_\psi - C_\psi S_\phi\\ C_\theta C_\phi\end{bmatrix} + \frac{1}{m}\begin{bmatrix}A_x & 0 & 0\\ 0 & A_y & 0\\ 0 & 0 & A_z\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} \tag{1.6}$$

from the previous equation we can get:

$$T_B = \begin{bmatrix}0\\0\\T\end{bmatrix} = R^{\mathrm{T}}\left(m\left(\ddot{\xi} + \begin{bmatrix}0\\0\\g\end{bmatrix}\right) + \begin{bmatrix}A_x & 0 & 0\\ 0 & A_y & 0\\ 0 & 0 & A_z\end{bmatrix}\dot{\xi}\right) \tag{1.7}$$

Thus,

$$\begin{bmatrix}\phi = \arcsin\left(\dfrac{d_x S_\psi - d_y C_\psi}{d_x^2 + d_y^2 + (d_z + g)^2}\right)\\ \theta = \arctan\left(\dfrac{d_x C_\psi + d_y S_\psi}{d_z + g}\right)\\ T = m\big(d_x\big(S_\theta C_\psi C_\varphi + S_\psi S_\varphi\big) + d_y\big(S_\theta C_\psi C_\varphi - C_\psi S_\varphi\big) + (d_z + g)C_\theta C_\varphi\big)\end{bmatrix} \tag{1.8}$$

Where d_x, d_y and d_z are:

$$\begin{bmatrix}d_x = \ddot{x} + \dfrac{A_x\dot{x}}{m}\\ d_y = \ddot{y} + \dfrac{A_y\dot{y}}{m}\\ d_z = \ddot{z} + \dfrac{A_z\dot{z}}{m}\end{bmatrix} \tag{1.9}$$

If ϕ and θ are known, we can get the angular velocity and angular acceleration by derivation $\dot{\phi} = \frac{d\phi}{dt}$, $\ddot{\phi} = \frac{d^2\phi}{dt^2}$, from these values; we can get torque values by substitute them into Euler-Lagrange equation.

$$\tau = J\ddot{\eta} + C(\eta, \dot{\eta})\,\dot{\eta} \tag{1.10}$$

When we obtain the torques and thrust, we can calculate the control inputs of the system using equation (1.5).

By using the Heuristic method, we can generate the trajectory (Luukkonen 2011, pp. 14-18). The third and fourth derivative of the position jerk and jounce are important for controlling the speed of the rotors. Jounce can affect the motor significantly because if the jounce value is high, this will affect the rotor responding as vibration or high sounds or harsh responds. The simulation results by Luukkonen (2011, pp. 16-17) shows that the value of the jounce is same as the control signal applied to the rotors, and the shape of the acceleration is same as the shape of the angle command. The author used the heuristic method to generate the desired jounce; after understanding and studying the Heuristic method and the desired acceleration, velocity and position can be calculated. But as the author mentioned: this method does not give optimal trajectories because it can be considered as a static method, but for applications like the quad-copter, a dynamical method is needed to achieve optimal trajectories. Another draw back in the study is that it only planned a trajectory for one axis, but considering a three axis will give more logical results. Another disadvantage is that it does not consider a lot of forces. It is supposed that there are no external forces affecting the trajectory, and if there is any deviation in the angles, it will result into a very big divergence in the trajectory.

From equation (1.4) the angular velocity of the rotors can be determined using equation (1.5). This assumption assumes that the desired angles are known and we can use a PD controller to control the angles, but if the desired angles are calculated from the trajectory planning, we can use two controllers: one for the desired angles and the one for the desired trajectory (desired acceleration, velocity and position), equation (1.11) shows how to control the acceleration command by a PDD controller,

$$\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} K_{x,P}(x_d - x) + K_{x,D}(\dot{x}_d - \dot{x}) + K_{x,DD}(\ddot{x}_d - \ddot{x}) \\ K_{y,P}(y_d - y) + K_{y,D}(\dot{y}_d - \dot{y}) + K_{y,DD}(\ddot{y}_d - \ddot{y}) \\ K_{z,P}(z_d - z) + K_{z,D}(\dot{z}_d - \dot{z}) + K_{z,DD}(\ddot{z}_d - \ddot{z}) \end{bmatrix} \tag{1.11}$$

Then the commanded angles $\emptyset_c, \theta_c$ and T_c can be determined from equation (1.7) and ψ_d is the desired Yaw angle.

$$\begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \left(K_{\varphi,D}(\dot{\varphi}_c - \dot{\varphi}) + K_{\varphi,P}(\varphi_c - \varphi)\right) I_{xx} \\ \left(K_{\theta,D}(\dot{\theta}_c - \dot{\theta}) + K_{\theta,P}(\theta_c - \theta)\right) I_{yy} \\ \left(K_{\psi,D}(\dot{\psi}_d - \dot{\psi}) + K_{\psi,P}(\psi_d - \psi)\right) I_{zz} \end{bmatrix} \tag{1.12}$$

Although, the angular velocities can be linearized and the low effective terms can be ignored, from equation (1.3), the linearized equation can be as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{\tau_\phi}{I_{xx}} \\ \frac{\tau_\theta}{I_{yy}} \\ \frac{\tau_\psi}{I_{zz}} \end{bmatrix}, \text{ Thus, } \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ddot{\varphi} I_{xx} \\ \ddot{\theta} I_{yy} \\ \ddot{\psi} I_{zz} \end{bmatrix} \tag{1.13}$$

Zuo (2010, p. 4) defined the position error as; $p_e = P_c - P$ where, $P_c = (x_c, y_c, z_c)$. So, the closed loop of the position will be as follows, $\ddot{P}_e + K_d \dot{P}_e + K_p P_e = 0$ where, K_d and K_p are positive matrices. The Routh-Hurwitz principle supposes that the error converges to Zero exponentially.

$$\ddot{P} = \ddot{P}_c + K_d\left(\dot{P}_c - \dot{P}\right) + K_p(P_c - P) \tag{1.14}$$

For stability the commanded or desired angular acceleration should be zero. With equation (1.14) as a base equation; same procedure can be done for the angular accelerations in equation (1.13)

$$\begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ddot{\varphi} I_{xx} \\ \ddot{\theta} I_{yy} \\ \ddot{\psi} I_{zz} \end{bmatrix} = \begin{bmatrix} \left(K_{\varphi,D}(\dot{\varphi}_c - \dot{\varphi}) + K_{\varphi,P}(\varphi_c - \varphi)\right) I_{xx} \\ \left(K_{\theta,D}(\dot{\theta}_c - \dot{\theta}) + K_{\theta,P}(\theta_c - \theta)\right) I_{yy} \\ \left(K_{\psi,D}(\dot{\psi}_d - \dot{\psi}) + K_{\psi,P}(\psi_d - \psi)\right) I_{zz} \end{bmatrix} \tag{1.15}$$

Gibiansky (2012, pp. 8-9) explained the relation between the torques, system and the Gyroscope sensor. The information that can be provide by the Gyroscope sensor is the angular velocities $(\dot{\varphi}, \dot{\theta}, \dot{\psi})$ or (p, q, r). If we combined equations (1.1) and (1.15) we got this result,

$$\begin{bmatrix} lk(-w_2^2 + w_4^2) \\ lk(-w_1^2 + w_3^2) \\ \sum_{i=1}^{4} \tau_{Mi} \end{bmatrix} = \begin{bmatrix} \left(K_{\emptyset,D}(\dot{\emptyset}_c - \dot{\emptyset}) + K_{\emptyset,P}(\emptyset_c - \emptyset)\right) I_{xx} \\ \left(K_{\theta,D}(\dot{\theta}_c - \dot{\theta}) + K_{\theta,P}(\theta_c - \theta)\right) I_{yy} \\ \left(K_{\varphi,D}(\dot{\varphi}_d - \dot{\varphi}) + K_{\varphi,P}(\varphi_d - \varphi)\right) I_{zz} \end{bmatrix} \tag{1.16}$$

In the equilibrium point, the control equation $U_1 = \Sigma T = c_T(\varpi_1^2 + \varpi_2^2 + \varpi_3^2 + \varpi_4^2)$ will equal the summation of the forces, and in hovering mode without disturbances the summation of forces will equal the gravity force: $F_g = mg$. In that case, we have four equations with four unknowns, we can obtain the squared angular velocities of the rotors of the quad-copter.

The cascaded control method (two PD controllers) gave better results, it can reach the desired position in 6 second, but the response time is slow and there are lots of oscillations in the angles before and after reaching the desired position as shown in figure (1.4). Therefore, the stabilization of the drone needs a lot of adjustment. As an example, the author didn't explain the method of choosing the PD controller parameters.

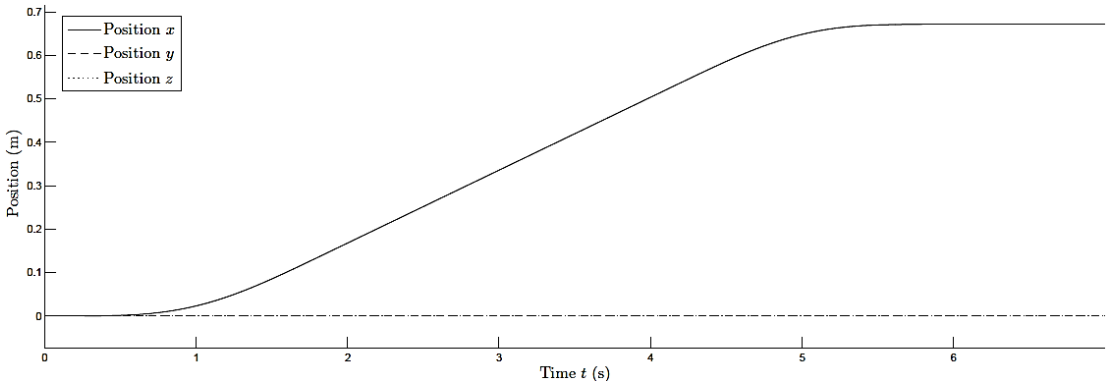


Figure 1.4: The result of cascaded PD controller for position vector

The conclusion of Luukkonen's (2011) study is: For controlling and stabilizing the quad-copter, a PD controller has been used; and for controlling the trajectory of the quad-copter, a heuristic method has been used. The author combined the PD controller and the heuristic method to improve the response of disturbances in different flight conditions. The PD controller gave a good result in stabilizing the attitude and the altitude of the drone, but the position of x and y was deviated. The conclusion of this study suggests that the combination of the PD control and

the heuristic method gave an overall acceptable result. The drawback of the study is as follows: the comparison between the two methods didn't consider the complexity of the system or the controller physical specifications. In this research, there is only one control method for testing the stability of the system; however, we can't judge the controller result if we didn't compare it with other controllers' results. The research didn't consider the wind disturbances and other dynamical disturbances. The system and the controller have been tested in simulation and they didn't consider the physical restrictions. Also, they didn't use feedback from sensors like gyroscope, accelerometer, and magnetometer. The research didn't mention about the gimbal lock problem of Euler angle (Newton-Euler) modeling method. The method of choosing the PD parameters was not mentioned in the study.

Another research conducted by Khuwaja, Tarca, Lighari, & Tarca, (2018) studied the quad-copter system controlled by PID controller and optimized by Genetic Algorithm. According to the author, the quad-copters are nonlinear dynamical unstable systems; and to maintain stability, the controller shouldn't be a classical linear controller. The authors aim is to improve the controller response and steady state error, reduce the overshoot, shorten the settling time, and decrease the rise time of the system. The idea is to determine the suitable PID parameters by tuning these parameters with Genetic Algorithm to reach the desired stability and accuracy.

In this study, they started by defining the model of the quad-copter as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{U_1}{m}\left(S_\phi S_\psi + C_\phi C_\psi S_\theta\right) \\ \frac{U_1}{m}\left(C_\phi S_\theta S_\psi - C_\psi S_\phi\right) \\ g - \frac{U_1}{m}\left(C_\theta C_\phi\right) \\ \frac{L}{I_{xx}}.U_2 - \frac{J_r}{I_{xx}}.\dot{\theta}.\Omega_r + \frac{I_{yy}-I_{zz}}{I_{xx}}.\dot{\theta}.\dot{\psi} \\ \frac{L}{I_{yy}}.U_3 - \frac{J_r}{I_{yy}}.\dot{\varphi}.\Omega_r + \frac{I_{zz}-I_{xx}}{I_{yy}}.\dot{\varphi}.\dot{\psi} \\ \frac{L}{I_{zz}}.U_4 + \frac{I_{xx}-I_{yy}}{I_{zz}}.\dot{\varphi}.\dot{\theta} \end{bmatrix} \quad (1.17)$$

After obtaining the mathematical model of the system; it can be represented in state space model form as shown in equation (1.18) (Khuwaja, Tarca, Lighari, & Tarca, 2018, pp. 3–4), also for

more about state space models and linearization we can return to Bouabdallah, Noth, & Siegwart (2004, p. 5)

$$a_1 = \frac{I_{yy}-I_{zz}}{I_{xx}}, a_2 = \frac{J_r}{I_{xx}}, a_3 = \frac{I_{zz}-I_{xx}}{I_{yy}}, a_4 = \frac{J_r}{I_{yy}}, a_5 = \frac{I_{xx}-I_{yy}}{I_{zz}}, b_1 = \frac{L}{I_{xx}}, \quad (1.18)$$

$$b_2 = \frac{L}{I_{yy}}, b_3 = \frac{L}{I_{zz}}$$

$$X^T = [\varphi \quad \dot{\varphi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad z \quad \dot{z} \quad x \quad \dot{x} \quad y \quad \dot{y}] \quad (1.19)$$

$$X^T = [X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \quad X_6 \quad X_7 \quad X_8 \quad X_9 \quad X_{10} \quad X_{11} \quad X_{12}] \quad (1.20)$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} \dot{\varphi} \\ \ddot{\varphi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{z} \\ \ddot{z} \\ \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix}, \qquad f(x,U) = \begin{bmatrix} x_2 \\ b_1U_2 - a_2x_4\Omega_r + a_1x_4x_6 \\ x_4 \\ b_2U_3 - a_4x_2\Omega_r + a_3x_2x_6 \\ x_6 \\ b_3U_4 - a_5x_2x_4 \\ x_8 \\ g - \frac{U_1}{m}(C_{x_1}C_{x_3}) \\ x_{10} \\ -\frac{U_1}{m}(C_{x_1}C_{x_5}S_{x_3} + S_{x_1}S_{x_5}) \\ x_{12} \\ -\frac{U_1}{m}(C_{x_1}S_{x_5}S_{x_3} - C_{x_5}S_{x_1}) \end{bmatrix} \quad (1.21)$$

Equation (1.21) is the state space model of the quad-copter system. A good point is that the authors didn't neglect the gyroscopic and centripetal terms despite their small values, but in the other hand, the external disturbances as wind speed and guest wind are neglected. On the contrary, Kurak & Hodzic, (2018, pp. 66–67) tried to exclude all low effective terms in an attempt to linearize the system, but in that way, the system model can lose a lot of terms that in some cases can affect the stability for the sake of linearity. The linearized model is explained in equation (1.22). The system is controllable and observable.

Khuwaja et al., (2018) used genetic algorithm to optimize the PID controller parameters; the controller is only applied to control the altitude of the quad-copter as shown in the study, but this can't be enough because the quad-copter has a six-degree of freedom, 3 variables for position, and 3 for orientation; all the variables should be considered.

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\\ \dot{x}_5\\ \dot{x}_6\\ \dot{x}_7\\ \dot{x}_8\\ \dot{x}_9\\ \dot{x}_{10}\\ \dot{x}_{11}\\ \dot{x}_{12}\end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7\\ x_8\\ x_9\\ x_{10}\\ x_{11}\\ x_{12}\end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0\\ 0 & b_1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & b_2 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & b_3\\ 0 & 0 & 0 & 0\\ \left(\frac{1}{m}\right) & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}U_1\\ U_2\\ U_3\\ U_4\end{bmatrix} \tag{1.22}$$

$$\begin{bmatrix}Y_1\\ Y_2\\ Y_3\\ Y_4\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7\\ x_8\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}U_1\\ U_2\\ U_3\\ U_4\end{bmatrix} \tag{1.23}$$

In the study, four fitness functions have been taken into account; the Integral Square Error (ISE), Integral Absolute Error (IAE), Integral Time multiple Square Error (ITSE) and Integral Time multiple Absolute Error (ITAE). (Kishnani et al., 2014, p. 2).

ISE: $J = \int_0^\infty e^2 dt$ (1.24)

IAE: $J = \int_0^\infty |e|\, dt$ (1.25)

$$ITSE: \quad J = \int_0^\infty te^2\, dt \tag{1.26}$$

$$ITAE: \quad J = \int_0^\infty t|e|\, dt \tag{1.27}$$

The cost function is the summation of three fitness functions multiplied by weights as in equation (1.28). From this cost function each individual can be evaluated.

$$Cost\ function = ISE\ x\ w_1 + IAE\ x\ w_2 + ITSE\ x\ w_3 \tag{1.28}$$

Nothing is mentioned in the paper about the Genetic Algorithm parameters, the number of individuals, the number of populations in the GA or the number of iterations. Also, the results of the optimized PID parameter values were not given. The displacement result of the open loop response is shown in figure (1.5), displacement response with classical PID and displacement of the system with GA are shown below:

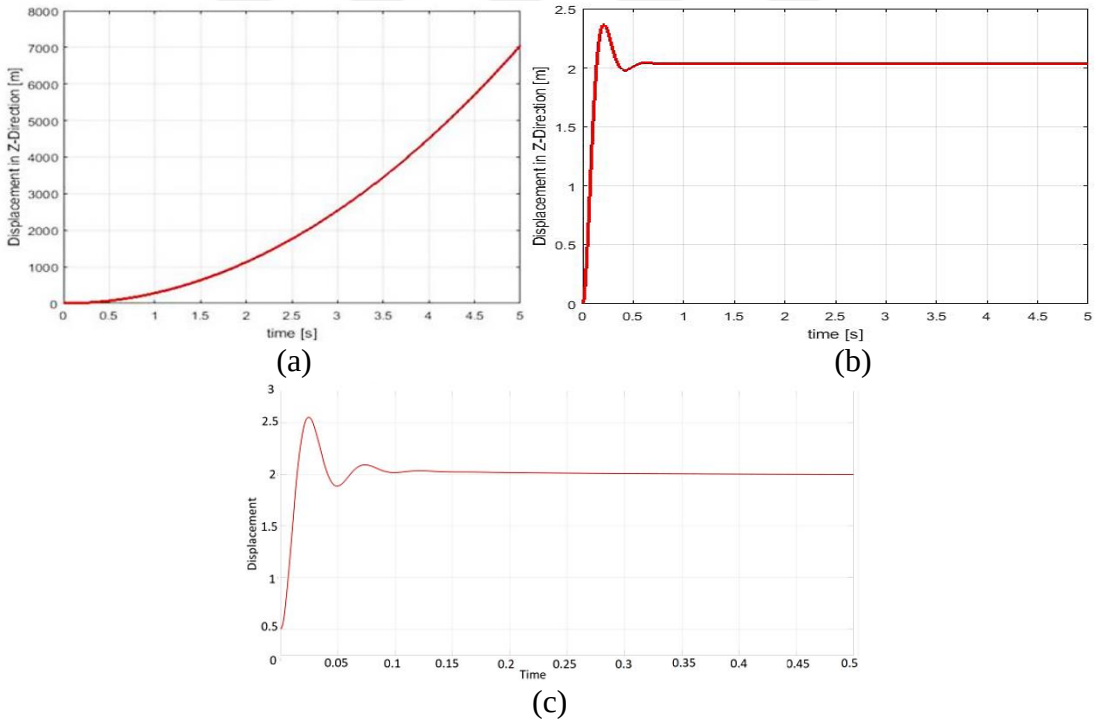


Figure 1.5: (a) displacement results of the open loop response, (b) displacement response with classical PID, (c) displacement of the system with GA

Another study conducted by Paiva et al. (2016) explained the quad-copter structure and simulated the model with a modified PID controller. In this paper, the modeling of the quad-copter is the same as the previous studies, but the ×-type configuration was explained. The ×-

type configuration refers to the axis represented over the quadcopter body. With +-type configuration the X-axis is over one arm of the quad-copter body frame and the Y-axis over different arm, but with ×-type configuration the X-axis and Y-axis is in the middle between quadcopter arms.

The forces represented in ×-type configuration will be as in equation (1.29) (Paiva, Soto, Salinas, & Ipanaque, 2016, p. 3),

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ b(\Omega_4^2 + \Omega_3^2 - \Omega_1^2 - \Omega_2^2) \\ b(\Omega_2^2 + \Omega_3^2 - \Omega_1^2 - \Omega_4^2) \\ d(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2) \end{bmatrix} \tag{1.29}$$

For the angular acceleration the authors considered the perturbations on the system axes (x, y, z) as A_p, A_q and A_r respectively as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{J_{TP}}{I_{xx}} q\Omega + \frac{lU_2}{I_{xx}} + \frac{A_p}{I_{xx}} \\ \frac{I_{zz} - I_{xx}}{I_{yy}} pr - \frac{J_{TP}}{I_{yy}} p\Omega + \frac{lU_3}{I_{yy}} + \frac{A_p}{I_{xx}} \\ \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{U_4}{I_{zz}} + \frac{A_r}{I_{zz}} \end{bmatrix} \tag{1.30}$$

One advantage in this study is the estimation of the parameters, so from practical experiments the mathematical equations of the propellers, brushless motors, and Inertia can be identified. The propellers equation will be,

$$\Omega^2 = a_\Omega * PWM - b_\Omega \tag{1.31}$$

Where Ω^2 is the angular velocity, PWM is the pulse width modulation, a_Ω and b_Ω are constants which completes the straight-line equation. The brushless motor equation is,

$$\frac{\Omega_{(s)}}{PWM_{(s)}} = \frac{K}{\tau_m s + 1} \tag{1.32}$$

The Inertia of the quad-copter was determined from the *SolidWorks* software; the software provide a model named *3DR Ardu Copter Quad-C* and provides the inertias and the weights of the quad-copter model.

The controller of the quad-copter was the modified PID controller. The block diagram represents the system and the controller as shown in Fig. 1.6 (Bresciani, 2008, p. 34; Paiva, Soto, Salinas, & Ipanaque, 2016, p. 3),

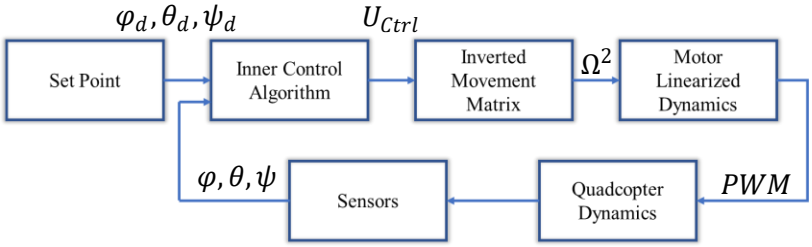


Figure 1.6: Block diagram of the system and controller

The Inner Control Algorithm block contains the modified PID controller which is presented in Laplace form in S-Plane as follows,

$$PID_{(s)} = E_{(s)}K_p + \frac{E_{(s)}K_I}{S} - Y_{(s)}K_dS - Y_{(s)}K_{d_2}S^2 \tag{1.33}$$

The mathematical structure of Inverted Movement Matrix can be found in Equation 1.29. After obtaining the values of the control inputs from the Inner Control Algorithm block, the four unknown - angular velocities squared – can be determined by solving the four equations; a Matlab command [sol] can help solving it easily. The Motor Linearized Dynamics block can be represented by Equation 1.31; it determines the Pulse Width Modulation signal PWM of the 4 motors. The sensor block provides the system with feedback information from the Gyroscope and accelerometer sensor to provide the estimated angular velocity [p q r] matrix.

The experimental and simulation results of the modified PID controller are shown in Fig 1.7 (Paiva *et al.*, 2016, p. 5).

For doing the experimental tests, the body frame was attached to a 3 degree of freedom joint to avoid accidents. The experimental results were closed to the simulation results, the differences in results were due to the friction of the drone body and the joint attached to the bench.

Paiva *et al.*, (2016) didn't explain the method of tuning the PID; instead, they just controlled three angles (Roll, Pitch and Yaw) over a fixed point and didn't consider the trajectory following variations and the integrated errors of using the drone with 6-degree of freedom.

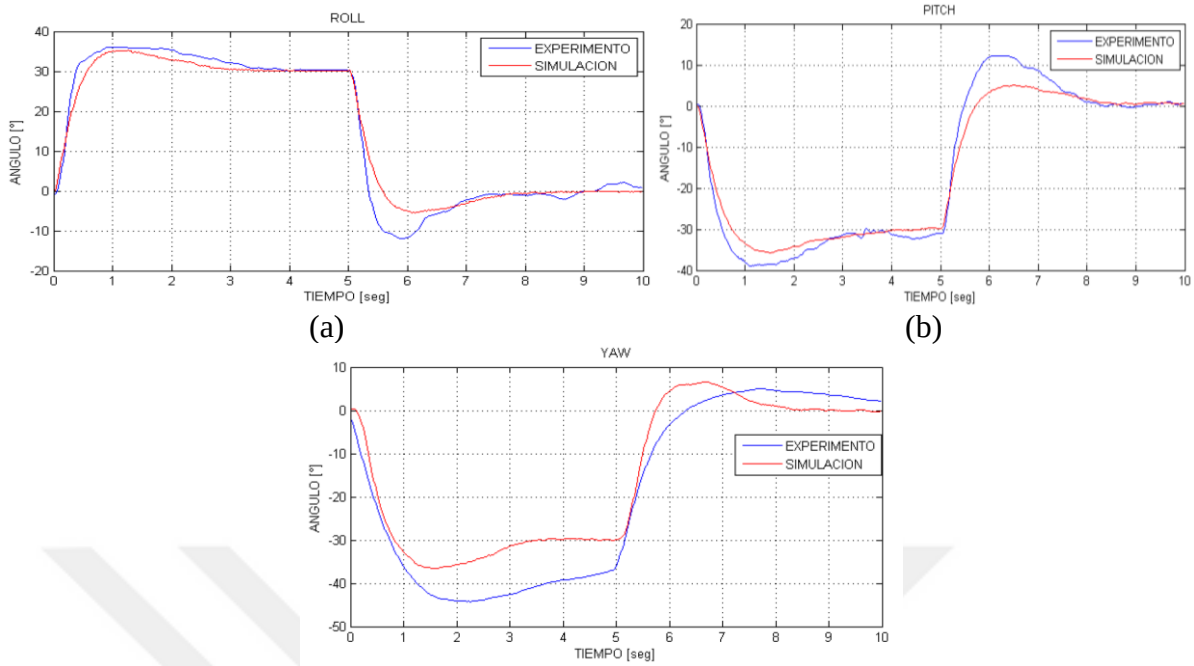


Figure 1.7: Compares ion between simulation and experimental results for the angels (a) Roll, (b) Pitch and (c) Yaw

Another study conducted by Waslander & Wang (2009) studies wind estimation and rejection to control the position of the quad copter. In this paper, they tried to model the wind effects on the quad-copter dynamics. The idea is to estimate wind velocity and try to control the quad-copter depending on the estimated wind velocities. Modeling the quad-copter dynamics, wind dynamics and considering the reading from the Inertial measurement unit (IMU) on-board can prepare a good base to apply a robust control algorithm. Going deep in fluid dynamics to make an exact calculation for wind effects on UAVs can take a lot of time and can consume a lot of efforts and equipment to do that, and the result will be a massive mathematical equations that can't be implemented on a relatively small microcontroller processors, but instead of that, wind velocity estimation algorithm can be used and considered to control UAVs.

According to Waslander & Wang (2009, p. 4), the wind vector was added to the moments and forces when modeling the quad-copter as in Equation 1.34.

$$ {}^{b}\dot{v}^{b}_{CM|i} = \frac{1}{m}\left(F_g + F_t + F_d + F_{dist}\right) - \omega^{b}_{b|i}\, x\, V^{b}_{CM|i} \tag{1.34} $$

Where, ${}^{b}\dot{v}^{b}_{CM|i}$ is the linear acceleration. The forces which have been considered are: the gravitational forces, the thrust forces, the drag forces, and the disturbances forces effects. $\omega^{b}_{b|i}$

is the angular velocity and $V_{CM|i}^{b}$ is the linear velocity. The cross product of the angular velocity and the linear velocity included the other effects on the system like the Coriolis effect and Centrifugal effect. Also, the disturbances will be added to the moments as shown in Equation 1.35.

$$\dot{v} = I^{-1}(-v\ x\ (Iv) - \Gamma + \tau_{thrust} + \tau_{dist}) \quad (1.35)$$

Where $v\ x\ (Iv)$ is the centripetal forces, Γ is the gyroscopic forces, τ_{thrust} is the thrust torque and τ_{dist} is the disturbances torque. A relation was developed to make a combination between the thrust velocity at the hovering mode and the freestream wind velocity.

The blade flapping can be considered, but the effect of the blade flapping for small UAVs can be neglected because it will add complexity to the mathematical model of the system without that significant modification in the control results or in error elimination. To estimate the wind velocities, two types of wind velocities were considered; first one is the static dominant wind direction and the second is wind gust model. For gust wind modeling Dryden wind gust model was used. The wind estimation was done in a memoryless way by only accounting the current state model. An accelerometer sensor was used to detect the acceleration of the UAV body. Also, the measured values were taken and processed with mathematical equations to give the wind velocity estimation.

There are some disadvantages in this study, one of which is math complexity because small microcontrollers can't hold to process complex math equations in the desired short time. On the other hand, some complex terms don't have noticeable effects in the results, and these terms can be neglected. The same drawback is the complexity added by the feedback sensors. These sensors will give a feedback as the actual measurements to the system; they include the GPS, accelerometer, magnetometer, barometer and ultrasonic sensors. However, the wind estimation itself needs intensive mathematical calculations. Adding to that, a Kalman filtering will be used to filter the measurement values from the sensors. So, let us imagine the amount of processing, solving, memory size, and time that these algorithms need.

Bao *et al.,* (2017) used a cascade PID to control the quad-copter attitude. The feature of this paper is that it considers a real test with electronics consideration so that the differences between theory and practice are considered in this work. The devices which have been used to build the

quad-copter were the sensors, microcontroller, electric speed controllers, actuators, and the communication module. The sensors were as follows: The Gyroscope sensor which is used to measure the quad-copter's angular velocities, the accelerometer sensor which is used to measure the acceleration on the three axes, and finally the magnetometer sensor which is used to act like magnetic compass. These sensors were connected to the main controller using the I^2C communication protocol. The wireless communication module was connected to the main controller by SPI communication protocol. Also, the motors were controlled by the electric speed controllers connected to the main controller. The main controller used in this article was *ARM-CortexM3* microcontroller. There are some advantages of using ARM to control a quad-copter, one of these advantages is fast processing that can yield a fast response for the controller. It's worth mentioning that this microcontroller can act like a small computer. Bao et al. (2017) considered the physical restrictions and designed a digital PID controller to control the system because when dealing with microcontrollers definitely, the system will be digital system. The control method was a cascade PID controller consisting of two stages. The first stage is controlling the angle position, and a PD controller was used for this purpose. The second stage was to control the rate of change in the Roll, Pitch and Yaw angles.

There are two types of digital PID controllers; the position PID and the incremental PID. The position PID can be described in time domain as follows,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{1.36}$$

Also, it can be described in discrete-time form as follows,

$$u_{(k)} = K_p e_{(k)} + K_i \sum_{j=0}^{k} e_{(j)} + K_D \left[e_{(k)} - e_{(k-1)} \right] \tag{1.37}$$

The summation in the previous form can be avoided by representing the PID form as follows (Toivonen, 2017, p.7),

$$\Delta u_k = u_k - u_{k-1} = K_p \Delta e_k + K_i e_k + K_d \Delta^2 e_k \tag{1.38}$$

Batmaz, *et al*., (2013, p. 2) and Basdogan, (2004, pp. 4–8) explained how to represent the PID controller with Z-Transform,

The derivative part is:

$$y_{(k)} = \frac{f_{(k)} - f_{(k-1)}}{T_s}, \qquad \frac{Y[z]}{F[z]} = \frac{z-1}{zT_s} \tag{1.39}$$

The integral part is:

$$y_{(k)} = y_{(k-1)} + \frac{f_{(k)} + f_{(k-1)}}{2} T_s, \qquad \frac{Y[z]}{F[z]} = \frac{T_s}{2}\frac{z+1}{z-1} \tag{1.40}$$

So, the PID controller in the Z-Transform will be,

$$\frac{U[z]}{E[z]} = K_p + K_i \frac{T_s}{2}\frac{z+1}{z-1} + K_d \frac{z-1}{zT_s} \tag{1.41}$$

The incremental PID was used to control the velocity of the motors. Bao et al. tried to simplify the quad-copter model and neglect any small values to avoid complex calculations; he also tried to minimize the math operations undergone by the microcontroller. The disadvantages are as follows: it didn't explain the connection between sensors in the system with the mathematical models; there was a significant difference between the simulation result and the practical result as shown in Fig. 1.8.

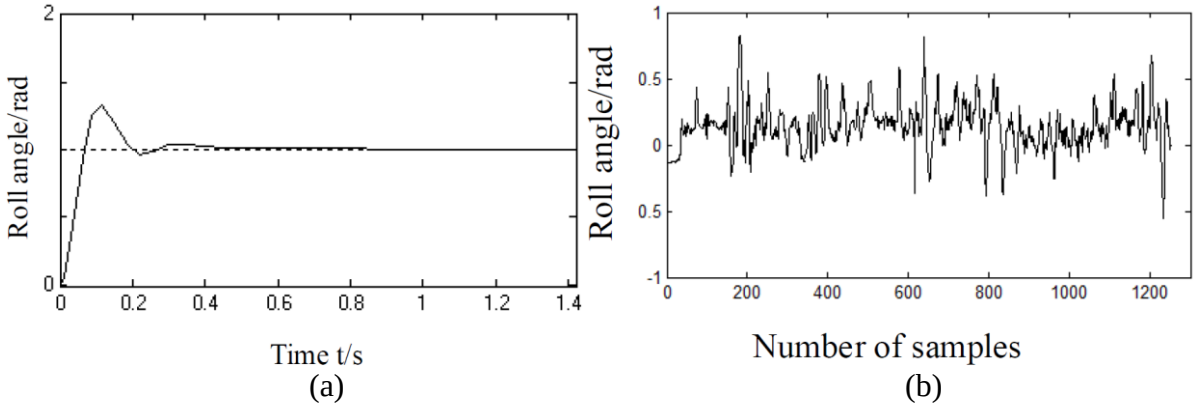


Figure 1.8: (a) Step response for Roll angle in simulation, (b) Response of Roll angle in flight test

Dikmen, *et al*., (2009) compared between different control algorithms for controlling the quad-copter. The authors compared four methods which are PD Control, Inverse Control, Backstepping Control, and Sliding Mode Control.

The PD control algorithm has been studied in this chapter; we will concentrate on the Inverse Control algorithm. The other two methods have complex math, so we will only mention the

results according to the authors. The block diagram of the Inverse control method can be seen in Fig. 1.9.

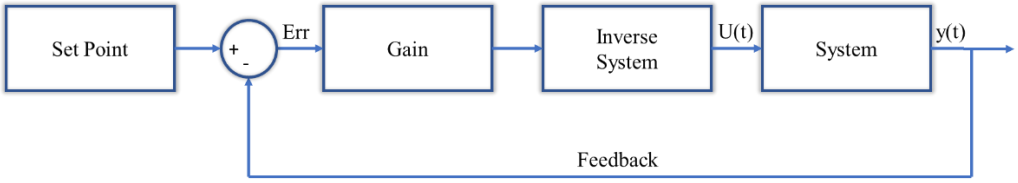


Figure 1.9: Inverse control block diagram

The relation between the angular acceleration and control inputs can be expressed as in Equation 1.3 but the notations can be rewritten in a simplified form as follows:

$$\begin{bmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix}=\begin{bmatrix}0 & a_1+a_2\Omega_r & a_1\\ a_3+a_4\Omega_r & 0 & a_3\\ a_5 & a_5 & 0\end{bmatrix}\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}+\begin{bmatrix}b_1 & 0 & 0\\ 0 & b_2 & 0\\ 0 & 0 & b_3\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \tag{1.42}$$

Where, $a_1=\frac{I_y-I_z}{I_x}$, $a_2=-\frac{J_r}{I_x}$, $a_3=\frac{I_z-I_x}{I_y}$, $a_4=\frac{J_r}{I_y}$, $a_5=\frac{I_x-I_y}{I_z}$, $b_1=\frac{l}{I_x}$, $b_2=\frac{l}{I_y}$, $b_3=\frac{l}{I_z}$

Now, by inverse control method, the state space can be presented as,

$$\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix}=\begin{bmatrix}b_1 & 0 & 0\\ 0 & b_2 & 0\\ 0 & 0 & b_3\end{bmatrix}^{-1}\begin{bmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} - \begin{bmatrix}b_1 & 0 & 0\\ 0 & b_2 & 0\\ 0 & 0 & b_3\end{bmatrix}^{-1}\begin{bmatrix}0 & a_1+a_2\Omega_r & a_1\\ a_3+a_4\Omega_r & 0 & a_3\\ a_5 & a_5 & 0\end{bmatrix}\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{1.43}$$

From the desired angular velocities and accelerations, the control inputs can be found. These are three equations with four variables; the authors didn't explain how to solve this problem. But Gibiansky (2012, pp. 8-9) provided the fourth equation; the summation of the forces in the equilibrium point will equal the gravity force.

$$U_1=\Sigma T=\ c_T(\varpi_1^2+\varpi_2^2+\varpi_3^2+\varpi_4^2) \tag{1.44}$$

The authors assumed that the angular velocities Ω_r are constant in hovering around the equilibrium point. One disadvantage of the state space model is that it needs to neglect a lot of terms, and it should assume some variables as constant. In that way, the linearized system will be different from the original system and it can fail in harsh conditions and strong disturbances.

The results of the inverse control algorithm and the other algorithms are shown in Fig. 1.10. From the figure, we can guess that the sliding mode is the best controller, but the sliding mode and Backstepping algorithms have heavy mathematical equations that can restrict the practical implementation of this method. Inverse control was the worst; that can be clear because the method neglects a lot of force terms. PD control gave the most logical and acceptable results because it is easy to implement in practical implementation.

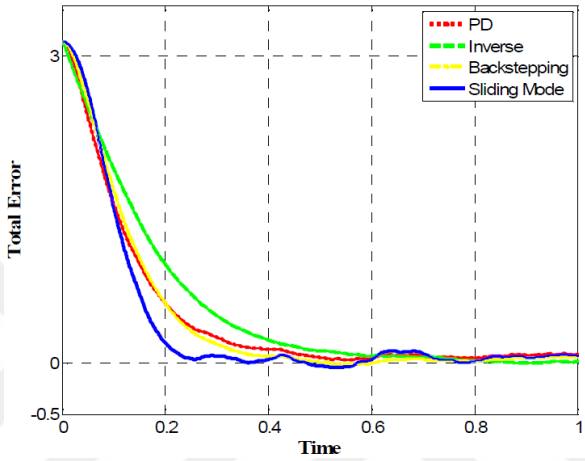


Figure 1.10: Comparison between PD, Inverse, Backstepping, and Sliding Mode control algorithms for controlling quad-copter

Masse, et al. (2018) compared between Linear Quadratic Regulator (LQR) synthesis and structured H_∞ synthesis to control and stabilize a quadcopter under windy weather condition. It considered the disturbances to the system with a wind speed up to 36 Km/h. As a previous work Waslander & Wang (2009) tried to make estimation for the wind, but estimating the wind speed is something very complicated and needs intensive math and a huge database. Instead of this, Masse et al. (2018) tried to construct a robust control to stabilize the UAV in confrontation of the sustained wind and wind gust.

Masse et al. (2018) used state-space model to describe the quad-copter model; it consists of the states of the kinematical, dynamical, force, moments, and motor's variables. The motor rotational speeds were the input state model of the state space model. Using a state space model, it's easier to check every state in the model and control it separately. Masse et al., (2018) supposed that there is an equilibrium position for the states X, Y, Z and Psi; the derivation of the states at this point should be zero. The same thing has been done for the motor parameters; the author considers an equilibrium point for the motor parameters to represent the motor dynamics.

Masse et al., (2018) modeled the wind as a nominal velocity plus random velocity. The wind velocity vector had been filtered with a White Gaussian Noise for the three axes. The result of wind disturbance modeling was the wind force vector and wind moment vector. These values will be integrated with the drone's force vector and moment vector. As it is mentioned in the paper, the two control methods will be LQR and the structured H_∞, but one drawback in the comparison is that when the LQR optimal control method is applied, the motor dynamics is neglected, but when applying the structured H_∞ optimal method, the motor dynamics is considered. Moreover, they didn't try to modify the parameter of the LQR method - Q and R weighing matrices – which can affect the performance. A Robust Control Toolbox in Matlab is used by Masse et al. to apply the structured H_∞ to the system and compare it with the results of LQR. The advantages of using the H_∞ control method, is the ability to apply constraints in the frequency domain.

First, an equilibrium point has been used to be the reference and try to stabilize the quad-copter over that point. Second, they provided a trajectory to compare between the results. The results show that the structured H_∞ control method responds faster to the disturbances, and it tries to compensate the error faster than the LQR method regardless of the performance compared to structured H_∞. But it is easier to understand and tune its weighing parameters (Q and R).

2. GENERAL SECTION

2.1. OVERVIEW

In this chapter, a brief introduction of the control theory is summarized including the differences between the open loop and closed loop systems, the representation of the physical systems in time domain and frequency domain, the stability criterion like Nyquist, Bode plot and Root-locus and some advantages and disadvantages of using these methods, and the definition of linear systems and non-linear systems. This chapter also sheds light on the definition of P, PD, PI and PID controllers, the differences between these controllers, where these controllers can be used, and how the improvements in the system respond after using these controllers. In this chapter, genetic algorithm optimization theory is explained, and the definition of genetic algorithm's parameters such as individual, population, iteration, cost-function and mutation are presented. In this research, Fuzzy algorithm and Neural Networks were used to improve controller performance; the basic concepts of these algorithms are explained in this chapter.

2.2. CONTROL THEORY

Control theory is a branch of mathematics that concentrates on controlling dynamical operating systems. The aim of control theory is to build a control model to control systems such as machines with fast response and without overshoot in addition maintain system stability. Automatic control has represented an important role in managing our modern life. As it is known, the control theory is the foundation stone of the industrial revolution; it is converting all industrial facilities to become fully automated. The aim was to change our life to fully automated and full-of-Robots life that can in turn serve the humanity. Therefore, robots now are used to do very sensitive medical operations, explore the space, manage the warehouses and factories, and for entertainment. All of such applications depend on strong controllers as a base for their operations. Thus, the quad-copter drones use automatic control as an essential part to make successful flight operations.

2.2.1. Open Loop and Closed Loop Systems

Control systems are divided into two categories: the open loop systems and the closed loop systems. In the open loop control systems, the action produced from the controller is

independent from the process output. An example of open loop control systems would be a simple irrigation operation: starting to irrigate the plants every specific time without taking information about the soil moisture. The controlled action opens or shuts the water pump, and the process output is the soil moisture, but neither is related.

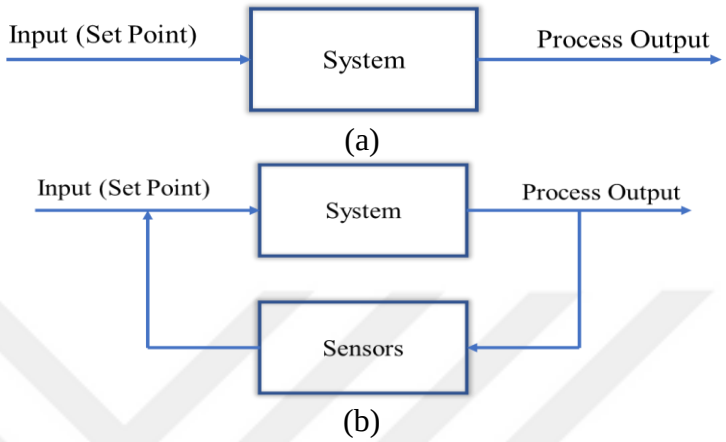


Figure 2.1: (a) Open Loop Systems, (b) Closed Loop Systems

In closed loop control systems, the action produced by the controller depends on the feedback coming from the process output. In the case of the irrigation operation, a closed loop control system would contain a soil moisture sensor to compare between a set point as an input to the system (desired soil moisture) with the reading from the soil moisture sensor (as a feedback). The controller tries to preserve the soil moisture at the desired moisture level by turning the water pump on and off. Mayr (1970) highlighted the definition of closed loop control system in his book as "a control system possessing monitoring feedback; the deviation signal resulting from this feedback is used to control the action of a final control element in a way to reduce the deviation to zero (Mayr 1970, P. 7)"

The close loop system depends on the feedback to compare between the set point as a desired input and the output of the system. The difference between the input value and the output value results from the amount of error; thus, close loop system helps to reduce the error value between the input and the output. In order to choose which control algorithm should be used; we need to determine the precision and the sensitivity of the output of the controlled system. So, if the desired application doesn’t consider the accuracy of the output, an open loop system can be used. But, in most cases, the open loop system can’t give guaranteed performance because it needs a lot of calibration to reach the desired output.

2.2.2. Time Domain and Frequency Domain Representation

In order to run a stable operating system, we should analyze the system stability. Generally, any controlled system is based on a physical structure. From the information of the physical structure, a transfer function can be obtained. The transfer function is the mathematical relationship between the input and the output of the system. Therefore, there are two methods through which we can present the transfer function of a system; in time domain or in frequency domain. In frequency domain, the values that present the inputs, output and feedback of the system are explained as functions of frequency.

The variables of the system can be converted from time domain to frequency domain by some transformation methods, some of these methods are: Laplace transformation technique, Fourier transformation technique, and Z transformation technique. The aim of these transformation techniques is to simplify the mathematical representations. When we simplify the mathematical model of a system, we will be able to get more information about it, and it will be easier for us to test the stability of these domains. However, the main drawback of these techniques is that they can only work with linear systems; if the system is nonlinear, it can't be directly converted to frequency domain.

In time domain, the state variables of the system can be expressed by differential equations as functions of time. Time domain is used to present non-linear systems because of the shortcomings of frequency domain methods. Although it is difficult to fix these non-linear systems, with the help of the current computer technologies, it has become a normal process. Some computer applications such as Matlab make it easier to solve mathematical operations. In time domains, a state space representation can be used to represent the state variables of the system. With this representation, it is possible to understand and study Multi Input Multi Output systems (MIMO) easily.

2.2.3. Stability Analysis Technique

There are some methods that can be used to analyze the stability of the system; some of these methods are Routh-Hurwitz, Nyquist plot, Bode plot, and root locus methods. So, what method should be used to analyze the system depends on how much simple and easy it is to design the controller. This also depends on the system there is no perfect method, but some techniques in

some cases can't describe the properties of the system as it should be. Every method has its advantages and disadvantages; we will give a brief review for each method.

Routh–Hurwitz is a simple and fast method used to determine the stability of any system by some mathematical operations. It is also used to test the stability of linear time invariant (LTI) control systems. It examines if all the roots of characteristic polynomial of a system are positive and if the system is stable. The features of this method are embodied in being simple and fast using only mathematical operations, but unfortunately the use of this method has decreased because of the rise in using computer programs.

Nyquist method is one of the frequency domain techniques that examine the stability of closed loop systems (Nyquist, 1932). Nyquist plot shows the values of the closed loop system roots as well as their transient response. Nyquist plot helps clarify the method and improve the transient response and steady state response of the system. The stability of the system can be determined by two values; gain margin and phase margin. One issue with the Nyquist plot is that if any small changes are made by integral gains, it will change the overall shape of the system curve.

Bode plot is a graphical method for examining the stability of an LTI system in different frequencies. Yarlagadda (2010) defines bode plot as "Bode plots use the asymptotic behavior of the amplitude and the phase responses of simple functions by straight-line segments and are then approximated by smooth plots with ease and accuracy (Yarlagadda 2010, P. 243)". Bode plot consists of two graphs, one is for gain response of the open-loop transfer function and the other is for the phase response. The horizontal axis represented in logarithmic scale represents the frequency; the vertical axis represents the magnitude of gain response in decibels and phase in degrees. The advantages of this technique are as follows: It is easy to predict the system behavior in different frequencies from its shape in system response plot; it is a very useful and powerful method in terms of electronics specially to test the stability and response of the transistors and op-amps in different frequencies. On the contrary, the frequency is factor in Nyquist plot, and it is difficult to find a frequency of a point in the plot.

Another famous method is root-locus discovered by Evans (1948). Exactly like Nyquist and Bode plot, root-locus depends on graphical representations; it describes the closed-loop transfer function of the system in s-domain. The reason for using the s-domain is the ability to represent the real and complex conjugate pairs of the poles and zeros of the transfer function. Root-locus

method determines the response of the system for specific system parameter changes. In addition, it can examine the stability of the system. This method has two important values that help design the controller the ratio, and the natural frequency of the closed-loop system. After constructing the graphical plot and lines of the system, choosing a point in the line with specific damping ratio and natural frequency can help find the required gain needed for the controller. Although, with root-locus, a lot of controlling techniques can be implemented to improve the stability, transient response and steady-state response of the system, some of these controllers are lag, lead, P, PD, PI and PID controllers. Fortunately, there are a lot of programming techniques such as *Matlab* that can make choosing the suitable controller much easier than before.

2.2.4. Linear Control Systems

Linear control systems consist of linear differential equations. These systems comply with the superposition principle, which means that the output is linearly related to the input. So, if the input is multiplied by a value, the output result will be multiplied by the same value, and the sum of input signals will result in sum of the output signals. One branch of linear control systems is the Linear Time Invariant (LTI) systems; the parameters of these systems don't change with time. LTI systems can be represented in frequency domain by mathematical conversion techniques such as Fourier transform, Laplace transform, Z transform, and other techniques. After the conversion of LTI systems with these methods, the stability can be easily checked with the stability examining methods as mentioned before, such as Nyquist plot, Bode plot, and root-locus. Although stability criterion techniques can give useful information about the system such as gain, frequency response, poles, and zeros of the system, these techniques help find suitable controllers to improve the overshoot, transient response, and steady state response of the system.

2.2.5. Non-Linear Control Systems

Non-Linear control systems consist of Non-linear differential equations. These systems don't comply with the superposition principle, which means that the output isn't linearly related to the input. Most of the real-world systems are non-linear; this adds difficulties to the use of the stability mechanisms to investigate the stability of the system and design suitable controllers to it. The way to use the traditional stability criterion is to apply some mathematical techniques (to do system linearization) by approximating the non-linear systems, but if the linearization

methods results were not acceptable and don't describe the system in an efficient way, some of the complicated non-linear stability criteria can be used such as Lyapunov stability theorem. These non-linear stability criterion methods are mathematically very complicated, and it is almost impossible to solve them manually, but with the use of computers and analytical programs like Matlab, it has become possible.

2.3. PID CONTROLLER

Proportional, Integral, Derivative (PID) controller is a controlling mechanism that depends on system feedback to improve the stability and performance of the system. PID controller is a very famous method for controlling the speed of motors, furnace heaters, industrial systems and a lot of other applications that need a continuous controlling mechanism. The idea of the PID controller is to get the error of the system by finding the difference between the required set point (SP) and the feedback - measured process variable (PV) - to improve the system reaching the desired set point as fast as possible without high overshoot and with zero steady state error.

In this section, we will consider a DC motor as an example to the development of system performance, and also to compare between the coming control methods. The transfer function of the DC motor is expressed in Equation. 2.1.

$$f(x) = \frac{18}{0.005\,s^2 + 0.06\,s + 324.1} \tag{2.1}$$

The closed loop response of the DC motor without applying any controller is shown in Fig. 2.2.

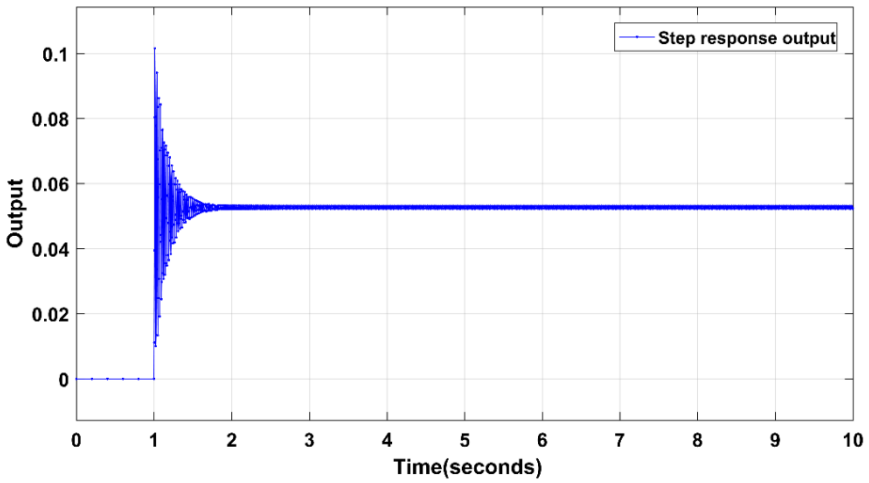


Figure 2.2: Closed loop response to step without controller

2.3.1. Proportional (P) Controller

The proportional (P) controller means proportional to the current error resulted by the difference between the set point and the process variable. So, if the error is large and positive, the output of the P controller will be proportional to this value, and it will be a large and positive value. This value will be the input of the system; this will result in a large PV value and will help reduce the error. The construction of the system blocks will be as shown in Fig. 2.3.

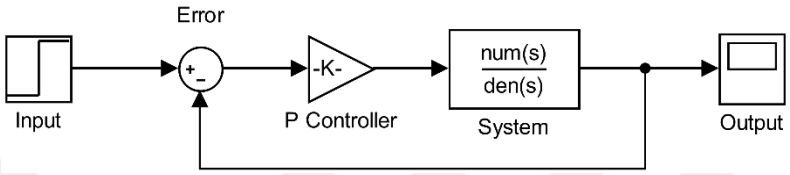


Figure 2.3: Closed loop system with P controller block diagram

Increasing the gain value will change the closed loop system dynamics. Increasing the gain can lead to the decrease of the steady state error of the system, but it will never eliminate it; it will increase the overshoot and decrease the rise time. P controllers can be useful and effective only for the systems with first order processes, but for higher order, processes P controller cannot be sufficient for the stabilization of such systems. As the proportional gain increases, the dynamics of the system will be faster with broader frequency band and more sensitive to noise. In addition to the amplification that happened to the process variable, it also amplifies the process noise. P control can result in oscillating the output, and the oscillation problem will increase by increasing the order of the system. For the DC motor example, we applied a proportional gain controller with a value of $k = 100$. The result of the controller is shown in Figure 2.4.

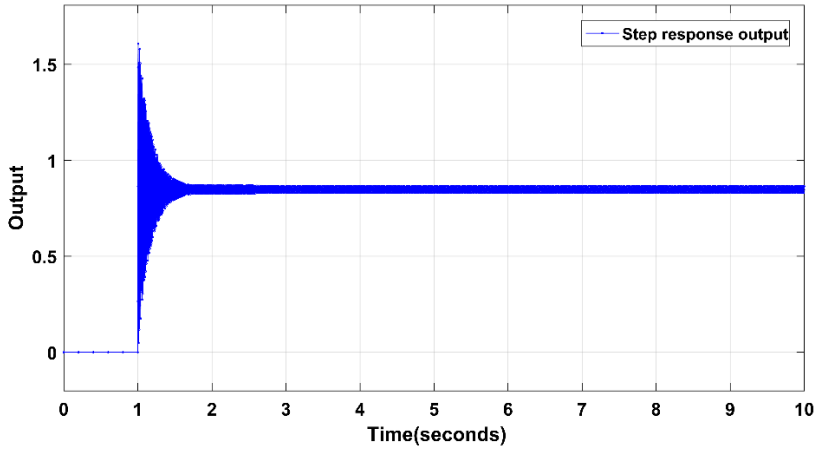


Figure 2.4: Step response of DC motor controlled by P controller

The transfer function of the DC motor example is classified as a second order system, and as mentioned above, the P controller can result in oscillation for systems higher than the first order. Also, the steady state error is improved but still not acceptable.

2.3.2. Proportional-Integral (PI) Controller

The integral part of the PI controller refers to the integration of the previous difference value of the set point and the process variable, so it integrates all the past errors over time. As mentioned in the proportional control section, the P controller can't eliminate the error; thus, the integral term will eliminate the residual steady state error by examining the cumulative error value. Also, it will inhibit the oscillation of the output if any. PI controller is a useful, low-cost, easy-to-implement controller. It can be used for some industrial applications, especially with applications that neglect time response, but the PI controller can give bad results when the nonlinearity of the system is large. One of the advantages of PI controller is the correction of the error when large noise and disturbances are present in the process. One of the disadvantages of the PI controller is slowing down the response; thus, in some applications, the response time is critical, and it can affect the stability of overall system.

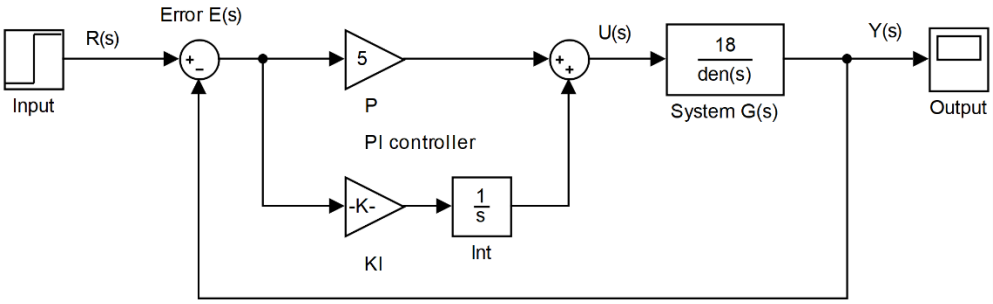


Figure 2.5: Closed loop system with PI controller block diagram

The formula of the PI controller in time domain can be found in Equation 2.2 and in S-domain in Equation 2.3, the block diagram of PI controller is shown in Fig 2.5.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt \tag{2.2}$$

$$U(s) = (K_p + \frac{K_i}{s})E(s) \tag{2.3}$$

Applying the PI controller to the DC motor example should eliminate the steady state error, and it can dampen the oscillation as shown in Figure 2.6. If we compare the results of using only P

controller shown in Figure 2.4 with PI controller shown in Figure 2.6, we will notice the improvement in the steady state response and system performance.

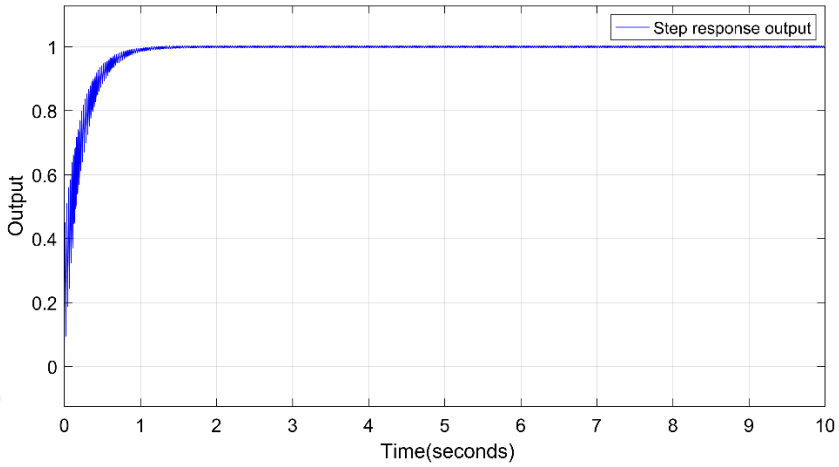


Figure 2.6: Step response of DC motor controlled by PI controller

2.3.3. Proportional-Derivative (PD) Controller

PD controller can be used to improve the stability of the system; the derivative part of the controller can predict the future errors of system response. Derivative controller deals with the change of error; however, it doesn't react with constant errors since the rate of change of a constant error with respect to time is zero. Some methods take the change of error from deriving the error signal, but to avoid sudden changes in the error signal, the derivative can be taken directly from the output of the system response. One of the advantages of derivative controller is that it can damp the oscillation in the output signal, increase the stability of the system and decrease overshoot, risetime, and settling time, but it can amplify the process noise.

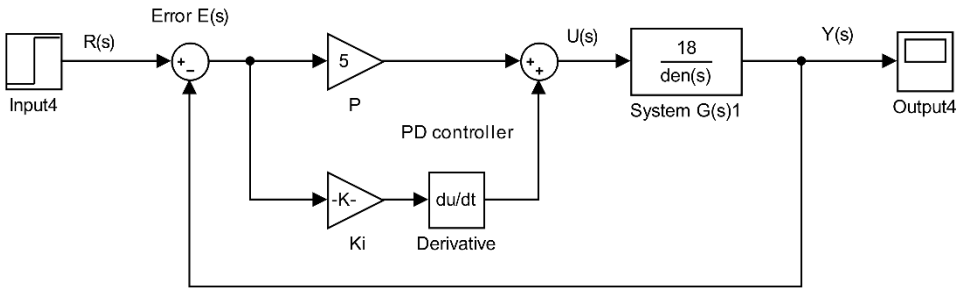


Figure 2.7: Closed loop system with PD controller block diagram

The formula of PD controller in time domain can be found in Equation 2.4 and in S-domain in Equation 2.5, the block diagram of PD controller is shown in Fig 2.7.

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t) \tag{2.4}$$

$$U(s) = (K_p + K_d s) E(s) \tag{2.5}$$

Applying the PD controller to the DC motor example should override the oscillation in the response of the system and decrease the overshoot if any as shown in Fig 2.8, therefore, the steady state error will remain not modified.

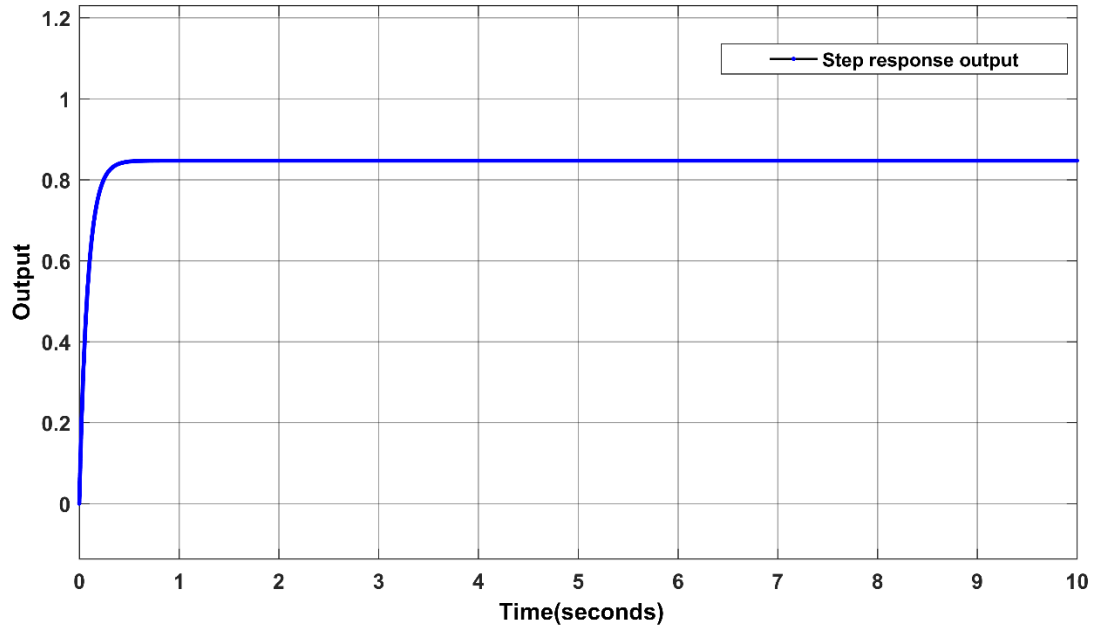


Figure 2.8: Step response of DC motor controlled by PD controller

2.3.4. Proportional-Integral-Derivative (PID) Controller

Many applications and machines use PID controllers to optimize the behavior of system response. PID controller is a most commonly applied algorithm and a well-known controller that can be used to control various machines such as {Furnace Temperature Control, motion control, Speed control in vehicles and a lot of other applications}.

Also, the structure of the controller and system can vary from application to another; thus, it can be used as a serial controller, parallel controller, serial-parallel controller, modified PID controllers or integrated with other optimization techniques.

PID controller has collected all advantages of proportional, integral, and derivatives controllers; thus, it responses quickly to any changes of the output (Derivative term), it eliminates the

constant error as to reach to a steady state error of zero (Integral term), and it increases the speed of response (Proportional term).

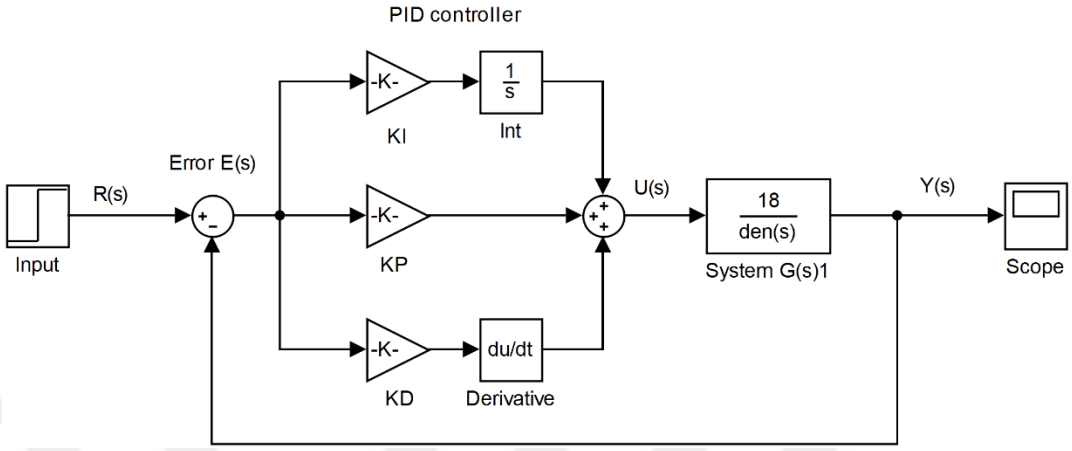


Figure 2.9: Closed loop system with PID controller block diagram

The formula of PID controller in time domain can be found in Equation 2.6 and in S-domain in Equation 2.7, the block diagram of PD controller is shown in Fig 2.9.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{d}{dt} e(t) \tag{2.6}$$

$$U(s) = (K_p + \frac{K_i}{s} + K_d s)E(s) \tag{2.7}$$

Applying the PID controller to the DC motor example should override the oscillation, eliminate the steady state error, improve settling time, rising time, and decrease the overshoot of system response as shown in Fig 2.10.

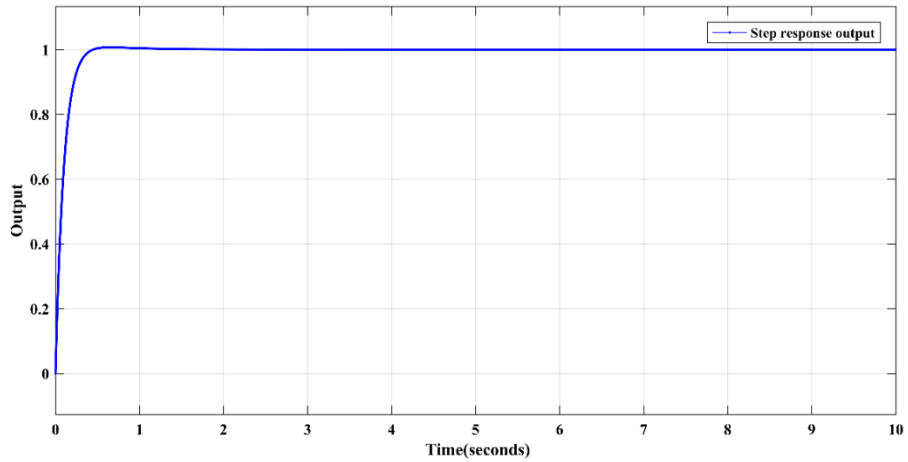


Figure 2.10: Step response of DC motor controlled by PID controller

The response of DC motor example with a serial PID controller shows fast response, no oscillation, and a zero steady state error. Compared to all previous trials of (P, PI, and PD) controllers, PID controller gave the desired performance due to its ability to overcome all kinds of errors in the system.

2.4. GENETIC ALGORITHMS

Genetic Algorithm (GA) was first suggested by Holland (1975) in his book Adaptation in Natural and Artificial Systems. The goals of his research have been twofold: first, to abstract and rigorously explain the adaptive processes of natural systems, and second, to design artificial systems software that retains the important mechanisms of natural systems. This approach has led to important discoveries in both natural and artificial systems science.

Over the last 40 years, GA's have been used to solve a wide range of search, optimization, and machine learning problems (Goldberg, 1989). Angeline et al. (1994) used GAs to constructs recurrent neural networks. Weller et al. (1995) used GAs to evolve an optimum input set for a predictive neural network. Jones & De Moura Oliveira (1995) produced a genetic tuning algorithm for PID controllers. Kim et al. (1995) designed fuzzy Neural controllers using GAs, and numerous others researches used GAs in control systems design.

As the name suggests, genetic algorithm is an optimization method that depends on the idea of biological evolutionary operation. Generally, searching algorithms was based on enumerative schemes and calculus-based, but these methods in a lot of cases can fail to find a solution. That's why there was a need to find alternatives to these methods such as random search algorithms. Unfortunately, random search algorithms also failed in finding solutions because they miss the efficiency requirement. GA's are one of the random search algorithms, but GAs successfully override the difficulties of other random search algorithms. GA's are different from normal optimization and search procedures in four ways (Alibeiki, 2010):

1- GAs deal with encoded data such as a chromosome containing three different variables.
2- GAs use objective function information, not derivatives or another auxiliary knowledge.
3- GAs use probabilistic transition rules, not deterministic rules.
4- Almost all conventional optimization techniques search from a single point, but GA always operates on a whole population of points (parallelism).

2.4.1. Basic idea of GAs

GA's are search algorithms that depend on the idea of natural selection as a base stone to the search process. Natural selection is the selection process of the fittest strings which are combined to form a new better generation. GA's start the search operation with a population of chromosomes, which are represented in a coded form. The chromosomes are evaluated with a fitness function that gives an index for selection and reproduction operations. GA should address five issues:

1- Initial population
2- Fitness Function
3- Selecting parents
4- Crossover operation
5- Mutation operation

The general structure of GA's stages is shown in Figure (2.11). At the end of this section, 8-queen problems were presented to show an example of GA solving mechanism.

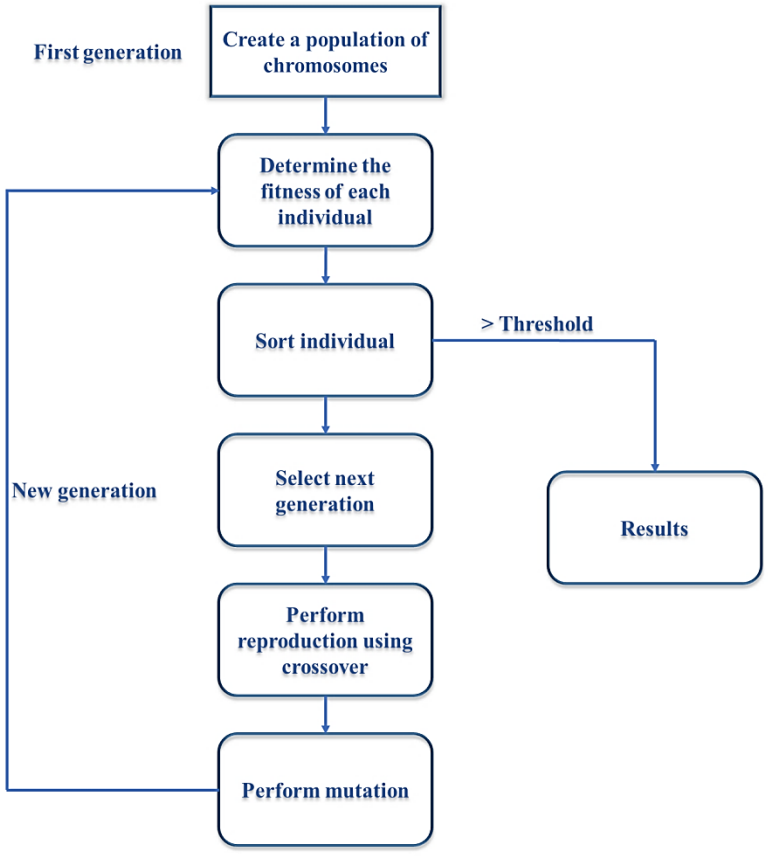


Figure 2.11: Sequence of Genetic Algorithm

2.4.2. Initial Population

The two important elements to initialize the GA are individuals and populations. An individual represents a single solution whilst the population represents a group of individuals cooperated in the search process.

2.4.2.1. Individual:

Now, we must construct the essential element in the GA which is the (individual); the individual contains the system's variables which represent one solution. Let us assume that our system has three variables, x, y and z. The object function is $f(x,y,z) = x^2 + 2y + 3z^{-1}$, so the individual will contain three variables together, as every variable represents a gene, and genes together in one row can be expressed by one chromosome, Figure (2.12) shows the structure of one chromosome.

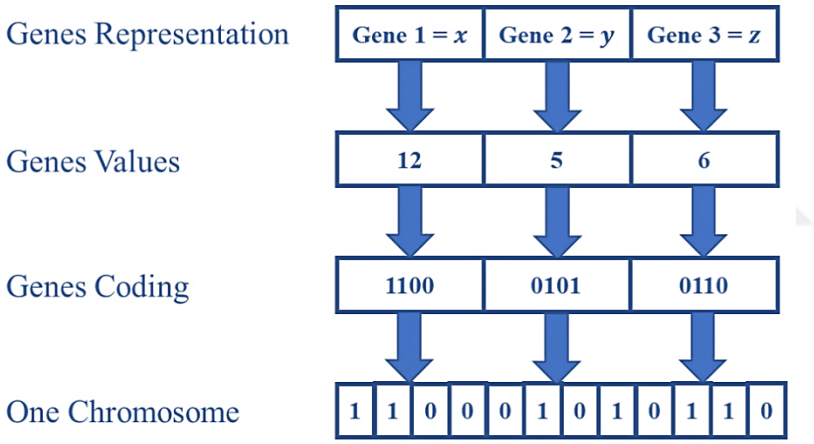


Figure 2.12: Chromosome representation

A chromosome is subdivided into genes. A gene is the GA's statement of a single variable for control variables. All the variables or genes are gathered to form one chromosome which can be defined as a unique solution. One chromosome stands for one solution, but in genetic algorithm there must be more than two solutions because as we've mentioned before, the GA method depends on parallelism, so we must have a population.

2.4.2.2. Population

A population is set of individuals. There are two important factors for defining the population, first is the population size, and second is the initial population. The size of the population depends on the specification of the problem, sometimes large number of chromosomes gives faster results, but in some cases; big populations can slow down the optimization process. In

that case, smaller populations will give faster results. Although the initial population is very important in GA's process, a good initial population will help the GA's to reach the optimum solution faster. Figure (2.13) shows that the population in binary coded consist of a group of individuals (a population of four chromosomes).

Chromosome 1	1	1	0	0	0	1	0	1	0	1	1	0
Chromosome 2	0	1	0	1	1	1	0	1	0	0	1	1
Chromosome 3	1	0	1	0	0	0	0	1	0	1	0	0
Chromosome 4	0	0	0	1	0	1	1	1	0	1	1	1

Figure 2.13: Population Representation

2.4.2.3. Encoding

Encoding is the method of representing the variables of the system in chromosomes. The encoding can be in the form of binary bits, decimal numbers, arrays, or any other form of encoding. So, which encoding method should we use to represent our chromosomes? The answer of this question is the way of solving our problem; sometimes it is efficient to use binary encoding. In other cases, it is better to use decimal encoding. For example, one can encode directly real or integer numbers, but we can ask a question why we need to convert the decimal number to binary or other form, and the answer is to make our scale larger and clearer. We will present four coding methods: Decimal numbers encoding, Binary numbers encoding, Octal numbers encoding and Hexadecimal numbers encoding. Decimal numbers are used to represent the numbers or symbols, often used in the problems of arrangement type as shown below in Figure (2.14).

Chromosome 1	8	5	11	20	35	41	12	7	1	3	6	19
Chromosome 2	14	1	0	8	13	28	30	95	88	5	61	77

Figure 2.14: Two chromosomes represented in Decimal numbers

At the beginning, binary numbers were the basic encoding method to be used for GA operations. Binary numbers are numbers expressed in base-2 numeral system, which declare the numbers using two symbols: typically, Zero (0) and One (1). Because of its simple implementation in digital electronics with logic gates, binary numbers are used by all electronic devices such as

computers, mobile phones, microcontrollers and other development devices. Binary numbers also known as ON or OFF, and is the first language that computers used to operate and communicate. Today's computers still employ binary numbers because it is the simplest and most effective method of detecting an electrical signal on or off, detecting magnetic poles with magnetic media like a hard drive, and because it is the most efficient way to control logic circuits. Below is an example of the maximum 8-bit value of 255, which is "11111111" in binary. To get this value add each position (column) as shown in Table (2.1), so that $1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$.

Table **2.1:** Binary number representation**.**

Value:	$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
ON/OFF:	1	1	1	1	1	1	1	1

Normally, counting in binary numbers starts with 0 rather than 1. As an Example, a binary number of 8 bits varies between 0 and 255 in decimal; however, it is 256 values. Table (2.2) is another example of binary number "10001101", which is 1 + 4 + 8 + 128 = 141.

Table 2.2: Binary number example.

Value:	128	64	32	16	8	4	2	1
ON/OFF:	1	0	0	0	1	1	0	1

When the computer reads binary, it is reading from the right to left, so in the above example the computer is actually reading "10110001" and not "10001101" as shown above.

Another method of encoding are Octal numbers. Octal numbers use the numbers between 0 and 7, in total it can represent 8 values in one bit as shown in Table (2.3).

Table 2.3: Chromosome consists of Octal numbers.

Chromosome 1	0	3	4	6	7	2	1	6
Chromosome 2	4	1	6	7	5	3	0	4

The last example of encoding variables is Hexadecimal numbers. Hexadecimal numbers use the numbers between 0 and 9 also characters between A and F, in total one bit can represent 16 values as shown in Table (2.4).

Table **2.4:** Chromosome consists of Hexadecimal numbers.

Chromosome 1	3	B	1	0	F	E	5	A
Chromosome 2	5	2	F	1	9	C	8	3

2.4.3. Fitness Function

After the initialization of the elements of the tuning parameters, the objective function is introduced and the value of the objective function is calculated using decoded values of the parameters in each string. In order to evaluate how good the individuals in the population are, a fitness function needs to be defined. A fitness function specifies each chromosome a value that reference how quite that chromosome resolve the given problem. Optimization is one of a popular application of genetic algorithm (Chen and Chang, 2006).

Here we display some of the famous fitness functions for control applications (Alexandru, 2008).

1- Integral of Squared Error (ISE)

$$Fitness.value = \int_0^\infty e^2(t)dt \tag{2.8}$$

Where e is the difference between the desired and actual signals (error signal), this function can detect errors quickly, but it can amplify the oscillation.

2- Integral of Absolute Error (IAE)

$$Fitness.value = \int_0^\infty |e(t)| \tag{2.9}$$

This fitness function can give pretty response, but it is not efficient for selection.

3- Integral of weighted Time Squared Error (ITSE)

$$Fitness.value = \int_0^\infty te^2(t)dt \tag{2.10}$$

This fitness function can give a quick tracking and good response, but it is definitely related to time.

4- Integral of weighted Time Absolute Error (ITAE)

$$Fitness.value = \int_0^\infty t|e(t)| \tag{2.11}$$

This fitness function can give fast tracking and sensitive to small errors.

Thus, for example, if minimum ISE is regarded for PID parameter tuning as the ultimate design requirement, GAs can be readily used to select the set $\{K_p, K_i, K_d\}$ for tuning parameters such that the generalized ISE is minimized. This performance index is computed by subjecting the plant concerned to a set-point change. In each case, the function in Equation (2.8) is evaluated. These criteria require that the ISE of the dynamic response be minimum, i.e. the area between the integral of the response curve and the set-point to be minimum. The ISE is only one criterion IAE, ITSE, ITAE or any other could equally be used depending on objective function. Indeed, actuator limits, or any limits can also be included in the cost function.

In general, genetic algorithm maximizes its cost function. However, for PID controller parameter tuning example and as in many other optimization problems; the logical process is to minimize a specific cost function, g(x), rather than maximize it. In normal operations, to convert a maximization function to a minimization function, the function can be multiplied by a minus one. However, this operation alone is insufficient because the objective function is not guaranteed to be non-negative in all instances. In the case of GAs, the most common objective function-to-fitness transformation is therefore of the form,

$$Fitness\ Function\ f(x) = \begin{cases} W_{max} - g(x), & g(x) < W_{max} \\ 0, & g(x) \geq W_{max} \end{cases} \tag{2.12}$$

Where W_{max} is design parameter. There are a variety of ways to choose the coefficient W_{max}, W_{max} may be taken as an input coefficient, as the largest $g(x)$ value observed so far, as the largest $g(x)$ value in the current population, or the largest of the last k generation.

2.4.4. Selecting Parents

Selecting parents or (reproduction process) is a process in which particular chromosomes are copied depending on their obtained fitness values. The Darwinian Theory explains the natural selection of the fittest creatures which is the ability of creatures to adapt and override life obstacles; thus, the same idea was used in selecting parents' process in GA's. The chromosomes with bad fitness values will be discarded, and the chromosomes with best fitness values will be selected and copied to

complete the population. There are a lot of methods for selecting and reproducing parents; we will highlight three of the known selecting algorithms: Roulette wheel selection, Random Selection, and Stochastic Universal Sampling.

Roulette wheel method depends on creating a partitioned roulette wheel where each chromosome in the population has a part of the roulette wheel. The size of each part depends on the fitness value of that chromosome. As an example, four chromosomes with different fitness values (the values in the table are chosen randomly) are presented in Table (2.5). Summing the fitness values for all the four chromosomes will give a total of 2000. From the total value each chromosome can take a percentage from Roulette wheel as shown in Figure 2.15.

Table 2.5: Sample Problem String and Fitness Value

Chromosome Number	Chromosome String	Fitness $f(x)$	% of Total
1	01101	400	20
2	11000	100	5
3	01000	500	25
4	10011	1000	50
Total		2000	100

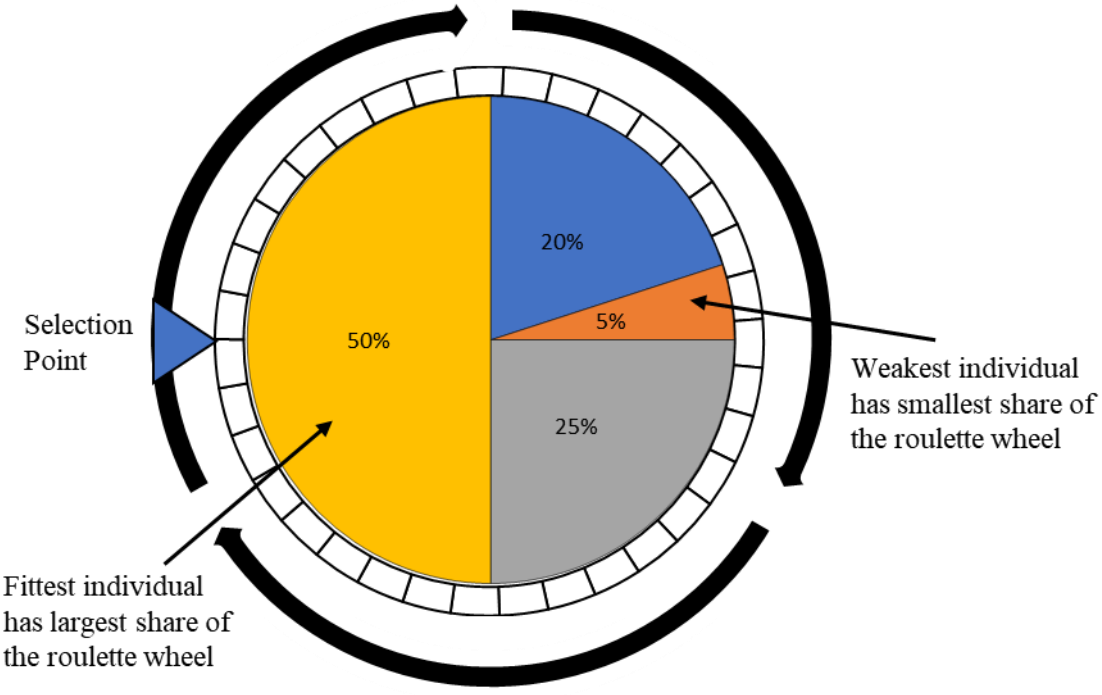


Figure 2.15: Roulette Wheel Selecting Method

For the example above, string number 1 has a fitness value of 400, which represents 20 percent of the total fitness. As a result, string 1 is given 20. In this way, the fittest chromosome will have a higher chance to be chosen for parents' reproduction process.

Random Selection method selects parents randomly to form a new population. For some systems, random process can give better performance for genetic operations, but sometimes it can be a little more disruptive. In comparison with Roulette wheel, Roulette wheel is much famous.

Stochastic Universal Sampling technique is similar to Roulette wheel method; the difference is in representing the number of adjacent segments into a line, such that every segment occupies the same fitness value size. Then, another line with deterministic space size pointers are placed above the divided segments to select parents, the number of pointers depends on required parents' numbers as shown in Figure 2.16. A random number will choose a starting point for the pointer line.

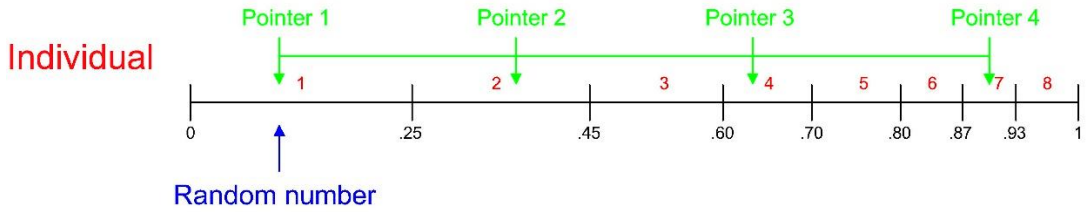


Figure 2.16: Stochastic Universal Sampling Technique

From the above figure, it can be seen that the reproduction process includes a number of steps. The first is the testing of each parameter against the fitness function so as to allocate fitness against the parameter. The second is the mapping of the objective function to the fitness function.

2.4.5. Crossover Operation

This process depends on the selection operation; after selecting the suitable parents and completing the population with the fittest individuals, a crossover operation is done between the chosen mating pool to generate new individuals and to complete a new better offspring. The crossover operation consists of three steps (Raiha, 2008):

1- Randomly select two individual strings from mating pool for mating.

2- A random crossover point should be chosen.

3- Finally, swapping between the two parents from crossover point is executed.

There are some known methods for carrying out crossover operations. These methods are: Single-Point Crossover, Two-Point crossover, Multi-Point Crossover, and Uniform Crossover.

Single-Point Crossover is the simplest method used in GA's because it depends on one crossover point. The two mating chromosomes will be cut from the crossover point and it will be divided into two sections, then the two sections from the mating chromosomes will be exchanged as shown in Figure (2.17). The crossover point is randomly chosen from every mating operation, which means that the next crossover point for the next pairs can be different than the previous pairs in the same population (Raiha, 2008).

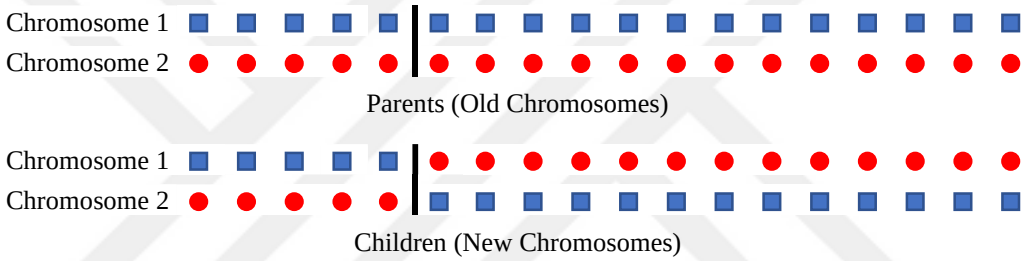


Figure 2.17: Single-Point Crossover Operation

Two-Point Crossover is similar to One-Point crossover, but the difference is in the region to be exchanged. In this method, two random points are necessary to determine the region; the area of the region can be expanded or collapse due to the random points that have been generated by a random function generator. The disadvantage of this method is that in some application the building blocks of their chromosome can be disrupted. The advantage of this method is that the search principle can be considered wider. The structure of two point's method is shown in Figure 2.18.

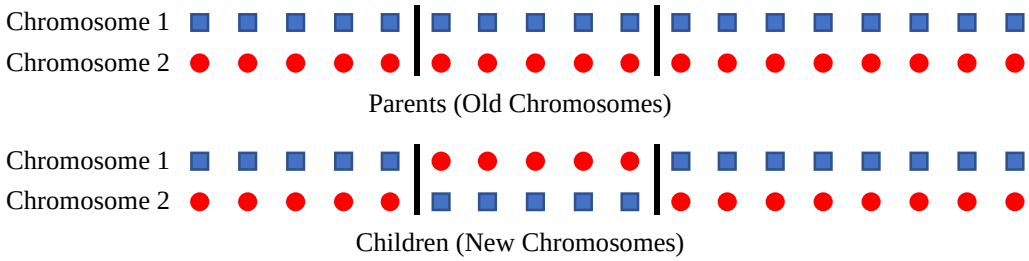


Figure 2.18: Two-Point Crossover operation

Multi-Point Crossover is somehow a complex method because the designer should understand the response of the optimization process before using this method. The multi-points can be either an odd number or even number, odd or even numbers can accelerate or decelerate the optimization process as explained by Spears and De Jong (1991) in their research "An Analysis of Multi-Point Crossover".

Uniform Crossover is one kind of Multi-Point crossover. This method differs from the previous methods; it uses a mask chromosome to complete the crossover operation. The mask chromosome consists of ones and zeros bits. For the first child, if mask bit value is one, it takes the bit value from first parent; if mask bit value is zero, it takes the bit value from the second parent. For the second child, if mask bit value is one, it takes the bit value from the second parent; if mask bit value is zero, it takes the bit value from first parent as shown in Figure 2.19.

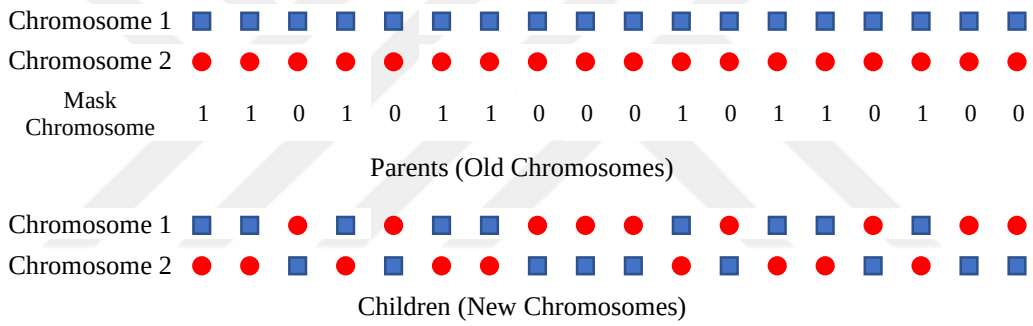


Figure 2.19: Uniform Crossover operation

2.4.6. Mutation

The last operation for producing a new generation is the mutation. The mutation operation is a method to prevent GA from being trapped by a local minimum solution. Mutation plays an important role in regaining some lost gene materials and disturbing the chromosomes information. Mutation operation is swapping one bit or one number in the chromosome to a different value randomly. As in binary numbers, we choose any bit randomly and swap it; if it is 'one', it will be swapped to zero and vice versa. If the coding is in decimal, the value of the swapped number can be maximum value of that number or any suitable value determined by the programmer. The mutation operation can happen with low probability; it can be every 5% of the new generation or one time every 500 chromosomes, this number can be determined by the programmer depending on the processed system.

2.4.7. Example for 8-queen problem

One example is solving 8-queen problem by Matlab software and Genetic Algorithm. 8-queen problem is to put 8queens in the board so that no queen attacks the other. The first step is to make the initial population; we preferred to start with 1000 chromosomes generated by a Matlab random function "*randperm*". Decimal numbers encoding was used to represent the chromosomes. A sample of the generated initial population is shown in Table 2.6

Table 2.6: Sample of Initial Population for 8-Queen Problem

Chromosome number	Chess row number	1	2	3	4	5	6	7	8	Fitness value
1	Chess column number	1	7	8	3	4	6	2	5	22
2		7	3	5	8	1	2	4	6	24
3		3	6	4	1	5	2	7	8	24
4		6	4	1	8	7	2	3	5	25
5		7	5	8	1	4	2	6	3	25
6		3	5	4	8	7	6	2	1	22
7		4	2	8	3	1	6	5	7	25
8		2	3	5	6	8	4	1	7	24
9		5	1	6	4	7	3	8	2	27
10		3	5	8	6	4	2	1	7	25

After generating the initial population, all the generated chromosomes will be evaluated with a fitness function. The fitness function depends on the number of no-attacking queens. It will test the position of the queen in the first column with the other columns; if the queen attacks another queen, it will not add a credit for that column, and then the test will be repeated for the next queen in the next column and so on. The fitness values of the initial population sample are shown in table 2.6. For example, let us test the first chromosome, the queen in the first column attacks the queen in the sixth column, so it will take 6 credits instead of 7, and we repeat this test for all the queens in all the columns. If the fitness value for the chromosome is 28, queens don't attack each other. This can be considered as a solution and will be added to solution matrix.

For selecting parents, we decided to duplicate the best chromosomes in the population. For crossover operation, we choose to use the simplest method as One-Point crossover. Mutation operation is executed every 5 chromosomes; the mutation is to swap the position of two queens.

The results of GA for solving the 8-queen problem are shown in Table 2.7, GA operation give an output of 15 solutions.

Table 2.7: GA results for 8-Queen problem

Chromosome number	Chess row number	1	2	3	4	5	6	7	8	Fitness value
1	Chess column number	4	6	1	5	2	8	3	7	28
2		6	3	7	2	4	8	1	5	28
3		6	4	1	5	8	2	7	3	28
4		5	2	4	7	3	8	6	1	28
5		6	3	1	7	5	8	2	4	28
6		4	2	8	5	7	1	3	6	28
7		5	2	6	1	7	4	8	3	28
8		4	2	7	3	6	8	1	5	28
9		1	5	8	6	3	7	2	4	28
10		5	1	8	6	3	7	2	4	28
11		4	8	1	3	6	2	7	5	28
12		4	2	7	3	6	8	5	1	28
13		3	6	8	1	5	7	2	4	28
14		4	2	7	5	1	8	6	3	28
15		5	7	2	6	3	1	8	4	28

2.5. FUZZY LOGIC CONTROL

Fuzzy logic is one form of many probabilistic and logic forms; its reasoning is to deal with issues as approximate way rather than certain and exact way. Traditional logic depends on binary sets, so that the logic variable could have a value of true or false, one or zero. Thus, the range of decision has only two options, even 0 or 1. Fuzzy logic has extended the traditional logic control to be wider in range; it considers the concept of partial truth: where the truth value can be either completely true, completely false or in between. Generally speaking, physical processes need precision for system modeling and controller designing. Therefore, most of the time we trust the complex calculations and computer processing to solve these issues and to give exact values, but in some processes and nonlinear systems, the interaction of human logic can be much useful. Fortunately, Fuzzy logic obtained the interaction between exact decisions and human logic to solve a lot of complex processes. Of course, fuzzy logic can’t be used for sensitive processes and applications that need complete precision, such as launching rockets to

space, shooting laser beams over tens of kilometers, dealing with nanotechnology applications, and other sensitive applications. Therefore, other application in industry don't require that much of precision, on contrary, those applications can require high cost, complex calculations, and long time for designing sensitive models and controllers. Instead, Fuzzy logic can come out with sufficient and satisfactory results.

Since the first appearance of fuzzy set was in 1965 (Zadeh, 1965), a lot of developments took place in this area. Fuzzy logic can be used in analysis design and system control, because it helps engineering developments to be faster; also, it can be more effective in cases of highly complex systems. The main concept of the fuzzy logic is that it solves the problems in a way similar to the human brain. When we say that, fuzzy logic will solve the problems similar to human brain logic; this doesn't mean that machine will act exactly like human as machine learning.

2.5.1. Historical Perspective

The major studies in fuzzy set assumption in late nineteenth century and twentieth century were the probability theories. However, the progressive development of uncertainty expression employing probability theory was first introduced by Black (1937) in his titled research "Vagueness", then, Fuzzy logic theory was introduced and explained by Zadeh (1965). Zadeh's paper had a deep impact on the concept of uncertainty, because it challenged the probability theory for representing uncertainty; also, it challenged the basic theory of the probability; the classical logic binary values zero and one. The mathematics of uncertainty was controlled by probability theory for more than five centuries, when the rules of probability in games of chance were recognized by Ore (1953).

The probability concept for explaining the uncertainty was still in the interface in 1684, when the Bishop of Wells wrote a paper including a discussion about two witnesses of two unreliable persons (Ross, 2010), the truth of their witness can only be considered with (P_1 and P_2) probabilities. Bishop's assumption was based on that the two persons got the information from independent sources (Lindley, 1987).

Different theories other than probability and classical logic theories started to appears on the surface to address additional types of uncertainty further than random kind. In 1930, a discrete multivalued logic was developed by Jan Lukasiewicz (Ross, 2010, P. 4). A theory of evidence

was developed by Arthur Dempster to cover for the first-time absence of information or assessment of ignorance (Ross, 2010, P. 4). Finally, after these developments, continuous-valued logic was introduced by Zadeh (1965) with the name of fuzzy set theory.

2.5.2. Fuzzy Logic and Fuzzy Sets

Dealing with uncertain processes is complex, especially when making decisions. The idea proposed by Zadeh (1965) to use set membership for making decision was a good idea to face the uncertainty of the systems. In ordinary crisp sets, the membership of any element is well-defined in the universe; only there are two decisions for it: the element is a membership or the element is not a membership in the universe such as:

$$\mu A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \qquad (2.13)$$

$$\mu A(x) \in \{0,1\}$$

On the other hand, the essence of Fuzzy Set is that it widens the range of the membership to be an infinite range as shown in Equation 2.14.

$$\mu A(x) \in [0,1] \qquad (2.14)$$

To clarify the idea of Fuzzy member ship, two sets are shown in Figure 2.20. Let us consider that the universe contains Set A as a crisp set and set B as a Fuzzy set.

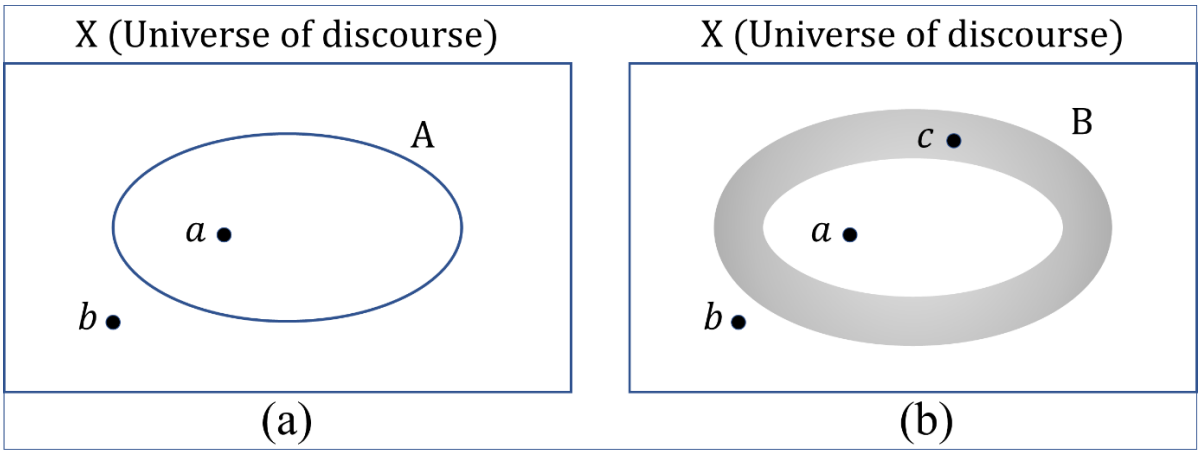


Figure 2.20: (a) crisp set and (b) fuzzy set

As shown in Figure (2.20(a)), universe (X) contains one set (A), and two elements $\{a$ and $b\}$. Element (a) can be considered to have a full membership in set (A); it is completely a member

in set (A). Element (b) can be considered to have no membership in set (A); thus, the membership of element (b) in set (A) is Zero. Therefore, set (A) is a crisp set and any element can have a full membership (1) or no-membership (0).

In Figure (2.20(b)), universe (X) contains three elements {a, b, and c} and one set (B). It is clear that element (a) has a full membership in set (B); however, element (b) has no-membership in set (B). Element (c) is the most important element that explains the main idea of fuzzy membership, because element (c) is not full membership in set (B) and can't be considered to have no-membership, but it has a partial membership in set (B). Thus, if the element is approaching the center of the set, the membership increases to reach full membership, but if it is going away from the center of the set, it will reach a point to have no-membership in that set. This is the importance of the fuzzy logic theory; any element has infinite probability to be a member or not a member of a set.

Fuzzy logic theory is based on linguistic values to describe linguistic variables; the linguistic values are words that explaining the set. For example, temperature normally can be expressed in degrees, i.e. 70 C°, the traditional sets can be numerical values such as specific temperatures {20, 55,80}. In traditional logic, an exact temperature number should be provided to make a decision for a specific output. But, with Fuzzy logic, the variables are words and the sets can be {very cold, cold, normal, hot, very hot}. For example, a temperature with a numerical value of 60 C° can have a membership of 0.85 in a linguistic value of "very hot" and a membership of 0.40 in a linguistic value of "hot". So, the numerical value of 60 C° expressed with two linguistic values {very hot (0.85), hot (0.40)}. Fuzzy set is clarified in a mathematical formula as shown in Equation

$$E = \{(x, \mu E(x)) \mid x \in F, \mu E(x) \in [0.1]\} \tag{2.15}$$

$\mu \mathrm{E}(x)$ is the membership degree of the x element in the fuzzy set E. If the value of the membership is $\mu \mathrm{E}(x) = 0$, the element x doesn't belong to set E at all. If the value of the membership is $\mu \mathrm{E}(x) = 1$, the element x totally belongs to set E.

2.5.3. Fuzzy Set Operations

There are some basic operators in traditional set theory or crisp ordinary set theory; these sets are intersection, union, and complement. In fuzzy set theory, there are much more operators

than traditional theory; therefore, we will explain the known operators only. For example, let us consider two fuzzy sets (membership functions) A and B as shown in Figure 2.21.

Intersection operator in fuzzy sets was explained by Zadeh (1965). He suggested using ($min(x)$) operator as the algebric notation for product (and). After the suggestion of Zadeh, a lot of ideas appeared to show and use other notations (Zimmermann, 1990). Let us suppose that the two sets A and B have two membership functions as μA and μB respectively. The most used intersection operator is shown in Equation 2.16.

$$\mu A(x)\ and\ \mu B(x) = min\{\mu A(x), \mu B(x)\} \tag{2.16}$$

The result of intersection between set A and set B can been seen in Figure 2.22.

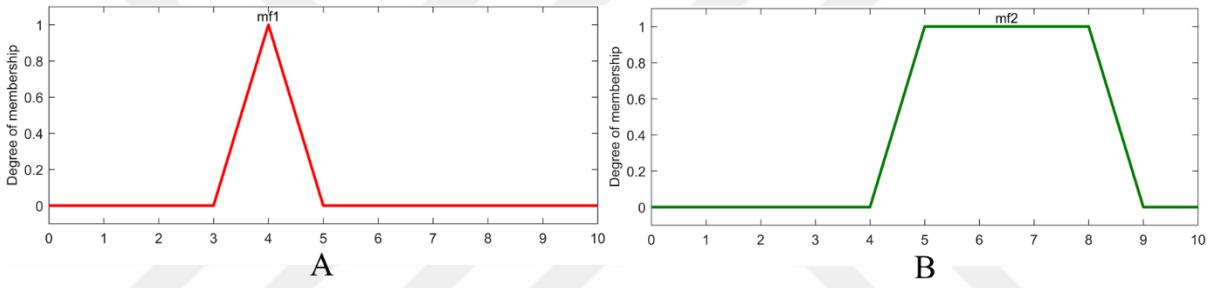


Figure 2.21: Fuzzy sets A and B

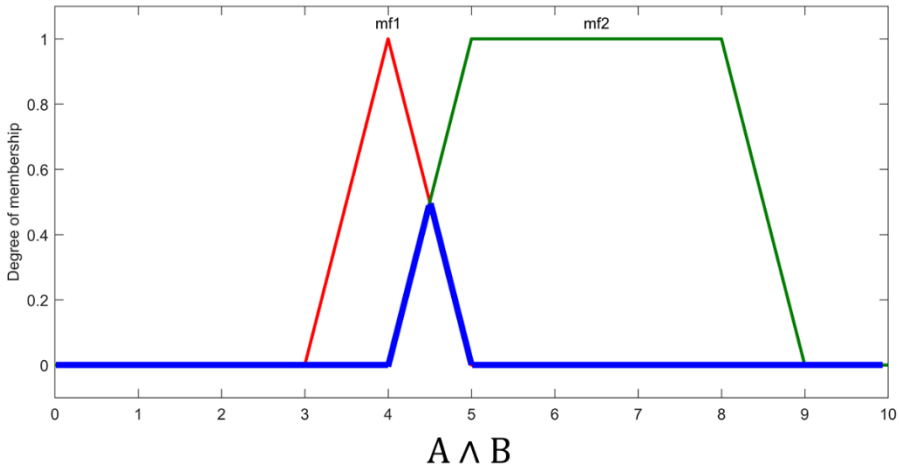


Figure 2.22: Intersection Operation for two fuzzy sets

Union Operation or Max operator for two fuzzy sets is launched by the logic operator (or), the result of the operation is, maximum for both fuzzy sets. The algebraic notation of union operator is shown below,

$$\mu A(x)\ or\ \mu B(x) = max\{\mu A(x), \mu B(x)\} \tag{2.17}$$

The output result of the union operation can be seen in Figure 2.23.

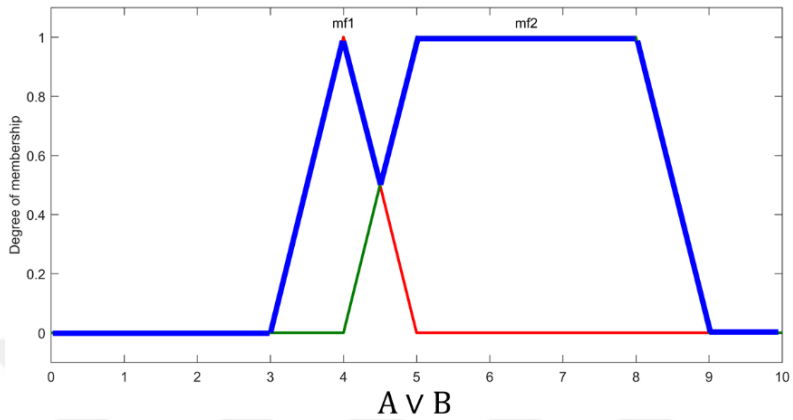


Figure 2.23: Union Operation for two fuzzy sets

Complement Operation in fuzzy theory is somehow different than in classical logic theory. For example, an element has a membership of 0.7 in fuzzy set A, meaning that this element doesn't belong to set A with a level of 0.3. In mathematics, it can be expressed as in Equation 2.18.

$$\mu\bar{B}(x) = not\big(\mu B(x)\big) = 1 - B(x) \tag{2.18}$$

Complement operation can be shown in Figure 2.24.

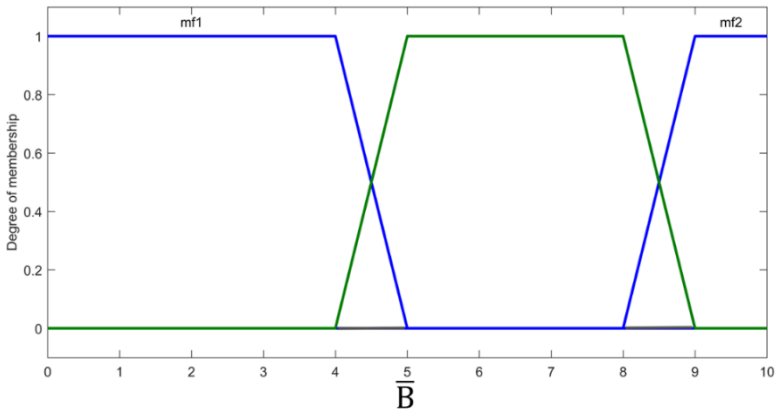


Figure 2.24: Complement Operation for one fuzzy set

2.5.4. Membership Function

Vagueness of fuzzy sets is characterized in membership functions. There are a lot of ways to specify the fuzzy membership function. Membership function mainly includes all fuzziness in

a specific fuzzy set; its characterization is fundamental to fuzzy operation and property. Therefore, the shape of the membership is determined by the process and application which is used for. Trapezoidal, Triangular, Gaussian, and Bell-shaped are some of the famous membership functions as shown in figure 2.25 (Jantzen, 1998). Membership function is a graphical shape that relates the value of an input to be described with a specific level of a membership function, or, if it was a membership of an output; then the weighted value of a membership will be transformed to a crisp value as an output for a specific problem. The shape of the membership function has an important role for taking decisions, especially when an input or an output has a lot of linguistic variables; also, determining the structure of membership shape changes the percentage of overlapping between two memberships which affect the sensitivity of the fuzzy process.

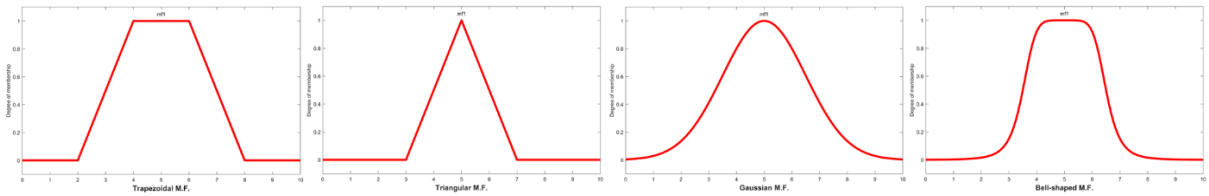


Figure 2.25: Different shapes of membership functions

2.5.5. Fuzzy Logic Concept

Fuzzy Logic control consists of three main stages: Fuzzification, Inference Engine, and Defuzzification. All of the three stages depend on a predefined fuzzy knowledge base. The structure of the Fuzzy Logic and its phases are shown in Figure 2.26.

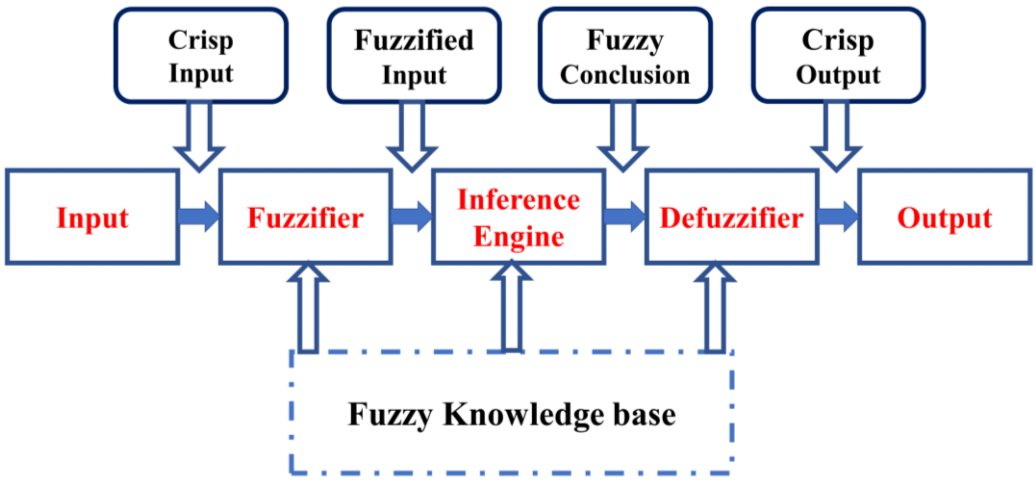


Figure 2.26: Fuzzy Logic Process

2.5.5.1. Fuzzification

Normally, the inputs of any physical system are numerical numbers, it can be in binary, decimal or hexadecimal, but the base of fuzzy logic is to judge on every input by a linguistic value. Thus, fuzzification is the process of converting crisp values to linguistic fuzzy values employing the predefined membership functions in fuzzy knowledge base. All physical systems are not actually fully deterministic. Every system has a considerable amount of uncertainty. The fuzziness, vagueness, ambiguity and imprecision of the system can be avoided by converting the crisp variable to a fuzzy value represented by a membership function.

For example, if we have a control system of a furnace, and the input of the control system is heat degree inside the furnace, we normally get a signal from a sensor with a specific crisp value such as 30 C°; fuzzification convert this value to be represented in a fuzzy set such as "Very Hot" with a membership of 0.66. The idea of fuzzification can be shown in Figure 2.27.

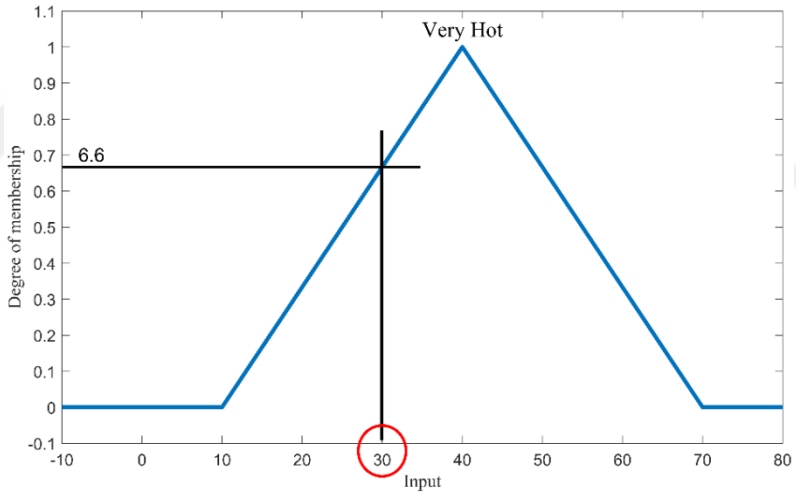


Figure 2.27: Fuzzification methodology

2.5.5.2. Knowledge Base

Knowledge base is the representation of expert knowledge based on fuzzy set theory and fuzzy logic control rules. Fuzzy control rules consist of linguistic variables which are the approximate reasoning of fuzzy sets of a system. So, if a variety of system conditions were satisfied by an expert, a result of these conditions can be used to infer system consequences. There are some important terms that should be clarified for fuzzy control rules, in system domain the conditions are named as antecedent; the action applied to the controlled system is the consequent. Thus, the antecedent and consequent are linked with fuzzy concepts to form the linguistic notation of fuzzy control rules.

Generally speaking, knowledge base is made up of two parts, the first part is Database; database consists of fuzzy sets linguistic labels “the membership functions with linguistic terms” which are used for fuzzy control rules. Then second part is Rule base; rule base is a group of fuzzy control rules “expert knowledge”. Fuzzy rules are in the form of (if-and/or-then) format, i.e. “if x is A and y is B then z is C”. There are some sources to obtain fuzzy control rules,

1) Heuristic method, this method depends on testing the response of a controlled system and based on it the fuzzy control rules can be formed.
2) Deterministic method, this method can produce fuzzy control rules by systematical method i.e., by using artificial intelligence techniques.
3) Fuzzy control rules can be obtained by an expert in the desired system, the experience of the expert can be transformed to fuzzy control rules.
4) System observations, fuzzy control rules can be obtained by input-output observations.
5) System self-learning, by this method, we can build the rules by offline self-tuning depending on pre-defined input-output information, some algorithms can be used to achieve this purpose such as Neural networks.

The above methods of determining fuzzy control rules aren’t mutually-exclusive, but they can be used together to obtain an effective and optimal fuzzy control rules. There are two methods of fuzzy control rules representations: State evaluation and Objective evaluation.

State evaluation: In the rules, the antecedent part can be thought as state variables, and the consequent part can be thought as control variable (Ross, 2010, P. 72). For example, let us consider a two-input system with a single output, the general rules will be as follows,

$$\begin{aligned} &Rule\ 1: \boldsymbol{IF}\ x\ is\ A_1\ \boldsymbol{AND}\ y\ is\ B_1\ \boldsymbol{THEN}\ z\ is\ C_1 \\ &Rule\ 2: \boldsymbol{IF}\ x\ is\ A_2\ \boldsymbol{AND}\ y\ is\ B_2\ \boldsymbol{THEN}\ z\ is\ C_2 \\ &\vdots \\ &Rule\ n: \boldsymbol{IF}\ x\ is\ A_n\ \boldsymbol{AND}\ y\ is\ B_n\ \boldsymbol{THEN}\ z\ is\ C_n \end{aligned} \tag{2.19}$$

Where x, y and z are state variables as linguistic variables, and A_n, B_n and C_n are the control variables as linguistic values.

Objective evaluation: It can be thought as a predictor, it can predict the present or future control response, and the mathematical structure of the rules will be as follows,

$$\begin{aligned} &Rule\ 1: \boldsymbol{IF}\big(z\ is\ C_1 \rightarrow (x\ is\ A_1 \boldsymbol{AND}\ y\ is\ B_1)\big)\boldsymbol{THEN}\ z\ is\ C_1 \\ &Rule\ 2: \boldsymbol{IF}\big(z\ is\ C_2 \rightarrow (x\ is\ A_2 \boldsymbol{AND}\ y\ is\ B_2)\big)\boldsymbol{THEN}\ z\ is\ C_2 \\ &\vdots \\ &Rule\ n: \boldsymbol{IF}\big(z\ is\ C_n \rightarrow (x\ is\ A_n \boldsymbol{AND}\ y\ is\ B_n)\big)\boldsymbol{THEN}\ z\ is\ C_n \end{aligned} \tag{2.20}$$

Based on objective assessment control action can be determined to satisfy objectives and desired states.

2.5.5.3. Fuzzy Inference Engine

Fuzzy inference engine is the process of connecting all fuzzy control parts together to complete the whole fuzzy control process. Fuzzy inference engine is the way of relating fuzzified values with fuzzy control rules and delivering the results of rules by defuzzification. For example, let us examine two inference methods (Generalized Modus Tollens (GMT) and Generalized Modus Pones (GMP)). Generalized modus tollens depends on inverse fuzzy theory as shown in equation (2.21).

$$\begin{aligned} &Fact: y\ is\ \bar{b} \\ &Rule: \boldsymbol{IF}\ x\ is\ a\ \boldsymbol{THEN}\ y\ is\ b \\ &Result: x\ is\ \bar{a} \end{aligned} \tag{2.21}$$

GMT considers the reverse operation; it checks control action and assumes that the inverse of the action value is a result of inverse input. On the other hand, GMP is a forward inference method such as,

$$\begin{aligned} &Fact: x\ is\ a \\ &Rule: \boldsymbol{IF}\ x\ is\ a\ \boldsymbol{THEN}\ y\ is\ b \\ &Result: y\ is\ b \end{aligned} \tag{2.22}$$

GMP is one way of inference methods. The famous and most used inference methods are Mamdani method, Takagi-Sugeno method, and Tsukamoto method. Mamdani method is used

widely in fuzzy control techniques and is the one that we counted on here for this research. Mamdani and Assilian (1975) used fuzzy control for controlling steam engine and boiler combination, it was one of the first applications to be controlled by fuzzy controller. Mamdani constructed the fuzzy rules by human operators' experience and the inference method was Mamdani method. The method depends on four steps: Fuzzification, Rule evaluation, Aggregation of the rule outputs, and Defuzzification.

Fuzzification is the first step in fuzzy control algorithm as aforementioned; it takes the crisp values and translates it into a linguistic value with a specific membership function. Thus, let us consider a cooling fan system as an example. In this example, fuzzy logic has two inputs, the first one is sensor temperature, while the second is the rate of temperature change inside the room. The output is fan speed. The first input of the system is a specific degree such as 32 degrees, Figure 2.28 explains the process of representing the crisp value to a fuzzy linguistic value.

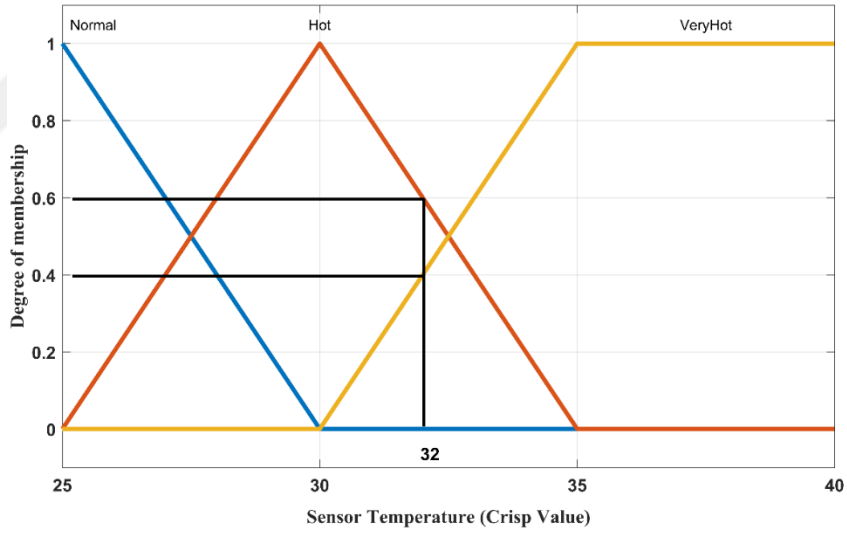


Figure 2.28: Fuzzification process

The input has three linguistic values representing the condition of the crisp input of the temperature sensor such as (Normal, Hot, Very Hot); these three sets can represent the input of 32 degrees as follows: a membership of (0.0) in the (Normal) fuzzy set; because there is no intersection between the value and the (Normal) fuzzy set , a membership of 0.6 in (Hot) fuzzy set and a membership of 0.4 in (Very Hot) fuzzy set.

$$\mu_{(x=Normal)} = 0, \quad \mu_{(x=Hot)} = 0.6, \quad \mu_{(x=Very\ Hot)} = 0.4 \tag{2.23}$$

The second input has three fuzzy sets (Slow, Medium, Fast), if we supposed that the value of rate of change is 0.5 then, we have only a membership of (1.0) in the (Medium) fuzzy set.

$$\mu_{(y=Slow)} = 0, \ \ \mu_{(x=Medium)} = 1.0, \ \ \mu_{(x=Fast)} = 0 \tag{2.24}$$

Rule evaluation is the second step of fuzzy logic control. In this step, we will gather the fuzzified values to be evaluated by fuzzy logic rules. For the above example, let us consider the following rules:

$$\begin{aligned}
&Rule\ 1: \boldsymbol{IF}\ (TS)\ is\ Normal\ \boldsymbol{THEN}\ (FS)\ is\ Slow\\
&Rule\ 2: \boldsymbol{IF}\ (TS)\ is\ Hot\ \boldsymbol{AND}\ (ROC)\ is\ Slow\ \boldsymbol{THEN}\ (FS)\ is\ Fast\\
&Rule\ 3: \boldsymbol{IF}\ (TS)\ is\ Hot\ \boldsymbol{AND}\ (ROC)\ is\ Medium\ \boldsymbol{THEN}\ (FS)\ is\ MediumSpeed\\
&Rule\ 4: \boldsymbol{IF}\ (TS)\ is\ Hot\ \boldsymbol{AND}\ (ROC)\ is\ Fast\ \boldsymbol{THEN}\ (FS)\ is\ Slow\\
&Rule\ 5: \boldsymbol{IF}\ (TS)\ is\ VeryHot\ \boldsymbol{THEN}\ (FS)\ is\ Fast
\end{aligned} \tag{2.25}$$

Where, Ts is the membership value of temperature sensor, ROC is the membership value of rate of change in temperature, Fs is the membership value of the fan speed.
Depending on the fuzzified values the rules can give the following results:

$$\begin{aligned}
&Rule\ 1: \boldsymbol{IF}\ (TS)\ is\ Normal\ (0)\boldsymbol{THEN}\ (FS)\ is\ Slow(0)\\
&Rule\ 2: \boldsymbol{IF}\ (TS)\ is\ Hot\ (0.6)\boldsymbol{AND}\ (ROC)\ is\ Slow(0)\ \boldsymbol{THEN}\ (FS)\ is\ Fast\ (0)\\
&Rule\ 3: \boldsymbol{IF}\ (TS)\ is\ Hot\ \boldsymbol{AND}\ (0.6)(ROC)\ is\ Medium\ (1.0)\ \ldots\\
&\qquad\qquad \boldsymbol{THEN}\ (FS)\ is\ MediumSpeed\ (0.6)\\
&Rule\ 4: \boldsymbol{IF}\ (TS)\ is\ Hot\ (0.6)\boldsymbol{AND}\ (ROC)\ is\ Fast\ (0)\boldsymbol{THEN}\ (FS)\ is\ Slow(0)\\
&Rule\ 5: \boldsymbol{IF}\ (TS)\ is\ VeryHot\ (0.4)\boldsymbol{THEN}\ (FS)\ is\ Fast\ (0.4)
\end{aligned} \tag{2.26}$$

The active rules are rule (3 and 5); thus, the output of the Fan speed is Medium speed with a membership of 0.6 and Fast speed with a membership of 0.4. The result contains two actions: Medium speed and Fast speed. In that case, we need to apply aggregation of the rules results to be combined and to form one fuzzy set as shown in Figure (2.29). After aggregation process, we should apply the defuzzification process to give a decision in a crisp value for the control action.

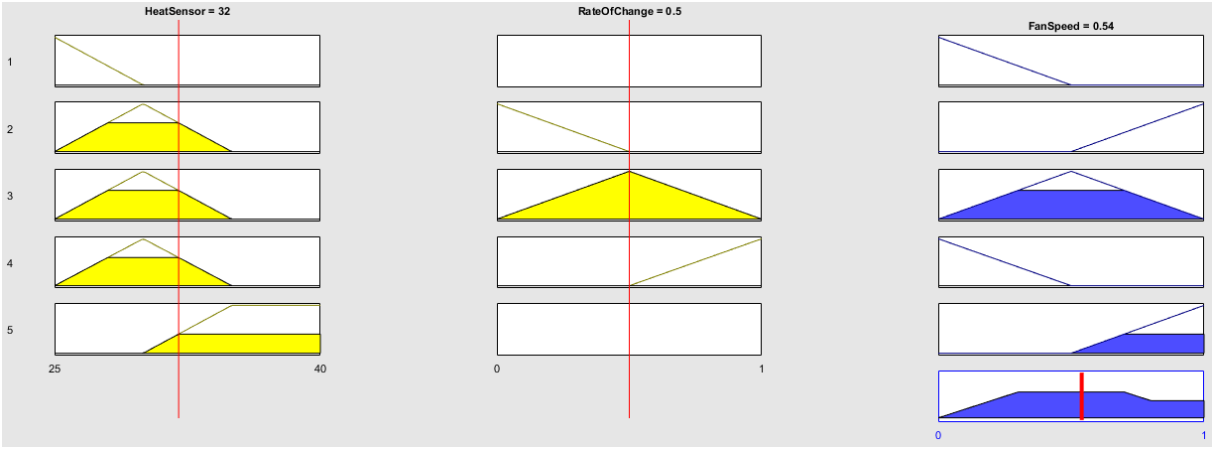


Figure 2.29: Rules implementation results

2.5.5.4. Defuzzification

As it is known, we can't deal with physical systems by giving word commands or linguistic values; thus, as we converted the crisp value to a linguistic value in the fuzzification process, we should make reverse process by converting the resultant linguistic values to a crisp value to be transformed to a physical quantity. There are several defuzzification methods such as Weighted Average (WAM) method, Mean of Maximum (MOM) method, and Center of Gravity (COG) method. Every method gives a different output value, it depends on system application and output requirements, and Center of Gravity is the chosen method in our research.

Center of Gravity (or sometimes named Centroid of area) is the most used defuzzification method. It finds out the value of the center of the gravity over the abscissa, Equation (2.27) explain the mathematical derivation of the method.

$$z = \frac{\int \mu_B(z) z . dz}{\int \mu_B(z) . dz} \tag{2.27}$$

For our example, the result of COG will be (0.54) which means 54% of Fan speed.

2.6. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANN) offer a perfect methodology in machine learning. ANN is ready to learn from example for real, discrete, vector valued functions. ANN proved its ability to solve problems such as robotics control scheme, speech recognition, and image recognition. ANN learning technique gave a robust response to errors depending on training data provided. One of the important features of ANN is that it can act like a predictor to the real-life problems

such as sensor readings. It can provide effective learning procedures. One of the ANN algorithms is the well-known “Backpropagation” algorithm. Backpropagation has been successfully applied in learning problems such as handwritten characters recognition by LeCun et al. (1989), spoken words recognition by Lang et al. (1990), and faces recognition by Cottrell (1990).

The idea of ANN algorithm was inspired from the observation of biological neural networks in humans’ brain; biological neural networks are complex networks of coordinated neurons. ANN are built of several interconnected set of neurons in its basic unit, each unit has a real valued input and a real valued output, gathering these units in one big network is the main idea of ANN algorithm, in that way, ANN will be able to learn from example as the real neural networks. Each neural unit contains inputs and outputs; but, to be able to learn and to be effective neural unit; it should contain numerical variable weights for connecting the inputs and the outputs of the neuron. These weights make the neural network to be adaptive network and have the ability to approximate nonlinear processes. These weights are variables which can be adjusted during training process of the neural networks to give suitable connections between neurons. Pomerleau’s (1993) tried to apply the ANN algorithm to an autonomous vehicle, in this project, ANN will be learned to steer the vehicle in a highway with a normal speed. The general structure of ANN for autonomous vehicle is shown in Figure (2.30).

2.6.1. ANN Learning Methods

There are a lot of methods for ANNs to be trained; some of these training methods are Self-organizing Map (SOM), Real coded Genetic Algorithm (RGA), and Back Propagation Algorithm (BPA) (Srinivasulu and Jain, 2006). Self-organizing map training method depends on unsupervised learning, which means the inclusion of a set of data patterns that haven’t been declared by human beings. This method maps the input into two-dimensional representation, thus, in that way it reduces the dimension of input space. This method relies on the competitive learning rather than error correction. One of the advantages of this method is the ability to arrange small or large number of nodes to make a cluster of data. The second method is Real coded Genetic Algorithm (RGA). This method integrates ANN with Genetic Algorithm (GA) to avoid some drawbacks of other training methods such as Backpropagation training method. To reach efficient and optimized learning techniques, we need to well-determine the number of

hidden layers, recognize the number of nodes inside each hidden layer, define a suitable value of learning rates, and specify good starting weights between nodes (Sohail et al. 2006).

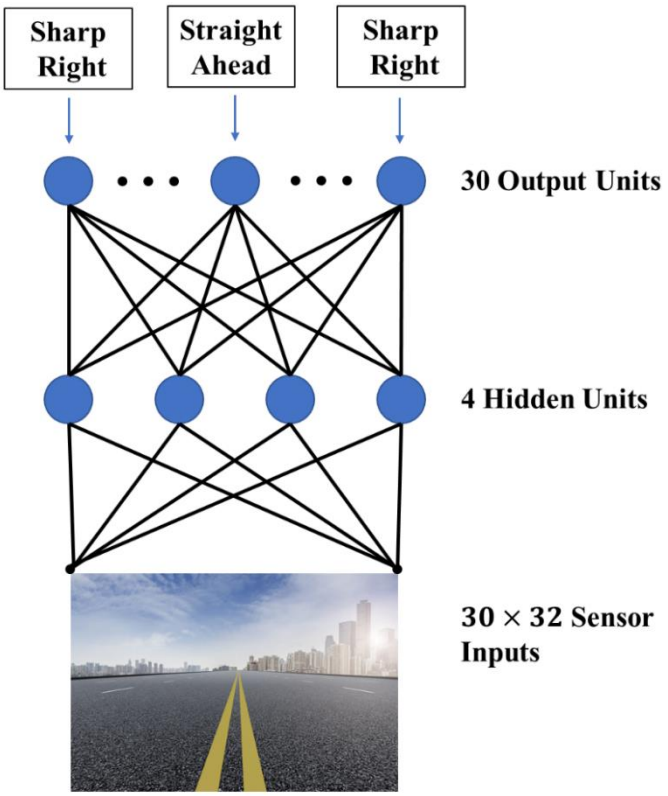


Figure 2.30: ANN for autonomous vehicle project

In some applications, these variables can affect the training process and can eventually miss the global optimum; for this reason, GA plays an important role in helping ANN to reach its optimum performance.

The third method is the Back-Propagation Algorithm (BPA), this method is the most used method and can give satisfactory results in most of machine learning applications. In the next section we will examine Back-Propagation Algorithm in details.

2.6.2. Back-Propagation Algorithm

Back-Propagation is the way of evaluating each node. The evaluation process is to get the output error of the nodes in a specific layer, then evaluate nodes in the previous layer, and so on. Thus, the evaluation process is moving backward from the output as the first layer to be evaluated to the first layer just after the input layer, that's why it is called Back-Propagation algorithm. The main idea of this learning algorithm is to learn from examples like humans, a child can learn to identify a cat from examples of cats, and the same thing with ANN, it can learn from examples.

So, Back-Propagation algorithm is based on examples, hence, we need a set of data as an example to be trained with. This learning method is to learn by supervising a set of data and learn from it, after the training process the network can be able to detect objects.

Therefore, Back-Propagation algorithm is effective when ANN has multi internal layers. For simple application, it can be enough to use one hidden layer to reach the desired output, but, if we are talking about image recognition for animal kinds, in that case, we should use multi-layers to give the correct output. For example, if we want to recognize animal name, we should build an ANN with the ability to recognize the details of the animals, such as, the number of limbs, size, length of the animal, and other features, so that each layer can match the feature of the specified animal and give correct output result.

Back-Propagation Algorithm consists of two steps. The first step is to make a forward and backward propagation in continuous circles until we reach the desired results. Forward propagation is from the inputs to the hidden layers and reaching the output to evaluate all neural nodes. Backward propagation is to calculate the error in each node from the output layer to the first layer in the network. Second step is to update the interconnection weights between neural nodes. The update is based on the multiplication of the differential value with the input of the node, then, weight value is subtracted with a ratio, and this ratio affects the learning speed and quality, so it should be chosen carefully.

2.6.3. ANN Mathematical Model

The general structure of a neural node contains inputs (x) multiplied by weights (w), the results of multiplied values will be summed. If the summation value is greater than a threshold (θ), it will pass to the next neural node as shown in Figure 2.31.

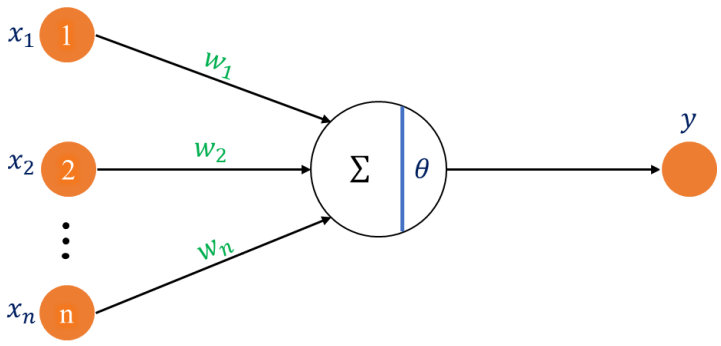


Figure 2.31: General structure of artificial neural cell

Several neural nodes with several layers construct the artificial neural network as shown in Figure 2.32.

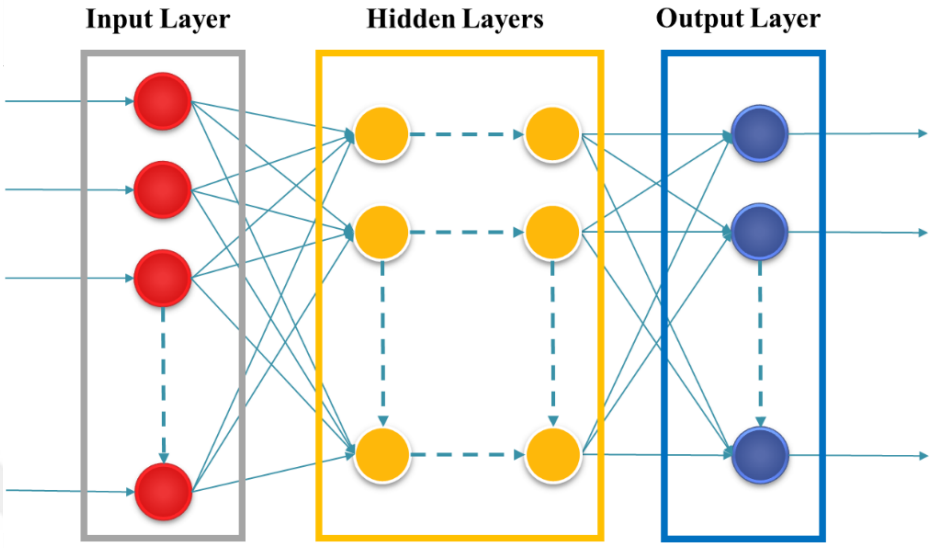


Figure 2.32: General structure of Artificial Neural Network

Each node will have a value; the basic element is the multiplication between inputs and weights; thus, all the multiplied values will be subtracted by a threshold (θ) then the result of all multiplication from all inputs will be summed together. This result can be evaluated by one of these functions: step, sign, linear, or sigmoid function, in our case we choose sigmoid to complete the equation.

$$O_k^i = \frac{1}{1 + e^{-\left(\sum_k w_{jk}^{i-1} * O_j^{i-1} - \theta_k^i\right)}} \tag{2.28}$$

Graphical explanation of the mathematical calculation is shown in Figure 2.33.

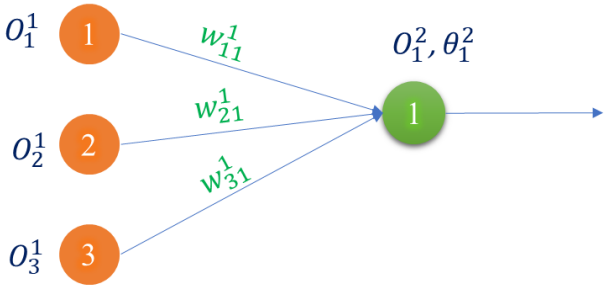


Figure 2.33: Neural node evaluation process

Where (O) is the neural node output value, (w) is the weight of the connection between two nodes, (θ) is the threshold value, (i) is the number of the layer, (j) is the number of the input

node, (k) is the number of the output node. This process is the forward propagation. After completing forward propagation, we start the backward propagation to evaluate the error in each node.

First step in evaluation process starts with resolving the final node (output node) error as follows:

$$E^i_{k(p)} = O^i_{k(p)}(actual) - O^i_k(output) \tag{2.29}$$

Where (p) is the number of iterations.

Second step in evaluation process is to determine the error for each node rather than the output nodes as presented in Equation 2.30. A graphical explanation is shown in Figure 2.34.

$$E^i_j = O^i_j(1 - O^i_j) * \left(\sum_k w^i_{jk} * E^{i+1}_k\right) \tag{2.30}$$

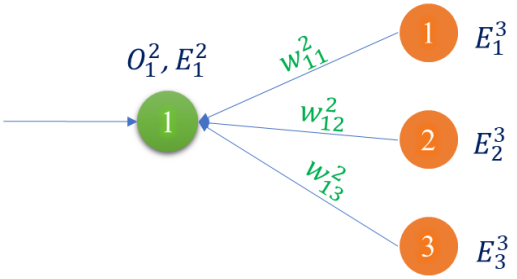


Figure 2.34: Back-Propagation error evaluation

After calculating the error and the output for each node, we calculate the weights and thresholds between every two nodes by the following equations.

$$w^i_{jk}(p) = w^i_{jk}(p-1) + \mu O^i_j E^{i+1}_k \tag{2.31}$$

$$\theta^i_j(p) = \theta^i_j(p-1) + \mu(-1)E^{i+1}_k \tag{2.32}$$

Where, μ is the change ratio of the weights and thresholds. The final general structure of the neural network will be as shown in Figure 2.35.

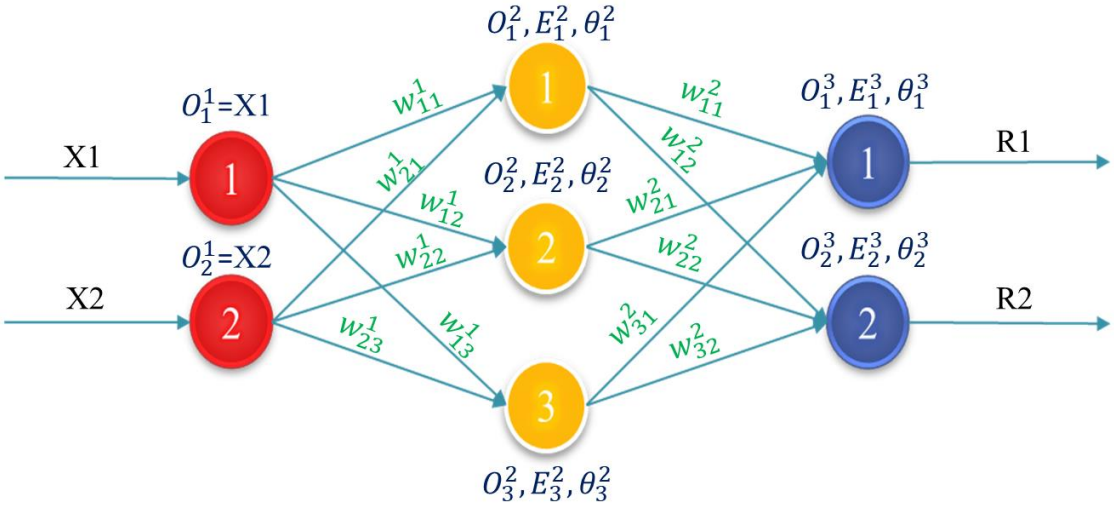


Figure 2.35: General structure of the Neural Network

After we calculate all these variables, we complete one iteration: therefore, to reach the desired results, we should do a lot of iterations.

3. MATERIALS AND METHODS

3.1. QUAD-ROTOR MODELING

The mechanical and electrical structures should be understood to perform a quad-rotor modeling. A quad-rotor system modeling is performed to explain the mathematical equations of the physical quantities, forces, inertia, and other variables that affect the system. When designing an aircraft, there are many aspects to consider, especially when it comes to passenger transport aircrafts. A lot of forces affect the aircraft. Some examples of these forces include the aerodynamic effects, the metal used in the construction of the aircraft, its ability to function under different weather conditions, actuators design, the potential to face years of continuous operation, and other safety constraints. All these issues must be discussed during aircraft modeling. When designing quad-rotor, we don't need that kind of complexity since we're dealing with an unmanned aircraft; however, the safety restrictions will of course be a little lower than the manned aircraft. In the upcoming quad-rotor modeling process, we will neglect some complicated and extensive mathematical equations; on the other hand, we will use intelligent control techniques to build the controller to achieve satisfactory and robust drone control.

We must understand some concepts and definitions to complete the quad-rotor modeling such as System kinematics, Forward and Inverse kinematics, kinetic and potential energies, Newton-Euler equations, and system dynamics. Kinematics is the study of the motion of a point, object, or multi-objects without the explanation of the causes of motion. In 3D Coordinate system, when considering an object kinematics, we must study its position as (x, y, z) and its rotational angles (θ, φ, ψ) rotating about (x, y, z) axes referenced to a specific initial frame i.e. Earth-Centered Coordinate system. The study of kinematics starts by defining the system geometry i.e. object location; this object in our study is the quad-rotor and its velocity wherein acceleration should be studied. The study can be undertaken for the center point of quad-rotor; the result of kinematics study is the transformation matrices. By the transformation matrices, we can determine position, velocity, and acceleration of any point on the system. Additionally, system dynamics should be stated since any system movement is induced by some kind of forces and torques. Time is important in dynamics study because each action in this moment will affect the coming action. Therefore, in this analysis, two aspects are important: Linear

dynamics and rotational dynamics. Linear dynamics can be applied to the objects that are moving in one direction such as a quad-rotor when it's moving in x-axis only. Rotational dynamics can be applied to objects traveling around a curved track. After we studied the concept of kinematics, let's now examine the kinematics of an object in more depth.

There are two kinds of kinematics; forward and inverse. Forward kinematics is the aim of determining the location of a point in an object by using object's kinematic equation. For instance, let's consider that we're dealing with a robotic arm and it is a three degree of freedom (DOF), assume that the three degrees of freedom are three joints which are translational and rotational joints. If we imagine a robotic arm with three joints and an end-effector, the first joint is rotational joint that is rotated by 90 degrees, the second joint is a translation joint that can move straightly by 30 centimeters, and the third joint is rotational that is rotated by 45 degrees. How do we get the end-effector location? Via Forward kinematics equations, we can obtain the new end-effector position from the values of the joints. Our system is a quad-rotor with six DOF, three position information (x, y, z) and three rotational angles(θ, φ, ψ). If we have the values of the translations and the angles of the rotations, we can determine the new location and orientation of the quad-rotor. Inverse Kinematics is the opposite concept of Forward Kinematics; we have the final position and rotations, but to achieve the defined location and orientation, we need to know what the suitable values of the joints are.

The importance of the previous definitions is to be prepared to represent the necessary mathematical formulation of quad-rotor modeling. Thus, the representation of the modeling will start with transformation matrices to construct a relationship between inertial frame and body frame.

3.1.1. Definitions of Quad-rotor's Variables

There are important base terms and symbols that need to be defined as they will be used in equations and matrices; the terms are:

$$\boldsymbol{\xi} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \boldsymbol{\eta} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}, \boldsymbol{V_b} = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \boldsymbol{V_i} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} \text{ or } \dot{\boldsymbol{\xi}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \ddot{\boldsymbol{\xi}} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \text{ or}$$

$$\dot{\boldsymbol{V}}_{\boldsymbol{i}} = \begin{bmatrix} \dot{u}_\iota \\ \dot{v}_\iota \\ \dot{w}_\iota \end{bmatrix}, \dot{\boldsymbol{\omega}} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}, \ddot{\boldsymbol{\eta}} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}.$$

Table 3.1: UAV modeling terms and symbols

Description	Symbol	Unit
Absolute linear position of the quad-rotor defined in the inertial frame.	(ξ)	meters (m)
Angular position referenced to the inertial frame.	(η)	radians(rad)
Linear velocities of the body frame.	(V_b)	meter/second (m/s)
Linear velocities of the body expressed in the inertial frame coordinate.	$(\dot{\xi}, V_i)$	meter/second (m/s)
Angular velocities of the body frame.	(ω)	radians per second (rad/s)
Angular velocities of the body frame expressed in the inertial frame.	$(\dot{\eta})$	radians per second (rad/s)
Linear acceleration of the body frame expressed in the inertial frame.	$(\ddot{\xi}, \dot{V}_i)$	m/s^2
Angular acceleration of the body frame.	$(\dot{\omega})$	rad/s^2
Angular acceleration of the body frame expressed in the inertial frame.	$(\ddot{\eta})$	rad/s^2

3.1.2. Transformation Between Frames

To explain the kinematics of the quad-rotor, it's necessary to understand the transition between the inertial frame and the body frame. The transition between frames may be represented by a set of rotations, so we can use the rotation matrices to transform any point from the inertial frame to the body frame. The rotation matrix can be easily explained by the vector projections of each axis as follows,

$$\boldsymbol{R} = \begin{bmatrix} x_1.x_0 & y_1.x_0 & z_1.x_0 \\ x_1.y_0 & y_1.y_0 & z_1.y_0 \\ x_1.z_0 & y_1.z_0 & z_1.z_0 \end{bmatrix} \tag{3.1}$$

Let's consider a rotation about Z-axis by angle ψ as shown in Figure 3.1, we get this result,

Table 3.2: Vectors projections

$x_1.x_0 = \cos(\psi)$	$y_1.x_0 = -\sin(\psi)$	$z_1.x_0 = 0$
$x_1.y_0 = \sin(\psi)$	$y_1.y_0 = \cos(\psi)$	$z_1.y_0 = 0$
$x_1.z_0 = 0$	$y_1.z_0 = 0$	$z_1.z_0 = 1$

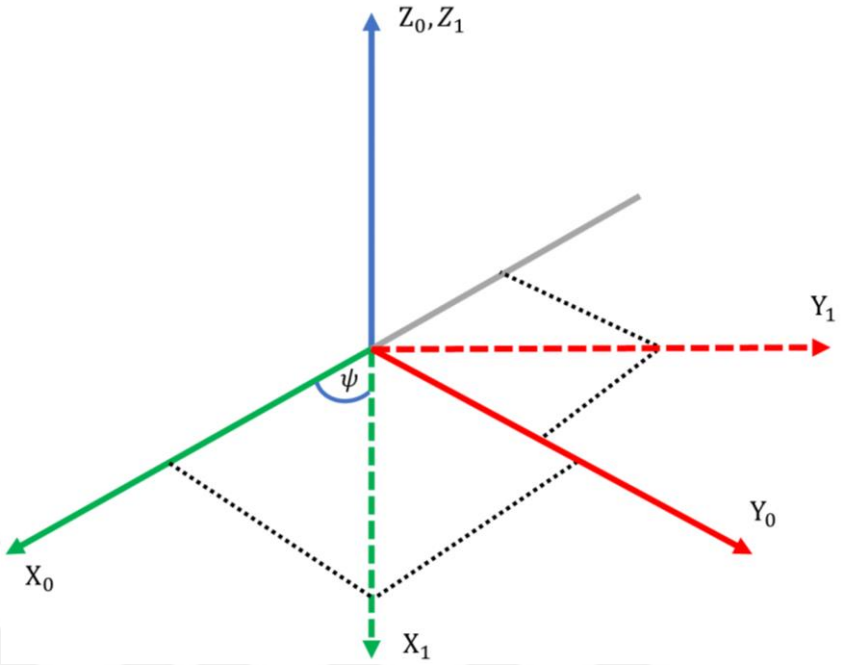


Figure 3.1: Vector projections

We can define rotation around Z-axis below,

$$\boldsymbol{R}_{Z,\psi} = \begin{bmatrix} cos\,(\psi) & -sin\,(\psi) & 0 \\ sin\,(\psi) & cos\,(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The same principle for other axes, the rotation matrix about the Y-Axis by an angle of θ will be as follows,

$$\boldsymbol{R}_{Y,\theta} = \begin{bmatrix} cos\,(\theta) & 0 & -sin\,(\theta) \\ 0 & 1 & 0 \\ sin\,(\theta) & 0 & cos\,(\theta) \end{bmatrix} \tag{3.3}$$

The rotation matrix about the X-Axis by an angle of ϕ will be,

$$\boldsymbol{R}_{X,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,(\phi) & sin\,(\phi) \\ 0 & -sin\,(\phi) & cos\,(\phi) \end{bmatrix} \tag{3.4}$$

Any transformation from a base frame to a body frame or vice versa can be represented by three successive rotations. There are three methods to make the transformation between the frames:

1- ZYZ Euler Angle representation.
2- Roll-Pitch-Yaw representation (ZYX Euler angles).

3- Axis/Angle representation

The Roll-Pitch-Yaw representation (ZYX Euler angles) is the method used for representing the rotation between frames in this study. Roll-Pitch-Yaw or ZYX rotation angles means a rotation around Z-axis, then a rotation around the *current* Y-axis, and then a rotation about the *current* X-Axis. *Current* axis means the new specified axis after the previous rotation in the body frame and not the inertial frame; thus, matrix multiplication will be after the first rotation matrix. The same thing is done for other rotations. Rotation about *fixed* axis is different than the rotation about *current* axis; therefore, a rotation about *fixed* axis means a rotation about the base frame or inertial frame. The multiplication of the rotation of the *fixed* axis will be before the rotation of the first rotation matrix - and the same rule is applied for other rotations. Figure 3.2 shows the steps of the Roll-Pitch-Yaw rotation angles.

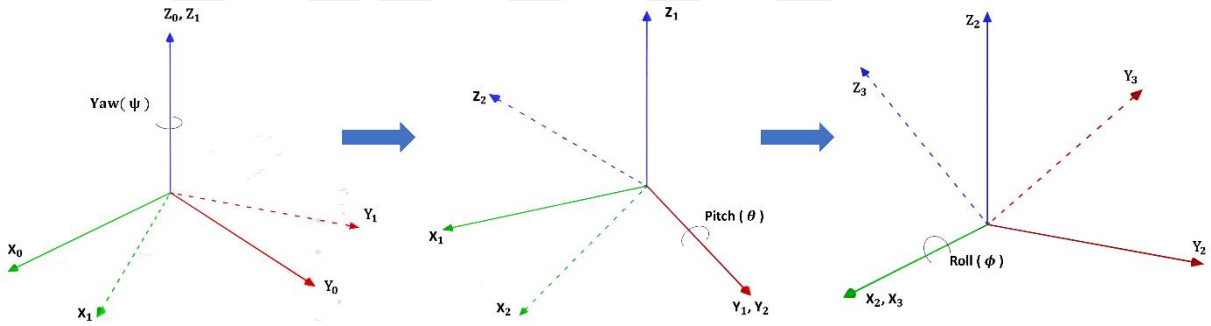


Figure 3.2: Roll-Pitch-Yaw rotations

The mathematical representation of rotational transformation matrix for Roll-Pitch-Yaw representation (ZYX Euler angles) will be as follows:

$$\boldsymbol{R}_{Z,Y,X} = \boldsymbol{R}_{Z,\psi}\boldsymbol{R}_{Y,\theta}\boldsymbol{R}_{Z,\phi} \tag{3.5}$$

$$\boldsymbol{R}_{Z,Y,X} = \begin{bmatrix} cos\,(\psi) & -sin\,(\psi) & 0 \\ sin\,(\psi) & cos\,(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\,(\theta) & 0 & -sin\,(\theta) \\ 0 & 1 & 0 \\ sin\,(\theta) & 0 & cos\,(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,(\phi) & sin\,(\phi) \\ 0 & -sin\,(\phi) & cos\,(\phi) \end{bmatrix} \tag{3.6}$$

$$\boldsymbol{R}_{Z,Y,X} = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\theta C_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{3.7}$$

Where *C* represents *cosine* and *S* represents *sine*.

3.1.3. Translational and Rotational Kinematics

To obtain linear velocity vector in the inertial frame, we multiply linear velocity vector of body frame by rotation matrix. Rotation matrix is based on (ZYX Euler angles) transformations. The quad-rotor velocities in body frame coordinates can be taken from sensors attached to the quad-rotor. The gyroscope sensor is the sensor which is responsible for providing the angles values. The linear velocity vector in the inertial frame can be represented as $\dot{\xi} = [\dot{x} \quad \dot{y} \quad \dot{z}]^T$ and can be found by multiplying body frame vector $V_b = [u_b \quad v_b \quad w_b]^T$ and rotation matrix as shown in Equation 3.8.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \tag{3.8}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\theta C_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \tag{3.9}$$

In case of linear velocity, transforming from the body frame to the inertial frame is not very difficult since it deals with linear quantities; thus, the velocity in a particular coordinate in the body frame can be easily converted into inertial frame (reference frame) by rotational matrix, just like position transformations. Linear transformation angles of position and velocity are time-independent, but each time step affects all angles for angular velocity. Therefore, the angular velocity of each coordinate depends on the velocity of another angle. Therefore, rotation matrix can't be sufficient to undergo angular velocity transformations between frames.

If we consider the general case of the angular velocity about an arbitrary possibly moving axis, the rotation matrix $\boldsymbol{R}$ is time-varying; the time derivative $\dot{\boldsymbol{R}}(t)$ can be derived from the rotation matrix as,

$$\dot{\boldsymbol{R}} = \boldsymbol{S}\big(w(t)\big)\boldsymbol{R}(t) \tag{3.10}$$

Where, the matrix $\boldsymbol{S}(\boldsymbol{\omega}(t))$ is skew-symmetric for a unique vector $\boldsymbol{\omega}(t)$. This vector $\boldsymbol{\omega}(t)$ is the angular velocity of the rotating frame with respect to the fixed frame at time t. $\boldsymbol{S}(\boldsymbol{\omega}(t))$ can also be represented as $\dot{\theta}\boldsymbol{S}(i)$ where θ is the angle of rotation. Equation 3.10 can be reformed as,

$$\boldsymbol{S}\big(w(t)\big)\boldsymbol{R}(t) = \dot{\theta}\boldsymbol{S}(i) \tag{3.11}$$

Where $\dot{\theta}\boldsymbol{S}(i)$ is the derivative of a rotating angle θ over time. $\omega = i\dot{\theta}$ is the angular velocity and i = $(1,0,0)^T$. Hence, equation 3.11 can be formed as follows:

$$\boldsymbol{S}\big(w(t)\big)\boldsymbol{R}(t) = \frac{d\theta}{dt}.\frac{d\boldsymbol{R}}{d\theta} = \frac{d\boldsymbol{R}}{dt} \tag{3.12}$$

Hence, Equation 3.12 can be presented as $\boldsymbol{S}\big(w(t)\big)\boldsymbol{R}(t) = \dot{\boldsymbol{R}}$ as a proof to Equation 3.10. From the above base equations, we can implement a suitable relevant rotating angles formulation as follows:

$$\dot{\boldsymbol{R}}_n^0 = \boldsymbol{S}(\omega_{0,n}^0)\boldsymbol{R}_n^0 \tag{3.13}$$

The notation $\omega_{0,n}^0$ relates the base frame with the current and next frames. Figure 3.3 explains the notation of angular velocity and the related frames. $\boldsymbol{R}_n^0$ is the rotation matrix between many frames.

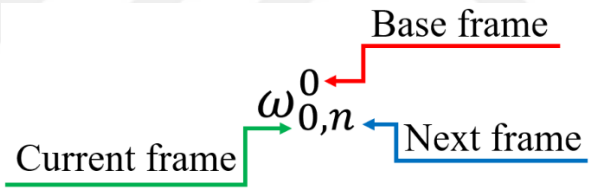


Figure 3.3: Explanation of angular velocity notation

Thus, the relationship between frames for rotation and angular velocities are as follows:

$$\boldsymbol{R}_n^0 = \boldsymbol{R}_1^0\boldsymbol{R}_2^1 \cdots \boldsymbol{R}_n^{n-1} \tag{3.14}$$

$$\boldsymbol{\omega}_{0,n}^0 = \boldsymbol{\omega}_{0,1}^0 + \boldsymbol{R}_1^0\omega_{1,2}^1 + \boldsymbol{R}_2^0\boldsymbol{\omega}_{2,n}^2 + \cdots + \boldsymbol{R}_{n-1}^0\boldsymbol{\omega}_{n-1,n}^{n-1} \tag{3.15}$$

$$\boldsymbol{\omega}_{0,n}^0 = \boldsymbol{\omega}_{0,1}^0 + \boldsymbol{\omega}_{1,2}^0 \quad + \boldsymbol{\omega}_{2,3}^0 \quad + \cdots + \boldsymbol{\omega}_{n-1,n}^0 \tag{3.16}$$

In order to get the equations and transformation matrices in three dimensions, the changing in three angular angles at the same moment should be considered because every angle changing in one axis will influence the other angles. Let us consider a sequence of rotations as ZYX, the rotation about the Z-axis ($\dot{\psi}$) will be affected by the rotation about Y and X axes, the rotation about Y-axis ($\dot{\theta}$) will be affected by the rotation about X-axis, while the rotation about X-axes

($\dot{\phi}$) will not be affected by any rotations. For the quad-rotor, the angular velocity vector will be related to three angular velocities as follows:

$$\boldsymbol{\omega}_{0,3}^{0} = \boldsymbol{\omega}_{0,1}^{0} + \boldsymbol{\omega}_{1,2}^{0} + \boldsymbol{\omega}_{2,3}^{0} \tag{3.17}$$

$$\boldsymbol{\omega}_{0,3}^{0} = \boldsymbol{\omega}_{0,1}^{0} + \boldsymbol{R}_{1}^{0}\boldsymbol{\omega}_{1,2}^{1} + \boldsymbol{R}_{2}^{0}\boldsymbol{\omega}_{2,3}^{2} \tag{3.18}$$

$\boldsymbol{\omega}_{0,3}^{0}$ (the angular velocity vector of the body frame) $= \begin{bmatrix} p \\ q \\ r \end{bmatrix}$. The first term in Equation 3.18 can be explained as follows:

$$\omega_{0,1}^{0} = i\dot{\phi} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\dot{\phi} \tag{3.19}$$

The second term in Equation 3.18 as follows:

$$R_{1}^{0}\omega_{1,2}^{1} = R_{x}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\dot{\theta} \tag{3.20}$$

Where R_x is the rotation matrix about X-axis. The third term in Equation 3.18 is shown below:

$$R_{2}^{0}\omega_{2,3}^{2} = R_{1}^{0}R_{2}^{1}\omega_{2,3}^{2} = R_{x}R_{y}\omega_{2,3}^{2} \tag{3.21}$$

$$R_{2}^{0}\omega_{2,3}^{2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,(\phi) & sin\,(\phi) \\ 0 & -sin\,(\phi) & cos\,(\phi) \end{bmatrix}\begin{bmatrix} cos\,(\theta) & 0 & -sin\,(\theta) \\ 0 & 1 & 0 \\ sin\,(\theta) & 0 & cos\,(\theta) \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dot{\psi} \tag{3.22}$$

$$\begin{aligned} R_{2}^{0}\omega_{2,3}^{2} &= \begin{bmatrix} cos\,(\theta) & 0 & -sin\,(\theta) \\ sin\,(\phi)sin\,(\theta) & cos\,(\phi) & cos\,(\theta)sin\,(\phi) \\ cos\,(\phi)sin(\theta) & -sin\,(\phi) & cos\,(\phi)cos\,(\theta) \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dot{\psi} \\ &= \begin{bmatrix} -sin\,(\theta) \\ cos\,(\theta)sin\,(\phi) \\ cos\,(\phi)cos\,(\theta) \end{bmatrix}\dot{\psi} \end{aligned} \tag{3.23}$$

The resultant angular velocity vector expressed in body frame can be taken from equations 3.19, 3.20, and 3.23, and can be reformed as follows:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\phi} + \begin{bmatrix} 0 \\ cos\,(\phi) \\ -sin\,(\phi) \end{bmatrix} \dot{\theta} + \begin{bmatrix} -sin\,(\theta) \\ cos\,(\theta) sin\,(\phi) \\ cos\,(\phi) cos\,(\theta) \end{bmatrix} \dot{\psi} \tag{3.24}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin\,(\theta) \\ 0 & cos\,(\phi) & cos\,(\theta) sin\,(\phi) \\ 0 & -sin\,(\phi) & cos\,(\phi) cos\,(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.25}$$

$$\boldsymbol{\omega} = \boldsymbol{W}_{\eta}\, \dot{\boldsymbol{\eta}} \tag{3.26}$$

In order to obtain the angular velocities represented in the inertial frame from the body frame, we should use the inverse of the angular velocity matrix. Thus,

$$\dot{\boldsymbol{\eta}} = \boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\, \boldsymbol{\omega} \tag{3.27}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\,(\phi) tan\,(\theta) & cos\,(\phi) tan(\theta) \\ 0 & cos\,(\phi) & -sin\,(\phi) \\ 0 & \dfrac{sin\,(\phi)}{cos\,(\theta)} & \dfrac{cos\,(\phi)}{cos\,(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.28}$$

3.1.4. Translational and Rotational Dynamics

The translational dynamics is the study of motion of an object; in our case, it's the study of quad-rotor motion. Translational dynamics focus on the terminology that considers time varying interaction between forces and movements. There are a lot of forces that affect the linear and rotational motion of the drone, such as Thrust forces, Gravitational force, Centrifugal force, Centripetal force, Aerodynamic disturbance forces, Coriolis force and the Aerodynamic drag force. Some of these forces have no noticeable impact when compared with other forces; thus, we will consider the Thrust force, Gravitational force, and the Aerodynamic drag force in our modeling procedures.

First, we will begin to examine the dynamics of translation by considering Newton's second law of motion as the basis for our explanation. The basic equation of Newton's law is shown below,

$$\vec{F} = m \frac{d\vec{V}}{dt} \tag{3.29}$$

In the above equation, the force equals mass (m) multiplied by the derivative of linear velocity vector. To clarify the idea of the derivative of a rotating frame, we will go a little deeper in the derivative to proof the idea. Chain rule for a rotating frame can give the following general explanation (Tytler, 2019):

$$\frac{d\boldsymbol{V}}{dt} = \dot{\boldsymbol{V}} + \boldsymbol{\omega} \times \boldsymbol{V} \tag{3.30}$$

$$\dot{\boldsymbol{V}} = \boldsymbol{a} - 2\left[\boldsymbol{\omega} \times \left(\dot{\boldsymbol{V}} + (\boldsymbol{\omega} \times \boldsymbol{V})\right)\right] + [\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{V})] \tag{3.31}$$

$$\dot{\boldsymbol{V}} = \boldsymbol{a} - 2\left[\boldsymbol{\omega} \times \dot{\boldsymbol{V}}\right] - [\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{V})] \tag{3.32}$$

The first term is the Inertial acceleration, the second term is the Coriolis acceleration, and the third term is the Centrifugal acceleration. The result of the derivative which is based on quad-rotor pre-defined notations will be as follows:

$$\boldsymbol{F} = \frac{d}{dt}(m\boldsymbol{V}_B) + \boldsymbol{\omega} \times (m\boldsymbol{V}_B) \tag{3.33}$$

By using the Chain Rule for a rotating frame formula (Broccoli, 2012), the moving object's velocity (quad-rotor's velocity) derivation will be as shown in Equation 3.33. Equation 3.33 can be reformed as follows:

$$\boldsymbol{F}_b = m\dot{\boldsymbol{V}}_b + m(\boldsymbol{\omega} \times \boldsymbol{V}_b) \tag{3.34}$$

$$\boldsymbol{F}_b = m\dot{\boldsymbol{V}}_b + m(\boldsymbol{\Omega}\,.\,\boldsymbol{V}_b) \tag{3.35}$$

Where $\boldsymbol{\omega}$ is the angular velocity vector of the body frame, and $\boldsymbol{\Omega}$ is the angular velocity matrix.

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{3.36}$$

$$\boldsymbol{F}_b = m\begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{w}_b \end{bmatrix} + m\begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} . \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \tag{3.37}$$

It's worth noting that the forces affecting the quad-rotor are Thrust force, Gravitational force, Drag force, and Disturbance force.

$$\boldsymbol{F_b} = \boldsymbol{F_t^b} + \boldsymbol{F_g^b} + \boldsymbol{F_d^b} + \boldsymbol{F_{dist}^b} \tag{3.38}$$

$$\boldsymbol{F_t^b} = \begin{bmatrix} \boldsymbol{F_x^b} \\ \boldsymbol{F_y^b} \\ \boldsymbol{F_z^b} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{F_z^b} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ k\sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \tag{3.39}$$

Where $\boldsymbol{F_t^b}$,is the Thrust Force, k is lift constant, ω_i is speed of the rotor, (i) is rotor number, and (b) indicates that the force is represented in body frame. The Thrust Force acts on the z-Axis of the body frame, the body frame of quad-rotor is supposed to be a rigid body and the direction of the thrust force vectors from the rotors are exactly parallel to the Z-Axis of the body frame. Therefore, the gravitational force is aiming towards z-axis of the inertial frame as follows:

$$F_g^i = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{3.40}$$

It's worth noting that F_g^i expresses the force in inertial frame. To represent the gravitational force in the body frame, we need to multiply the force by the inverse of the pre-defined rotation matrix as shown below.

$$R^{-1} = R^T = \begin{bmatrix} C_\theta C_\psi & C_\theta C_\psi & -S_\theta \\ C_\psi S_\theta S_\phi - C_\phi S_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\theta S_\phi \\ S_\phi S_\psi + C_\phi C_\psi S_\theta & C_\phi S_\theta S_\psi - C_\psi S_\phi & C_\theta C_\phi \end{bmatrix} \tag{3.41}$$

Hence, the rotation matrix is an orthogonal matrix, and the inverse of rotation matrix equals the transpose of rotation matrix.

$$\boldsymbol{F_g^b} = \boldsymbol{R^{-1}}.\boldsymbol{F_g^i} = \boldsymbol{R^{-1}} \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{3.42}$$

Drag force can be represented in the inertial frame below,

$$\boldsymbol{F_d^i} = \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \frac{1}{2}\rho_{air} I \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \begin{bmatrix} V_x^2 & 0 & 0 \\ 0 & V_y^2 & 0 \\ 0 & 0 & V_z^2 \end{bmatrix} \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{3.43}$$

Where, ρ_{air} density of fluid (Air) kg/m^3, A is the cross-section of wind-exposed field, V is speed of quad-rotor relative to wind, and C is drag coefficient in the inertial frame (Moyan Cano 2013, P.50). To represent Drag force in the body frame, the force should be multiplied by an inverse rotational matrix. A lot of considerations should be taken to model exposed wind, and in order to simplify quad-rotor mathematical modeling, we merged two forces, drag force and disturbances, to be under a general term named Disturbance forces. Thus, Disturbance force vector will contain the forces in inertial frame multiplied by inverse rotation matrix to maintain the force vector in body frame.

$$\boldsymbol{F}_{dist}^{b} = \boldsymbol{R}^{-1}.\begin{bmatrix} F_{dist,x} \\ F_{dist,y} \\ F_{dist,z} \end{bmatrix} \tag{3.44}$$

The result of a detailed mathematical explanation from Equation 3.29 to Equation 3.44 is shown below. First, we will represent the equation in the body frame.

$$m\dot{\boldsymbol{V}}_b + m(\boldsymbol{\omega} \times \boldsymbol{V}_b) = \boldsymbol{F}_t^b + \boldsymbol{F}_g^b + \boldsymbol{F}_{dist}^b \tag{3.45}$$

Thus, the linear acceleration in body frame is

$$\dot{\boldsymbol{V}}_b = \frac{1}{m}\left(\boldsymbol{F}_t^b + \boldsymbol{R}^{-1}.\boldsymbol{F}_g^b + \boldsymbol{R}^{-1}.\boldsymbol{F}_{dist}^b\right) - \boldsymbol{\omega} \times \boldsymbol{V}_b \tag{3.46}$$

$$\begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{w}_b \end{bmatrix} = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ k\sum_{i=1}^{4}\omega_i^2 \end{bmatrix} + g\begin{bmatrix} -S_\theta \\ C_\theta S_\phi \\ C_\theta C_\phi \end{bmatrix} + \frac{1}{m}\begin{bmatrix} C_\theta C_\psi & C_\theta C_\psi & -S_\theta \\ C_\psi S_\theta S_\phi - C_\phi S_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\theta S_\phi \\ S_\phi S_\psi + C_\phi C_\psi S_\theta & C_\phi S_\theta S_\psi - C_\psi S_\phi & C_\theta C_\phi \end{bmatrix}\begin{bmatrix} F_{dist,x} \\ F_{dist,y} \\ F_{dist,z} \end{bmatrix} - \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}\begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \tag{3.47}$$

At last Equation 3.47 represents the linear acceleration of quad-rotor in body frame. In other way, linear acceleration can be rewritten to be represented in inertial frame as follows:

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{1}{m}\left(k\sum_{i=1}^{4}\omega_i^2\right)\begin{bmatrix}S_\phi S_\psi + C_\phi C_\psi S_\theta\\ C_\phi S_\theta S_\psi - C_\psi S_\phi\\ C_\theta C_\phi\end{bmatrix} + \begin{bmatrix}0\\0\\-g\end{bmatrix} + \frac{1}{m}\begin{bmatrix}F_{dist,x}\\F_{dist,y}\\F_{dist,z}\end{bmatrix} - \begin{bmatrix}S_\phi S_\psi + C_\phi C_\psi S_\theta\\ C_\phi S_\theta S_\psi - C_\psi S_\phi\\ C_\theta C_\phi\end{bmatrix}\begin{bmatrix}0 & -r & q\\ r & 0 & -p\\ -q & p & 0\end{bmatrix}\begin{bmatrix}u_b\\v_b\\w_b\end{bmatrix} \quad (3.48)$$

Equation 3.47 and 3.48 summarize the definition of translational dynamics. Now, we will discuss the rotational dynamics. Two methods can be used to study rotational dynamics, Newton-Euler method and the Euler-Lagrange method (Alaimo et al., 2013, P.4). We will start to analyze Newton's second law of rotation (Tytler, 2017, P.6-7).

$$\boldsymbol{\tau} = \frac{d\boldsymbol{H}}{dt} = \frac{d(\boldsymbol{I\omega})}{dt} \quad (3.49)$$

Where τ is Torque, H is Angular Momentum, I is Moment of Inertia, and ω is Angular Velocity. The derivative of angular momentum is

$$\boldsymbol{\tau} = \boldsymbol{I\dot{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I\omega} \quad (3.50)$$

This torque is represented in the body frame; thus, the angular velocity of the body frame is $\boldsymbol{\omega} = [p \quad q \quad r]^{\mathrm{T}}$. Hence, to complete the study of the rotational dynamics, the external torques affecting the quad-rotor should be defined, and these torques are gyroscopic torque vector and external torque generated by rotors (García Carrillo et al., 2013, P.29-31) (Artale et al., 2013, P.7).

$$\boldsymbol{I\dot{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I\omega} = \boldsymbol{\tau}_b - \boldsymbol{\Gamma} - \boldsymbol{\tau}_{dist} \quad (3.51)$$

Where τ_b is external torque and Γ is the gyroscopic force or sometimes named as gyroscopic precession. Every term in the above equation will be clarified in details,

$$\boldsymbol{I\dot{\omega}} = \boldsymbol{I}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} \quad (3.52)$$

Where $\mathbf{I\dot{\omega}}$ is the angular acceleration of the inertia in the body frame, $\boldsymbol{I}$ is the Moment of inertia in which $I_c = m\,x\,r^2$, I_c is the moment of inertia relative to the center of mass (m) $I_c = \sum_i m_i \Delta r_i^2$.

$$\boldsymbol{\omega} \times \boldsymbol{I\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix} \tag{3.53}$$

$$\Gamma = \frac{I_r \pi}{30} \begin{bmatrix} p \\ q \\ r \end{bmatrix} x \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_\Gamma \tag{3.54}$$

$$\omega_\Gamma = \omega_1 - \omega_2 + \omega_3 - \omega_4 \tag{3.55}$$

Gyroscopic force (Γ) is an event that occurs when a rotating body switches its axis of rotation; and the observations are usually unintuitive for those unfamiliar with its results. Gyroscopic forces are generated by the inertia (I_r) of each rotating rotor (ω_Γ) in the quad-rotor. Hence, the change of sign for rotors in equation 3.55 is due to different rotating directions. Thus, to prevent the quad-rotor from rotating around itself, each neighboring motor has to spin opposite one another as shown in Figure 3.4. Also, the rate of change of the quad-rotor's angles $[p \quad q \quad r]$ affects the gyroscopic force. The numeric number $\left(\frac{\pi}{30}\right)$ in the above equation is for converting the value from revolution per minute (rpm) to radians per second(rad/s).

Figure 3.4: Rotors spins in opposite direction

$$\boldsymbol{\tau}_{dist} = \begin{bmatrix} \tau_{dist,x} \\ \tau_{dist,y} \\ \tau_{dist,z} \end{bmatrix} \tag{3.56}$$

The disturbance torque is applied directly to the body frame coordinate system. Accordingly, the external and disturbance torques are affecting the same frame. Depending on the quad-rotor configuration, the external torques are divided into two types, it can be either "×" configuration or "+" configuration as shown in Figure 3.5.

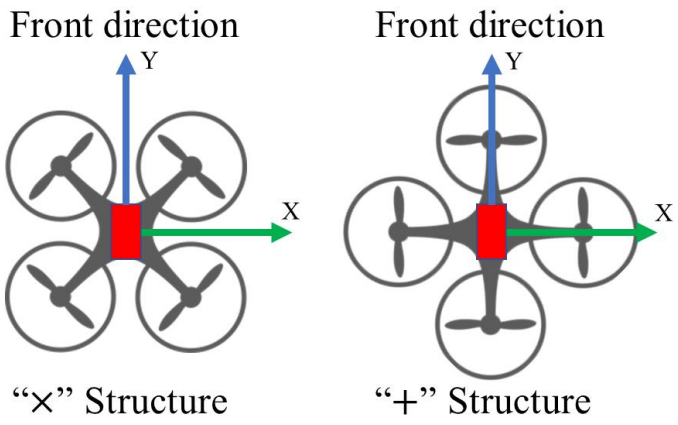


Figure 3.5: Quad-rotor "×" and "+" structures

For quad-rotor with "×" structure, the mathematical expression for external torques will be as follows,

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_t & c_t & c_t & c_t \\ -lc_t & lc_t & lc_t & -lc_t \\ -lc_t & -lc_t & lc_t & lc_t \\ -c_q & c_q & -c_q & c_q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.57}$$

$$l = d * sin(45) \tag{3.58}$$

In which T is the summation of thrust applied by the rotors, the base equation for thrust is as follows: $T = c\rho A_r r^2 \omega^2$, c is Thrust factor, ρ is air density, A_r is the cross-sectional area that the propellers covers, r is rotor's radius, and ω is rotor's speed. Therefore, all these values can be gathered in a lumped coefficient such as c_t. c_t Represents the lumped coefficient for thrust formula which affects $roll$ (φ) and h (θ). c_q Represents the thrust coefficient that affects yaw (ψ) angle. l is the distance between the rotor center and the corresponding rotational axes.

For "×" configuration, the distance (d) - between the center of the rotor - and the center of the quad-rotor should be multiplied by $\sin(45)$ to obtain the distance l over the corresponding axes. Conversely, for "+" configuration, the distance between the rotor and the center of quad-rotor is completely over x-axis or y-axis. Thus, the distance (d) equals exactly the distance (l) which simplifies the matrix of thrust torques.

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_t & c_t & c_t & c_t \\ 0 & dc_t & 0 & -dc_t \\ -dc_t & 0 & dc_t & 0 \\ -c_q & c_q & -c_q & c_q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.59}$$

All of the above equations explain the base Equation 3.51. Note that these equations were gathered to give the following result:

$$\boldsymbol{I\dot{\omega}} = \boldsymbol{\tau}_b - \boldsymbol{\Gamma} - \boldsymbol{\omega} \times \boldsymbol{I\omega} - \boldsymbol{\tau}_{dist} \tag{3.60}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \boldsymbol{I}^{-1} \left(\begin{bmatrix} 0 & dc_t & 0 & -dc_t \\ -dc_t & 0 & dc_t & 0 \\ -c_q & c_q & -c_q & c_q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} - \frac{I_r \pi}{30} \begin{bmatrix} p \\ q \\ r \end{bmatrix} x \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_\Gamma - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx} p \\ I_{yy} q \\ I_{zz} r \end{bmatrix} - \begin{bmatrix} \tau_{dist,x} \\ \tau_{dist,y} \\ \tau_{dist,z} \end{bmatrix} \right) \tag{3.61}$$

Equation 3.61 is the base equation to get the angular velocity values in body frame.

3.1.5. Quad-rotor Modeling Result

After we finished quad-rotor mathematical models' representation, we will take the main equations to be integrated in a *Matlab* block, so that they're ready for simulation and controller design. The main parameters are the location of the quad-rotor$[x, y, z]$, velocity$[u, v, w]$, angles$[\varphi, \theta, \psi]$, and angular velocity$[p, q, r]$. These parameters can be found by integrate linear velocity, linear acceleration, angular velocity, and angular acceleration as shown below,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\theta C_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \tag{3.62}$$

$$\begin{bmatrix}\dot{u}_b\\ \dot{v}_b\\ \dot{w}_b\end{bmatrix} = \frac{1}{m}\begin{bmatrix}0\\0\\k\sum_{i=1}^{4}\omega_i^2\end{bmatrix} + g\begin{bmatrix}-S_\theta\\ C_\theta S_\phi\\ C_\theta C_\phi\end{bmatrix} + \frac{1}{m}\begin{bmatrix}C_\theta C_\psi & C_\theta C_\psi & -S_\theta\\ C_\psi S_\theta S_\phi - C_\phi S_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\theta S_\phi\\ S_\phi S_\psi + C_\phi C_\psi S_\theta & C_\phi S_\theta S_\psi - C_\psi S_\phi & C_\theta C_\phi\end{bmatrix}\begin{bmatrix}F_{dist,x}\\F_{dist,y}\\F_{dist,z}\end{bmatrix} - \begin{bmatrix}0 & -r & q\\ r & 0 & -p\\ -q & p & 0\end{bmatrix}\begin{bmatrix}u_b\\v_b\\w_b\end{bmatrix} \tag{3.63}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & sin\,(\phi)tan\,(\theta) & cos\,(\phi)tan(\theta)\\ 0 & cos\,(\phi) & -sin\,(\phi)\\ 0 & \frac{sin\,(\phi)}{cos\,(\theta)} & \frac{cos\,(\phi)}{cos\,(\theta)}\end{bmatrix}\begin{bmatrix}p\\q\\r\end{bmatrix} \tag{3.64}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \boldsymbol{I}^{-1}\left(\begin{bmatrix}0 & dc_t & 0 & -dc_t\\ -dc_t & 0 & dc_t & 0\\ -c_q & c_q & -c_q & c_q\end{bmatrix}\begin{bmatrix}\omega_1^2\\\omega_2^2\\\omega_3^2\\\omega_4^2\end{bmatrix} - \frac{I_r\pi}{30}\begin{bmatrix}p\\q\\r\end{bmatrix} x \begin{bmatrix}0\\0\\1\end{bmatrix}\omega_\Gamma - \begin{bmatrix}p\\q\\r\end{bmatrix}\times\begin{bmatrix}I_{xx}p\\I_{yy}q\\I_{zz}r\end{bmatrix} - \begin{bmatrix}\tau_{dist,x}\\\tau_{dist,y}\\\tau_{dist,z}\end{bmatrix}\right) \tag{3.65}$$

By using equation 3.62 to 3.65, we can start testing the quad-rotor system and implement these equations in *Matlab* software as a system block to be able to design the controller and examine the behavior of the quad-rotor system under different conditions. The system block consists of four inputs which are angular velocities of the rotors, and twelve outputs consisting of position, linear velocity, angles, and angular velocities as shown in Figure 3.6.

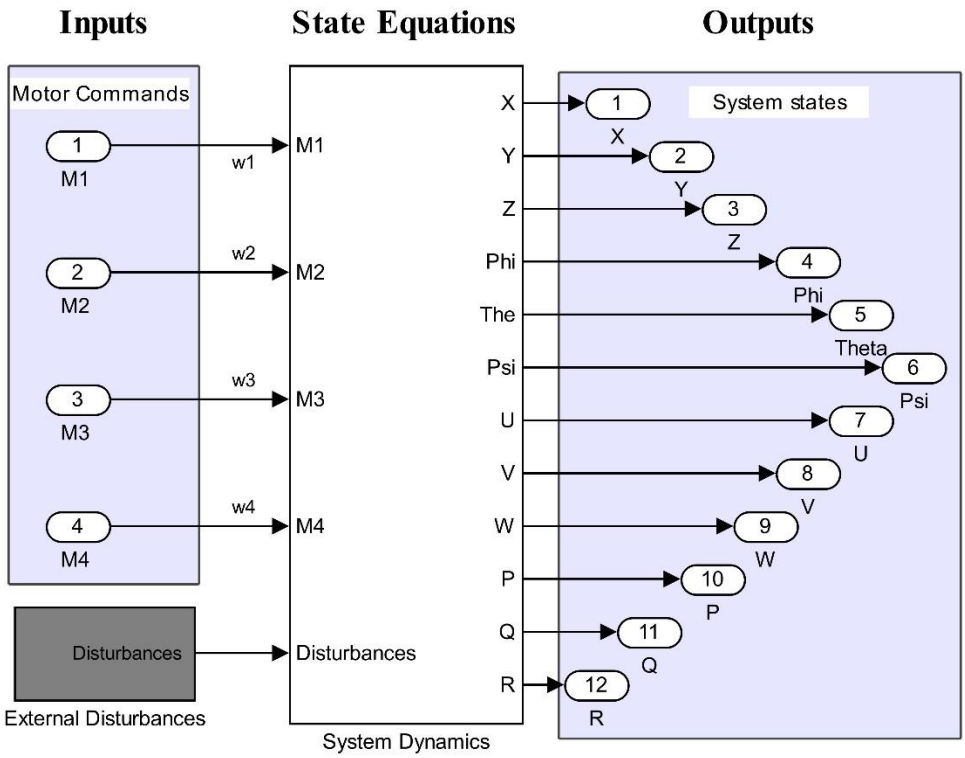


Figure 3.6: System Dynamics Block Inputs & Outputs

The method used to implement system model equations is *S-Function*. In this method, we can write our state equations and provide them with inputs, so that we can get the desired outputs as shown in Figure 3.7.

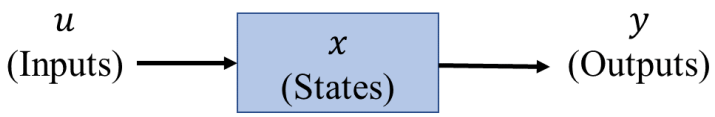


Figure 3.7: S-Function general structure

S-Function depends on 3 main general formulas as described in *Matlab*, these formulas are as follows,

$$y = f_0(t, x, u) \qquad \text{(Outputs)} \tag{3.66}$$

$$\dot{x} = f_d(t, x, u) \qquad \text{(Derivatives)} \tag{3.67}$$

$$x_{d_{k+1}} = f_u\left(t, x_c, x_{d_k}, u\right) \qquad \text{(Update)} \tag{3.68}$$

3.2. NON-INERTIAL FRAME OF REFERENCE

All the relations and transformation matrices were based on Inertial Frame of Reference (IFR); thus, the transformation matrix in Equation 3.7 and 3.41 are the relationship between IFR and body frame of the quad-rotor. IFR can be one of the known frames such as earth – Centered coordinate system, geodetic coordinate system (Longitude, Latitude and altitude) and Local North-East-Up or Down (NEU/NED) coordinate system. Cai et al. (2011) summarized most of the famous coordinate systems that can be used as IFR. On the other hand, if we want to consider a moving object as our reference frame, some of the state equations should be changed to achieve the desired transformation. To transform from an inertial frame of reference to a non-inertial frame of reference, we need to make a relation between the position of the body frame "quad-rotor" and the moving object such as car, ship or even a plan. Normally, in order to make the mathematical model of any system, we need to obtain the state equation of this system, and in order to obtain position equation, we need to obtain the velocity since the position state is the body frame velocity.

$$\begin{bmatrix} \dot{x}_b^i \\ \dot{y}_b^i \\ \dot{z}_b^i \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} \tag{3.69}$$

A relation between the velocity vector of the body frame referenced to the inertial frame and the velocity of the reference moving object referenced to the inertial frame should be established. This will provide the velocity of the body frame referenced to the moving object as a non-inertial frame as shown in the equation below:

$$\dot{\boldsymbol{\xi}}_b^m = \dot{\boldsymbol{\xi}}_b^i - \dot{\boldsymbol{\xi}}_m^i = \begin{bmatrix} \dot{x}_b^i \\ \dot{y}_b^i \\ \dot{z}_b^i \end{bmatrix} - \begin{bmatrix} \dot{x}_m^i \\ \dot{y}_m^i \\ \dot{z}_m^i \end{bmatrix} = \begin{bmatrix} \dot{x}_b^m \\ \dot{y}_b^m \\ \dot{z}_b^m \end{bmatrix} \tag{3.70}$$

$$Velocity\ error = \dot{\boldsymbol{\xi}}_{desired}^m - \dot{\boldsymbol{\xi}}_b^m \tag{3.71}$$

In which $\dot{\boldsymbol{\xi}}_b^i$ is the position state vector of the body frame referenced to the inertial frame, $\dot{\boldsymbol{\xi}}_m^i$ is the position state vector of the moving object related to the inertial frame, and $\dot{\boldsymbol{\xi}}_b^m$ is the resultant position state vector of the body frame related to the moving object.

The output state vector of the system model should be modified to make a suitable transition from the inertial frame of reference to a non-inertial frame of reference. The normal output state vector inertial referenced can be expressed as:

$$\boldsymbol{X}^T = [x\ \ y\ \ z\ \ u\ \ v\ \ w\ \ \phi\ \ \theta\ \ \psi\ \ p\ \ q\ \ r]^T \tag{3.72}$$

The state vector of the inertial referenced frame contains the information of position, velocity, angular position, and rotational velocity of the body frame. In case of non-inertial reference frame, the information of the position and velocity of the moving body should be included in the state vector as follows:

$$\dot{\boldsymbol{X}}^T = [x_b\ \ y_b\ \ z_b\ \ u_b\ \ v_b\ \ w_b\ \ x_m\ \ y_m\ \ z_m\ \ u_m\ \ v_m\ \ w_m\ \ \phi\ \ \theta\ \ \psi\ \ p\ \ q\ \ r]^T \tag{3.73}$$

The new information in the state vector is the position and velocity of the moving body; these two quantities can be obtained by providing the acceleration of the moving object to the state equation. Therefore, the block of the state equation must have additional input; this input is the acceleration of the moving object. The changes of the system block are shown in Figure 3.8.

The error of position in body frame will be a result of relating position feedback of body frame with position feedback of moving frame as follows:

$$x_{err}^b = x_d^m + x_m^i - x_b^i \tag{3.74}$$

Where x_d^m are the desired position referenced to moving object (non-inertial frame referencing), x_m^i is the feedback position of the moving object, and x_b^i is the feedback position of the body frame referenced to the inertial frame.

The result of this modification will convert the desired position to be the actual desired position referring to the moving object non-inertial frame and any movement of the non-inertial frame. The position and velocity state vector of the non-inertial frame will be derived from its acceleration. The acceleration information of the moving object can be obtained directly from the moving object system or by external sensors mounted over it.

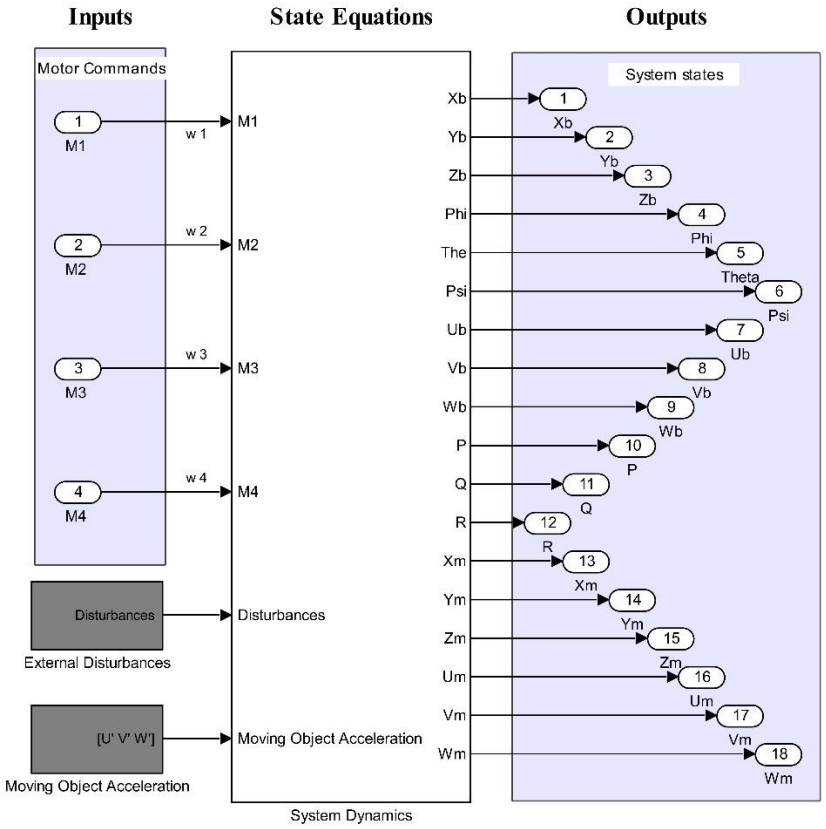


Figure 3.8: System Dynamics Block for NIFR referencing

3.3. OPEN-LOOP SYSTEM STABILITY CHECKING

To check the stability of an open loop system; we examined the flight's stability from an equilibrium state, if the system's response did not change its equilibrium state, then the open-loop system is stable, but if the system's response changes exponentially over time, it indicates the system is not stable. Thus, we launch the flight to reach an altitude of three meters without changing the quad-rotor orientation and without applying any external forces to the system. Simulink code was written to leave the system out of control after a specific period of time in which to analyze its response.

The Open-Loop system response indicates instability for angles Phi, Theta, and Psi. Figure 3.9 shows the Phi, Theta, Psi responses and the quad-rotor position over x-axis. The problem of an Open-Loop system's instability can be solved by designing an effective controller that can stabilize the system the following section addresses controller design.

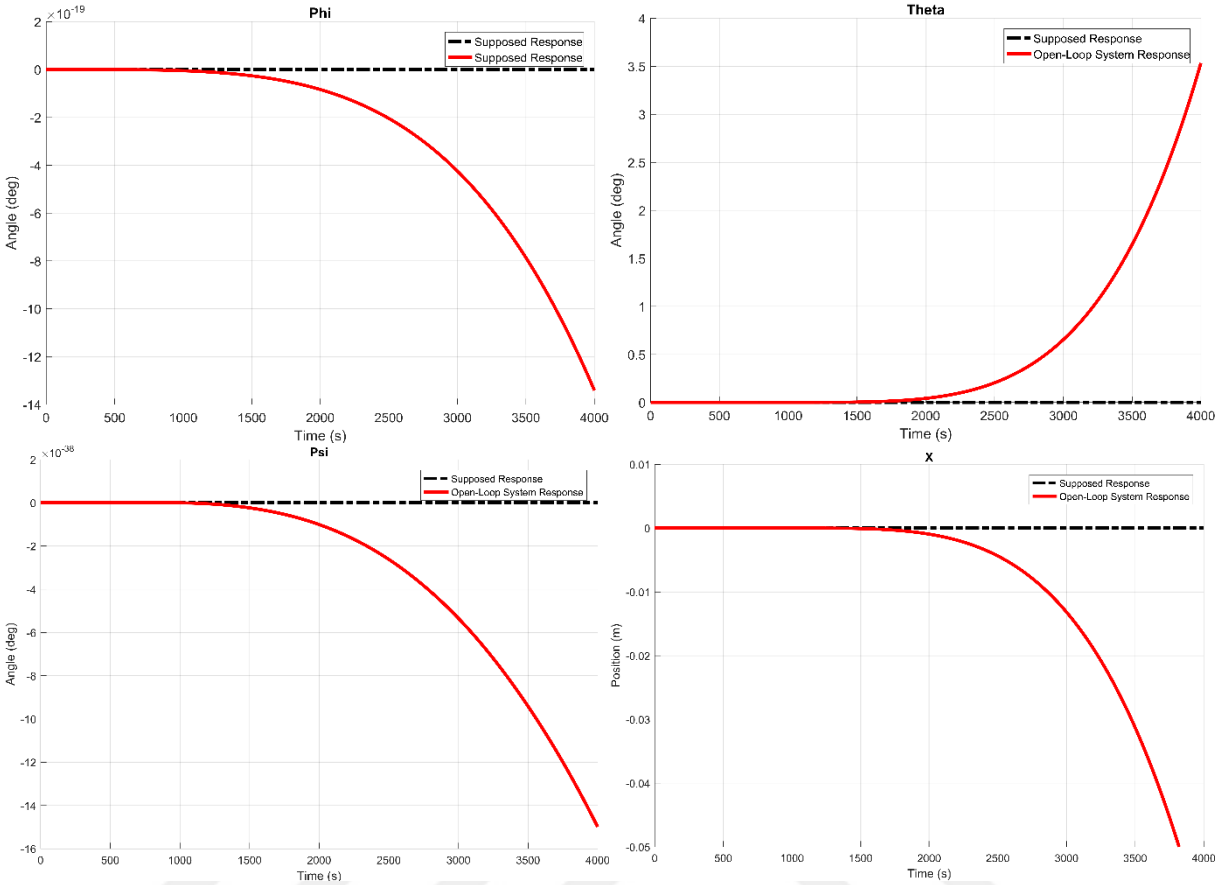


Figure 3.9: System Open-Loop stability checking

3.4. CONTROLLER DESIGN

3.4.1. Multi-Stage PID Controller

The main control algorithm used for quad-rotor control is PID control algorithm. PID controller was selected since minimal mathematical calculations would provide the necessary stability. The simplicity of the controller is important due to the limitation of processor, memory and physical microcontroller specifications. The mathematical formulation of the PID expressed in time-domain is as follows,

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{3.75}$$

The assumption in this study is to take a signal from a specific path as a commanded input to the system. The path gives the information of X, Y and Z position represented in time-series as an input to the system; also, the path command provides the required Psi angle of the drone.

The Phi and Roll angles command are given after processing X and Y command. The structure of the overall system and controller is shown in Figure 3.10.

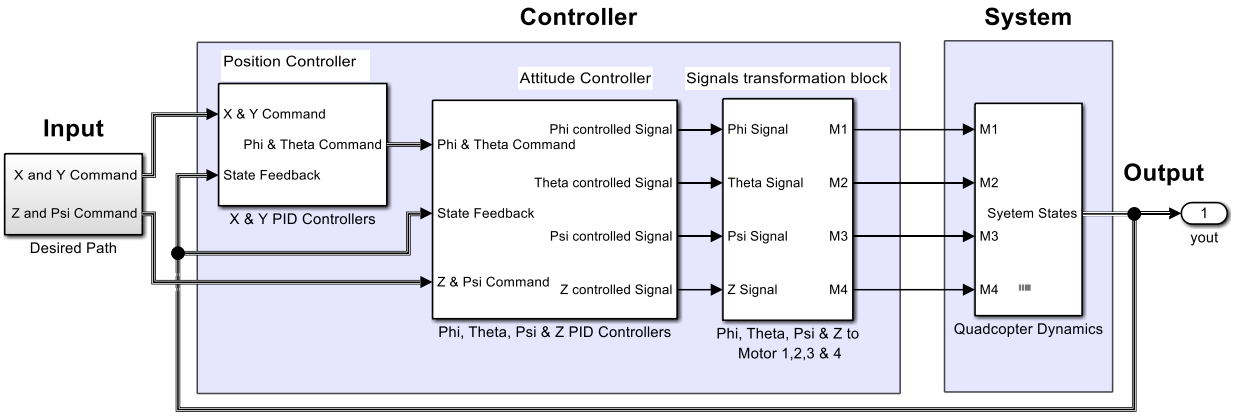


Figure 3.10: The structure of the UAV System and controller

The system block contains quad-rotor dynamics. In the previous section, the dynamics of the system was explained, regardless of the complicated mathematical equation, the system block can be considered as a black box with inputs and outputs. The inputs of the system block are angular velocities of the rotors $(\omega_1, \omega_2, \omega_3, \omega_4)$ and disturbances$\left(F_x, F_y, F_z, \tau_x, \tau_y, \tau_z\right)$, the outputs of the system block are the states of the system and contains the quad-rotor's velocity vector(u, v, w), angular velocity vector(p, q, r), position vector (x, y, z), and angular position vector (φ, θ, ψ) as shown in Figure 3.6.

The controller consists of three stages, the first stage is X and Y position controller, the second stage is attitude controller to control phi, theta, psi and Z commands, the third stage is translation from phi, theta, psi and altitude correction signals to a throttle signal as an input to four rotors.

3.4.2. First Stage Position Control

First-stage controller inputs are X, Y and Psi angle commands, X and Y feedback states in inertial frame, u and v velocities in body frame as the change of position feedback. The outputs are Phi and Theta angles command as shown in Figure 3.11. It should be noticed that the state equation block provides the position with respect to inertial frame reference, and the velocity in body frame reference. Position error in the inertial frame can be represented as follows,

$$x^i_{err} = x^i_{cmd} - x^i_{state} \tag{3.76}$$

$$y_{err}^{i} = y_{cmd}^{i} - y_{state}^{i} \tag{3.77}$$

Where x_{cmd}^{i} is the command position in the inertial frame.

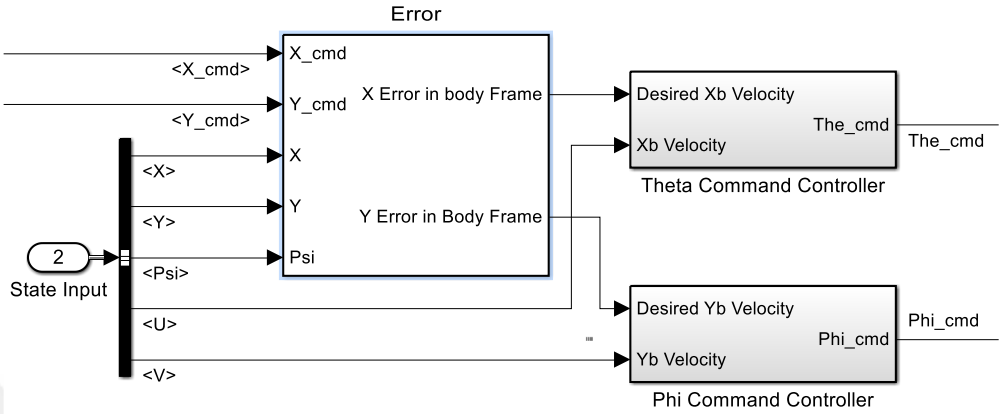


Figure 3.11: The structure of First Stage PID controller

The error signal should be multiplied by the rotation matrix to transform the error signal from the inertial frame to body frame; it can be represented as,

$$x_{err}^{b} = x_{err}^{i}\, cos(\psi) + y_{err}^{i}\, sin(\psi) \tag{3.78}$$

$$y_{err}^{b} = y_{err}^{i}\, cos(\psi) - x_{err}^{i}\, sin(\psi) \tag{3.79}$$

x and y errors in the body frame should be converted in the appropriate axis as a velocity signal to solve the error in that direction.

$$u_{desired}^{b} = x_{err}^{b} \tag{3.80}$$

$$v_{desired}^{b} = y_{err}^{b} \tag{3.81}$$

Velocity error will be the difference between the desired velocity $u_{desired}^{b}$ in X-axis and the state feedback velocity u_{state}^{b}. The output of the controller will be angle command θ_{cmd}. Maximum and minimum angle of inclination "bank angle" should be considered to avoid losing control during flight.

$$\theta_{cmd} = \begin{cases} -12^{o}, & \theta \le -12^{o} \\ K_p\left(U_{desired}^{b} - U_{state}^{b}\right) - K_d U_{state}^{b}, & -12^{o} < \theta < 12^{o} \\ 12, & \theta \ge 12^{o} \end{cases} \tag{3.82}$$

Of course, the bank angle varies from one application to another, even some drones are designed for acrobatic flights and sports competitions, in that case, bank angle will be unlimited. The structure of the controller can be seen in Figure 3.12. The same procedure will be done to obtain φ_{cmd}.

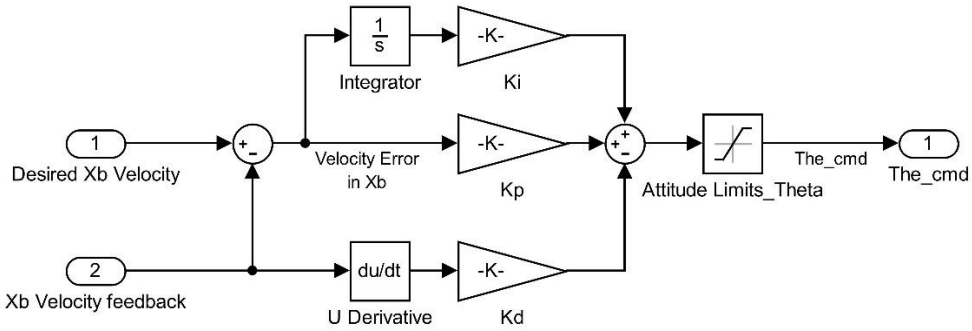


Figure 3.12: The structure of first stage position controller

In the beginning, all six controllers were PID controllers, but during the optimization process with AGA, the controller of x and y axes were changed to PD controller. The optimization process changed the integral parameter for x and y axes controllers to zero. PD controller gave a better performance than PID. Therefore, after we added disturbances, the integral part got a small value again.

3.4.3. Second Stage attitude and altitude Control

The inputs of the second stage controller are φ_{cmd}, θ_{cmd}, ψ_{cmd} and z_{cmd} command signals, φ_{state}, θ_{state},ψ_{state}, P_{state}, Q_{state} and R_{state} are feedback states to the controller. The controller compensates the angle commands and altitude (z) command as a correction signal. The controller structure is shown in Figure 3.13.

The mathematical formula of the second stage controller for θ angle is shown below; the same formula for φ, ψ angles and z altitude were considered.

$$\theta_{Comp}(t) = K_p\big(\theta_{cmd}(t) - \theta_{state}(t)\big) + K_i\int\big(\theta_{cmd}(t) - \theta_{state}(t)\big)dt + K_d * Q_{state}(t) \tag{3.83}$$

The output of the second stage will be input to the third stage; in this part, the controller will translate the compensation values to throttle command “Rotors command”.

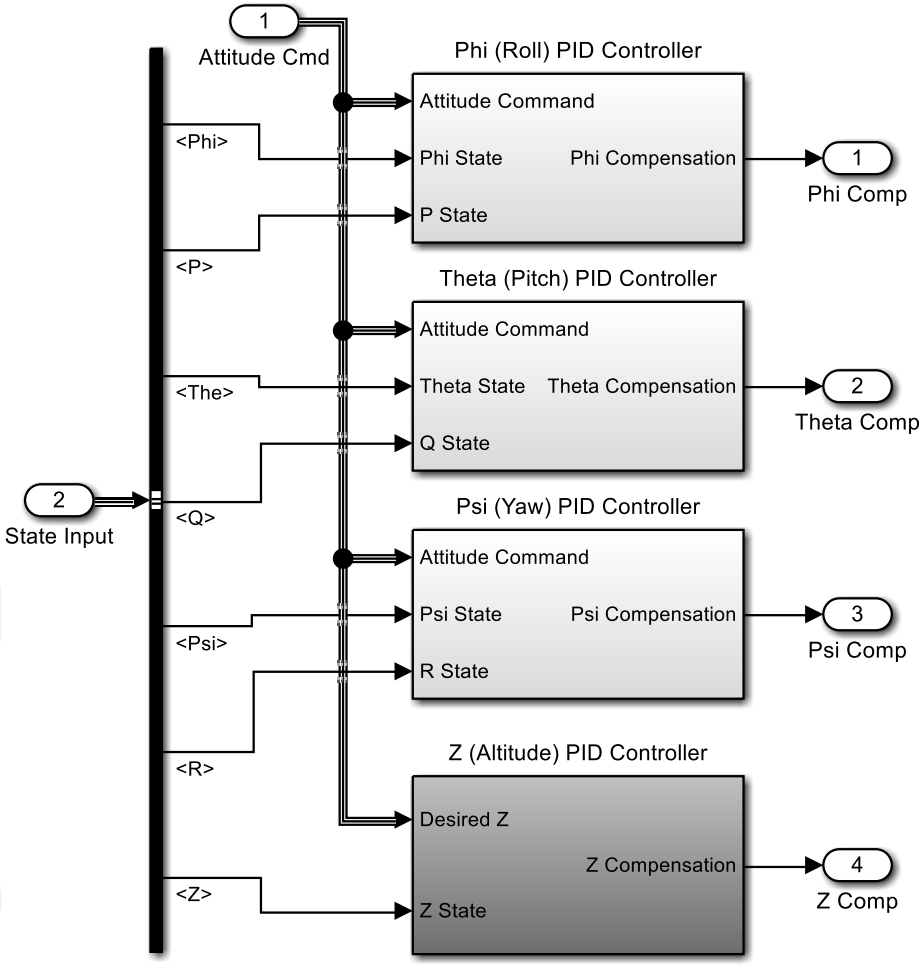


Figure 3.13: The structure of Second Stage, attitude and altitude controllers

3.4.4. Third Stage Rotors Controller

At this point, the result of position, attitude, and altitude controllers can be converted to rotors speed signals. Controllers' output signals are Phi, Theta, Psi, and Z compensations. Compensation signals can be transformed depending on quad-rotor configuration, as indicated in system modeling. These two forms are (+) configuration and (×) configuration. The general structure of third stage rotor controller is shown in Figure 3.14.

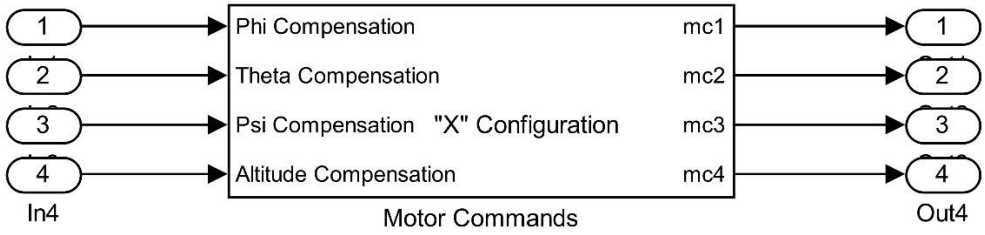


Figure 3.14: General structure of motors controller

First let us represent motor commands for (+) configuration.

$$\begin{bmatrix} mc_1 \\ mc_2 \\ mc_3 \\ mc_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z_{comp} \\ \varphi_{comp} \\ \theta_{comp} \\ \psi_{comp} \end{bmatrix} \tag{3.84}$$

To explain the concept of motor commands, we may take an example of going straight in a positive x-axis direction. The assumed quad-rotor structure is $(+)$ configuration. The first stage controller must raise the velocity in that direction in order to travel straight in the positive direction of the x-axis; hence, increasing the velocity in that direction would increase the angle of Theta (θ). Figure 3.12 helps us understand this process. Two motors are responsible for the adjustment of the Theta angle, Motor 1 and Motor 3 as shown in figure 3.15. We will reduce the velocity of motor 1 and increase the velocity of motor 3 to relate the increasing in Theta (θ) angle to motor commands. For that consequence, reducing the velocity of motor 1 can be formulated as:

$$mc1 = Z_{comp} - \theta_{comp} - \psi_{comp} \tag{3.85}$$

Increasing the velocity of motor 3 can be formulated as:

$$mc3 = Z_{comp} + \theta_{comp} - \psi_{comp} \tag{3.86}$$

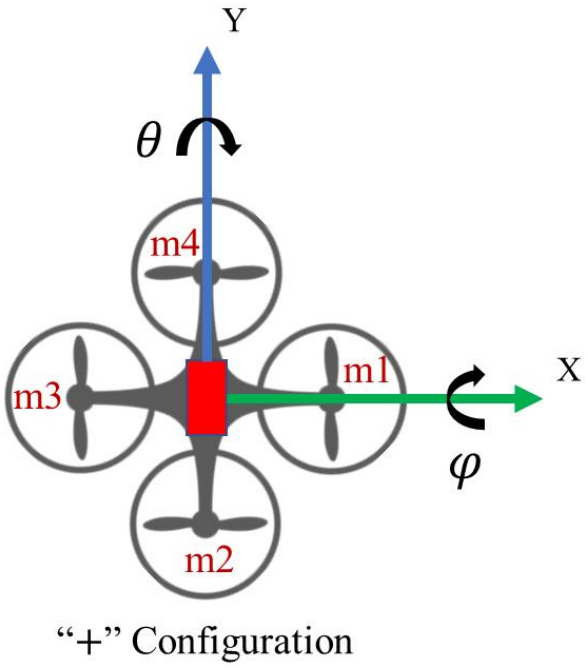


Figure 3.15: Motor/Angles relation

Equation 3.84 is for $(+)$ configuration, for $(\times)$ configuration; the axes will be between the arms of the quad-rotor, which means, to move straight in x-axis direction, the four motors will be changed. Motor commands formula is shown in Equation 3.87.

$$\begin{bmatrix} mc_1 \\ mc_2 \\ mc_3 \\ mc_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} Z_{comp} \\ \varphi_{comp} \\ \theta_{comp} \\ \psi_{comp} \end{bmatrix} \tag{3.87}$$

Throttle signals of motor commands are constrained by minimum and maximum values. The mc_{min} and mc_{max} values can be obtained from quad-rotor's and motors' characteristics. Throttle signals are calculated by interpolation formula to be consistent with rotor specifications.

$$Th_{cmd}^{i} = \begin{cases} 0, & mc_i \leq mc_{min} \\ a * mc_i + b, & mc_{min} < mc_i < mc_{max} \\ a * mc_{max} + b, & mc_i \geq mc_{max} \end{cases} \tag{3.88}$$

Where Th is motor's throttle signal, a and b are constants and can be obtained from Motor's specification, i is motor number, mc_{min} is the minimum allowable throttle the motors can operate with, and mc_{max} is the maximum throttle which motors can use. Motor's throttle can be considered as an input to motor's dynamics system block, the output is rotor angular velocity in revolution per minute (rpm) signal, and motor block system is shown in Figure 3.16.

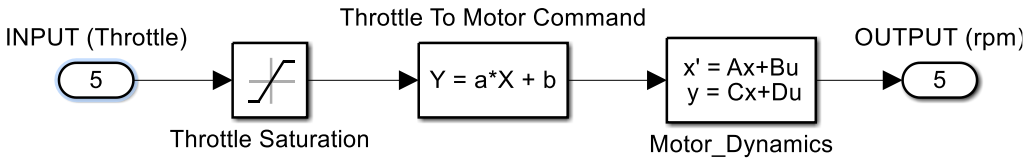


Figure 3.16: The structure of Motor's system and controller

In the end of control system, the output of the rotors in (rpm) will be the input of the system dynamics block as angular velocities as shown in Figure 3.6.

3.5. GA OPTIMIZATION PROCESS

Genetic Algorithm (GA) is an example of an optimization technique that uses random selection as a tool to guide a strictly exploitative search through parameter space coding. The advantages of using GA that it searches for the solution randomly; this method of searching can be useful

especially when the system is nonlinear. GA generally follows these steps: Genetic representation, Initial population, Fitness Function, and Genetic operations.

3.5.1. Chromosome Structure

The first step in GA is to determine the Genes/individuals; in our case, every gene represents the variable of the PID controllers. Each PID controller contains 3 genes represented as: K_p, K_i and K_d values. Thus, we have 6 PID controllers for$(x, y, z, \varphi, \theta, \psi)$, all of these genes' variables will construct one chromosome consisting of 18 parameters as shown in Table 3.3.

Table 3.3: GA Chromosome structure

1	2	3	4	5	6	7	8	9
K_P^x	K_I^x	K_D^x	K_p^y	K_i^y	K_d^y	K_p^z	K_i^z	K_d^z
10	11	12	13	14	15	16	17	18
K_p^φ	K_i^φ	K_d^φ	K_p^θ	K_i^θ	K_d^θ	K_p^ψ	K_i^ψ	K_d^ψ

There is more than one method to represent GA chromosome; it can be represented by binary, octal, hexadecimal or decimal numbers. Here we preferred to use decimal numbers to represent the chromosome. Decimal numbers have been used to hold each gene separate from the other, making it simpler to produce restricted initial population using this approach, and getting to the answer would be much quicker.

3.5.2. Initial Population

We designed the initial population to be 100 chromosomes. We did not increase the number of chromosomes in one population because of the huge processing time *Matlab* takes for each iteration (about 15 minutes), but instead we decided to play with iteration numbers. The base values of PID parameters were taken from a previous study done by Hartman et al. (2014). Then, a constrained random function with different weights produced 100 chromosomes as an initial population around the base parameters. Each gene is summed with a separated constrained weighted random number; the formula of generating the initial random population is shown below,

$$k_m^n(l) = k_m^n(base) + (R * a_m^n - b_m^n) \tag{3.89}$$

Where k is gain parameter, m specifies the gain type (P, I, D), n is the axis or angle $(x, y, z, \varphi, \theta, \psi)$, l is the number of the chromosomes in the initial population, R is a random number between zero and one, a is the width of random values, and b is the offset of random values. From a and b values, the range and limits of each gain value can be specified as a random value summed with the base gain value; the principle is clarified in Figure 3.17. The aim of restricting the gain values is to prevent the system from reaching an unstable area, as it will prevent *Matlab* from continuing the process normally, it will enter infinite processing loops.

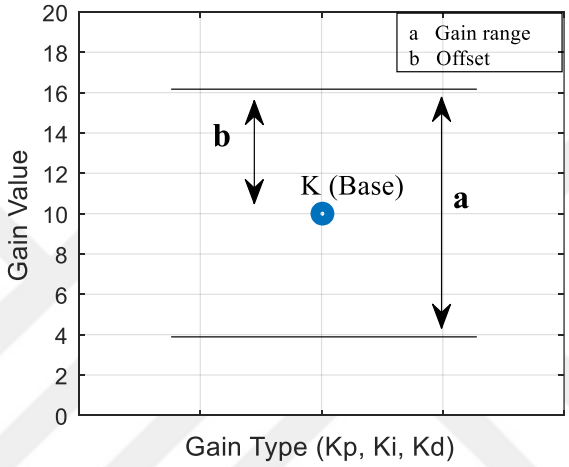


Figure 3.17: PID random values constraints

After building the initial population, a *Matlab* script command runs the Simulink model, and then a comparison between the result of the command signals and the simulated signal are performed to determine the amount of error.

3.5.3. Fitness Function

Fitness function examines how well the chromosome approaches the required solution, and then it gives a value to the chromosome which indicates its rank in the solution pool. It can be Integral of Squared Error, Integral of Absolute Error, Integral of Time Weighted Squared Error, or other functions. In this study, the optimization process is divided into two steps: the first step is to optimize the parameters to obtain a low error value, while the second step in optimization process is for fine tuning to obtain the optimum solution. For the first step, we used Integral of Squared Error and Max of Squared Error as follows:

$$Fitness\ value = \int_0^\infty e^2(t)dt \qquad \text{(Integral of Squared Error)} \tag{3.90}$$

$$Fitness\ value = max\left(e^2(t)\right) \qquad \text{(Max of Squared Error)} \tag{3.91}$$

Fitness function is mixed between Integral of Squared Error for each PID controller, Max of Squared Error for each PID controller and Integral of Squared Error for all PID controller results. Each fitness function is multiplied by weight value; these weight values can be changed by the user depending on the response and performance of GA operations. The overall fitness value will be as follows:

$$F.V. = w_1 * \sum_{i}^{6}\left(\int_0^{\infty} e_i^2(t)dt\right) + \sum_{i}^{6}\left(w_2 * \int_0^{\infty} e_i^2(t)dt + w_3 * max\left(e_i^2(t)\right)\right) \tag{3.92}$$

Where i represents PID controller such as: $i = [x, y, z, \varphi, \theta, \psi]$, e_i represents the error for each PID controller.

Each chromosome will be evaluated depending on its fitness value, then a sorting operation sorts the chromosomes; thus, the best chromosome with the best fitness value will be the first chromosome in the population. The second-best fitness value will be the second chromosome and so on. By this way, the upper half chromosomes in the population are the fittest solutions and the lower half will be the worst solutions.

3.5.4. Reproduction Process

The process of reproduction consists of three steps: the selection of parents, the cross-over process, and the mutation. For selection of parents, the developer will choose the pairs of parents that will be crossed depending on the response of the optimization process. The new parents are used to make a new offspring to create the next population. The method of selecting parents is used to duplicate the best solutions to produce parent chromosomes.

Then a cross-over operation is undertaken to generate new children (solutions) from the pre-selected parents. Since we chose encoding the genes in decimal coding, we preferred using a single point crossover to complete the cross-over operation.

At last, a Mutation operation is done. The importance of using mutation operation is to escape a local minimum solution. Thus, in that way, the solution can be completely changed, compared to the base solution. GA will therefore arrive at a better answer by using mutation. After a certain number of generations of children, we applied a mutation with a percentage of 1 percent

from all over the population. A gene will be changed from decimal representation to 16-bit binary number; then, a randomly chosen bit will be flipped.

3.5.5. Adaptive Genetic Algorithm

Due to long processing time taken by *Matlab* and *Simulink* to simulate and execute all the operations related to multi-copter flight system, and due to long mathematical operations needed to solve system dynamics and controller response, the normal GA is moved with slow steps toward finding the solution. In that case, we tried to modify some parts of GA process for the sake of accelerating the search principle. The adaptive GA is similar in the main procedure logic, but the difference is in generating new populations. Normally, generating new population consists of selecting parents and crossover process, but the new method is to divide the generation process to three steps:

1- Normal process: First of all, the solutions are tested, then they are sorted such that the first chromosome is the best solution, then the second-best solution is the second chromosome and so on. The selection process is then accomplished by choosing the best half of the sorted population. The best half sorted population will be the new parents' chromosomes. Then by crossover process, we generate 50% of the new population.
2- Generate constrained random solutions around one third of the best sorted solutions of the old population. The generation function will be same as in equation 3.89. Thus, the range of random values (a) will be small, so that we are searching for the solution near best old solutions. By this method, 30% of the new generation will be produced.
3- Generate constrained random solutions around one fifth of the best sorted solutions of the old population. The difference between this step and the previous step is that the range of random values (a) will be larger. The aim of this step is avoiding fall in local minimum solution. This step can be considered as a mutation to refresh the optimization process.

By this new generation method, we can move with faster steps towards the best solution. To evaluate the efficacy of this new method, a comparison has been done between conventional GA and adaptive GA. For the comparison to be fair, the initial population of both methods is the same, the number of chromosomes in the initial population is 100 chromosomes, and the number of iterations of both methods is 10 iterations. The evaluation of each iteration depends

on squared total error for all PID controllers. The results of both methods can be shown in Table 3.4.

Table 3.4: Comparison between Conventional GA and Adaptive GA

Number of Iterations	Conventional GA	Adaptive GA
	Total Integrated Square Error	
1	395.87	395.87
2	397.198	363.929
3	388.363	354.026
4	384.874	337.825
5	384.803	318.168
6	384.49	310.48
7	384.106	314.313
8	384.106	307.316
9	381.619	290.028
10	381.625	286.261

Adaptive GA advantages are:

1- It has small number of chromosomes inside the population which minimize processing time for each iteration.

2- Adaptive PID gains values towards better solution which means a dynamic range based on Equation 3.89, a graphical explanation of the idea is shown in Figure 3.18.

3- Avoid falling in a local minimum by widening the range of the gains for 20% of new generations.

4- Each iteration gains range change depending on best solutions.

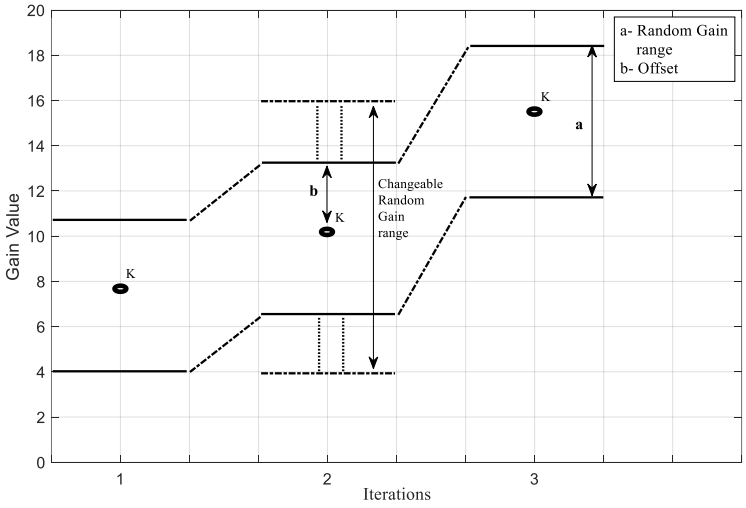


Figure 3.18: Adaptive random gain value range

3.6. ADAPTIVE CONTROL DESIGN

After doing much iteration for the optimization process with many different cases including wind disturbances, we figured that the optimum solution can't be defined as a single solution. The optimization process gave two different solutions for PID parameters: in no wind condition it gave optimum PID parameters, but with strong wind disturbances, it gave robust PID parameters to maintain stability. The conclusion of the tests is shown in Table 3.5.

Table 3.5: Disturbance vs. PID parameters

	No wind weather	Windy weather (Max dist.)
Optimum PID parameters	Perfect response	Unstable
Robust PID parameters	Bad response	Stable

In those cases, we performed a large number of simulations with different levels of disturbances and different PID values to test the response of the system. In the appendix 1 and 2, the data obtained from the simulation process are documented in a matrix.

The data obtained indicates that the z-controller is the one responsible for quad-rotor stability, especially K_p gain value. K_p Gain value has an effect on the quad-rotor stability as it amplifies the difference between the desired altitude and the quad-rotor real height, or in other words, it amplifies the error. Hence, if the error is large and the gain value of K_p is big, the action will be strong and cause a sudden response with a probability to enter unstable region. Also, K_i gain affects the response of the quad-rotor. Adjusting K_i value makes a fine tuning for the response to reach the best solution. Therefore, the gain values of K_p and K_i for different cases are not the same. The simulations we performed to get the best K_p and K_i values found out that the gain values could be modified for different conditions. The curve of the disturbances vs gain values was as shown in Figure 3.19 and 3.20.

As shown in the figures, the gain values change depending on the disturbance strength; moreover, and the disturbance versus gain values models a non-linear curve. In that case, the parameters of z-controller should be modified according to the current disturbance, which means that the controller needs to be an adaptive controller. Detailed values for K_p and K_i parameters versus disturbance values are represented in Appendix 1 and Appendix 2 respectively.

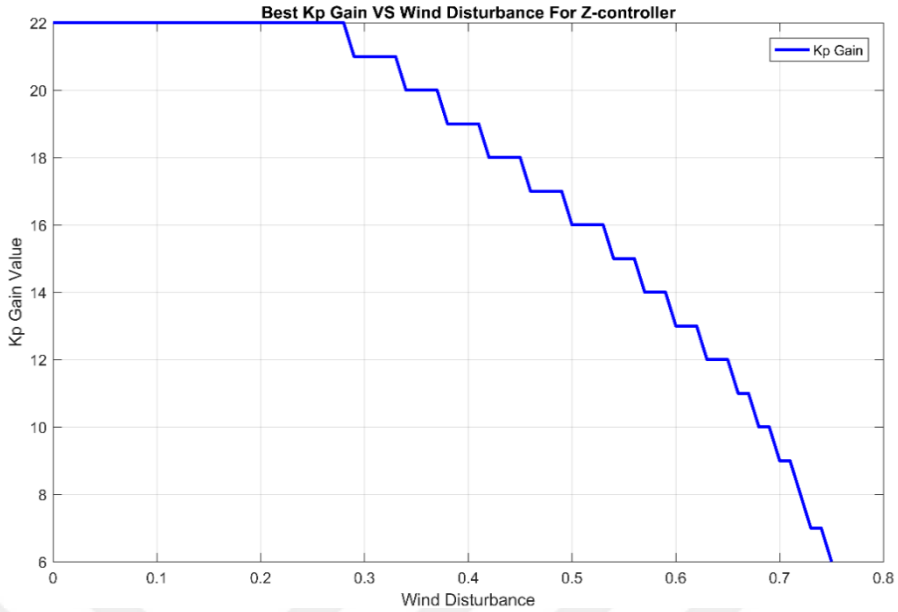


Figure 3.19: Best K_p gain values vs**.** disturbances

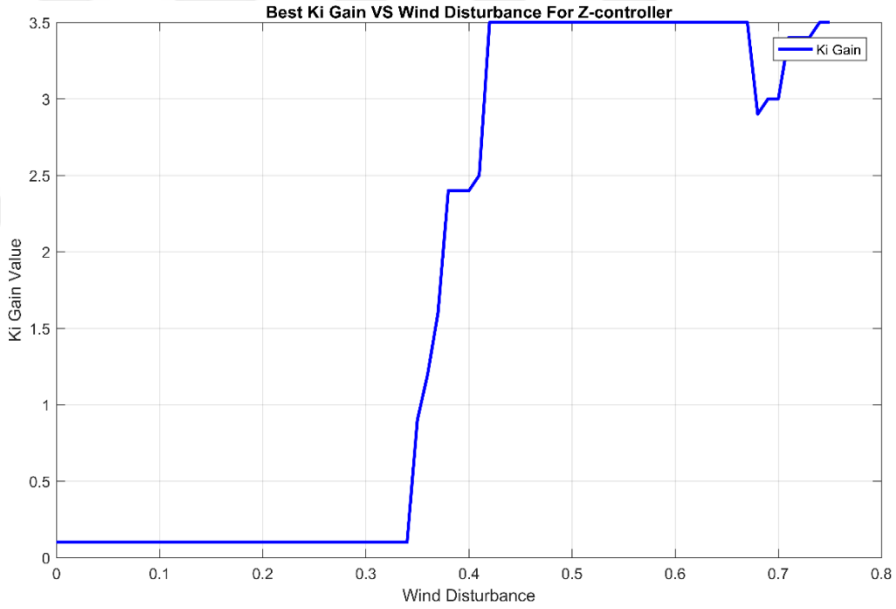


Figure 3.20: Best Ki gain values vs**.** disturbances

In order to design adaptive controller for a non-linear function, we designed two artificial intelligent techniques. We proposed two techniques to evaluate their efficiency compared to the desired non-linear function. The first method is Fuzzy control algorithm, the second one is Neural Networks. The main purpose of these techniques is to provide the best PID parameter according to the current wind disturbance. Those controllers' decisions are based on the non-linear curve obtained in Figure 3.19 and Figure 3.20. The best PID values obtained from a large number of simulations do not cover all possible disturbance values, so the intelligent techniques

must address this issue and be able to deal with all possible disturbance values. The intelligent controllers will tune and adapt the parameter of the main PID controller automatically.

3.7. FUZZY CONTROLLER DESIGN

Fuzzy control is one of the intelligent control algorithms. The advantages of using Fuzzy control are the ability to deal with complex non-linear functions, the way Fuzzy logic operates is similar to human, and the flexibility of making appropriate decisions depending on the shape of membership functions. In our case, the input of Fuzzy controller is disturbance value, and the output is the PID gain value. Generally, Fuzzy controller can be single input, single output (SISO), or multi-input single output (MISO). Our input is one input "the disturbance value", the output is two values K_p and K_i; thus, our controller should be single input multi output. But unfortunately, Fuzzy controller can't provide multi-output, and because of this, we will split the controller into two controllers, one for K_p values, and one for K_i values.

The first step in Fuzzy logic is to perform fuzzification to the input; the fuzzification process will transform the crisp value "disturbance value" to a linguistic value, such as No disturbances, small disturbances, strong disturbances, and so on. But in our case, the names of the membership functions are not very important, hence, we will give the names such as: membership function 1 (MF1), MF2, and so on.

In the fuzzification process, the non-linearity of the best value curve in Figure 3.19 and 3.20 will be considered. The shape and width of each membership will differ from in order to solve the problem of nonlinearity. In the fuzzification process, nine membership functions were used to overcome the non-linearity of the desired curve. Disturbance values were measured between 0 and 0.75. These values are a factor of the effective torque applied to the quad-rotor. The value of these disturbances is connected directly to system dynamics block to be included in system dynamics calculation. The regions of the membership functions were divided according to the strength of the disturbance as follows: $MF1 = [0,0.25]$ and the shape of the membership function was Trapezoidal, $MF2 = [0.25,0.45]$ Triangular, $MF3 = [0.35,0.53]$ Triangular, $MF4 = [0.45,0.6]$ Triangular, $MF5 = [0.53,0.65]$ Triangular, $MF6 = [0.6,0.7]$ Triangular, $MF7 = [0.65,0.73]$ Triangular, $MF8 = [0.7,0.75]$ Triangular, and $MF9 = [0.73,0.8]$ Trapezoidal. The width of first membership function is large, and then the width of membership

functions is getting smaller to address the nonlinearity of the curve, the membership functions of the fuzzification process are shown in Figure 3.21.

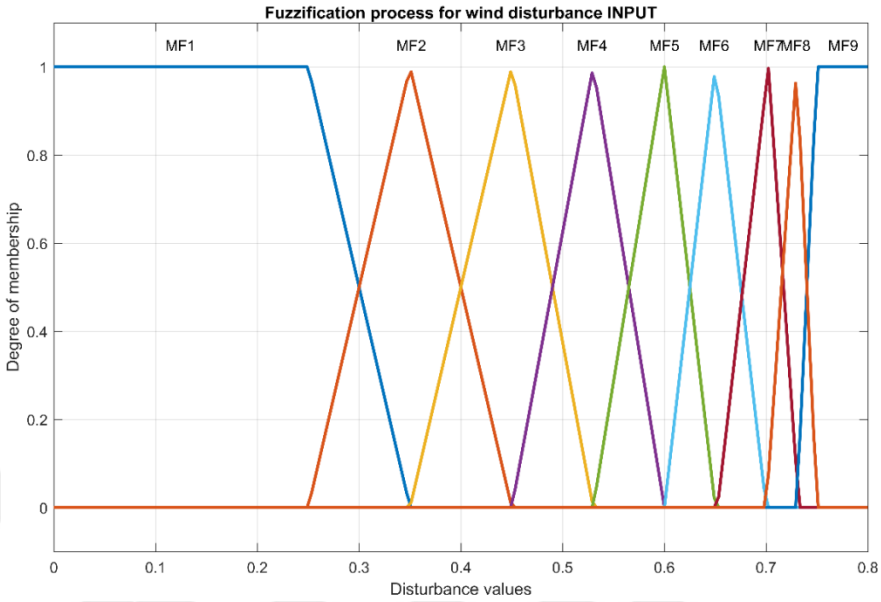


Figure 3.21: Membership functions of fuzzification process for $\boldsymbol{K_p}$ gain values

The output of Fuzzy controller is the value of K_p gain; therefore, we need a defuzzification process to transform the linguistic values to a crisp value. The shape of the output membership functions is almost typical because fuzzification process linearized the input. The range of the crisp output is between the optimum K_p gain value and the robust K_p gain value, $K_p = [4,23]$. The shape of defuzzification membership functions is shown in Figure 3.22.

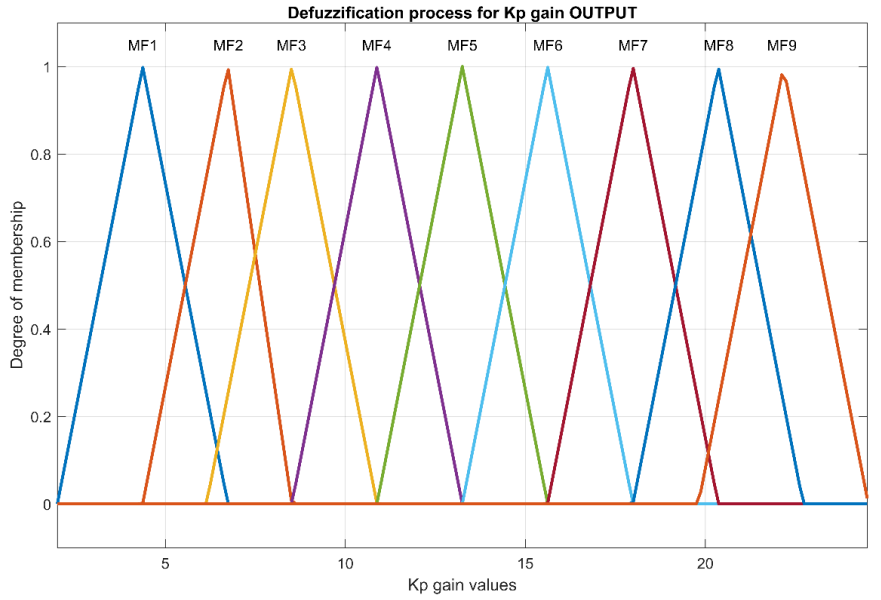


Figure 3.22: Membership functions of defuzzification process for $\boldsymbol{K_p}$ gain values

After we introduced the fuzzification and defuzzification processes, a connection between them should be established by Fuzzy rules representation. Hence, we have one input and one output which will be arranged with a set of rules. The input is the disturbance value; thus, if a strong disturbance is presented, a Fuzzy controller should provide a small K_p gain value, and if no disturbance is presented, then Fuzzy controller should provide a large K_p value. The noticeable issue here is the inverse relation between the input and output; thus, the rules will solve this issue as follows:

Table 3.6: Fuzzy control rules for $\boldsymbol{K_p}$ gain values

1. If (input1 is MF1) then (output1 is MF9) (1)
2. If (input1 is MF2) then (output1 is MF8) (1)
3. If (input1 is MF3) then (output1 is MF7) (1)
4. If (input1 is MF4) then (output1 is MF6) (1)
5. If (input1 is MF5) then (output1 is MF5) (1)
6. If (input1 is MF6) then (output1 is MF4) (1)
7. If (input1 is MF7) then (output1 is MF3) (1)
8. If (input1 is MF8) then (output1 is MF2) (1)
9. If (input1 is MF9) then (output1 is MF1) (1)

For Fuzzy rules inference method, we considered Mamdani method, and for defuzzification we considered Center of Gravity method. The result of using Fuzzy controller is to provide the optimum K_p gain values based on disturbance input is shown in Figure 3.23.

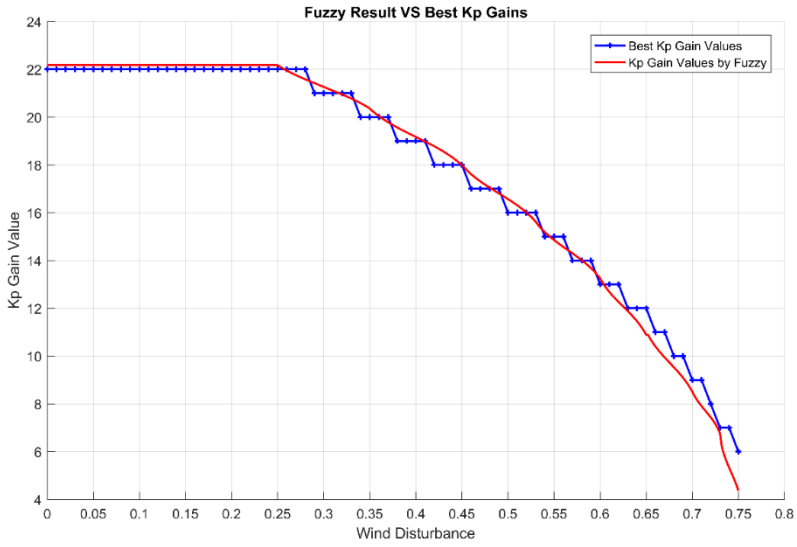


Figure 3.23: Best $\boldsymbol{K_p}$ gain value vs disturbance using Fuzzy controller

Compared to the desired curve, the result of Fuzzy controller shown in Figure 3.23 is reasonable and can be considered for being integrated with z-controller.

The same process is used to obtain the optimum K_i gain values. Fuzzification and defuzzification memberships are shown in Figure 3.24 and 3.25 respectively.

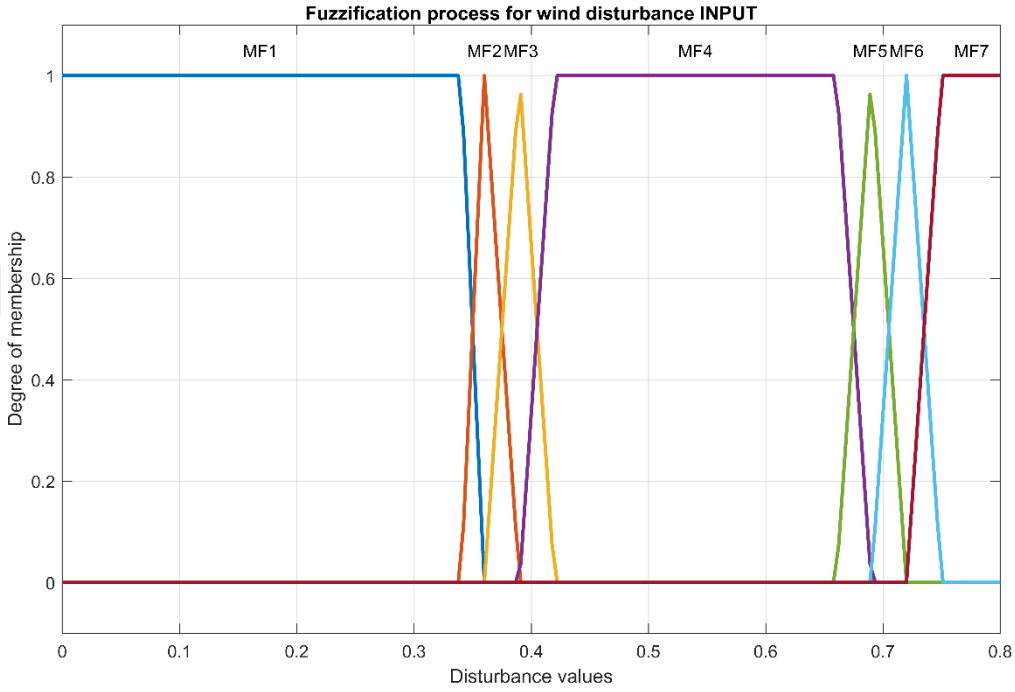


Figure 3.24: Membership functions of fuzzification process for $\boldsymbol{K_i}$ gain values

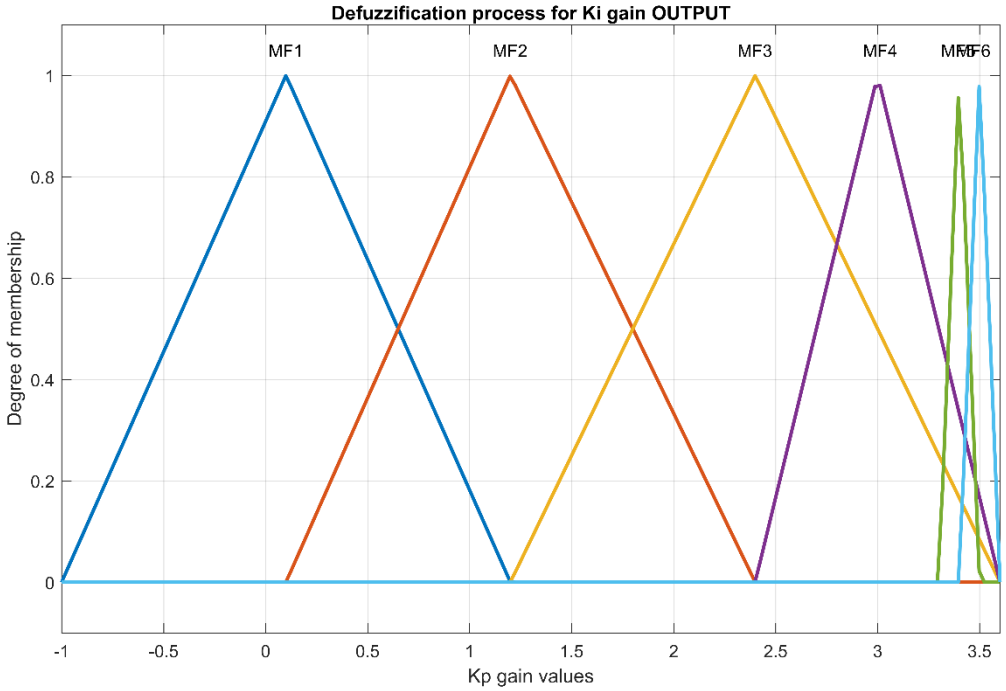


Figure 3.25: Membership functions of defuzzification process for $\boldsymbol{K_i}$ gain values

For K_i gain parameter, the relation between the input as a disturbance and the output is a proportional relation. As the error increases, the value of the K_i gain will increase to override the long-term effect of the disturbance, since the integral term is responsible for steady state error. Thus, fuzzy rules will arrange the proportional relation between the input and the output as follows:

Table 3.7: Fuzzy control rules for $\boldsymbol{K_i}$ gain values

1. If (input1 is MF1) then (output1 is MF1) (1)
2. If (input1 is MF2) then (output1 is MF2) (1)
3. If (input1 is MF3) then (output1 is MF3) (1)
4. If (input1 is MF4) then (output1 is MF6) (1)
5. If (input1 is MF5) then (output1 is MF4) (1)
6. If (input1 is MF6) then (output1 is MF5) (1)
7. If (input1 is MF7) then (output1 is MF6) (1)

Figure 3.26 shows the effect of using the Fuzzy controller to provide the optimum k_i gain values based on input disturbance.

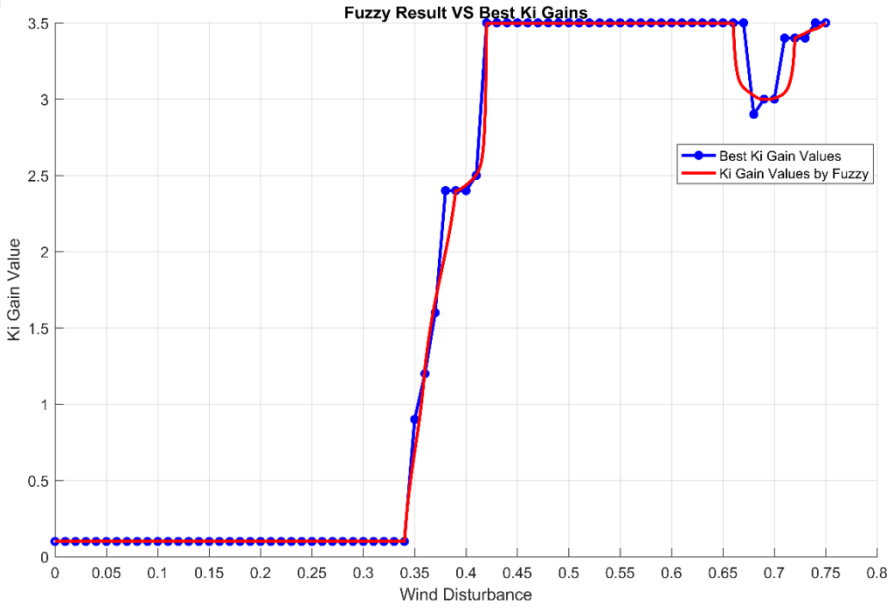


Figure 3.26: Best $\boldsymbol{K_i}$ gain value vs disturbance using Fuzzy controller

We integrated fuzzy controller with PID z-controller in *Simulink* software, the input of Fuzzy controller is the maximum disturbance of (x, y, z) axes. The modified adaptive z-controller is shown in Figure 3.27.

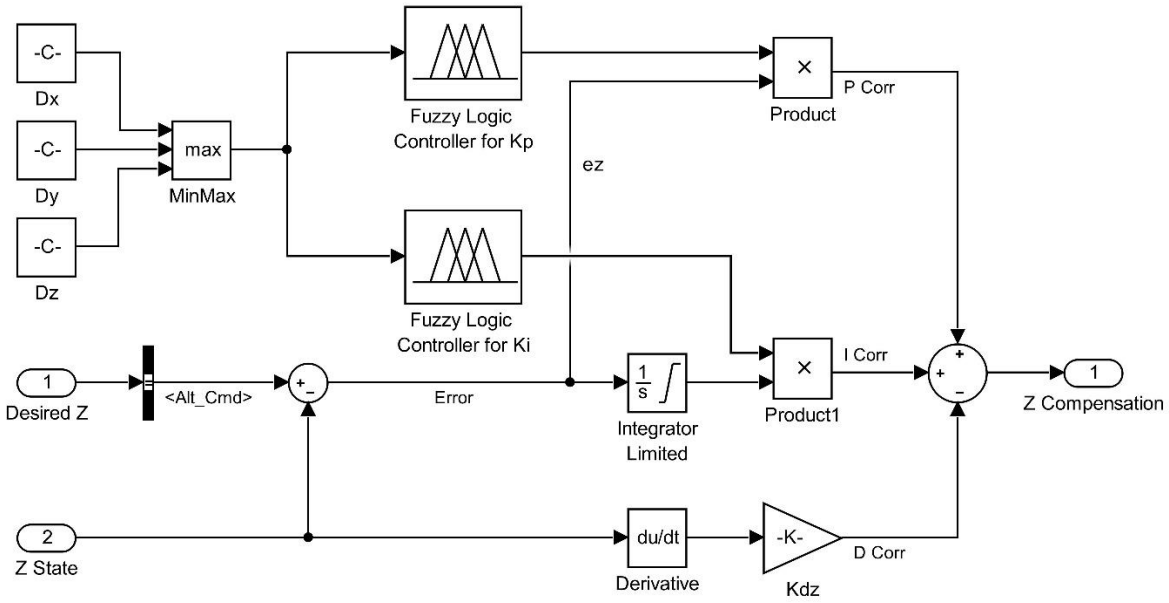


Figure 3.27: Adaptive Fuzzy-PID controller for altitude control

Adaptive controller can also have optimum response in the absence of wind disturbances and robust response in maximum wind disturbances as well as best response between no wind and maximum wind. The adaptive Fuzzy-PID controller performed very well for various disturbance values. Now we're going to analyze another approach “Neural Networks” to see which system manages to better dampen the disturbances.

3.8. NEURAL NETWORKS DESIGN

Neural Network (NN) is one of artificial intelligence's best approaches because of its ability to learn from data collection. As described in the previous chapter, NN procedure is to do training on a set of pre-collected data, learning from example, and then testing different data to check the wellness of recognizing new data based on the training phase. NN can be used to recognize images, sounds, and for other applications. Here we use NN to adapt the best predefined z-controller curves for K_p and K_i parameters, similar to what we did with Fuzzy controller. The object of using NN structure is to compare it with Fuzzy controller outcomes. The input of the NN is disturbance value, and the output is K_p or K_i parameter values. Thus, the input layer will contain one node, and the output layer will also be one node. The hidden layers are the key of reaching the optimum result. First, we tried to construct the network with one hidden layer with different node numbers, but it didn’t give us the desired performance. thus, we increased the number of hidden layers. By trial and error method, we reached a reasonable construction of

the NN as follows: one node for the input layer, two hidden layers each layer contains six nodes, and one node for the output layer. The construction of the NN is shown in Figure 3.28.

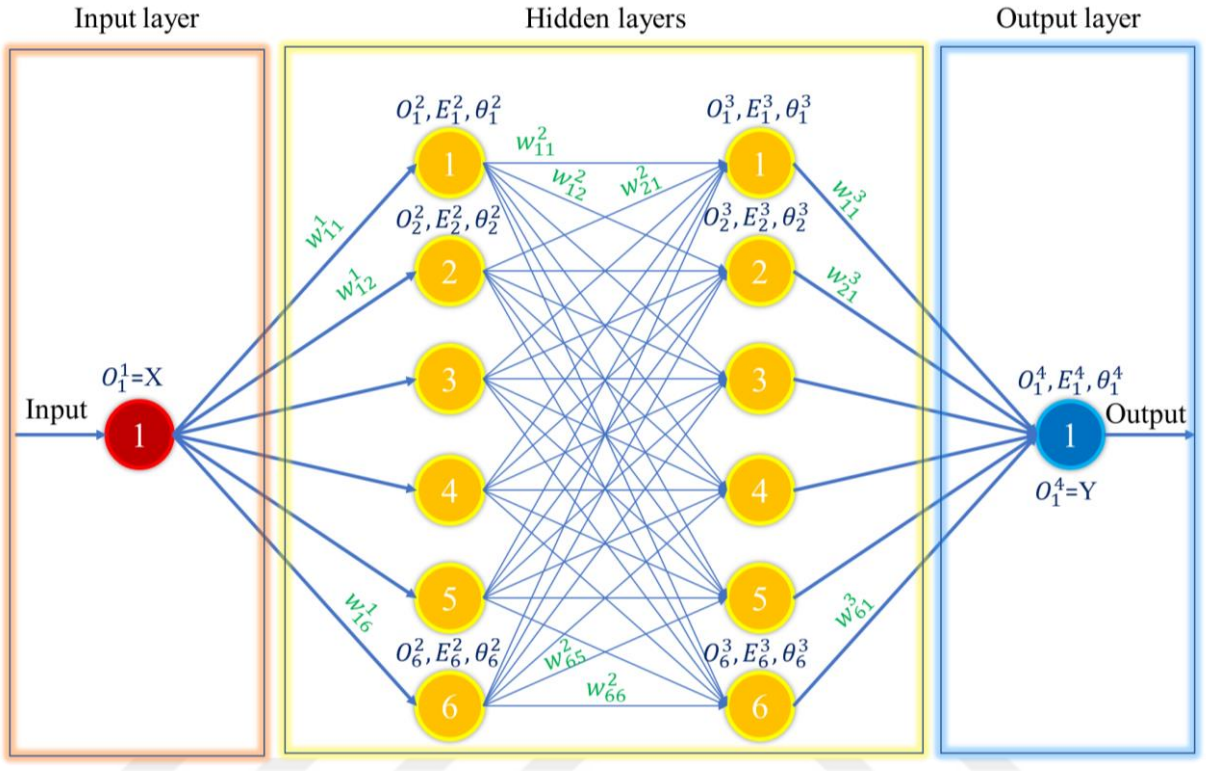


Figure 3.28: Neural Network controller internal structure

All the relative equations for the above structure are explained in the previous chapter from Equation 2.28 to 2.32. The correction rate μ is chosen to be 0.25, the number of iterations is 10000 to reach optimum solution; therefore, after 1000 iterations, NN reached 7% steady state error, and after 9000 iterations, it reached 3% steady state error with Integral Squared Error of 0.8. Of course, 1000 iterations result can be acceptable, but to obtain more precise solutions, we increased the number of iterations, Figure 3.29 shows the curve of error vs. iteration numbers. Hence, the result of training process can be summarized in two important parameters: weights and thresholds. The weight values between all nodes are expressed in three-dimensional matrix, the rows are for the input nodes, the columns are for the output nodes, the pages vector is for the layer number, and by this representation we can obtain the trained weight values. The threshold values are for each node in the network. The results of neural network controllers for K_p and K_i parameter values versus desired values are shown in Figure 3.30 and 3.31 respectively.

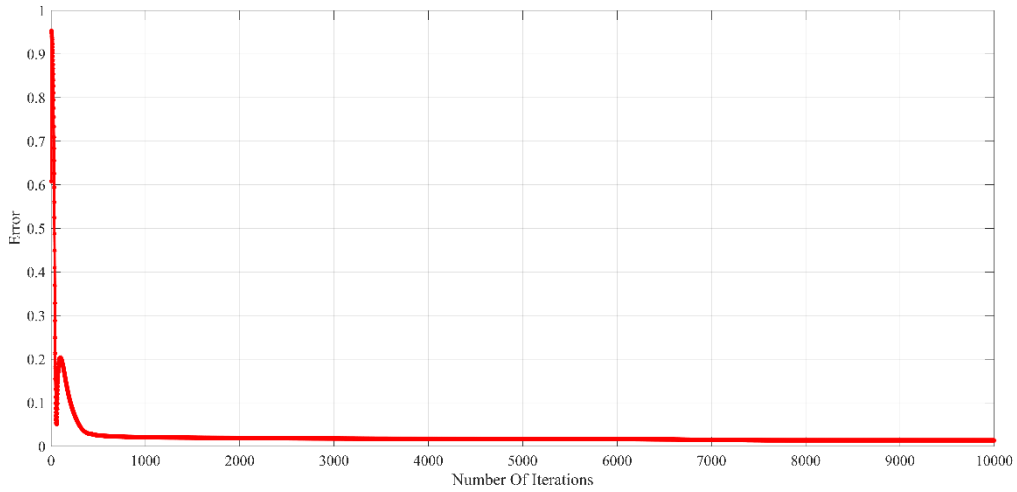


Figure 3.29: Neural Network training error vs number of iterations

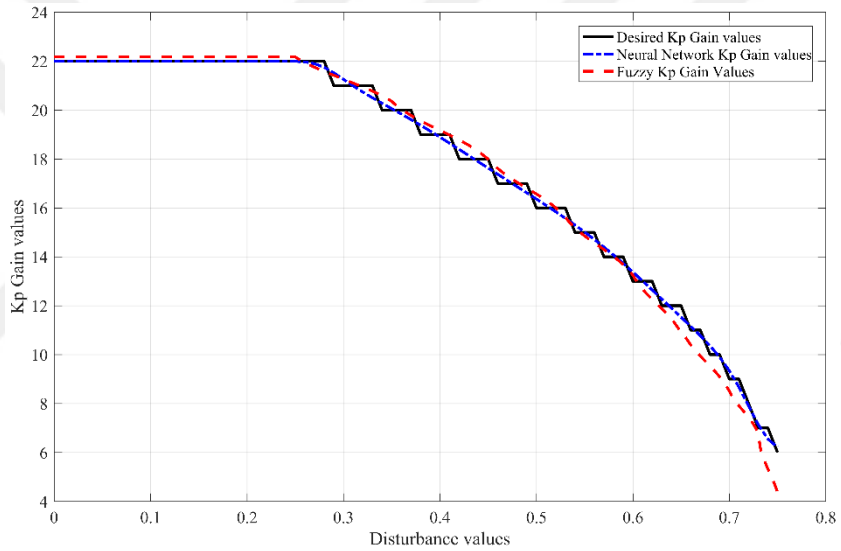


Figure 3.30: Neural Network vs Fuzzy results for K_p Gain values

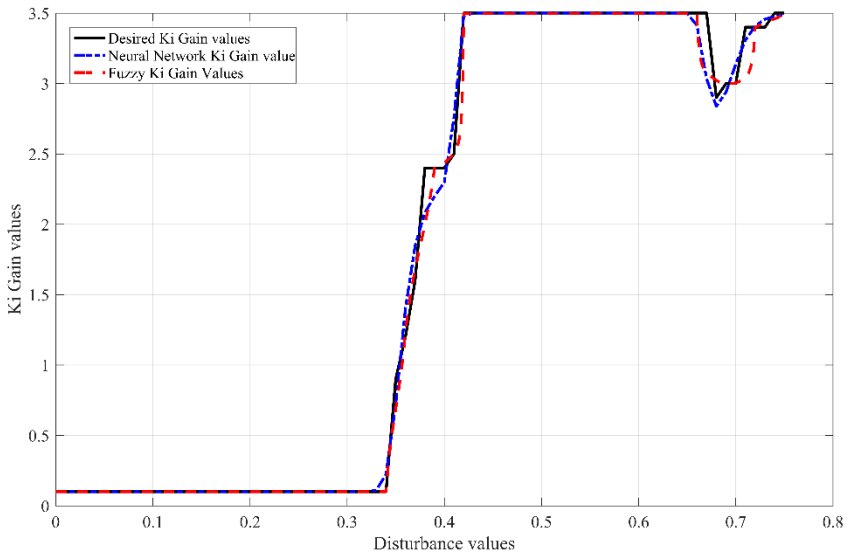


Figure 3.31: Neural Network vs Fuzzy results for K_i Gain values

The results of both Neural Networks and Fuzzy controllers are acceptable and gave the desired response. But Neural Network shows the ability to adapt the desired curve more accurately than Fuzzy controller. This is because of the nature of NN construction and training process.

To complete the design of the Neural Network controller, we integrated it with PID controller for altitude controlling in *Simulink* to be Adaptive NN-PID controller as shown in Figure 3.32.

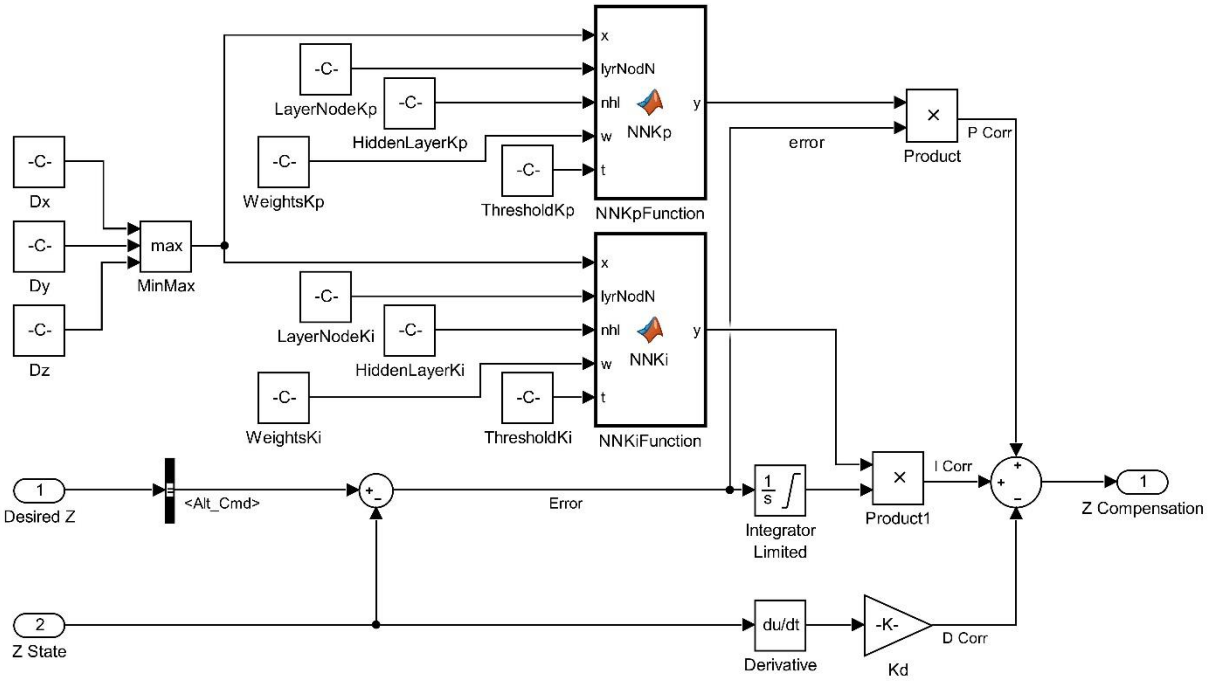


Figure 3.32: Adaptive NN-PID controller for altitude control

The controller gave the desired results; the result and response of the controller over the quad-rotor flight simulation is presented in next/results chapter.

3.9. DISCRETE CONTROLLER DESIGN

After we designed and constructed the controller, we need to prepare it to be implemented in microcontroller devices. Therefore, microcontroller devices are not continuously processing; that's why we need to design and check the system with discrete controllers. Although the microcontroller devices do a very fast processing nowadays, it will also still affect the performance. We changed the continuous processing controllers in three steps to implement discrete controllers and test their responses. The first step, we assumed the multi-stage controller as a black box controller that contains 6 PID controllers. The black box controller has the following inputs and outputs: desired inputs (x_d, y_d, z_d, ψ_d), state feedback

inputs$(x_f,\ y_f,\ z_f,\ \varphi_f,\ \theta_f,\ \psi_f, u, v, w, p, q, r)$, and motor command signals as outputs$(\omega_1, \omega_2, \omega_3, \omega_4)$. In the second step, we performed discretization for the inputs and outputs of the black box controller as shown in Figure 3.33. The sample time is 2 milliseconds; this processor speed can be considered to be very slow, but we preferred to examine the response with low frequency to check the robustness of the controller. The results show some oscillations about the desired path; the oscillations were about 15 cm. Thus, we increased the sampling time to be 20 microseconds and decreased the oscillations to be less than 1 cm.

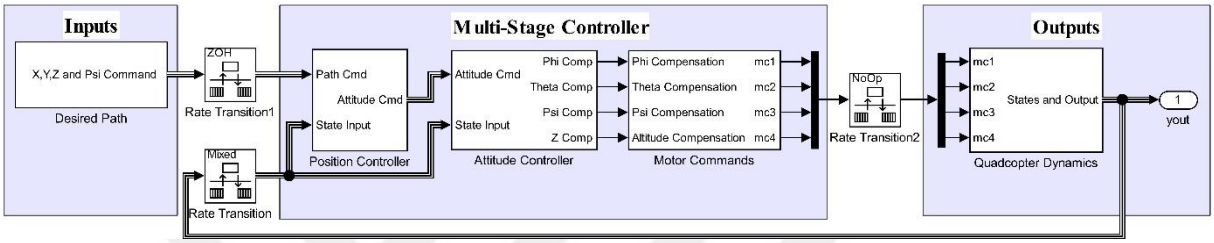


Figure 3.33: Discretization of controller's inputs and outputs

The third step is to transform all controllers in a discrete form; thus, all six PID controllers represented in z-domain will be transformed by Z-transform. All related transform equations are represented in Equations 1.36 to 1.41. A sample controller represented in Z-domain is shown in Figure 3.34.

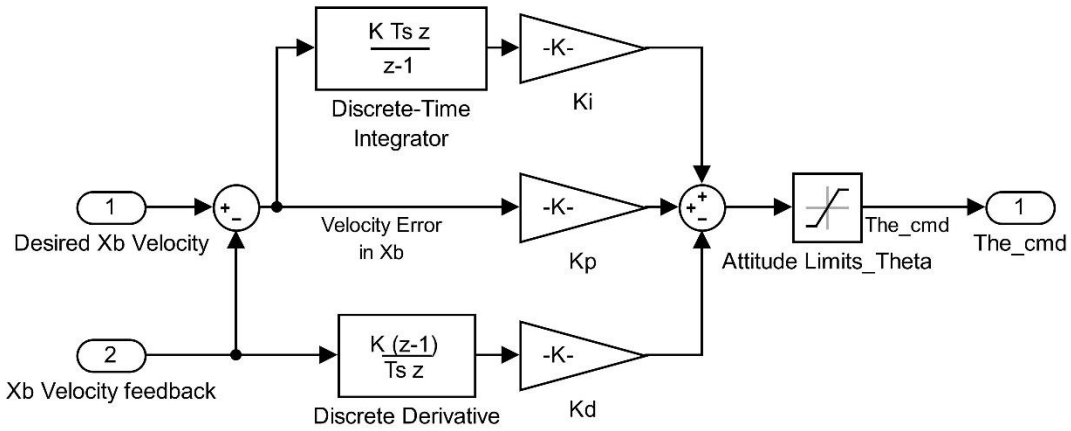


Figure 3.34: X-axis discrete PID controller

Discrete controller results are discussed in next chapter.

4. RESULTS

The parameters of the quad-rotor system analyzed in this thesis were based on practical tests and observations from a previous study done by Hartman et. al. (2014). We chose to make our simulation based on real practical values to be ready to implement our controller in real projects, and to have results approximately near the real outcomes. All programs, codes, and results are included in appendix 4. In this chapter, the results of optimization process, controller's response, and command references are presented. The simulations included the following topics:

1- Optimization process by AGA for multi-stage PID controllers.

2- The result of relating body frame with a Non-Inertial Frame of Reference.

3- The effect of Linear wind disturbances where a comparison was done between controller's response before disturbance, after disturbance, and after re-optimization by AGA for PID parameters with linear wind disturbances.

4- The effect of angular wind disturbances; comparison between controller's response with and without disturbance before re-optimizing parameters by AGA, Obtaining stability by AGA, and a comparison between controller's response before and after re-optimization by AGA.

5- Two optimization results, Optimum parameters and Robust parameters.

6- A comparison between Adaptive Fuzzy-PID controller response and Adaptive NN-PID controller response.

7- Examining the response of Adaptive Discrete controller.

4.1. ADAPTIVE GENETIC ALGORITHM OPTIMIZATION RESULTS

In the beginning, we started testing our controller with PID parameters that were obtained from a previous study by Hartman et. Al. (2014), then we went to run our Adaptive Genetic Algorithm to optimize the PID parameters. The optimization process attempted to optimize all

six parameters of the PID controllers simultaneously to achieve an optimum overall result based on a predefined path.

We chose to do the optimization simultaneously for all controllers because we assumed that each controller influences the other. The results of the optimization process for $(\varphi, \theta, \psi, x, y, z)$ controllers are shown in the following figures.

A comparison between the desired path and simulated path before and after AGA optimization was done. Figure 4.1 shows the results in 2-D for x and y axes; the results show that the quad-rotor missed the path after a sharp turning by 1.77 meters before trying to recover and follow the path again but after the AGA optimization quad-rotor missed the path with only 0.76 meters. After optimization, the quad-rotor can retrieve to the desired position faster with better performance and less power consumption. Although the improvement in response will significantly affect the command and results of Phi and Theta angles as shown in Figure 4.3 and 4.4, also, it will increase the stability of the system.

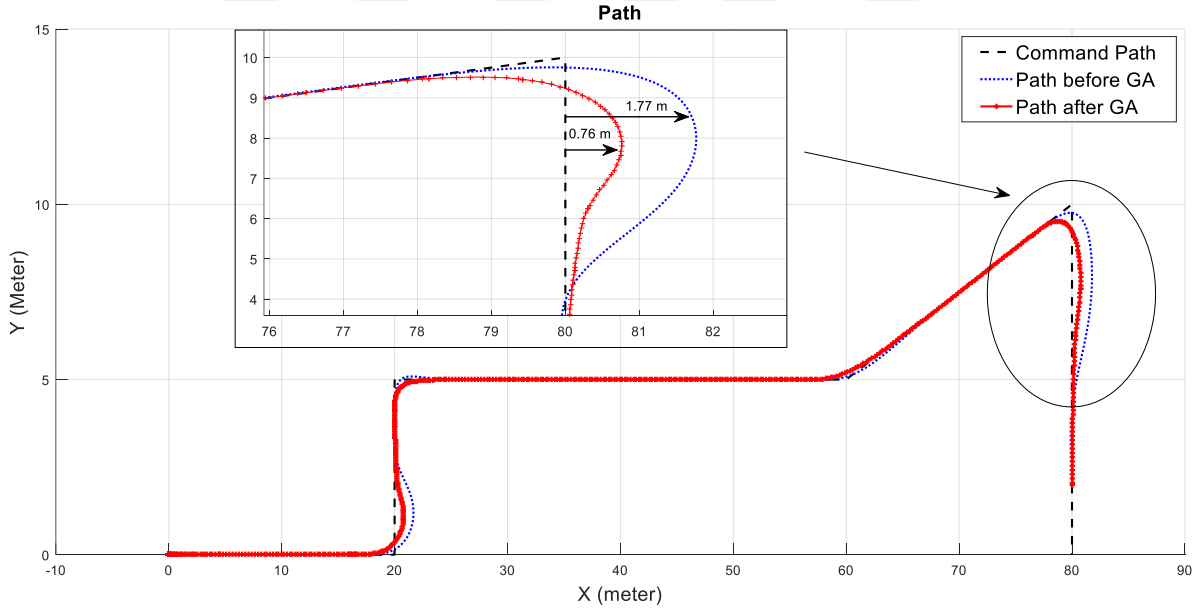


Figure 4.1: Flight path represented in x-axis and y-axis

Figure 4.2 shows the response of Z-axis controller response; great improvement can be seen in overshoot and in transient response as well as the overall performance and power consumption. It is nearly zero overshoot; this result is clear because the command signal for take-off is a sharp step signal over Z-axis without any movement in other directions.

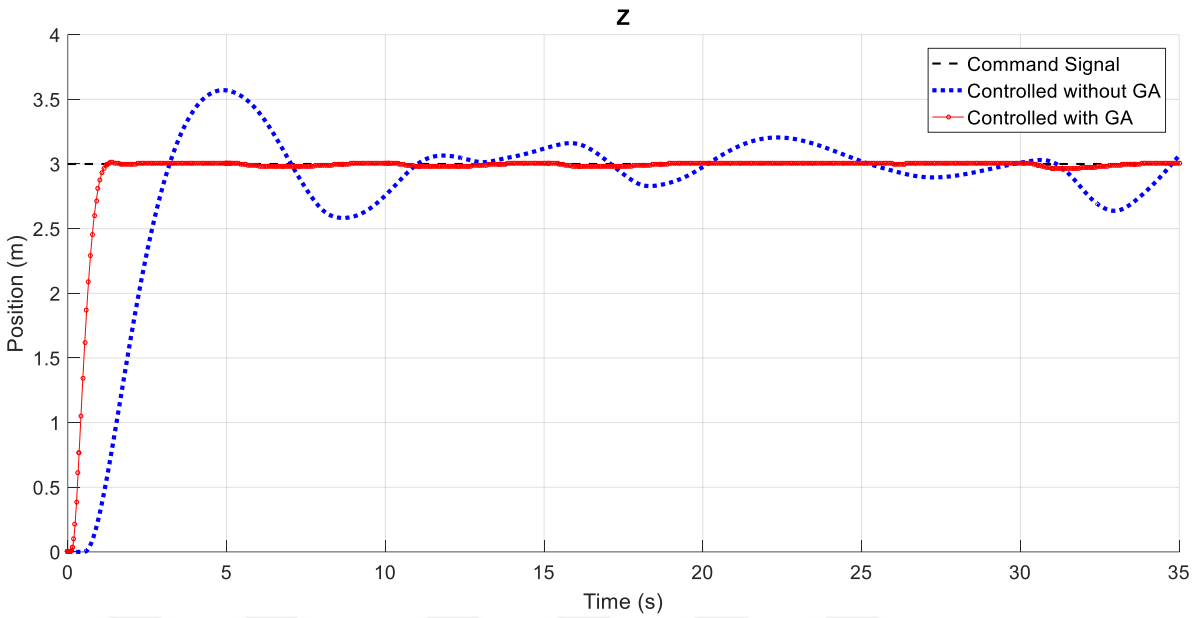


Figure 4.2: Z-axis controller response

Optimization process for Z-axis controller can be considered easier than other axes because in this interval, there is only one movement in one axis. However, the next steps affect the altitude, but the controller is robust and can recover very fast. This result was reached because in this research the optimization was done for all PID parameters simultaneously.

The output of the Y-axis controller has actually improved the command of Phi angle. Figure 4.3 shows controller commands and responses for Phi angle. Sample interval was taken between the seconds 15 and 17 of flight simulation to show the differences between the command signals and response signals before and after AGA. In this interval, to reach the desired position, the controller gave a command to the Phi angle to turn with 6.6 degrees, but after AGA optimization, the command signal has decreased with about 30% to be 4.8 degrees. So, the quad-rotor doesn't need to turn Phi angle with a big degree to reach the desired position that provides more stability and less power consumption. The maximum delay between command and controlled signal before AGA was 0.6 second; in some cases, this delay can affect the stability, but after AGA, the maximum delay between command and control signal becomes 0.14 second which means 75% improvement.

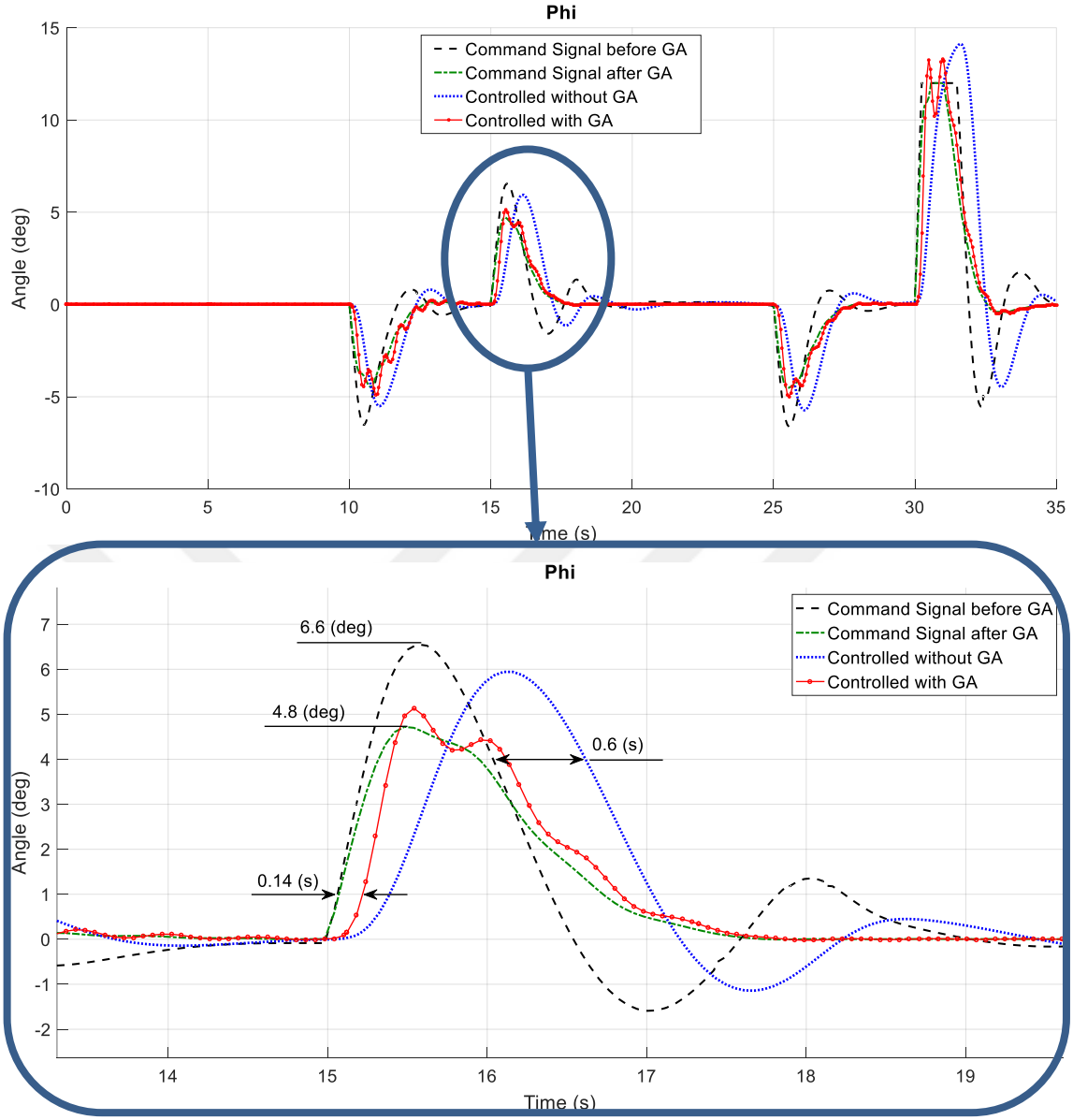


Figure 4.3: Phi angle controllers' commands and responses

Figure 4.4 shows controller commands and responses for Theta angle; a scoped interval was taken to study the characteristics of commands and response signals. At the fifth second of flight simulation, a sharp step command was given by the X-axis controller trying to follow the desired position. The command signal has reached its predefined limit by 12 degrees. This action acts as a step input to Theta angle, so from this step system characteristics can be examined. Before the optimization process, rising time was t_r = 0.84 s, while AGA optimization improved rise time to t_r = 0.28 s. It is clear that, prior to optimization; the response was very slow, as it could not catch the command signal. The same improvement can be seen for the overshoot as it was improved from P.O = 18.5% to P.O = 5.2%.

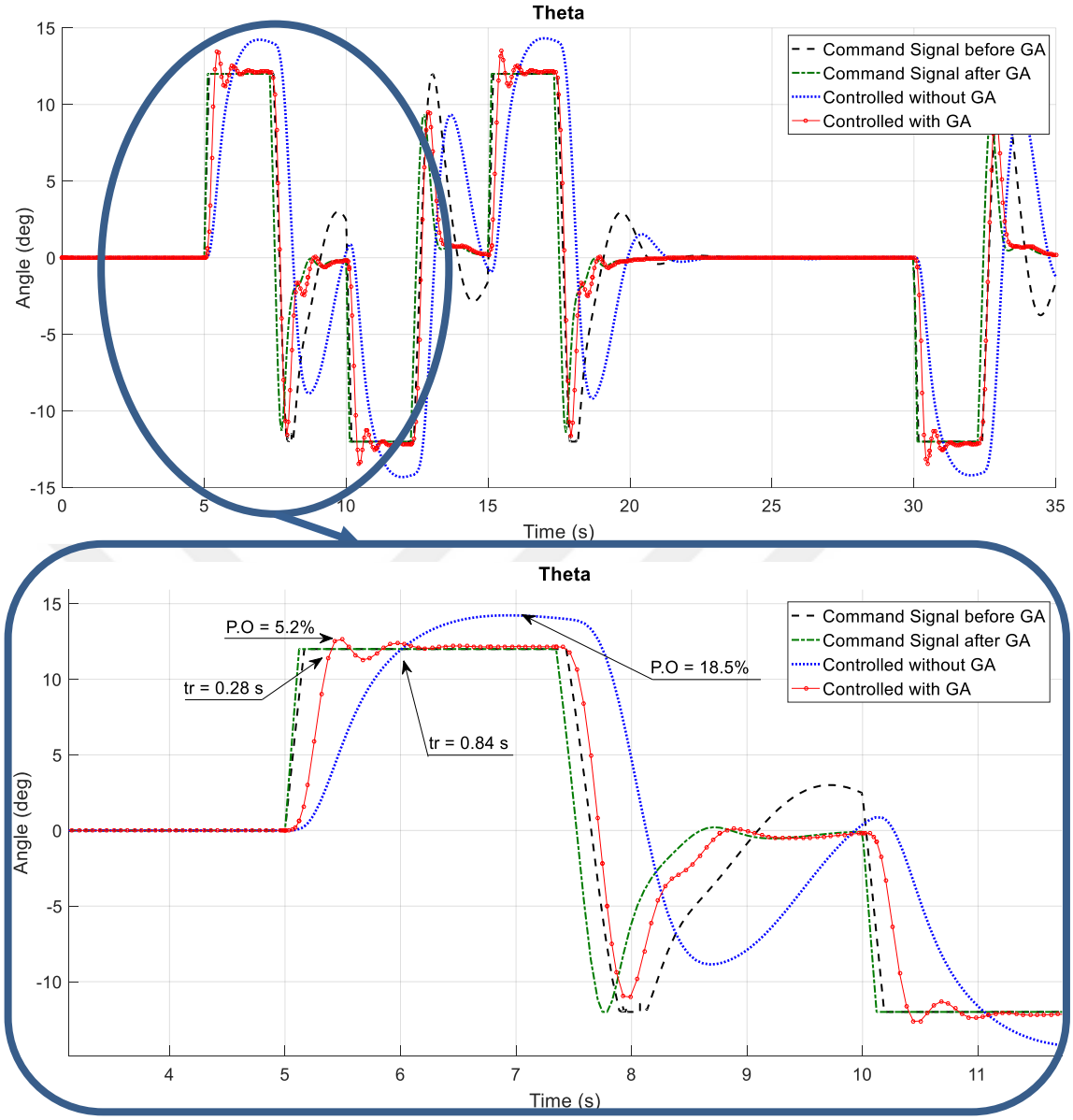


Figure 4.4: Theta angle controllers' commands and responses

For Psi angle, the command signal over the flight interval was zero; therefore, it has been affected by other coordinates, as shown in Figure 4.5. In this case, the controller compensates for small variations. The response of Psi controller before optimization process shows that there is a possibility of entering in an unstable region due to continuous increasing in oscillations. However, AGA optimization successfully damped those oscillations to give robust response.

The summation of thrust consumed without using AGA was around $1.26*10^5$; however, with AGA, it became $1.05*10^5$ for the same path command. As it can be seen, the AGA provides 16.2% improvement in overall power consumption.

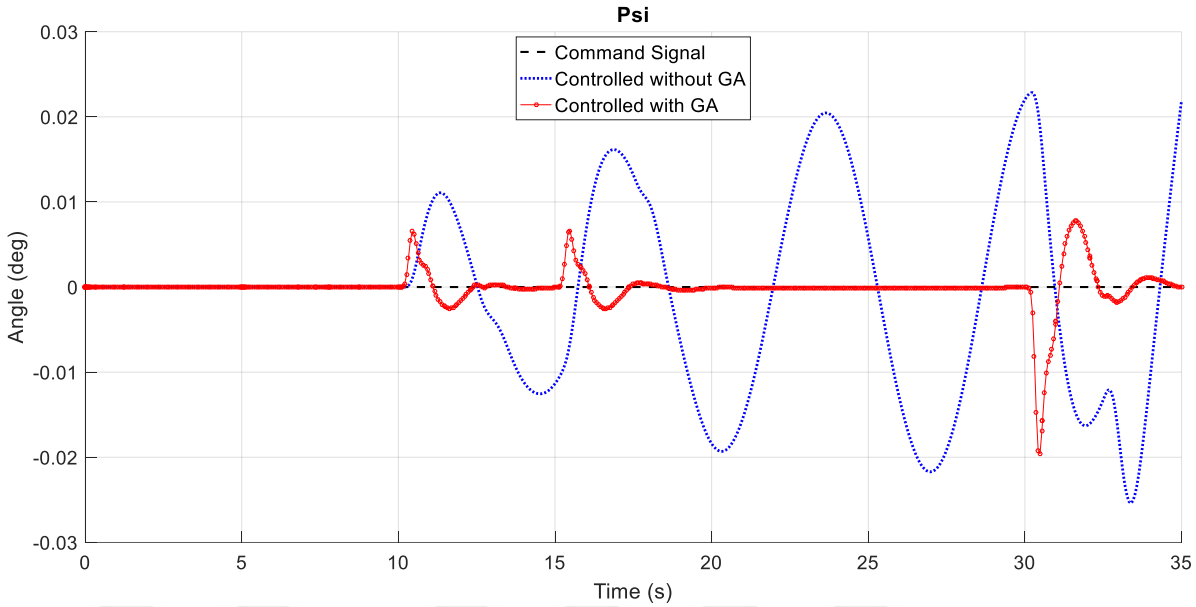


Figure 4.5: Psi angle controllers' command and responses

4.2. NON-INERTIAL OF REFERENCE REPRESENTATION

After we successfully made a relationship between quad-rotor and a moving body frame in system equations, we simulated a flight to examine the results of a relative motion.

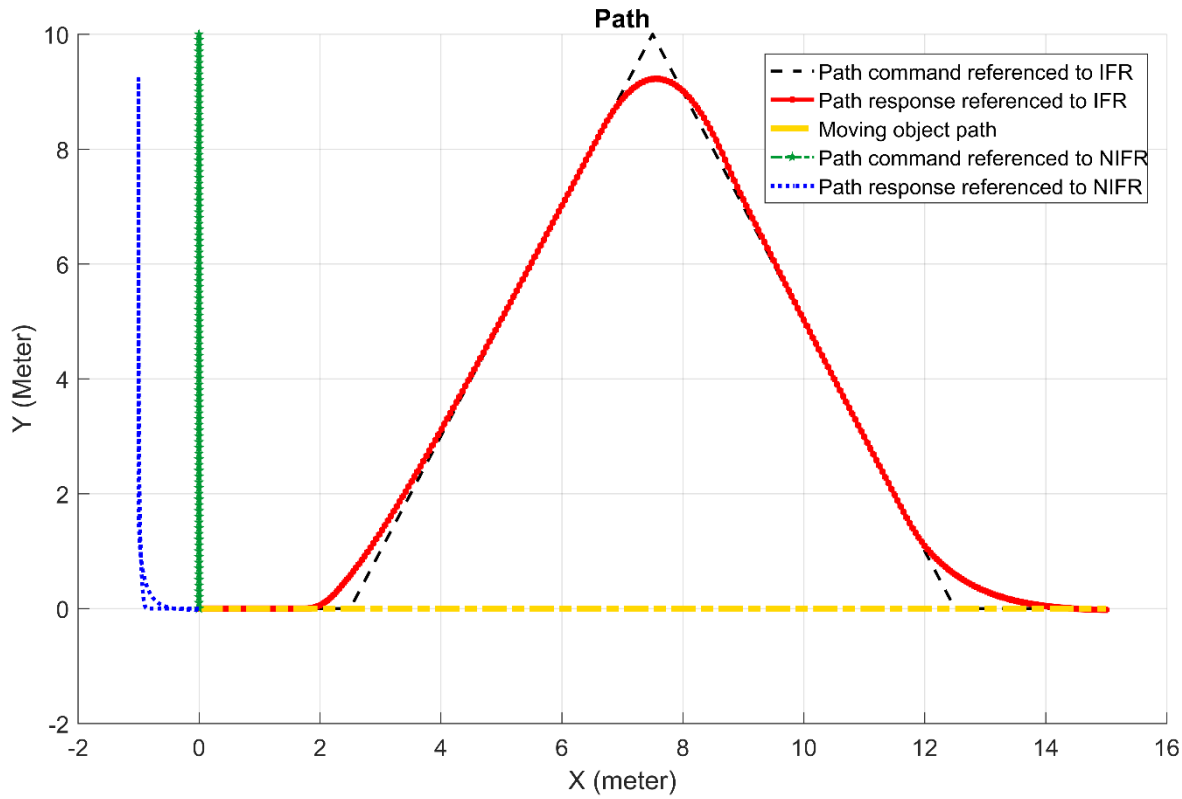


Figure 4.6: The result of relating quad-rotor with moving frame

The simulation was based on the following scenario: we related the quad-rotor to a moving body as NIFR. This moving body was considered as a ship on the sea; the ship moves in one direction over x-axis to a distance of 15 meters as shown in Figure 4.6. The command signal to the quad-rotor was to move 10 meters in y-axis direction without any movement in x-axis referenced to the ship. Thus, the quad-rotor will move in x-axis because it should follow the reference 'ship'. It should also move in y-axis following the desired movement referenced to the ship. Therefore, the movement of the quad-rotor can be represented in two cases: the first case is referenced to IFR, the command signal is the black line color, and the response is the red line color, The second case is referenced to NIFR, the command signal is the green line color and the response is the blue line color. However, there is a difference between the command and response with NIFR due to the delayed position feedback of the ship and because the ship is moving and the quad-rotor is following it. If the ship stopped, there will be no difference in position between quad-rotor and ship.

4.3. FLIGHT SIMULATION DURING LINEAR WIND DISTURBANCE

Wind disturbance can be divided into two categories: one can be considered as a linear wind disturbance; the other one is angular wind disturbance. Linear wind disturbance is an external **Force** applied to quad-rotor; thus, this disturbance will try to deviate the quad-rotor from its planned path. Figure 4.7 shows the response of the quad-rotor after applying a linear wind disturbance in the direction of x-axis and y-axis equally. From the figure, we can see that the quad-rotor deviates strongly in the direction of the wind, and that's because of more than one reason. The first reason is that we are not using well-planned commands to move the quad-rotor in a specific path. We used timeseries command signal from *Matlab* to implement quad-rotor commands according to a pre-defined position over specific time intervals. As an example, if we want to move forward in the x-axis from point 0 to point 20 in 10 seconds, a command signal will be delivered from *Timeseries* function giving the desired position, but it doesn't provide the desired velocity, acceleration, and deceleration. Thus, the desired velocity and acceleration will be supposed maximum. However, when we reach the target position, the quad-rotor will obtain the maximum speed, and the quad-rotor's inertia will prevent it from a sudden stop. The second reason is wind direction, so if the wind is very strong in the direction of the desired position, the quad-rotor speed will increase, especially if the speed is not planned or limited. This can be avoided by using one of path-planning programs.

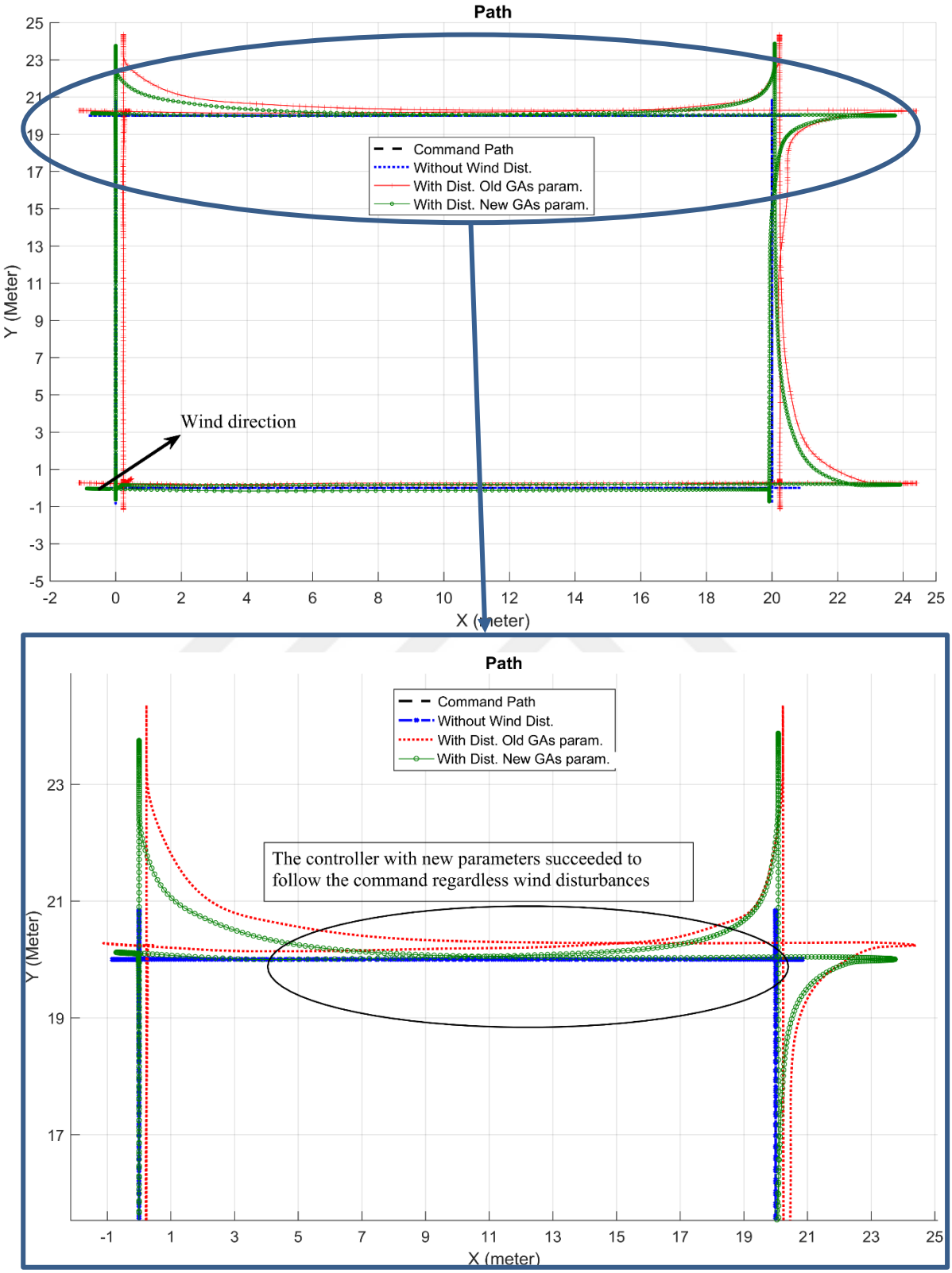


Figure 4.7: Linear wind disturbance effect on quad-rotor planned path

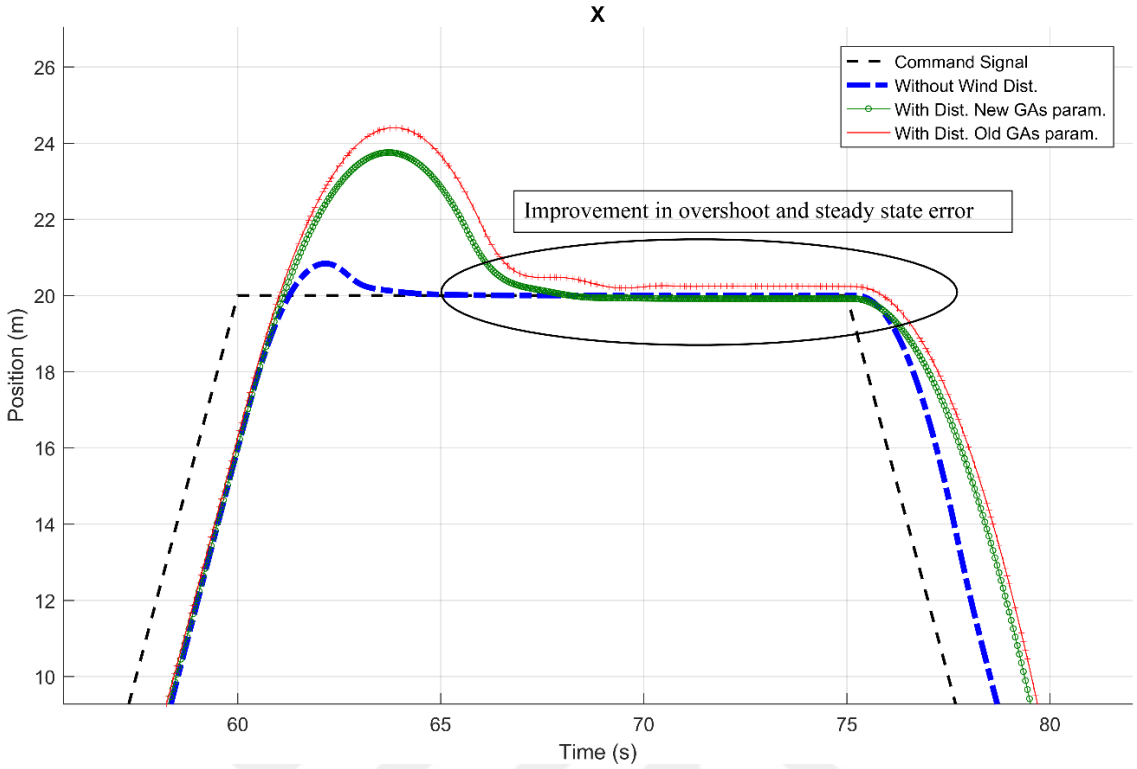


Figure 4.8: Linear wind disturbance effect on x-axis controller

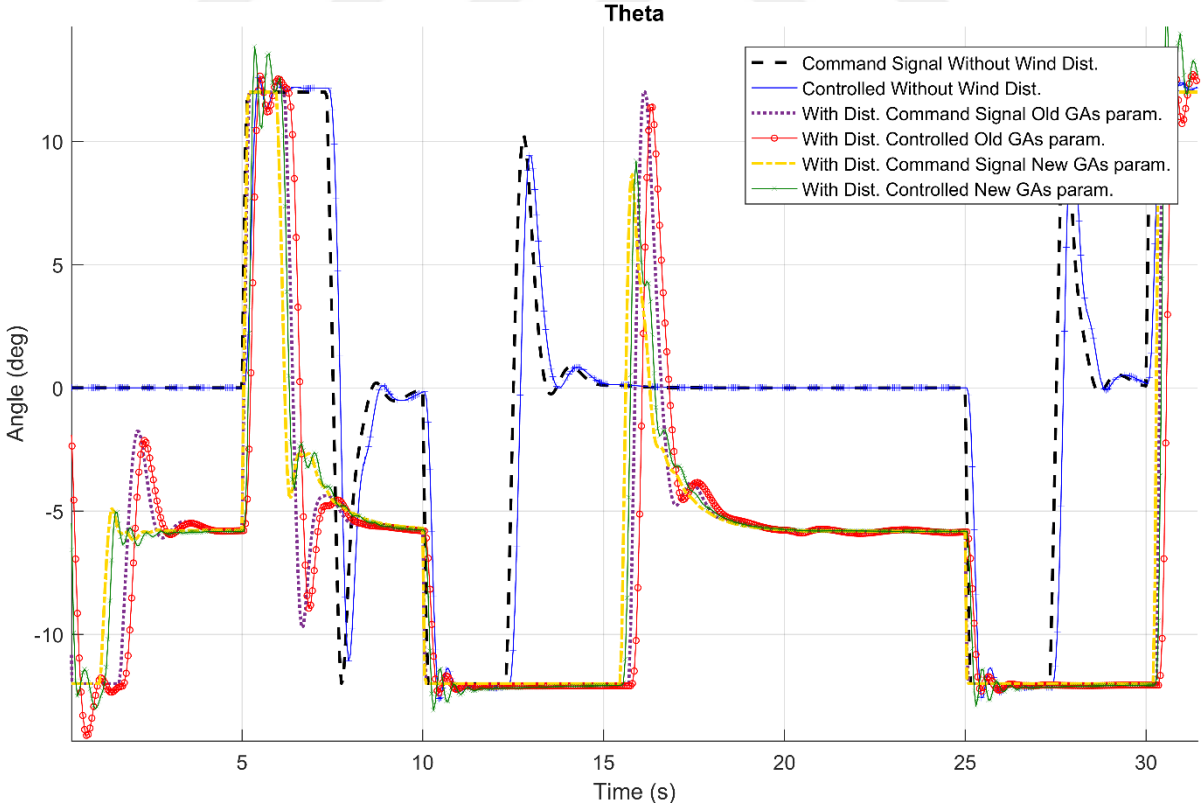


Figure 4.9: Linear wind disturbance effect on Theta angle controller

After we applied linear wind disturbance to quad-rotor simulation, we supposed that a re-optimization process by AGA can improve the performance of the multi-stage PID controllers. AGA optimization process improved the response of x and y axes controllers as shown in Figure 4.8, however, overshoot decreased and the steady state error reached zero. Also, the response of Phi, Theta, and Psi controllers improved, Figure 4.9 shows a decreasing in the overall oscillation for Theta controller response.

4.4. FLIGHT SIMULATION DURING ANGULAR WIND DISTURBANCE

Angular wind disturbance is an external **Torque** applied to quad-rotor; thus, this disturbance will try to deviate the quad-rotor in an angular form. This disturbance can affect the stability much more than the linear disturbance, because it can affect all the six PID controllers at the same time.

4.4.1. Controller Response Before and After Angular Wind Disturbance

A simulation was done to compare between controller response before and after angular wind disturbance. The torque "angular wind disturbance" applied to quad-rotor is the highest torque which can be tolerated by quad-rotor. Maximum angular wind disturbance "maximum torque" was obtained by a lot of trails and a lot of re-optimization process.

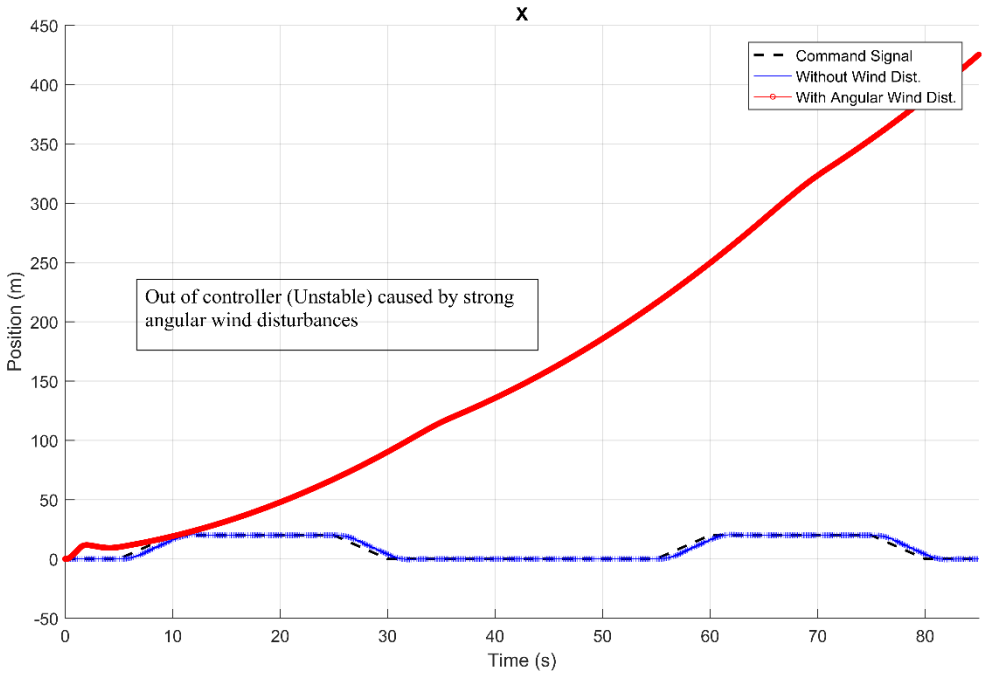


Figure 4.10: Angular wind disturbance effect on x-axis controller

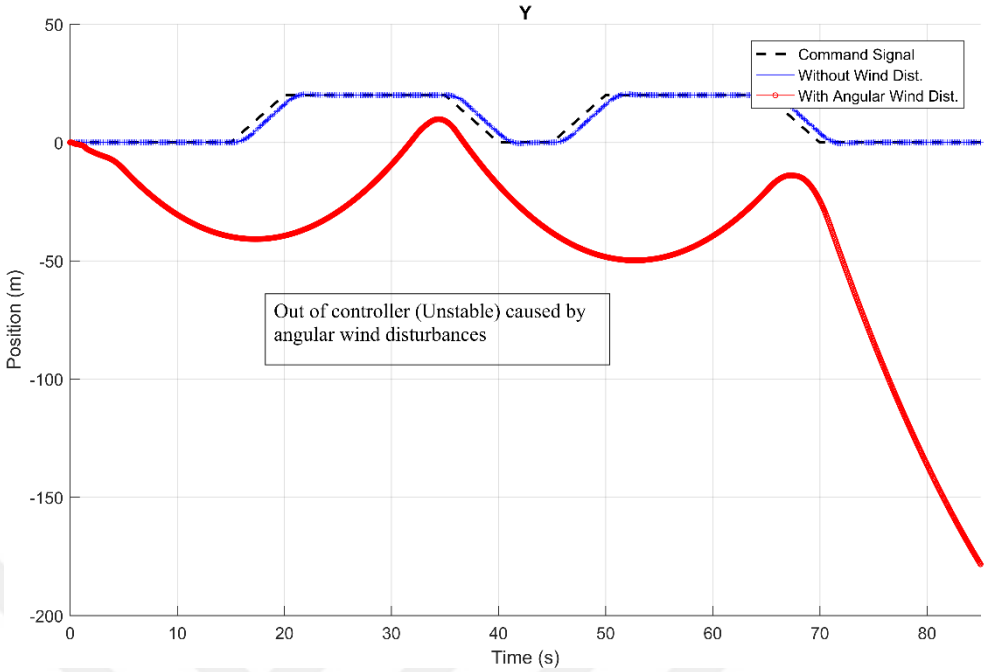


Figure 4.11: Angular wind disturbance effect on x-axis controller

The result of the comparison for x-axis and y-axis are shown in Figures 4.10 and 4.11. The Figures show that the quad-rotor missed the path and became unstable and uncontrollable. Figure 4.12 shows the response of the Psi angle controller. From the figure, we can see that the controller is unable to catch the command signal which causes the quad-rotor to rotate around itself continuously.

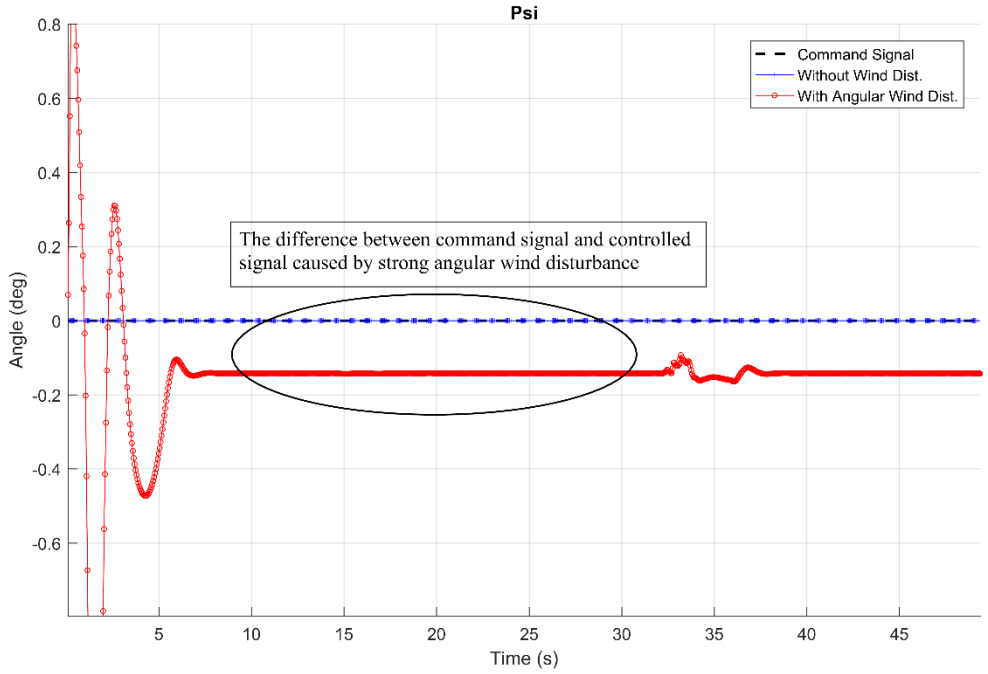


Figure 4.12: Angular wind disturbance effect on Psi angle controller

4.4.2. AGA Optimization Process for Obtaining Stability

To stabilize quad-rotor again, we run the AGA optimization process. In each iteration, we recorded all PID parameters to check which iteration AGA would maintain stability. Detailed values for PID parameter values versus total error are represented in Appendix 3.

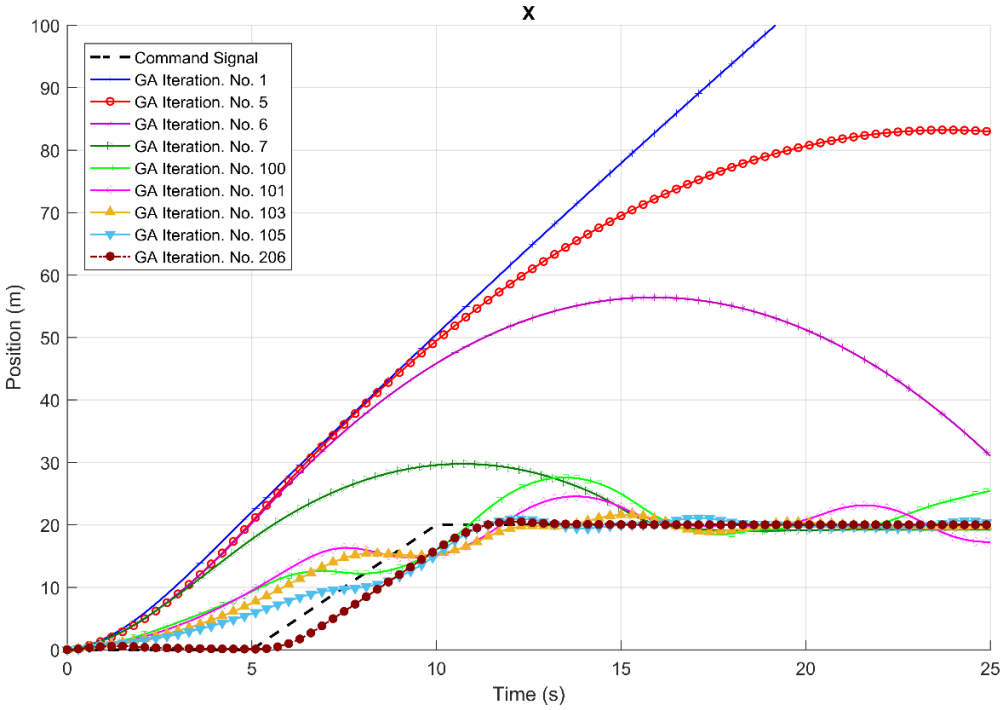


Figure 4.13: AGA iterations to maintain stability for x-axis controller

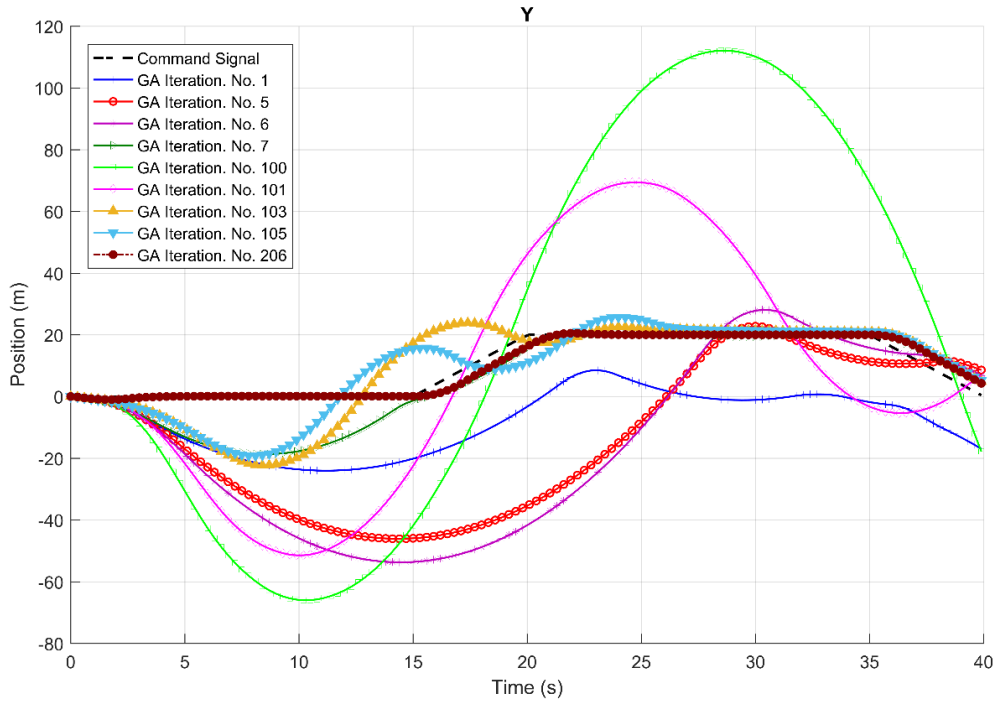


Figure 4.14: AGA iterations to maintain stability for y-axis controller

We have chosen to display in the figures some important critical iteration to provide a clear understanding of the optimization process. Figure 4.13 and 4.14 show controller's response for different iterations. The first iteration presented with blue line shows complete instability, then each iteration of optimization improves system's stability. Following seven iterations, the controller's response began to follow the desired command but still with unsatisfactory results. Stability has continued to be achieved through the optimization process; iteration 105 shows a stable flight but with a large error. After 206 iterations, we reached a stable condition with optimum response and a very small error. We stopped the optimization algorithm at this point because the improvement in the controller response has almost no effect on the flight.

Figures 4.15 and 4.16 show Theta and Psi controller commands and responses. The Figures show that the controller response is improved with each iteration. We added constraints to the controller to improve stability, so the controllers' constraints were to limit the desired angle signals between upper and lower limits. For the Theta angle at iteration 206; the controller achieved optimum response with small reasonable errors. The errors occurred the beginning of the flight before the controller adapted the disturbances. One of the most affected controllers by angular disturbances was the Psi controller, because disturbances always try to turn quad-rotor around itself. The controller successfully damped the oscillations after about 3 seconds.

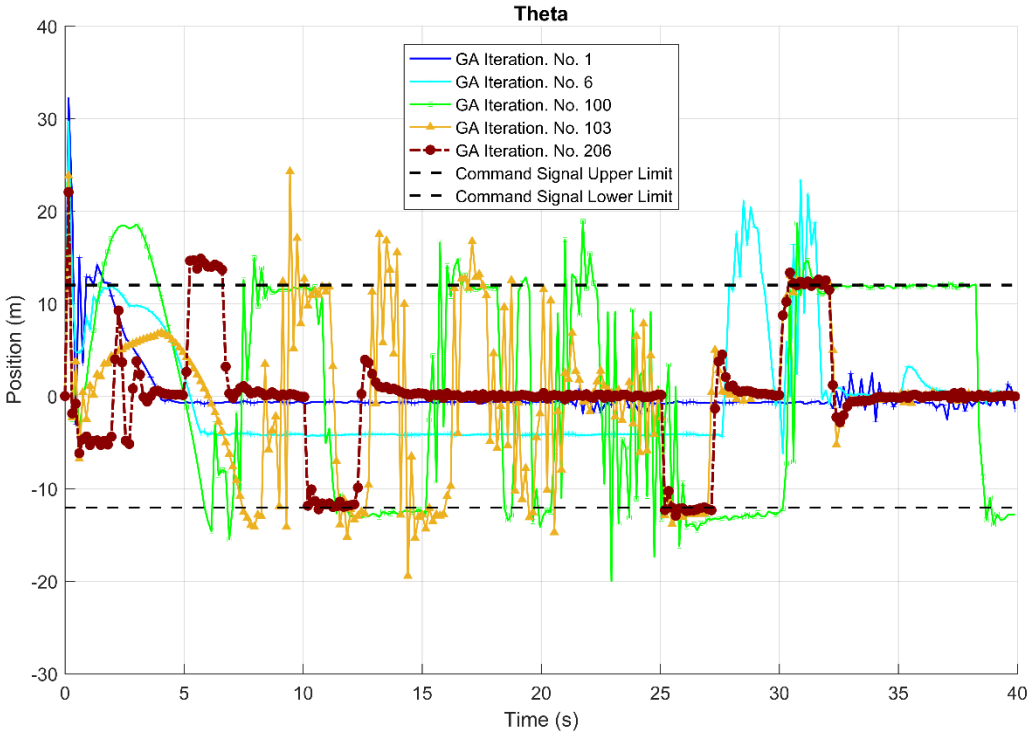


Figure 4.15: AGA iterations to maintain stability for Theta angle controller

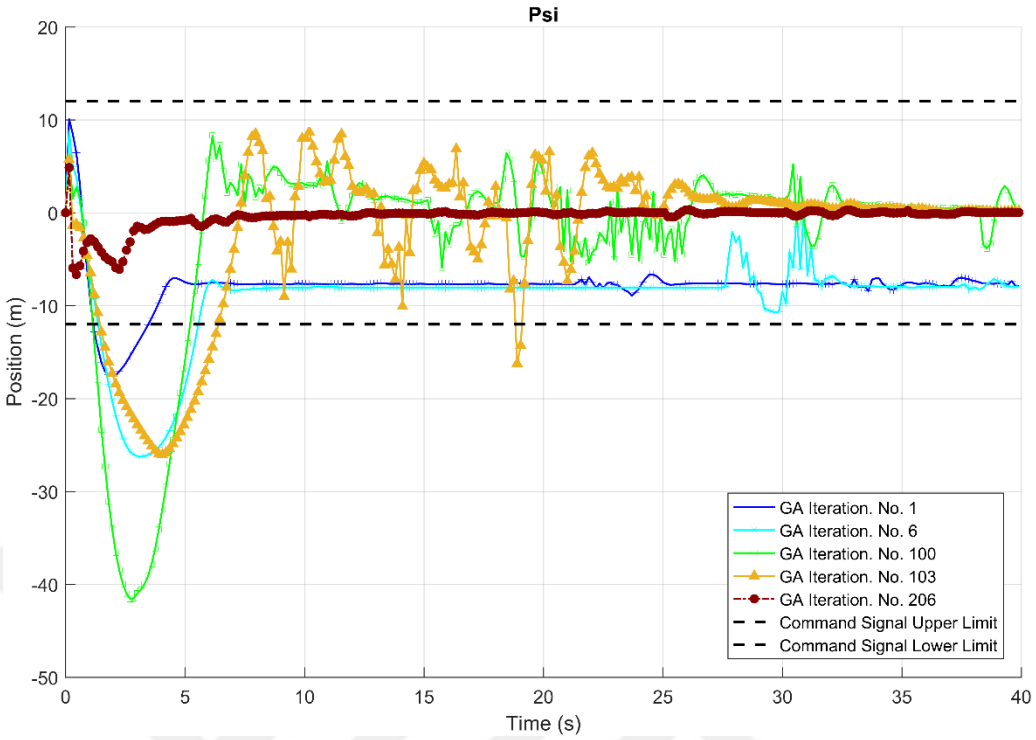


Figure 4.16: AGA iterations to maintain stability for Psi angle controller

Consequently, after several investigations and comparisons between the responses of all controllers, we found out that z-controller is the main controller responsible for the stability.

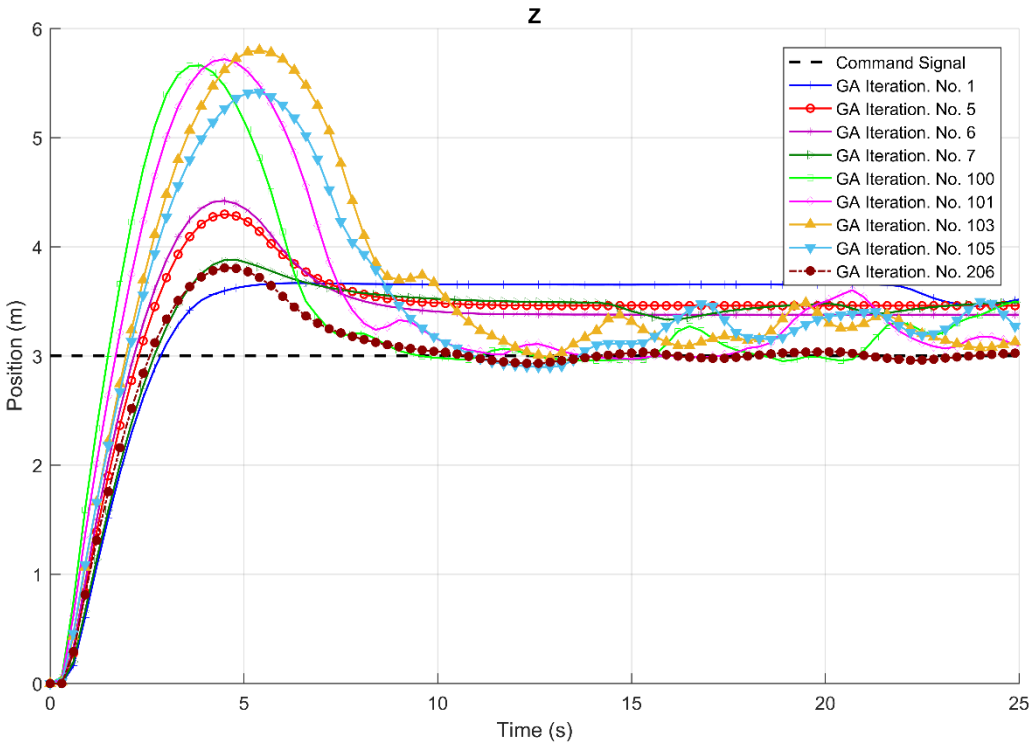


Figure 4.17: AGA iterations to maintain stability for z-axis controller

Figure 4.17 displays AGA iterations for maintaining z-axis controller stability. It is reasonable that the z-axis controller is responsible for flight stability since the z-controller controls the quad-rotor altitude which links all motor commands to this controller. The outcome of the optimization process provides the following: if the gain value of the PID controller of z-axis decreases, system stability increases. So, to dampen the disturbances, the controller provides slow planned moves, but it will lose control if it acts quickly.

4.5. COMPARISON BETWEEN OPTIMUM AND ROBUST PID PARAMETERS

By providing the controllers with robust PID parameters, AGA optimization process maintained quad-rotor stability during strong angular wind disturbances. But what if there were no wind disturbances, do robust PID parameters provide best flight performance? The answer can be seen in Figures 4.18. Z-controller is key to stability; thus, the performance will be tested according to it. We applied these cases: (No-Wind, Windy) weather with robust PID parameters "robust PID parameters obtained by AGA after applying wind disturbances", and (No-Wind, Windy) weather with optimum PID parameter "optimum PID parameters obtained by AGA before applying wind disturbances".

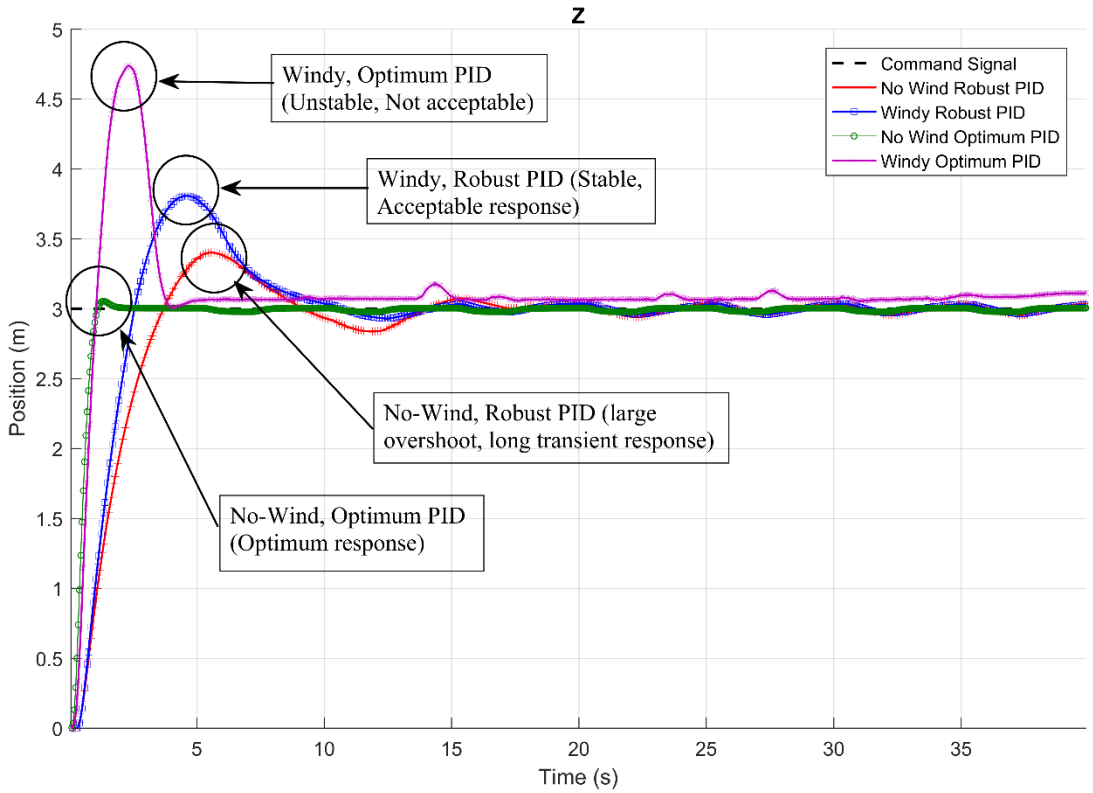


Figure 4.18: Z-controller response in different cases

While the quad-rotor was affected by strong angular wind disturbance, we maintained stability by using Robust PID parameters. In case of no-wind disturbance controlled by Robust PID parameters, the response may be regarded as a bad result (large overshoot, long transient response). Optimum PID parameters failed to stabilize the quad-rotor during strong wind disturbances. Therefore, optimum PID parameters give perfect response when there are no wind disturbances.

In this case, PID parameters must be a function of disturbance values, since each disturbance value requires different PID parameter values. The controller's response was affected by two parameters in the z-axis controller; the K_p and K_i values. In the previous chapter K_p and K_i curves were obtained by performing many simulations. In addition, two controllers were combined with PID z-controller to make the controller adaptive.

4.6. ADAPTIVE FUZZY-PID VS ADAPTIVE NN-PID CONTROLLER RESPONSES

In the previous chapter, we designed Fuzzy and Neural Networks controllers to be integrated with PID controller for z-axis. Figure 4.19 shows a comparison between using Adaptive Fuzzy-PID and Adaptive NN-PID controllers.

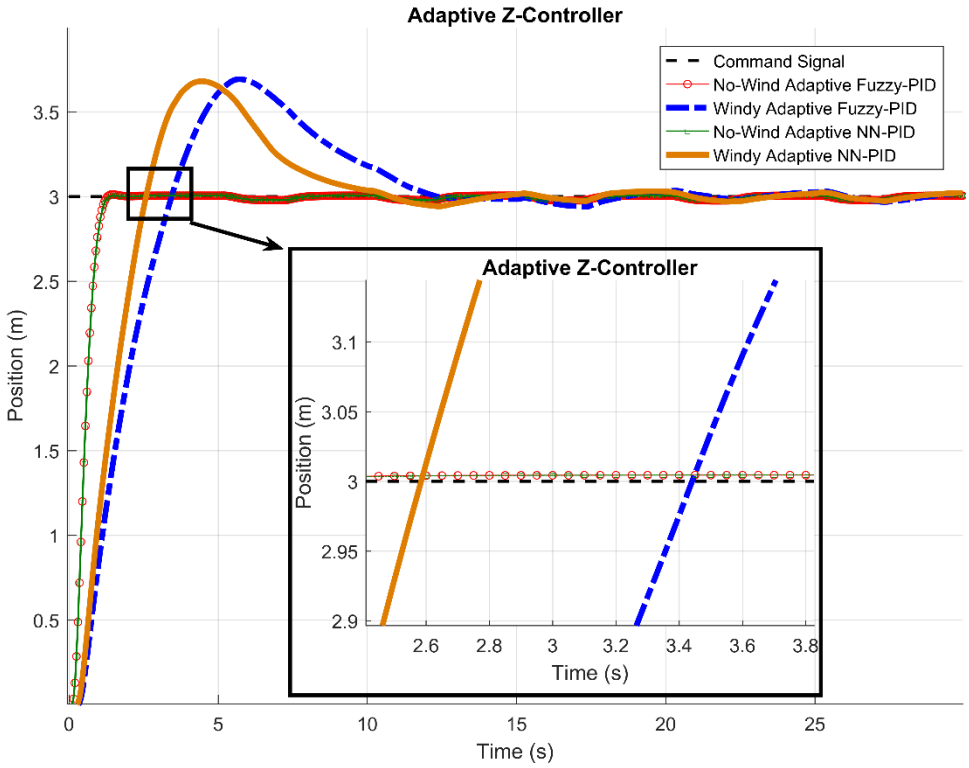


Figure 4.19: Adaptive Fuzzy-PID and NN-PID controllers for z-axis

The comparison result shows that both controllers preserved stability and provided optimum response. However, Figure 4.19 shows that the Adaptive NN-PID controller responded faster by 0.8 seconds than the Adaptive Fuzzy-PID controller.

Figure 4.20 shows the quad-rotor flight path controlled by Adaptive controllers. Due to strong angular wind disturbances, the quad-rotor is drifted from its planned path during takeoff time. The reason for the drifting is that the controller started to adapt and dampen the disturbances after the disturbances started to hit the quad-rotor, at that time, the controller should follow two issues: the first is making a successful takeoff from the ground, the second is maintaining stability during take-off. Both adaptive controllers showed reasonable results with small differences. However, Adaptive Fuzzy-PID controller has less error in y-axis by about 0.2 meter. Figure 4.21 shows the response of Psi angle controllers, Adaptive NN-PID controller showed faster response than Fuzzy-PID, but Fuzzy-PID had less error than NN-PID. Thus, both controllers have advantages and disadvantages. Therefore, both controllers maintained stability during strong weather disturbances for Psi angle.

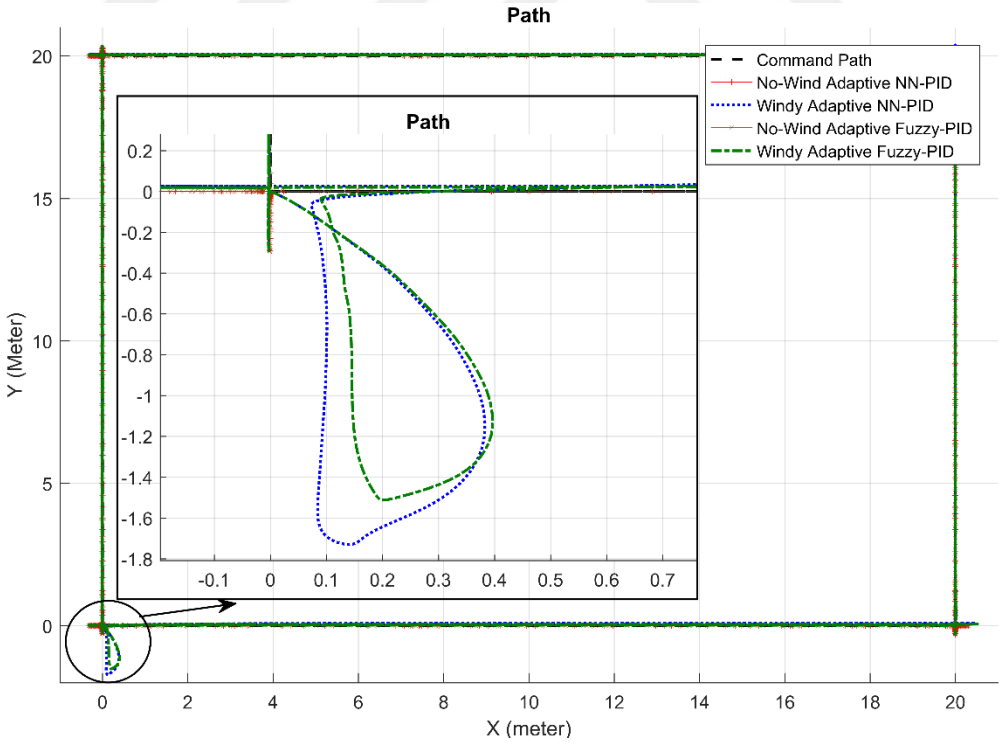


Figure 4.20: Quad-rotor planned path controlled by Adaptive controllers

Figure 4.22 shows the response of Theta controllers, between seconds 0 and 3; there were some oscillations, but after that the controllers successfully dampen the oscillations. Between seconds

5 and 7, some limit exceeded, angle limit was assumed to be 12-degrees, but the response reached an angle of 14-degrees. Due to the strong wind disturbances, the exceeding of the limit can be understandable. Both controllers maintained a roughly similar ratio of stability.

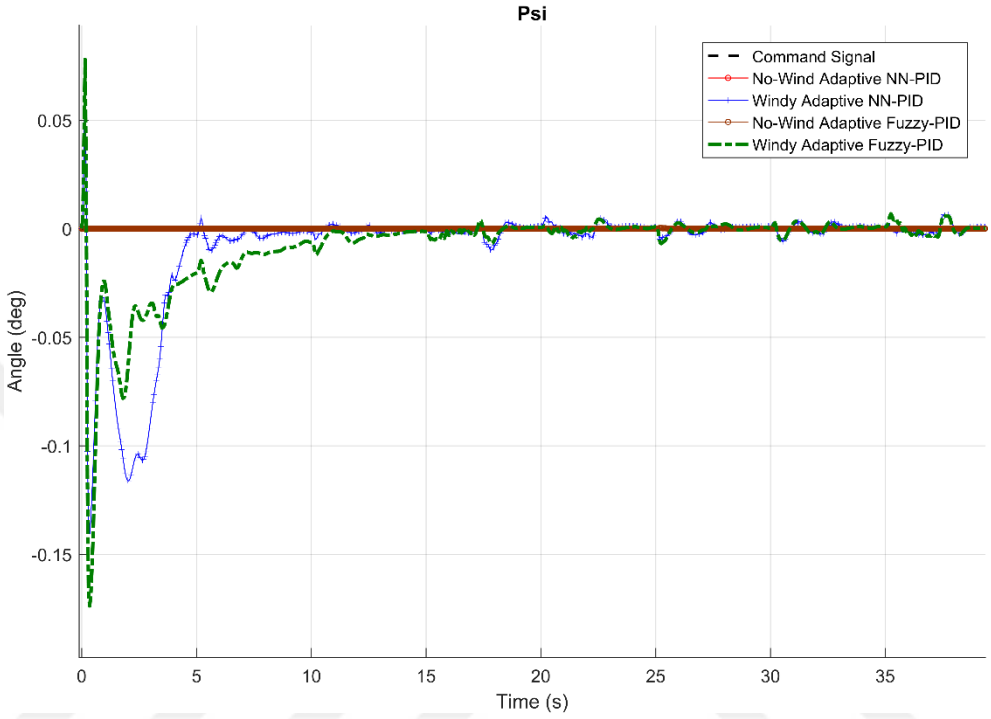


Figure 4.21: Adaptive controllers' response for Psi angle

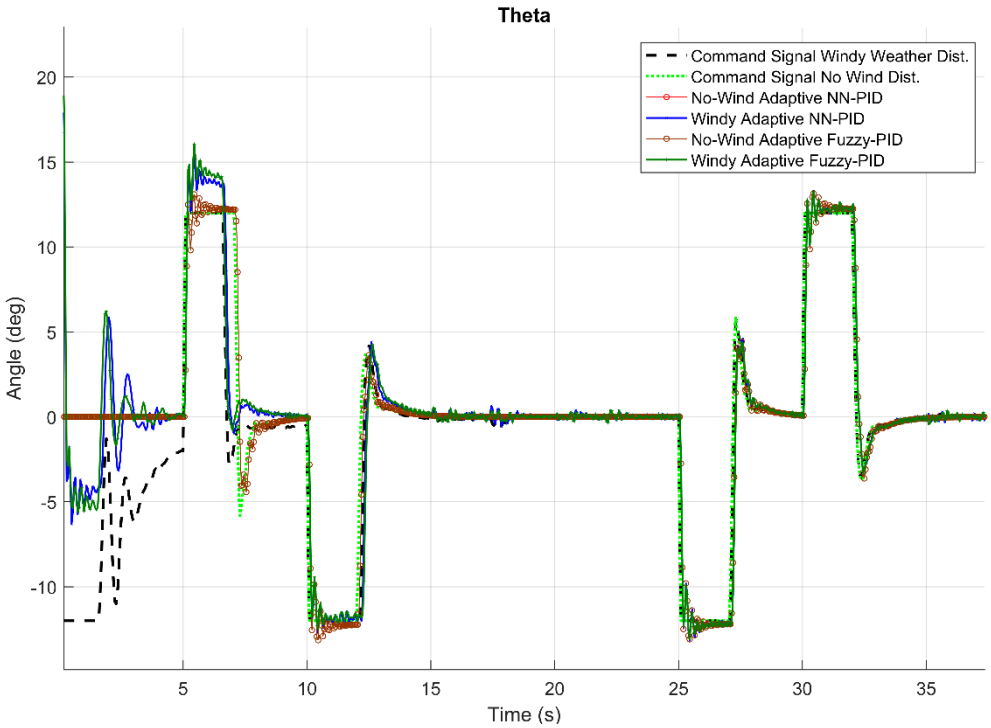


Figure 4.22: Adaptive controllers' response for Theta angle

To ensure that controllers are doing well for any other disturbance values between minimum and maximum disturbance; we simulate the flight with different disturbance values. Disturbance values are torque values applied to the system of quad-rotor as an external continuous torque. The figures below show the response of y-axis and z-axis controllers.

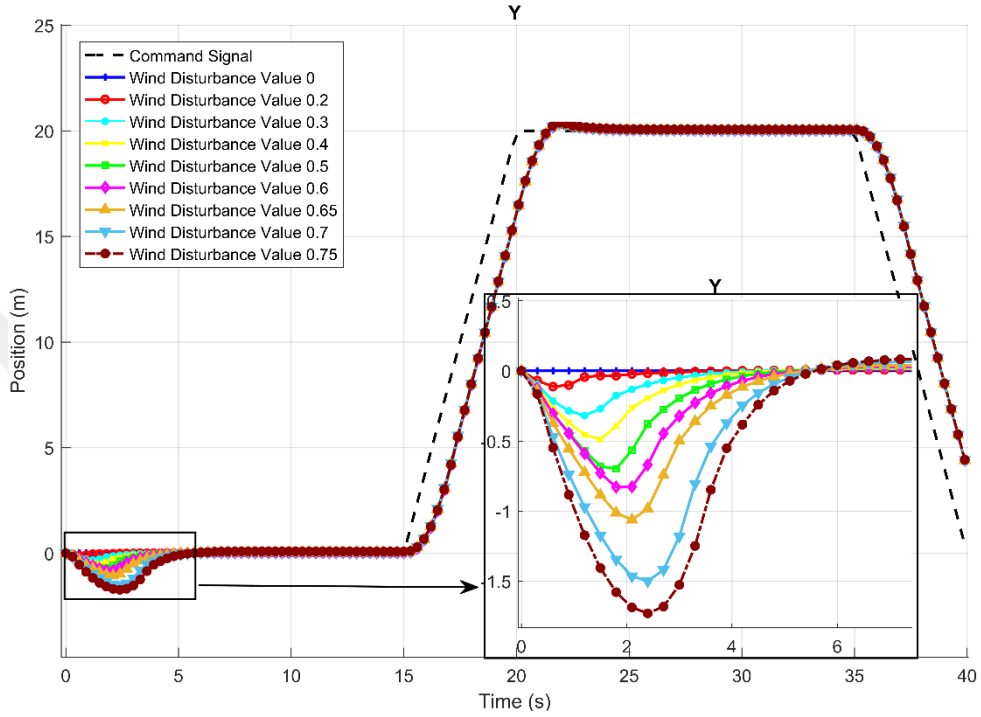


Figure 4.23: Y-axis adaptive controller response for different disturbance values

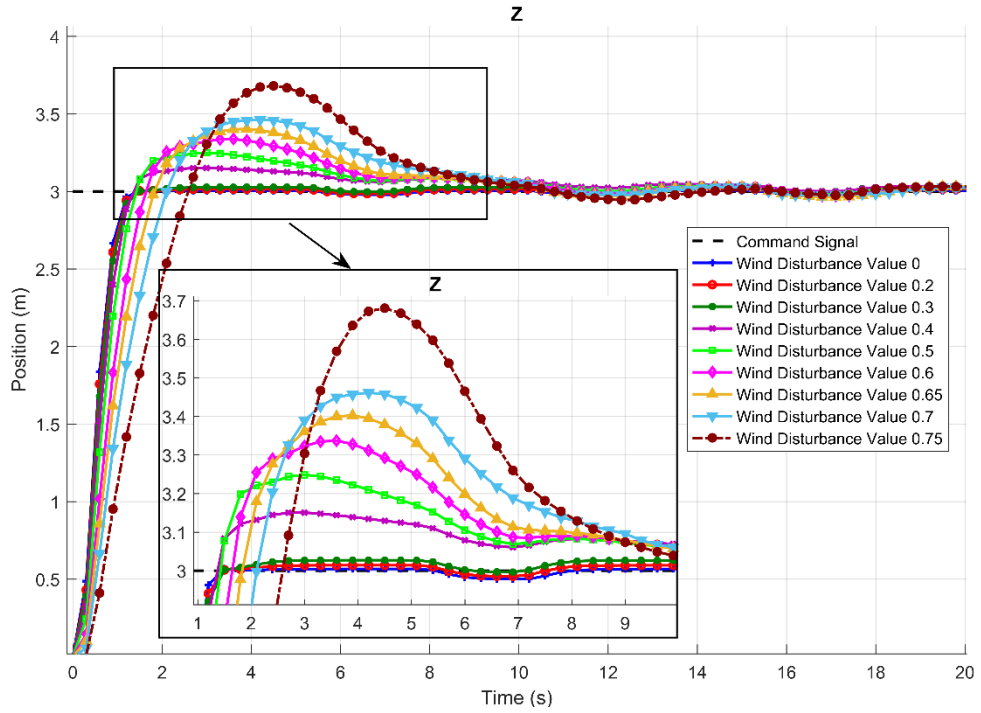


Figure 4.24: Z-axis adaptive controller response for different disturbance values

Figure 4.23 and 4.24 show that increasing angular wind disturbance increases the error, but after the take-off phase, the controller successfully dampened the error to be almost zero to the end of the flight. Therefore, the controllers maintained stability in all disturbance cases between the minimum and maximum disturbance values.

4.7. ADAPTIVE DISCRETE CONTROLLER RESPONSE

The results of the previous simulations indicate that we successfully designed an Adaptive controller which gives Optimum-Robust performance. For now, we need to implement our controller in physical microcontroller devices to make real flight tests. Before programming the controller, we performed a last simulation; thus, we discretized the controller with different sample rates to make sure that the controller would function as desired. The results below are for two sample rate values; 2 milliseconds and 20 microseconds. However, the processors of the current available devices in the market are faster than 20 microseconds which provide better performance. For example, the *Pixhawk* controller can work with a sampling rate of up to 6 ns, 168 Mhz. Figure 2.25 shows the response of a simulation with sample rates of 2 ms and 20 us.

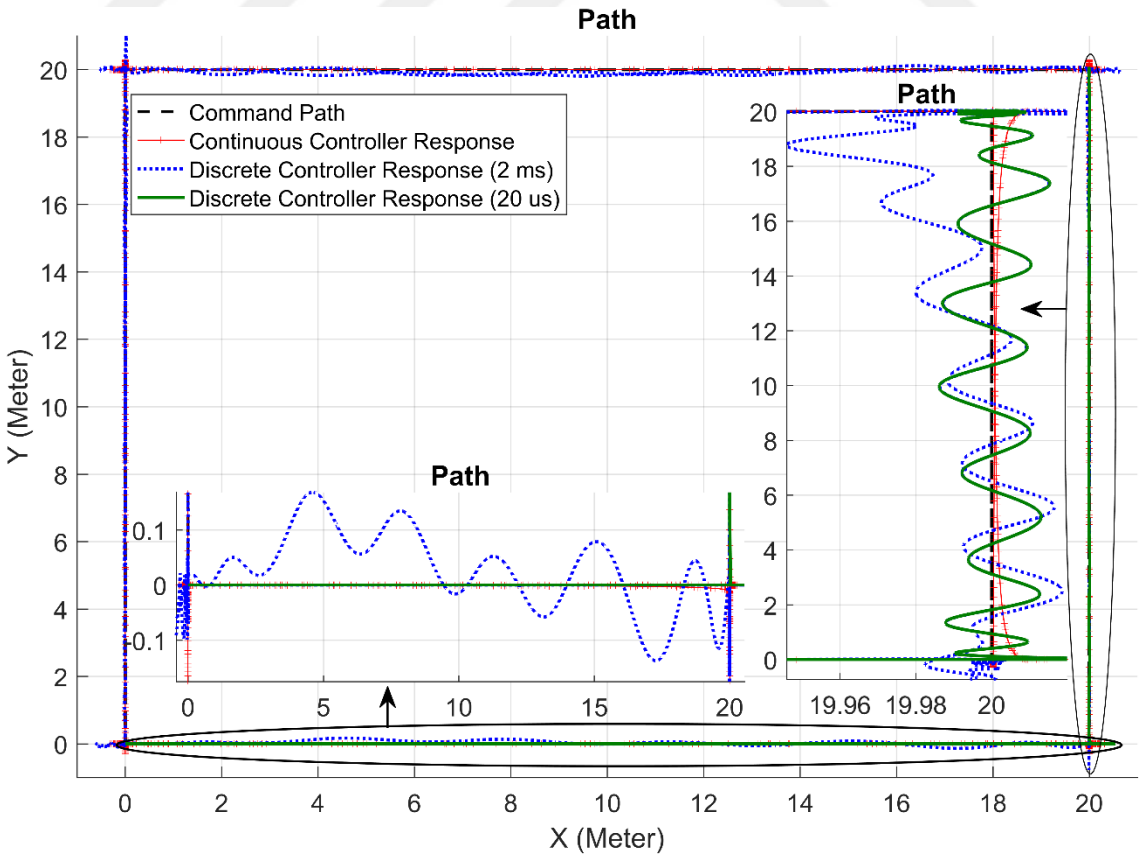


Figure 4.25: Comparison between different sample times for discrete controller

Figure 4.25 shows that the oscillations generated by discrete controller with sampling time of 2 milliseconds didn't exceed 20 cm; thus, oscillations generated by discrete controller with sampling rate of 20 microseconds didn't exceed 2 cm. accordingly, increasing sampling time increases the stability of the flight.

The simulation was done under these parameters: Solver type (Fixed-step), Solver (ode-Auto), Fixed-step size (fundamental sample time) is 0.05. The simulation was done by a computer with these specifications: Processor (Intel® Core™ i3-6006U CPU @ 2.00GHz 2.00 GHz, Memory (RAM) 8.00 GB, System type 64-bit Operating System, x64-based processor.

Also, to be sure that the quad-rotor will hold under different weather conditions; we added load to the drone as a force applied to the z-axis, the results of adding load with about 30% of quad-rotor weight is shown in Figure 4.26.

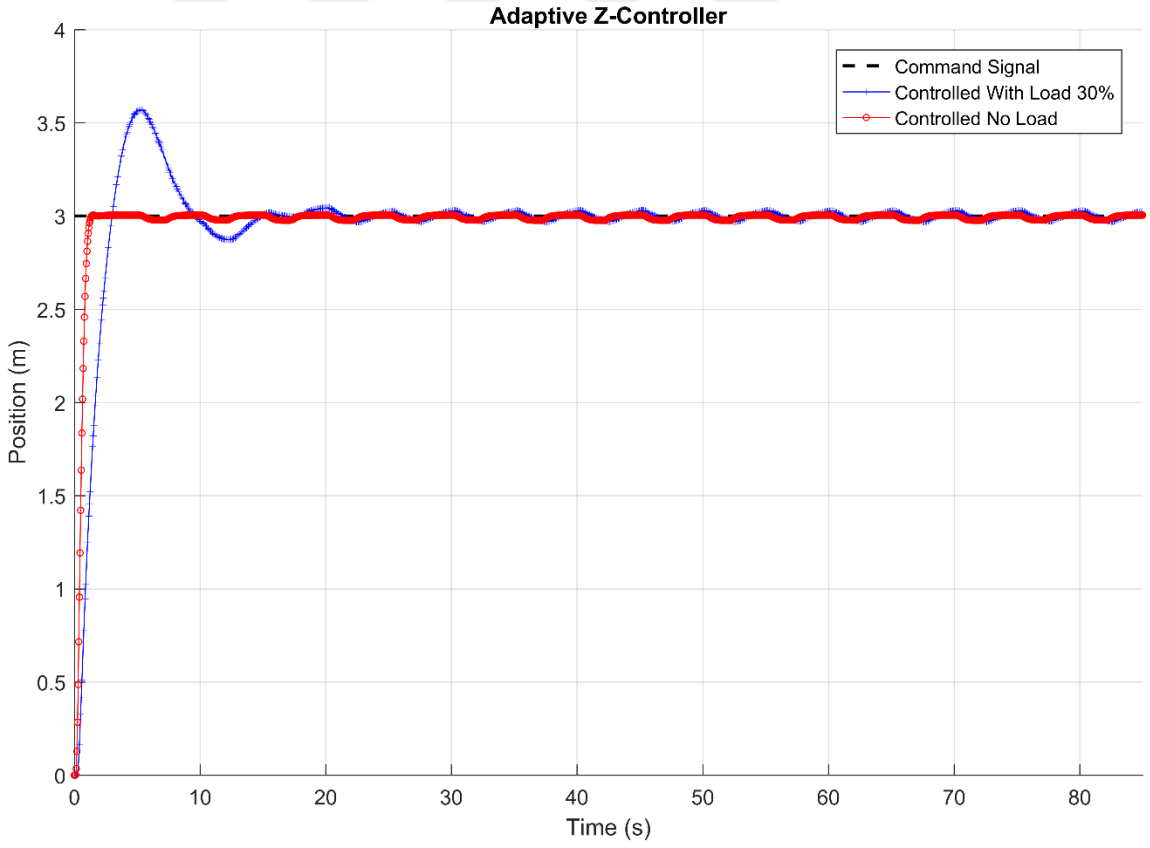


Figure 4.26: Controller response with load

The response of the controller shows a stable flight after adding a load to the quad-rotor which means that the drone can maintain stability even with load.

5. DISCUSSION

First, let's begin with the quad-rotor system modeling. According to the previous studies summarized in the first chapter; there are many methods of modeling the system, but all these models are based on two main theories: Newton-Euler and Euler-Lagrange. The difference between these two approaches is the way we want our system to be analyzed. So, if we want to use Euler angles and rectangular co-ordinate system, we will need to use Newton-Euler. But if our co-ordinate system is expressed in Axis/Angle representation, it is better that we use the Euler-Lagrange method. Therefore, an analysis based on the Newton-Euler method considers the equations of force and torque as basic equations for modeling the system, whereas the Euler-Lagrange method analyzes the total value of potential and kinetic energy. So, it is possible to use both approaches to model the quad-rotor system. Newton-Euler method was used in this research to model quad-rotor system.

Quad-rotor system can be analyzed on the basis of different considerations. Accordingly, in some studies, they found that all forces and torques that affect the system should be included in the analysis of the system; it does not matter whether it has a small or large effect on the system. However, after we have performed many simulations and tests over system model, some terms of external forces and torques can be ignored, particularly with unmanned aerial vehicles. We find that these terms have very little impact and don’t affect the flight system. Neglecting certain complex terms has the advantages of decreasing mathematical complexity which helps accelerate the simulation process. The most affective terms are: Inertial acceleration, Coriolis acceleration, Centrifugal acceleration, Gyroscopic force, Gravitational force, Drag force, Thrust force, and disturbance forces “as a lumped term”.

We found that previous studies could not easily perform the issue of stability testing due to the system's non-linearity. Some studies suggested linearization of the quad-rotor system to use linear algorithms that test system stability, but for the linearization of the system, they ignored several terms that could affect the system. On the other hand, one of the useful ideas for testing system stability in open loop form is to hold the system in a point of equilibrium, then if the system's response goes to infinity, it means the system is unstable. The results show that the quad-rotor system is unstable in open-loop; hence, our new controller handled the instability.

We started designing quad-rotor controller after we finished the modeling of the system. Based on previous studies, many controllers can be used to control quad-rotor. Some of these projects used intelligent controls like the Fuzzy controller, but when it comes to the practical side, these projects encountered some problems. Intelligent controllers like Fuzzy controller require heavy processing, so controllers like *ArduPilot* can't handle controller processing operations. Other devices like *Pixhawk* can handle such controllers, but they didn't produce very satisfactory results in real-time operations. The outcomes of previous projects did not promote the use of intelligent controllers to control multi-copters, and that is why we chose a guaranteed controller to be our quad-rotor 's main controller. On the other hand, we have integrated other intelligent controllers as tuning controllers with the main PID controller.

PID parameter tuning is one of the challenges presented in previous studies. In this field, we preferred using the Adaptive Genetic Algorithm as a method to provide optimum PID parameter. Adaptive Genetic Algorithm is an enhancement to the Genetic Algorithm to speed up the optimization process and extend the search area. Compared to GA, AGA made great results, so we considered AGA as our optimization method. We have included two principles for optimizing the parameters: firstly, to tune all six PID controllers simultaneously; secondly, to make the optimization according to complete planned flight. This can be considered as half online-tuning and half offline-tuning. The optimization result yielded very satisfactory results because we obtained what we were looking for; optimizing for a complete flight. Adaptive Genetic Algorithm was a very useful technique because of its nature of randomity. Two stages of optimization have been adopted to optimize and stabilize system performance using two fitness functions; the first approach is to use square error as a basis for the fitness function: this approach has been useful for large errors. The second method is absolute error as a basis for fitness function; this has been useful for fine tuning. As seen in the previous chapter, the findings were quite satisfactory. In this way, we succeed in overcoming all prior difficulties faced by previous studies related to parameter tuning of the controller.

Our aim was to design a controller that is able to tolerate strong disturbances such as strong wind as well as maintain stability under specific constraints. Previous studies applied wind disturbances but without considering the non-linear nature of this wind disturbance. We chose to split wind disturbances in this area into linear wind disturbance and angular wind disturbance. In that way, we can understand in more details the characteristics of the wind's impact. After

doing several simulations as seen in the previous chapter, we found out that the optimum PID parameters can be used to encounter linear wind disturbances. As a result, if linear wind disturbance increases; there will be no need to modify PID parameters. On the other hand, angular wind perturbance has non-linear effects on the system. Consequently, if the disturbance increases, the controller parameters should be updated with a non-linear function. We found out in overall that the controller should be a non-linear controller to be able to withstand wind disturbances.

To design a non-linear controller, we decided to integrate an intelligent controller to act as a tuning controller for the PID parameters in the main controller. In order not to experience the mistakes of other projects, we can program those intelligent controllers in separated devices. According to the disturbance value, the intelligent controller will tune PID parameters. However, the operation of these intelligent controllers won't affect the main PID controllers' work. In this way, we override the disadvantages of using intelligent controllers in standard programmable devices.

We chose two artificial intelligent techniques to tolerate the non-linearity, Fuzzy control and Neural Networks. These controllers should address the non-linearity of PID parameters that have happened because of wind disturbances. Both controllers can deal with the non-linear functions, but as we've seen in the results, there are some differences between those controllers. One of the advantages of Fuzzy is that we can solve the non-linearity by playing with the width of the membership functions in fuzzification or defuzzification according to the application. But the disadvantage is that if the non-linearity of the system isn't serialized, it will be difficult to distribute membership functions as we faced with K_i parameter values. On the other hand, Neural Networks automatically adapt the non-linearity by continuously correcting and modifying weight values by Back-Propagation method. The challenge in designing Neural Networks was to prepare input data, identify the number of network nodes and decide the number of hidden layers. Fuzzy control and Neural Networks therefore maintain system stability and have delivered great performance.

Based on the results of the flight simulations and the responses of the controllers; the controller is ready for testing in actual flight operations. Also, we checked the discrete controller efficiency, and the results were reasonable. For future work, the prototype controller is to be implemented in physical programmable devices. However, we should learn the new

characteristics of the quad-rotor that we want to control. First, we should change system parameters such as quad-rotor weight, length, motor speed, and other parameters, then re-run the optimization process to get the optimum parameters. After updating the controller's parameters, we can program our controller in physical controller device.

As we proposed, the physical controller can be divided into two units, one for the main PID controller and another for tuning controllers. The tuning controllers will provide the main controller with updated parameter values.

6. CONCLUSION AND RECOMMENDATIONS

The purpose of this research was to prepare a quad-rotor for search and rescue operations during harsh weather conditions. SAR operations are considered the most important sensitive operations, and if we use a quad-rotor controlled by intelligent controllers to do these operations; we can get perfect results. It's worth noting that using quad-rotors in these operations can prevent life loses or injures when they're turned on. To achieve this goal, we have taken several steps to prepare a well-designed system capable of carrying out SAR operations under bad weather conditions. One of the essential steps needed for SAR operations is to link the quad-rotor to a non-inertial reference frame. We have shown in our thesis how we can use a Non-inertial frame as a quad-rotor reference. We also compared the relationship between the inertial frame and the non-inertial frame, and how a non-inertial frame can be easily used in special cases such as SAR operations.

A complete modeling of non-linear 6-DOF quad-rotor system including the relationship between quad-rotor body frame and non-inertial reference frame was adopted. We succeeded in designing the quad-rotor in a way that does not ignore essential and effective forces and terms, as well as removing other terms that do not impact small planes like quad-rotors. SAR operations overwater usually occur in harsh weather conditions, so we need advanced control techniques to apply to our system. We preferred using a multi-stage PID controller as it will give us better results than a single-stage control system.

We have improved the Genetic Algorithm to be an Adaptive Genetic Algorithm for PID parameter tuning. GA development was for the sake of fast optimization process and avoiding falling to a local minimum. The results indicate a great improvement in the optimization process with a small number of iterations. PID tuning parameters weren't enough to withstand harsh weather conditions, so we improved the controller to be multi-stage adaptive PID controller. Adaptive PID controller may include Fuzzy controller or Neural Networks to tolerate weather disturbance non linearity. Adaptive Fuzzy-PID and Adaptive NN-PID both preserved stability and delivered optimal performance.

Following these developments, the controller can be applied to practical projects, so many further improvements can be added to the quad-rotor project, such as filtering sensors, camera system, and other extensions.

REFERENCES

Abaunza, H., Castillo, P., and Lozano, R., 2018, *Quaternion Modeling and Control Approaches*, Handbook of Unmanned Aerial Vehicles, Springer, ISBN: 978-3-319-32193-6.

Akhil, M., Anand, M.K., Sreekumar, A., and Hithesan, P., 2012, Simulation of the mathematical model of a quad rotor control system using matlab Simulink, *Applied Mechanics and Materials*, 110-116 (2012), 2577-2584.

Alexandru, C., 2008, Aspects Regarding the Mechatronic Tracking Systems Used for Improving the Photovoltaic Conversion, *Fascicle of Management and Technological Engineering*, 7(17), 695 – 704.

Alibeiki, A., 2010, Genetic Algorithm and Comparison with Usual Optimization Methods, *World Applied Sciences*, 11(6), 752–754.

Angeline, P. J., Saunders, G. M., and Pollack, J. B., 1994, An Evolutionary Algorithm that Constructs Recurrent Neural Networks, *IEEE Transactions on Neural Networks*, 5(1), 54 – 65.

Artale, V., Milazzo, C., and Ricciardello, A., 2013, Mathematical modeling of hexacopter, *Applied Mathematical Sciences*, 7(97), 4805–4811.

Bao, N., Ran, X., Wu, Z., Xue, Y., and Wang, K., 2017, Research on attitude controller of quadcopter based on cascade PID control algorithm. *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 15-17 Dec. 2017, Chengdu, IEEE, ISBN: 978-1-5090-6414-4, 1493-1497.

Basdogan, C., 2004, *Discrete PID Controller*, http://portal.ku.edu.tr/%7Ecbasdogan/Courses/Robotics/lab.htm, [Visiting date: 23 June 2019].

Batmaz, A. U., Elbir, O., and Kasnakoglu, C., 2013, Design of a Quadrotor Roll Controller Using System Identification to Improve Empirical Results. *International Journal of Materials, Mechanics and Manufacturing*, 1(4), 347–349.

Black, M., 1937, Vagueness. An Exercise in Logical Analysis, *Philosophy of Science*, 4(4), 427 – 455.

Bouabdallah, S., Noth, A., and Siegwart, R., 2004, PID vs LQ control techniques applied to an indoor micro quadrotor, *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 28 Sept.-2 Oct. 2004, Sendai, IEEE, ISBN: 0-7803-8463-6

Bresciani, T., 2008, *Modelling, Identification and Control of a Quadrotor Helicopter*, Master Thesis, Department of Automatic Control, Sweden, Lund University.

Broccoli, A.J., 2012, *Total Differentiation of a Vector in a Rotating Frame of Reference*, http://people.envsci.rutgers.edu/broccoli/dynamics_lectures/, [Visiting date: 1 March 2019].

Cai, G., Chen, B. M., Lee, T. H., 2011, *Unmanned Rotorcraft Systems*, Springer-Verlag London, ISBN: 978-0-85729-635-1.

Chen, H.C., Chang, S.H., 2006, Genetic Algorithms Based Optimization Design of a PID Controller for an Active Magnetic Bearing, *IJCSNS International Journal of Computer Science and Network Security*, 6(12), 95 – 99.

Chovancová, A., Fico, T., Chovanec, Ľ., and Hubinsk, P., 2014, Mathematical Modelling and Parameter Identification of Quadrotor (a survey), *Procedia Engineering*, 96(2014), 172–181.

Cottrell, G., (1990), Extracting features from faces using compression networks: Face, identity, emotion, and gender recognition using hotons, *Proceedings of the 1990 Connectionist Models Summer School*, 328 – 337.

Dallal Bashi, O.I., Wan Hasan, W.Z., Azis, N., Shafie, S. and Wagatsuma, H., 2017, Quadcopter sensing system for risky area, *2017 IEEE Regional Symposium on Micro and Nanoelectronics (RSM)*, Batu Ferringhi, 23-25 August 2017, ISBN: 978-1-5090-4029-2, 216–219.

Dikmen, I. C., Arisoy, A., and Temeltas, H., 2009, Attitude control of a quadrotor. *2009 4th International Conference on Recent Advances in Space Technologies*, 11-13 June 2009, Istanbul, IEEE, ISBN: 978-1-4244-3626-2, 722-727.

Evans, W. R., 1948, Graphical Analysis of Control Systems, *Transactions of the American Institute of Electrical Engineers*, 67(1), 547 – 551.

García Carrillo, L.R., Dzul López, A.E., Lozano, R., and Pégard, C., 2013, *Quad Rotorcraft Control Vision-Based Hovering and Navigation*, Springer-Verlag London, ISBN: 978-1-4471-4398-7.

Gibiansky, A., 2012, *Quadcopter Dynamics, Simulation, and Control*, http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/, [Visiting date: 10 September 2019].

Goldberg, E. D., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley P.C., ISBN: 0201157675.

Hartman, D., Kim, J. M., Landis, K., Mehrer, M., and Moreno, S., 2014, *Quadcopter dynamic modeling and simulation for control system design*, MSc. Thesis, Drexel University, Philadelphia.

Hemingway, E. G., and O'Reilly, O. M., 2018, Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments, *Multibody System Dynamics*, 44(1), 31–56.

Holland, J., 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan, ISBN: 0262581116.

Jantzen, J., 1998, *Design Of Fuzzy Controllers*, Tech. Report, Technical University of Denmark.

Jones, A. H., and De Moura Olivera, P. B., 1995, Auto-Tuning of PID Controllers, *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 12-14 September 1995 Sheffield, ISBN: 0852966504, 141 – 145.

Khuwaja, K., Tarca, I. C., Lighari, N.-Z., and Tarca, R. C., 2018. PID Controller Tuning Optimization with Genetic Algorithms for a Quadcopter, *Recent Innovations in Mechatronics*, 5(1).

Kim, J., Moon, Y., and Zeigler, B. P., 1995, Designing fuzzy net controllers using genetic algorithms, *IEEE Control Systems Magazine*, 15(3), 66 – 72.

Kishnani, M., Pareek, S., and Gupta, R., 2014, Optimal Tuning of PID controller by Cuckoo Search via Lévy flights, *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*, 1-2 Aug. 2014, Unnao, ISBN: 978-1-4799-6393-5.

Kotarski, D., Benic, Z., and Krznar, M., 2016, Control design for unmanned aerial vehicles with four rotors, *Interdisciplinary Description of Complex Systems*, 14(2), 236-245.

Kumar, P.V., Challa, A., Ashok, J. and Narayanan, G.L., 2015, GIS based fire rescue system for industries using Quad copter – a novel approach, *2015 International Conference on Microwave, Optical and Communication Engineering (ICMOCE) IEEE*, Bhubaneswar, 18-20 December 2015, ISBN: 978-1-4673-6981-7, 72–75.

Kurak, S., and Hodzic, M., 2018, Control and Estimation of a Quadcopter Dynamical Model. *Periodicals of Engineering and Natural Sciences (PEN)*, 6(1), 63–75.

Lang, P.J., Bradley, M.M., and Cuthbert, B.N., 1990, Emotion, Attention, and Startle Reflex, *Psychological Review*, 97(3), 377 – 395.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D., 1989, Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation*, 4(1), 541 – 551.

Lindley, D.V., 1987, The Probability Approach to the Treatment of Uncertainty in Artificial Intelligence and Expert Systems, *Statistical Science*, 2(1), 17 – 24.

Luukkonen, T., 2011, Modelling and Control of Quadcopter, Independent Research Project in Applied Mathematics, Espoo, Aalto University.

Mahmoud, O. E., Roman, M. R., and Nasry, J. F., 2014, Linear and Nonlinear stabilizing control of Quadrotor UAV, *2014 International Conference on Engineering and Technology (ICET)*, 19-20 April 2014, Cairo, ISBN: 978-1-4799-5807-8.

Mamdani, E.H., and Assilian, S., 1975, An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal of Man-Machine Studies*, 7(1), 1 – 13.

Masse, C., Gougeon, O., Nguyen, D., and Saussie, D., 2018, Modeling and Control of a Quadcopter Flying in a Wind Field: A Comparison Between LQR and Structured $\mathscr{H}\infty$ Control Techniques. *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 12-15 June 2018, Dallas, IEEE, ISBN: 978-1-5386-1354-2, 1408-1417.

Mayr, O., 1970, *The Origins of Feedback Control*, M.I.T. Press, ISBN: 0262630567.

Moyano Cano, J., 2013, *Quadrotor UAV for wind profile characterization*, M.Sc. thesis, Universidad Carlos III de Madrid, Institute for Wind Energy and Energy System Technology.

Musa, S., 2017, Techniques for Quadcopter Modelling & Design: A review, *Journal of Unmanned System Technology*, 5(3), 66-75.

Najm, A. A., and Ibraheem, I. K., 2019, Nonlinear PID controller design for a 6-DOF UAV quadrotor system, *Engineering Science and Technology, an International Journal*, 22(4), 1087–1097.

Nyquist, H., 1932, Regeneration Theory, *The Bell System Technical Journal*, 11(1), 126 – 147.

Ore, O., 1953, *Cardano - The Gambling Scholar*, Princeton University Press.

Paiva, E., Soto, J., Salinas, J., and Ipanaque, W., 2016, Modeling, simulation and implementation of a modified PID controller for stabilizing a quadcopter, *2016 IEEE International Conference on Automatica (ICA-ACCA)*, 19-21 Oct. 2016, Curico, ISBN: 978-1-5090-1147-6.

Pomerleau, D.A., 1993, Neural Networks For Intelligent Vehicles, *Proceedings of the Intelligent Vehicles '93 Symposium*, 14-16 July 1993 Tokyo.

Raiha, O., 2008, *Applying Genetic Algorithms in Software Architecture Design*, M.Sc. thesis, University of Tampere, Department of Computer Sciences.

Ross, T.J., 2010, *Fuzzy Logic with Engineering Applications, Third Edition*, John Wiley & Sons, Ltd.

Sinivasulu, S., and Jain, A., 2006, A comparative analysis of training methods for artificial neural network rainfall-runoff models, *Applied Soft Computing*, 6(3), 295 – 306.

Sohail, A., Watanabe, K., and Takeuchi, S., 2006, Stream flow forecasting by artificial neural network (ANN) model trained by real coded genetic algorithm (GA), *Journal of Groundwater Hydrology*, 48(4), 233–262.

Spears, W.M., De Jong, K.A., 1991, An Analysis of Multi-Point Crossover, *Foundations of Genetic Algorithms*, 1(1), 301 – 315.

Stevens, B. L., Lewis, F. L., and Johnson, E. N., 2015, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems, 3rd Edition*, Hoboken, NJ: J. Wiley, 768 p., ISBN: 978-1-118-87098-3.

Toivonen, H., 2017, *Discrete-time systems and digital controllers*, http://users.abo.fi/htoivone/courses/isy/, [Visiting date: 5 October 2019].

Tytler, C., 2017, *Modeling Vehicle Dynamics – Quadcopter Equations of Motion*, https://charlestytler.com/quadcopter-equations-motion/, [Visiting date: 1 April 2019]

Tytler, C., 2019, *Vehicle Dynamics – Coriolis' Theorem and the Cross-product Matrix*, https://charlestytler.com/vehicle-dynamics-coriolis-theorem-and-the-cross-product-matrix/, [Visiting date: 1 March 2019].

Waslander S., and Wang, C., 2009, Wind disturbance estimation and rejection for quadrotor position control, *AIAA Infotech at Aerospace Conference and AIAA Unmanned Unlimited Conference*, 6-9 April 2009, Seattle.

Weller, P. R., Summers, R., and Thompson, A. C., 1995, Using a genetic algorithm to evolve an optimum input set for a predictive neural network, *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 12-14 September 1995 Sheffield, ISBN: 0852966504, 256-258.

Yarlagadda, R. K., 2010, *Analog and Digital Signals and Systems*, Springer, ISBN: 1441900330.

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

Zhang, X., Li, X., Wang, K., and Lu, Y., 2014, A Survey of Modelling and Identification of Quadrotor Robot, *Abstract and Applied Analysis*, 1–16.

Zimmermann, H.-J., 1990, Fuzzy Sets Theory - and Its Applications, 2nd ed., Springer Science + Business Media, New York, ISBN: 9401579512.

Zuo, Z., 2010, Trajectory tracking control design with command-filtered compensation for a quadrotor, *IET Control Theory & Applications*, 4(11), 2343–2355.

APPENDICES

Appx. 1. K_p Values versus Disturbance Values

Disturbance values	Kp (Jan Values)																			
0.75	3.67	3.66	4.14	5.07	5.62	5.84	5.85	5.82	5.74	5.66	5.6	5.53	5.48	5.42	5.37	5.09	4.84	4.75	4.7
0.74	3.66	3.62	3.61	4.18	4.91	5.47	5.64	5.66	5.62	5.57	5.53	5.48	5.43	5.37	5.34	5.29	4.89	4.75	4.69
0.73	3.66	3.62	3.58	3.64	4.24	4.92	5.31	5.45	5.47	5.47	5.45	5.4	5.36	5.34	5.29	5.26	4.99	4.76	4.69
0.72	3.65	3.62	3.57	3.53	3.89	4.39	4.87	5.13	5.28	5.34	5.36	5.35	5.32	5.29	5.26	5.25	5.01	4.78	4.69
0.71	3.63	3.6	3.56	3.52	3.48	3.95	4.41	4.76	5.03	5.17	5.23	5.26	5.25	5.24	5.23	5.21	5.17	4.79	4.68
0.7	3.62	3.59	3.55	3.52	3.48	3.67	4.06	4.43	4.76	4.98	5.1	5.16	5.19	5.19	5.18	5.17	5.14	4.86	4.68
0.69	3.62	3.6	3.56	3.52	3.48	3.43	3.78	4.17	4.51	4.77	4.95	5.08	5.09	5.13	5.13	5.13	5.12	5.09	4.69
0.68	3.62	3.6	3.57	3.54	3.5	3.45	3.58	3.96	4.26	4.58	4.8	4.93	5.01	5.04	5.07	5.09	5.08	5.07	4.72
0.67	3.62	3.6	3.57	3.53	3.49	3.46	3.4	3.69	4.12	4.4	4.65	4.81	4.92	4.98	4.99	5.01	5.03	5.04	4.76
0.66	3.64	3.61	3.58	3.54	3.51	3.47	3.43	3.58	3.99	4.26	4.52	4.72	4.84	4.91	4.95	4.98	5	5	4.99
0.65	3.62	3.6	3.57	3.53	3.5	3.46	3.43	3.42	3.71	4.08	4.37	4.58	4.73	4.83	4.89	4.91	4.95	4.97	4.96
0.64	3.61	3.59	3.55	3.52	3.49	3.46	3.43	3.38	3.53	3.87	4.2	4.45	4.62	4.75	4.83	4.87	4.9	4.93	4.93
0.63	3.59	3.57	3.54	3.51	3.48	3.45	3.42	3.39	3.4	3.7	4.01	4.31	4.52	4.66	4.75	4.81	4.85	4.89	4.9
0.62	3.58	3.56	3.53	3.5	3.47	3.44	3.42	3.39	3.35	3.56	3.85	4.19	4.4	4.56	4.68	4.75	4.79	4.84	4.86
0.61	3.57	3.55	3.52	3.49	3.46	3.43	3.41	3.38	3.35	3.47	3.75	4.05	4.29	4.47	4.6	4.69	4.74	4.79	4.82
0.6	3.56	3.54	3.51	3.48	3.45	3.42	3.4	3.38	3.35	3.4	3.62	3.91	4.18	4.37	4.51	4.62	4.69	4.74	4.78
0.59	3.55	3.53	3.5	3.47	3.44	3.41	3.39	3.37	3.35	3.31	3.51	3.84	4.08	4.27	4.43	4.54	4.63	4.69	4.73
0.58	3.53	3.52	3.49	3.46	3.43	3.41	3.38	3.36	3.34	3.31	3.42	3.72	3.98	4.18	4.34	4.46	4.56	4.63	4.69
0.57	3.52	3.51	3.48	3.45	3.42	3.4	3.38	3.36	3.34	3.32	3.33	3.59	3.88	4.09	4.26	4.39	4.49	4.57	4.64
0.56	3.51	3.5	3.47	3.44	3.42	3.39	3.37	3.35	3.33	3.31	3.29	3.49	3.77	3.98	4.15	4.29	4.41	4.5	4.57
0.55	3.51	3.49	3.47	3.44	3.41	3.39	3.36	3.34	3.33	3.31	3.29	3.41	3.67	3.89	4.07	4.22	4.34	4.44	4.51
0.54	3.5	3.48	3.46	3.43	3.4	3.38	3.36	3.34	3.32	3.31	3.28	3.34	3.54	3.8	3.99	4.14	4.26	4.37	4.45
0.53	3.49	3.47	3.45	3.42	3.4	3.37	3.35	3.33	3.32	3.3	3.28	3.28	3.46	3.72	3.91	4.06	4.19	4.3	4.39
0.52	3.48	3.47	3.44	3.42	3.39	3.37	3.35	3.33	3.31	3.3	3.28	3.26	3.4	3.63	3.84	3.99	4.12	4.24	4.34
0.51	3.47	3.46	3.44	3.41	3.38	3.36	3.34	3.32	3.31	3.29	3.28	3.26	3.33	3.54	3.77	3.92	4.06	4.18	4.28
0.5	3.46	3.45	3.43	3.4	3.38	3.36	3.34	3.32	3.3	3.29	3.27	3.26	3.28	3.47	3.7	3.85	3.99	4.11	4.22
0.49	3.46	3.44	3.42	3.4	3.37	3.35	3.33	3.32	3.3	3.28	3.27	3.26	3.24	3.4	3.62	3.79	3.93	4.05	4.16
0.48	3.45	3.44	3.42	3.39	3.37	3.35	3.33	3.31	3.3	3.28	3.27	3.25	3.24	3.35	3.55	3.69	3.83	3.99	4.1
0.47	3.44	3.43	3.41	3.39	3.36	3.34	3.32	3.31	3.29	3.28	3.26	3.25	3.24	3.3	3.47	3.63	3.77	3.93	4.05
0.46	3.43	3.42	3.41	3.38	3.36	3.34	3.32	3.3	3.29	3.27	3.26	3.25	3.24	3.25	3.41	3.57	3.72	3.84	3.99
0.45	3.43	3.42	3.4	3.38	3.35	3.33	3.31	3.3	3.28	3.27	3.26	3.24	3.23	3.22	3.36	3.51	3.66	3.79	3.94
0.44	3.42	3.41	3.39	3.37	3.35	3.33	3.31	3.29	3.28	3.27	3.25	3.24	3.23	3.22	3.32	3.45	3.61	3.74	3.86
0.43	3.41	3.4	3.39	3.37	3.34	3.32	3.31	3.29	3.28	3.26	3.25	3.24	3.23	3.21	3.27	3.4	3.56	3.69	3.81
0.42	3.41	3.4	3.38	3.36	3.34	3.32	3.3	3.29	3.27	3.26	3.25	3.24	3.23	3.21	3.24	3.38	3.51	3.64	3.76
0.41	3.32	3.3	3.3	3.29	3.27	3.26	3.24	3.23	3.22	3.21	3.2	3.19	3.18	3.17	3.17	3.16	3.27	3.39	3.53	3.65
0.4	3.32	3.28	3.28	3.28	3.27	3.25	3.24	3.22	3.21	3.2	3.19	3.18	3.17	3.17	3.16	3.23	3.35	3.48	3.6
0.39	3.31	3.28	3.28	3.27	3.26	3.25	3.23	3.22	3.21	3.19	3.19	3.18	3.17	3.16	3.15	3.2	3.3	3.44	3.56
0.38	3.29	3.25	3.23	3.23	3.23	3.21	3.2	3.19	3.18	3.17	3.16	3.15	3.15	3.14	3.13	3.15	3.25	3.37	3.5
0.37	3.27	3.23	3.2	3.2	3.2	3.19	3.18	3.17	3.16	3.15	3.14	3.14	3.13	3.12	3.12	3.11	3.19	3.31	3.44
0.36	3.24	3.21	3.18	3.16	3.15	3.15	3.15	3.14	3.13	3.12	3.12	3.11	3.11	3.1	3.1	3.09	3.16	3.25	3.37
0.35	3.18	3.17	3.15	3.13	3.12	3.11	3.1	3.1	3.09	3.09	3.08	3.08	3.08	3.07	3.07	3.07	3.1	3.19	3.31
0.34	3.11	3.1	3.09	3.08	3.08	3.07	3.06	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.04	3.04	3.13	3.24
0.33	3.11	3.09	3.08	3.08	3.07	3.07	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.04	3.04	3.03	3.1	3.21
0.32	3.1	3.09	3.08	3.07	3.07	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.1	3.18
0.31	3.09	3.08	3.07	3.07	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.08	3.16
0.3	3.09	3.08	3.07	3.06	3.06	3.06	3.05	3.05	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.05	3.14
0.29	3.08	3.07	3.07	3.06	3.06	3.05	3.05	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.04	3.13
0.28	3.07	3.07	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.12
0.27	3.07	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.1
0.26	3.06	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.09
0.25	3.06	3.05	3.05	3.05	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.08
0.24	3.05	3.05	3.05	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.07
0.23	3.05	3.05	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.06
0.22	3.04	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.05
0.21	3.04	3.04	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.04
0.2	3.04	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.03
0.19	3.03	3.03	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.04
0.18	3.03	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.01	3.03
0.17	3.03	3.03	3.03	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.01	3.01	3.01	3.04
0.16	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.15	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.14	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.13	3.02	3.02	3.02	3.02	3.02	3.02	3.02	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.12	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.11	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.1	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.02
0.09	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.08	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.07	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.06	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.05	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.03
0.04	3	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.02
0.03	3	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.02
0.02	3	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3	3.03
0.01	3	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3	3.03
0	3	3	3	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3.01	3	3.01	3.05
	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

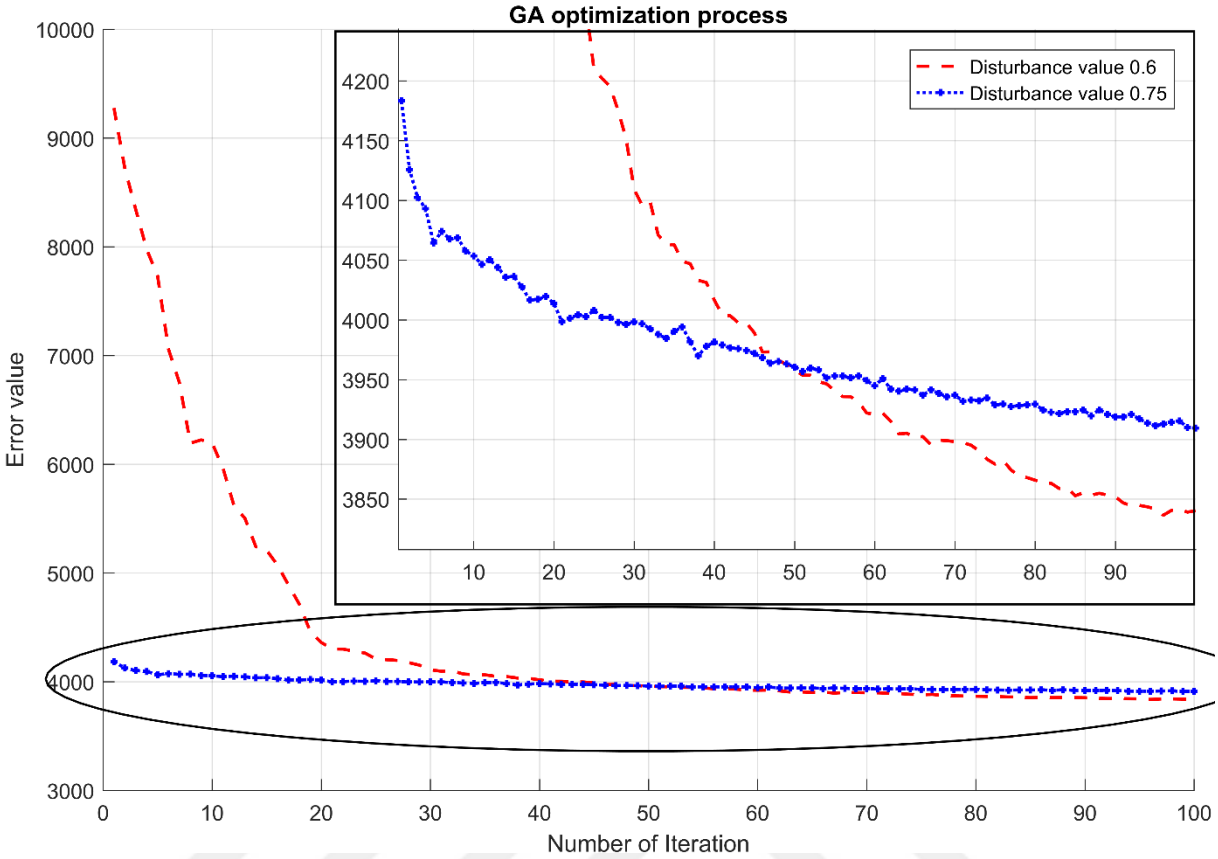
Appx. 2. K_i Values versus Disturbance Values

Appx. 3. GA Optimization Iterations

Iteration Number		Error	PID Parameters for Six Controllers																	
			PhiAngle			ThetaAngle			PsiAngle			Z-axis			X-axis			Y-axis		
			Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
1	Disturbance Value: 0.6	9276	41.11	7.098	5.274	28.72	7.831	3.564	25.26	0.025	7.578	4.585	0.375	10.06	0.624	0.005	0.018	0.628	0.004	0
2		8719	42.16	7.031	4.936	28.92	8.402	4.234	20.63	0.422	7.596	3.483	0	9.91	0.791	0.002	0.015	0.768	0.003	0
3		8341	39.59	7.625	4.912	31.19	8.048	4.054	25.26	0.248	6.99	7.093	0.917	9.443	0.905	0.007	0.037	0.645	0.007	0.006
4		7976	40.76	7.94	5.164	32.34	8.05	4.109	25.78	0	6.736	7.556	0.722	9.71	0.812	0.006	0.029	0.667	0.009	0.013
5		7731	42.22	7.356	4.953	31.87	8.236	4.312	23.49	0.786	8.119	5.519	0.813	9.133	0.503	0.004	0.048	0.515	0.005	0.016
6		7057	42.26	7.644	4.975	32.34	9.115	4.531	24.44	0	8.402	5.964	1.062	9.326	0.526	0.002	0	0.895	0.002	0
7		6750	41.32	8.189	4.913	30.97	9.72	4.087	25.75	0	7.433	4.855	0.878	10.49	0.744	0.004	0	0.607	0.001	0.008
8		6194	45.42	7.755	5.707	33.79	9.608	5.14	25.01	1.133	7.958	4.327	1.315	8.836	0.713	0.004	0	0.963	0.004	0
9		6223	45.79	7.953	5.513	33.36	9.444	4.769	23.36	1.182	7.591	8.687	0.612	9.008	0.419	0.003	0.025	0.731	0.001	0
10		6195	45.79	7.953	5.513	33.36	9.444	4.769	23.36	1.182	7.591	8.687	0.612	8.917	0.582	0.002	0	0.965	0.002	0
11		5961	44.85	8.452	5.724	32.24	9.586	4.39	23.59	1.47	7.505	7.021	0.699	9.056	0.432	0.004	0.054	0.861	0.003	0
12		5619	42.96	7.631	5.149	33.94	10.81	5.1	26.28	1.667	7.929	5.526	1.861	9.502	0.3	0.002	0.024	0.958	0.002	0
13		5503	43.9	7.765	5.432	32.08	10.41	4.642	24.92	2.002	7.989	4.214	2.299	9.544	0.432	1E-03	0.014	1.038	0.002	0
14		5242	42.22	9.308	5.431	34.24	10.33	5.059	25.15	1.644	6.801	9.961	0.938	9.481	0.824	0.004	0.018	0.73	0.001	0.033
15		5208	42	9.205	5.083	35.72	10.31	5.419	26.06	1.686	6.932	9.037	1.03	9.848	0.994	0.006	2E-04	0.519	0	0
16		5070	42.92	8.156	5.527	36.56	10.91	4.786	26.81	2.436	8.252	7.164	2.68	9.225	0.475	0.003	0.02	0.662	3E-04	0.011
17		4889	43.89	9.817	5.759	34.12	10.73	4.545	24.2	1.936	6.904	8.247	1.132	9.56	1.073	0.005	0	0.448	0.002	0.007
18		4710	43.89	9.817	5.759	34.12	10.73	4.545	24.2	1.936	6.904	8.247	2.297	8.934	0.471	0.003	0.015	0.73	0.001	0
19		4459	44.28	10.39	5.402	35.06	11.07	4.728	29.19	2.191	9.092	4.083	2.411	9.167	0.651	0.002	0.016	0.744	0.001	0.033
20		4361	40.71	10.25	5.522	39.49	11.07	4.728	29.19	2.191	9.092	4.083	2.411	9.167	0.651	0.002	0.016	0.744	0.001	0.033
21		4301	45.32	10.53	5.183	37.36	11	4.65	29.34	2.044	9.162	6.018	2.343	9.289	0.59	0.002	0.014	0.431	0.003	0.013
22		4300	44.22	10.36	4.7	37.03	11.32	4.991	27.68	2.514	8.813	7.122	2.346	9.514	0.703	0.004	0.019	0.353	0.005	0.007
23		4280	44.37	10.77	5.149	37.42	11.54	4.918	25.71	2.545	8.528	5.855	2.799	9.759	0.601	0.002	0.039	0.418	0.005	0
24		4263	45.69	10.48	4.642	37.64	10.93	5.203	26.09	2.581	9.051	8.798	2.725	9.385	0.674	0.002	0.012	0.498	0.006	0
25		4209	44.67	11	5.225	37.95	11.15	4.294	28.44	3.775	8.289	4.258	2.405	9.208	0.42	0.002	0.002	0.711	0.002	0.013
26		4202	44.67	11	5.225	37.95	11.15	4.294	28.44	3.775	8.289	4.258	2.405	9.208	0.594	0.003	0.002	0.493	0.005	0.018
27		4195	47.53	10.59	5.635	39.54	11.57	4.491	28.89	3.291	8.302	4.359	2.781	9.299	0.708	0.003	0	0.646	0.006	0.014
28		4175	45.43	10.68	6.006	39.77	11.27	4.768	28.12	2.778	8.842	8.179	2.257	9.262	0.585	8E-04	0	0.724	0.004	0
29		4152	42.84	11.43	5.381	40.67	10.98	4.991	27.11	2.849	6.997	7.424	2.462	9.194	0.628	0.001	0	0.469	0.004	0.019
30		4109	45.9	11.07	5.472	42.23	11.97	4.771	26.4	3.11	8.865	7.523	3.276	9.272	0.663	0.004	0	0.667	0.002	0.017
31		4096	47.5	11.05	5.786	41.49	12.39	5.07	24.92	2.947	8.351	7.851	3.384	9.068	0.502	9E-04	0.002	0.683	0.006	0.01
32		4099	47.5	11.31	5.743	42.23	11.97	4.771	26.4	3.11	8.865	7.523	3.276	9.272	0.663	0.004	0	0.667	0.002	0.017
33		4071	48.67	11.7	5.902	44.18	11.89	4.912	28.51	3.052	9.007	7.613	3.35	10.02	0.533	0.002	0.015	0.498	8E-04	0.006
34		4063	48.67	11.7	5.902	44.18	11.89	4.912	28.51	3.052	9.007	7.613	3.35	10.02	0.533	0.002	0.013	0.562	8E-04	0
35		4063	48.67	11.7	5.902	44.18	11.89	4.912	28.51	3.052	9.007	7.613	3.35	10.02	0.533	0.002	0.013	0.562	8E-04	0.007
36		4049	48.9	10.91	5.887	47.93	11.74	4.913	28.64	3.259	8.729	9.103	3.675	9.259	0.497	7E-04	8E-04	0.679	0	0.013
37		4047	47.03	11.02	5.619	47.79	11.47	5.306	28.51	3.052	9.007	7.613	3.35	10.02	0.675	8E-04	0	0.725	0.006	0
38		4033	47.66	11.69	5.921	45.11	12.05	5.154	26.11	3.462	9.399	8.443	3.301	9.523	0.602	6E-04	0	0.683	0.006	0.013
39		4031	47.66	11.69	5.921	45.11	12.05	5.154	26.11	3.462	9.399	8.443	3.301	9.523	0.602	6E-04	0	0.599	0.005	0
40		4017	48.77	11.12	5.601	48.59	12.05	5.154	26.11	3.462	9.399	8.443	3.301	9.523	0.602	6E-04	0	0.599	0.005	0
41		4004	48.81	11.01	5.482	50.46	12.43	5.528	27.94	3.462	9.399	8.443	3.301	9.523	0.602	6E-04	0	0.599	0.005	0
42		4003	51.82	12.04	5.781	48.47	12.05	5.948	31.76	3.632	8.903	9.018	3.558	9.152	0.687	0.002	0.004	0.525	0.006	0
43		3997	51.82	12.04	5.781	48.47	12.05	5.948	31.76	3.632	8.903	9.018	3.301	9.523	0.602	6E-04	0	0.593	1E-03	0
44		3997	51.82	12.04	5.781	48.47	12.05	5.948	31.76	3.632	8.903	9.018	3.301	9.523	0.602	6E-04	0	0.593	1E-03	0
45		3989	49.53	11.25	5.541	52.66	12.05	5.948	31.76	3.632	8.903	9.018	3.301	9.523	0.602	6E-04	0	0.593	1E-03	0
46		3973	49.91	12.53	5.738	52.66	12.05	5.948	31.76	3.632	8.903	9.018	3.301	9.523	0.602	6E-04	0	0.593	1E-03	0
47		3973	51.82	12.04	5.781	53.24	12.22	5.948	31.76	3.632	8.903	9.018	3.301	9.523	0.602	6E-04	0	0.593	1E-03	0
48		3966	51.91	12.73	5.414	50.53	13.13	6.215	31.16	4.395	9.066	9.373	3.675	9.259	0.586	#####	0.002	0.599	0.005	0
49		3963	50.06	13.13	5.663	50.71	13.31	5.621	27.78	4.019	9.436	8.845	3.301	9.686	0.554	4E-04	0	0.689	0.007	0.005
50		3958	50.06	13.13	5.663	54.01	12.3	6.6	31.46	4.981	8.959	9.788	2.182	10.53	0.645	0.005	0	0.641	0.005	0
51		3954	50.06	13.13	5.663	54.01	12.3	6.6	31.46	4.981	8.959	9.788	2.182	9.259	0.586	#####	0.002	0.599	0.005	0
52		3954	50.85	13.24	5.594	53.33	12.6	6.972	32.62	5.407	9.297	9.295	2.949	9.523	0.602	6E-04	0	0.593	1E-03	0
53		3948	51.98	12.72	5.765	55.33	11.85	6.3	31.86	5.162	9.185	8.939	3.009	9.594	0.702	2E-04	0	0.612	0.003	0
54		3946	50.78	13	6.335	52.82	13.3	6.344	34.7	5.207	9.523	9.11	3.109	9.481	0.755	0	0	0.636	0.01	0
55		3941	51.94	13.26	6.341	54.44	13.15	6.607	35.16	5.384	9.583	8.139	3.568	9.934	0.647	#####	0	0.646	0.012	0.007
56		3936	51.94	13.26	6.341	54.44	13.3	6.344	34.7	5.207	9.523	9.11	3.109	9.481	0.755	0	0	0.636	0.01	0
57		3936	51.98	13.26	6.341	54.44	13.3	6.344	34.7	5.207	9.523	9.11	3.109	9.481	0.755	0	0	0.636	0.01	0
58		3931	50.55	13.58	5.97	54.21	13.11	6.232	36.63	5.832	10.38	8.829	3.579	10.06	0.593	0.002	0.01	0.608	0.005	0
59		3922	50.05	13.58	5.844	53.49	13.65	6.135	38.39	5.956	9.358	9.922	3.618	9.359	0.691	0	0	0.526	0.012	0
60		3921	50.05	13.58	5.844	53.49	13.65	6.135	38.39	5.956	9.358	9.922	3.618	9.359	0.691	0	0	0.636	0.012	0.009
61		3922	50.84	14.07	6.251	53.28	13.35	6.179	38.94	6.049	10.34	8.557	3.053	9.445	0.703	0	0	0.69	0.011	0.006
62		3915	52.81	14.28	5.651	52.67	13.47	6.378	36.86	5.781	9.58	9.419	3.834	10.28	0.617	0	0	0.63	0.01	0
63		3904	50.56	14.85	5.909	53.16	13.55	6.038	36.86	5.781	9.58	9.419	3.834	10.28	0.617	0	0	0.63	0.01	0
64		3905	50.56	14.85	5.909	53.16	13.55	6.038	36.86	5.781	9.58	9.419	3.834	10.28	0.617	0	0	0.651	0.009	0
65		3902	52.96	14.28	6.403	56.92	13.89	6.351	31.81	7.103	9.686	9.246	2.625	10.12	0.692	6E-04	0	0.59	0.015	0
66		3902	51.55	14.51	5.844	53.49	13.65	6.135	38.94	6.049	9.728	9.612	3.347	9.328	0.663	0	0	0.63	0.014	0
67		3896	51.55	14.51	5.844	53.49	13.65	6.135	38.94	6.049	9.728	9.612	3.347	10.12	0.692	6E-04	0	0.59	0.015	0
68		3899	51.55	14.51	5.844	53.49	13.65	6.135	38.94	6.049	9.728	9.612	3.347	10.12	0.692	0.001	0.006	0.54	0.015	0
69		3899	50.09	15.19	5.844	53.49	13.65	6.135	38.94	6.049	9.728	9.612	3.347	10.12	0.692	0.001	0.006	0.54	0.015	0
70		3898	53.31	14.88	5.793	57.94	13.17	6.954	37.75	6.049	9.728	9.612	3.347	10.12	0.692	0.001	0.006	0.54	0.015	0
71		3897	54.26	14.59	5.731	53.28	13.47	5.949	39.03	7.103	9.686	9.246	2.625	9.328	0.663	0	0	0.599	0.018	0
72		3895	50.84	15.22	5.822	54.91	13.57	6.459	38.39	6.509	9.547	10.56	2.951	10.73	0.729	#####	0.002	0.54	0.018	0
73		3890	52	15.68	5.914	56.03	13.95	6.876	39.82	6.746	10.5	8.943	2.905	9.966	0.66	3E-04	0.003	0.609	0.015	0
74		3883	51.8	15.58	5.919	58.18	14.55	6.49	40.14	6.289	11.07	10.13	3.64	10.1	0.67	0	0	0.623	0.014	0
75		3879	51.8	15.58	5.919	58.18	14.55	6.49	40.14	7.636	9.623	9.025	2.28	10.44	0.605	0	0.004	0.553	0.016	0.002
76		3882	51.8	15.58	5.919	58.18	14.55	6.49	40.14	7.636	9.623	9.025	2.28	10.1	0.67	0	0	0.623	0.014	0
77		3874	51.8	15.58	5.919	58.18	14.55	6.49	40.14	7.636	11.07	10.13	3.64	10.44	0.605	0	0.004	0.553	0.016	0.002
78		3869	52.95	16.11	5.966	56.25	14.68	6.767	41.93	7.449	10.92	10.76	3.402	11.38	0.75	0.001	0.003	0.684	0.01	0
79		3868	54.5	16.41	6.157	57.77	14.9	6.826	40.23	7.605	10.44	9.028	3.816	11.7	0.701	3E-04	0.005	0.578	0.012	0.012
80		3866	54.5	16.41	5.966	56.25	14.68	6.767	41.93	7.449	10.92	10.76	3.402	11.38	0.75	0.001	0.003	0.684	0.01	0.002
81		3864	52.67	15.25	5.797	56.86	14.95	6.616	39.12	8.386	11.79	10.6	4.057	10.9	0.757	0.003	4E-04	0.564	0.019	0
82		3863	52.95	16.11	6.157	57.77	14.9	6.826	40.23	7.605	10.92	10.76	3.402	11.38	0.75	0.001	0	0.59	0.017	0
83		3859	54.5	16.41	5.966	56.25	14.68	6.767	41.93	8.386	11.79	10.6	4.057	10.9	0.757	0.003	0.004	0.553	0.016	0
84		3859	53.97	16.62	5.812	58.17	14.68	6.767	41.93	8.386	11.79	10.6	4.057	10.9	0.757	0.003	0.004	0.553	0.016	0
85		3853	53.97	16.62	5.812	58.17	14.68	6.767	41.74	7.605	10.92	10.76	3.402	11.38	0.75	0.001	0	0.59	0.017	0
86		3855	53.97	16.62	5.812	57.17	14.9	6.391	38.49	7.592	9.672	10.84	3.686	10.5	0.726	0.002	0	0.596	0.019	0
87		3853	53.97	16.59	5.705	58.17	14.68	6.767	41.74	7.605	10.92	10.76	3.402	11.38	0.75	0.001	0	0.59	0.017	0
88		3855	52.25	16.7	5.705	58.17	14.68	6.767	41.74	7.605	10.92	10.76	3.402	11.38	0.75	0.001	0	0.59	0.017	0
89		3853	55.6	16.7	6.048	58.12	15.04	6.994	38.92	6.962	10.77	10.72	4.061	11.5	0.763	0	0	0.591	0.019	0
90		3852	55.63	16.64	5.851	59.4	15.01	6.154	42.06	7.113	10.98	11.15	2.981	12.23	0.548	5E-04	0	0.399	0.019	0.012
91		3847	55.8	16.93	5.434	61.39	14.57	6.72	41.72	7.433	10.52	11.49	3.054	11.77	0.603	0.002	0	0.513	0.021	0
92		3845	55.63	16.64	5.851	59.4	14.68	6.767	41.74	7.605	10.92	10.76	3.402	11.7	0.664	0	1E-04	0.513	0.021	0
93		3845	53.49	17.11	5.851	59.4	14.68	6.767	41.74	7.605	10.92	10.76	3.402	11.7	0.664	0	1E-04	0.513	0.021	0
94		3844	54.36	17.03	5.626	60.25	14.68	6.767	41.74	7.605	10.92	10.76	3.402	11.7	0.664	0	1E-04	0.513	0.021	0
95		3842	53.49	17.11	5.851	59.4	15.22	6.767	41.74	7.605	10.92	10.76	3.402	11.7	0.664	0	1E-04	0.513	0.021	0
96		3837	54.77	17.23	5.705	58.17	15.49	6.154	41.74	7.605	10.92	10.76	3.402	11.7	0.664	0	1E-04	0.513	0.021	0
97		3841	51.44	17.84	5.594	58.63	15.71	6.119	40.82	7.745	11.14	10.05	3.402	11.7	0.664	0	1E-04	0.513	0.021	0
98		3841	54.77	17.23	5.705	58.17	15.49	6.154	41.74	7.605	10.92	10.76	3.983	11.74	0.701	9E-04	0	0.467	0.019	0
99		3839	56.05	17.16	6.064	59.83	15.46	6.946	41.7	7.947	10.56	10.75	2.848	11.47	0.676	0.002	0	0.573	0.02	0
100		3840	56.05	17.16	6.064	59.83	15.46	6.946	41.7	7.947	10.56	10.75	2.848	11.47	0.676	0	0	0.506	0.02	0
101	Disturbance Value: 0.7	4316	57.76	17.44	6.306	61.5	15.09	7.263	40.67	7.548	10.44	9.126	2.68	11.89	0.712	6E-04	0.014	0.519	0.02	0
102		4117	55.9	17.16	5.689	60.79	16.07	7.526	42.81	7.542	10.77	8.521	2.897	11.74	0.594	0	0	0.549	0.019	0.011
103		4045	58.89	17.74	6.132	62.57	14.89	6.616	39.45	7.804	10.28	6.682	2.876	11.21	0.842	0.002	9E-04	0.28	0.022	0.016
104		4011	58.68	18.24	6.535	61.28	14.79	7.048	41.44	7.463	10.77	4.894	3.172	11.45	0.918	0	0.002	0.4	0.02	9E-04
105		4013	52.53	17.77	6.669	62.58	15.56	6.796	47.03	7.599	10.56	6.951	3.343	11.85	0.673	0.003	0.013	0.419	0.02	0.024
106		4000	58.68	18.24	6.535	61.28	14.79	7.048	41.44	7.599	10.56	6.951	3.343	11.85	0.673	0.003	0.013	0.419	0.02	0.024

Iteration Number		Error	PID Parameters for Six Controllers																	
			PhiAngle			ThetaAngle			PsiAngle			Z-axis			X-axis			Y-axis		
			Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
107		4183	58.83	18.57	6.342	63.18	14.99	7.31	41.68	7.382	10.07	5.044	3.206	12.13	0.674	0.002	0.016	0.355	0.022	0.02
108		4126	60.19	18.93	6.781	61.72	15.26	7.574	39.78	7.327	10.04	4.38	3.254	11.59	0.7	0.004	0.013	0.699	0.019	0.011
109		4102	61.81	18.61	6.439	63.69	14.76	6.601	38.11	8.01	9.743	3.863	3.718	12.08	0.507	0.002	0.014	0.397	0.022	0.006
110		4093	57.24	18	5.972	60.88	15.5	6.564	38.11	8.01	9.743	3.863	3.718	12.08	0.507	0.002	0.014	0.397	0.022	0.006
111		4064	58.35	18.73	6.252	64.62	15.3	7.025	42.94	8.592	9.657	3.622	3.671	12.32	0.58	0.002	0	0.489	0.02	0.012
112		4074	58.35	18.73	6.252	64.62	15.3	7.025	42.94	7.14	9.743	3.863	3.718	12.08	0.507	0.002	0.014	0.397	0.022	0.006
113		4068	58.35	18.73	6.252	64.62	15.3	7.428	38.55	8.592	9.657	3.622	3.671	12.32	0.58	0.002	0	0.489	0.02	0.012
114		4069	59.81	19.16	7.756	63.01	16.39	6.886	41.53	6.95	11.03	3.218	3.36	10.69	0.699	0	0.009	0.398	0.025	0
115		4058	58.53	19.21	6.748	63.43	15.51	7.166	44.18	7.262	10.2	3.837	4.202	11.73	0.684	1E-04	0	0.604	0.02	0
116		4054	61.08	19.3	6.898	64.69	14.77	7.087	38.95	8.177	9.6	3.318	3.36	10.69	0.699	0	0.009	0.398	0.025	0
117		4046	61.33	19.14	7.634	65.34	16.1	7.117	38.6	7.368	11.07	3.837	3.36	10.69	0.699	0	0.009	0.398	0.025	0
118		4050	59.81	18.88	6.43	64.3	15.6	7.736	38.23	8.218	11.07	3.837	3.36	10.69	0.699	0	0.009	0.398	0.025	0
119		4044	56.53	19.92	6.089	66.47	15.64	7.745	38.2	8.397	10.28	3.478	4.173	10.8	0.724	0.003	0.014	0.461	0.019	0.009
120		4036	56.53	19.92	6.089	66.47	15.64	7.745	38.2	8.397	10.28	3.478	3.36	10.69	0.699	0	0.009	0.398	0.025	0.013
121		4036	57.69	18.86	6.089	66.47	15.64	7.745	38.2	8.397	10.28	3.478	3.36	10.69	0.699	0	0.009	0.398	0.025	0.013
122		4028	61.01	19.41	6.482	69.82	15.69	7.757	40.16	8.347	9.792	4.349	3.937	11.87	0.536	0	0	0.493	0.024	0.012
123		4017	61.01	19.41	6.482	69.82	15.69	7.757	37.57	8.205	9.249	3.935	4.045	10.32	0.625	0	0	0.379	0.028	0
124		4017	61.01	19.41	6.482	69.82	15.69	7.757	37.57	8.205	9.249	3.935	4.045	10.32	0.625	0	0	0.379	0.026	0
125		4020	61.3	19.41	6.809	69.02	15.62	7.989	37.65	8.508	9.249	3.935	4.045	10.32	0.625	0	0	0.379	0.026	0
126		4014	61.3	19.41	6.809	69.02	15.62	7.989	37.65	8.508	9.249	3.935	4.045	10.32	0.625	0	0	0.397	0.028	0
127		3998	62.88	19.47	6.827	71.39	16.19	7.936	37.75	8.508	9.249	3.935	4.045	10.32	0.625	0	0	0.397	0.028	0
128		4001	62.88	19.47	6.827	71.39	16.19	7.936	37.75	8.508	9.249	3.935	4.045	10.32	0.625	0	0	0.397	0.028	0.015
129		4004	62.88	19.47	6.827	71.39	16.19	7.936	37.75	8.508	9.249	3.935	4.045	10.32	0.625	0	0	0.397	0.027	0
130		4003	61.01	19.47	6.827	71.39	16.19	7.936	37.75	8.508	9.249	3.935	4.045	10.32	0.625	0	0	0.397	0.027	0
131		4008	61.01	19.47	6.827	71.39	16.19	7.936	37.75	9.477	11.15	3.412	3.513	9.79	0.744	0.002	0	0.428	0.028	0.015
132		4002	58.15	20.08	6.067	69.58	15.65	7.049	40.28	8.959	10.69	4.006	4.194	10.72	0.668	0	0	0.385	0.025	0
133		4002	58.15	20.08	6.067	69.58	15.65	7.049	40.28	8.959	10.69	4.006	4.194	10.72	0.668	0	0	0.385	0.025	0
134		3998	59.1	20.12	6.16	69.01	15.92	7.32	40.39	8.498	10.42	3.696	3.876	10.75	0.61	0	0	0.427	0.028	0
135		3996	63.65	19.56	6.702	71.64	16.38	8.55	35.16	8.513	10.24	3.928	3.496	10.14	0.61	0	0	0.427	0.028	0
136		3998	61.48	19.64	6.906	73.06	16.33	7.993	35.44	8.757	9.642	3.785	4.621	10.19	0.518	0	0	0.438	0.026	0.02
137		3997	62.5	18.99	6.928	74.45	16.5	8.172	40.8	9.385	10.97	4.076	4.085	10.16	0.701	0	0.011	0.37	0.029	0
138		3993	56.8	20.88	5.792	71.59	16.72	8.061	34.65	8.523	10.69	4.613	3.842	11.8	0.684	0.002	0.002	0.424	0.028	0.008
139		3988	57.66	20.33	6.076	68.04	16.82	7.491	40.93	10.09	11.88	4.409	3.472	10.65	0.811	6E-04	0	0.424	0.028	0.008
140		3984	56.8	20.88	6.076	68.04	16.82	7.491	40.93	10.09	11.88	4.409	3.472	10.65	0.811	6E-04	0	0.424	0.028	0.008
141		3990	63.97	19.73	7.393	72.36	17.38	7.614	39.91	8.766	10.34	4.095	3.805	10.65	0.811	6E-04	0	0.424	0.028	0.008
142		3994	56.8	20.88	6.076	68.04	16.82	7.491	40.93	8.766	10.34	4.095	3.805	10.65	0.811	6E-04	0	0.424	0.028	0.008
143		3982	64.98	20.07	7.634	72.49	17.15	8.038	37.11	10.24	11.48	4.767	3.788	11.57	0.629	2E-04	0	0.571	0.025	0.018
144		3970	65.46	20.17	7.308	73.66	17.61	7.839	40.39	10.31	12.57	3.68	3.703	9.838	0.695	0	0	0.512	0.027	0.008
145		3978	62.28	20.17	7.308	73.66	17.61	7.839	40.39	10.31	12.57	3.68	3.703	9.838	0.695	0	0	0.512	0.027	0.008
146		3981	58.28	20.88	6.076	68.04	16.82	7.491	39.91	10.09	11.32	3.489	4.222	9.49	0.524	0	0	0.422	0.028	0.006
147		3979	64.7	20.17	7.308	73.66	17.61	7.839	40.39	10.31	10.49	3.822	3.619	10.9	0.699	0.002	0.004	0.4	0.028	0.009
148		3977	64.7	20.17	7.308	73.66	17.61	7.839	40.39	10.31	11.32	3.489	4.222	9.49	0.524	0	0	0.422	0.028	0.006
149		3976	64.7	20.17	7.308	73.66	17.61	7.839	40.39	10.31	11.32	3.489	3.703	9.838	0.695	0	0	0.512	0.027	0.008
150		3974	64.7	20.17	7.308	73.66	17.61	7.839	40.39	10.31	11.32	3.489	3.703	9.838	0.695	0	0	0.512	0.027	0.007
151		3972	62.24	21.75	6.918	72.42	16.61	8.286	42.65	10.25	9.028	4.091	3.753	10.29	0.62	7E-04	0.004	0.44	0.03	0.017
152		3968	64.7	20.17	7.308	73.66	17.61	7.839	40.39	10.31	9.028	4.091	3.753	10.29	0.62	7E-04	0.004	0.44	0.03	0.017
153		3964	60.84	21.5	6.849	73.66	17.61	7.839	40.39	10.31	9.028	4.091	3.753	10.29	0.62	7E-04	0.004	0.44	0.03	0.017
154		3965	61.73	20.87	7.031	73.66	17.61	7.839	40.39	10.31	9.028	4.091	3.753	10.29	0.62	7E-04	0.004	0.44	0.03	0.017
155		3963	67.54	20.55	7.834	74.93	18.27	7.677	41.16	9.685	10.71	3.936	3.465	9.814	0.603	0	0.004	0.576	0.025	0
156		3961	67.54	20.55	7.834	74.93	18.27	7.677	41.16	9.685	10.71	4.091	3.753	10.29	0.62	7E-04	0.004	0.44	0.03	0.017
157		3957	60.52	21.82	6.156	72.49	17.61	7.839	40.39	10.31	9.028	3.936	3.465	9.814	0.591	0	0	0.397	0.03	0
158		3960	60.52	21.82	6.156	72.49	17.61	7.839	40.39	10.31	9.028	3.936	3.465	9.814	0.591	0	0	0.429	0.028	0.004
159		3958	61.39	21.92	6.682	74.93	18.27	7.677	41.16	9.685	10.71	4.091	3.753	10.29	0.484	0	0.004	0.44	0.03	0.017
160		3951	60.7	22.17	6.254	73.66	16.76	7.823	38.56	10.25	13.04	3.792	3.678	10.18	0.645	2E-04	0	0.428	0.031	0
161		3953	60.7	22.17	6.254	73.66	16.76	7.823	38.56	10.25	13.04	3.792	3.678	10.18	0.645	2E-04	0	0.428	0.031	0.017
162		3953	59.98	21.98	6.471	73.51	17.6	7.938	39.88	10.1	11.23	4.031	3.775	10.38	0.613	0	0	0.428	0.031	0.017
163		3951	58.39	22.23	6.625	75.33	18.67	8.132	40.91	9.976	11.33	4.547	3.646	10.25	0.727	0	0	0.397	0.03	0.004
164		3953	59.27	21.91	6.374	73.16	18.69	7.601	41.33	9.298	11.1	4.222	3.647	10.37	0.524	0	0	0.509	0.034	0
165		3949	61.41	22.23	6.593	74.98	18.13	8.378	40.59	9.759	11.37	4.584	3.326	10.92	0.597	3E-04	0	0.507	0.035	0
166		3945	59.98	21.98	6.254	75.01	17.99	7.951	38.28	9.873	10.69	3.78	3.531	9.785	0.695	0	0	0.507	0.035	0
167		3951	61.62	22.32	6.768	74.87	18.09	8.002	40.56	10.2	11.53	4.34	3.326	10.92	0.597	3E-04	0	0.389	0.03	0.004
168		3942	62.71	22.74	6.597	78.96	18.2	8.961	38.04	9.336	10.8	4.34	3.326	10.92	0.597	3E-04	0	0.389	0.03	0.004
169		3940	62.71	22.74	6.597	78.96	18.2	8.961	38.04	9.336	10.8	4.34	3.326	10.92	0.597	3E-04	0	0.389	0.033	0.003
170		3942	62.71	22.74	6.597	78.96	18.2	8.961	38.04	9.336	10.8	4.34	3.326	10.92	0.597	3E-04	0	0.389	0.03	0.004
171		3941	62.71	22.74	6.597	78.96	18.2	8.961	38.04	9.336	10.8	4.34	3.326	10.92	0.597	3E-04	0	0.509	0.036	0.012
172		3937	62.71	22.74	6.597	78.96	18.2	8.961	38.04	9.336	10.8	4.34	3.326	10.92	0.597	3E-04	0	0.426	0.033	0.011
173		3942	62.71	22.74	6.471	77.74	18.45	8.309	38.26	9.341	11.33	4.222	3.496	10.27	0.62	7E-04	0	0.469	0.036	0.016
174		3938	62.71	22.74	6.471	77.74	18.45	8.309	38.26	9.341	11.33	4.222	3.326	10.92	0.597	3E-04	0	0.426	0.033	0.011
175		3936	59.79	23.5	6.214	77.96	18.44	8.254	37.97	10.2	12	4.284	3.151	10.02	0.59	0.003	0.004	0.485	0.034	0
176		3937	59.79	23.5	6.214	77.96	18.44	8.402	36.9	9.856	11.75	4.159	3.631	10.43	0.738	0	0	0.389	0.03	0.004
177		3932	59.79	23.5	6.214	77.96	18.44	8.402	39.95	9.595	11.42	4.222	3.326	10.92	0.597	3E-04	0	0.426	0.033	0.011
178		3933	59.79	23.5	6.214	77.96	18.44	8.402	39.95	9.595	11.42	4.222	3.326	10.37	0.698	0.001	0	0.422	0.031	0.02
179		3932	59.84	23.1	6.386	78.83	19.17	8.038	41.66	10.14	11.59	3.904	3.999	10.47	0.64	0	0.002	0.47	0.034	0.017
180		3934	59.84	23.1	6.386	78.83	19.17	8.038	41.66	10.14	11.59	3.904	3.999	10.47	0.64	0	0.002	0.47	0.033	0
181		3929	60.77	23.33	6.621	75.85	19.22	8.392	39.52	9.326	11.24	4.733	3.238	10.81	0.617	0	0	0.526	0.036	0.016
182		3929	60.77	23.33	6.621	75.85	19.22	8.392	39.52	9.326	11.24	4.733	3.238	10.81	0.617	0	0	0.417	0.036	0.016
183		3927	57.91	23.56	6.139	79.39	18.86	8.173	40.49	9.533	10.63	4.047	3.721	10.81	0.617	0	0	0.417	0.036	0.016
184		3928	59.84	23.1	6.386	78.83	19.17	8.038	41.66	10.14	10.63	4.047	3.721	10.81	0.617	0	0	0.417	0.036	0.016
185		3929	62.08	22.55	6.95	80.45	18.46	8.417	41.39	9.861	11.09	4.193	3.613	10.25	0.722	0	0.002	0.471	0.036	0.015
186		3929	59.84	23.1	6.386	78.83	19.17	8.038	41.66	9.861	11.09	4.193	3.613	10.25	0.722	0	0.002	0.471	0.036	0.015
187		3925	58.21	23.86	6.129	77.62	18.87	8.268	42.54	10.2	10.79	4.362	3.364	10.57	0.67	0	2E-04	0.469	0.033	0.012
188		3923	57.91	23.56	6.139	79.39	19.17	8.038	41.66	9.861	11.09	4.193	3.613	10.25	0.722	0	0.002	0.421	0.035	0.004
189		3922	58.21	23.86	6.129	77.62	19.17	8.038	41.66	9.861	11.09	4.193	3.613	10.25	0.722	0	0.002	0.421	0.035	0.004
190		3923	58.21	23.56	6.139	79.39	18.87	8.268	42.54	10.2	10.79	4.362	3.364	10.57	0.67	0	2E-04	0.417	0.036	0.016
191		3923	58.21	23.56	6.139	79.39	18.87	8.268	42.54	10.2	10.79	4.362	3.364	10.57	0.67	0	2E-04	0.417	0.035	0.004
192		3925	57.91	23.86	6.129	77.62	19.17	8.268	42.54	10.2	10.79	4.362	3.364	10.57	0.67	0	2E-04	0.417	0.035	0.004
193		3920	59.03	23.85	6.115	80.67	18.65	8.262	40.09	10.42	12.55	4.242	3.364	10.57	0.67	0	2E-04	0.417	0.035	0.004
194		3925	57.37	24.36	6.37	78.61	19.44	8.672	39.12	10.1	11.61	4.232	3.721	10.81	0.617	0	0	0.403	0.036	0.013
195		3921	59.79	24.36	6.37	78.61	19.44	8.672	39.12	10.1	11.61	4.232	3.721	10.81	0.617	0	0	0.403	0.036	0.013
196		3919	59.79	24.36	6.37	78.61	19.44	8.672	39.12	10.1	11.61	4.232	3.721	10.81	0.617	0	0	0.409	0.033	0.014
197		3919	60.24	23.66	6.28	78.84	19.39	8.075	38.96	10.46	12.22	4.232	3.721	10.81	0.617	0	0	0.409	0.033	0.014
198		3921	61.21	24.14	6.376	78.62	18.59	8.102	38.89	10.66	11.59	4.253	3.525	10.38	0.619	0	0	0.417	0.035	0.004
199		3917	61.21	24.14	6.376	78.62	18.59	8.102	38.89	10.66	11.59	4.253	3.525	10.38	0.619	0	0	0.421	0.036	0.016
200		3914	60.31	24.85	6.547	79.1	19.05	8.456	37.93	9.949	10.46	4.461	3.405	10	0.688	0	0	0.446	0.037	0.014
201		3911	59.79	24.36	6.37	78.61	19.44	8.483	39.07	10.76	11.77	4.191	3.501	10.08	0.647	5E-04	0	0.477	0.036	0.013
202		3913	59.79	24.36	6.37	78.84	19.39	8.084	42	10.76	11.59	4.253	3.525	10.38	0.619	0	0	0.446	0.037	0.014
203		3914	59.79	24.36	6.28	78.61	19.44	8.483	39.07	10.76	11.77	4.191	3.501	10.08	0.647	5E-04	0	0.477	0.036	0.013
204		3915	60.24	23.66	6.37	78.84	19.39	8.084	42	10.76	11.59	4.253	3.501	10.08	0.647	5E-04	0	0.477	0.036	0.013
205		3910	60.11	25.27	6.308	76.86	19.48	7.963	40.52	10.97	11	3.965	3.598	9.679	0.724	0	0.003	0.391	0.037	0
206		3909	60.11	24.36	6.28	78.61	19.39	8.084	42	10.76	11.59	4.253	3.501	10.08	0.647	5E-04	0	0.477	0.036	0.013

Disturbance Value: 0.75



Appx. 4. Matlab and Simulink Codes

This appendix is actually included in CD drive contains all the modeling, animation, optimization, and artificial intelligent codes.

CURRICULUM VITAE

Personal Information	
Name Surname	Fady M. ALALAMI
Place of Birth	GAZA - PALESTINE
Date of Birth	02.02 1987
Nationality	❒ T.C. ☑ Other: PALESTINIAN
Phone Number	00905550016487, 00972599310417
Email	eng.fadi.alami@gmail.com
Web Page	

Educational Information B. Sc.	
University	The Islamic University of Gaza
Faculty	Faculty of Engineering
Department	Department of Electrical Engineering
Graduation Year	10.06.2009

M. Sc.	
University	The Islamic University of Gaza
Institute	Institute of Graduate Studies
Department	Department of Electrical Engineering
Programme	Electrical Engineering Programme
Graduation Year	01.06.2013

Ph. D.	
University	Istanbul University-Cerrahpasa
Institute	Institute of Graduate Studies
Department	Department of Electrical and Electronic Engineering
Programme	Electrical and Electronic Engineering Programme
Graduation Year	28.07.2020

Publications

HAMED, B., ALAMİ, F., 2015, Adaptive Hierarchical Fuzzy controller for HVAC Systems in Low Energy Buildings, *Akademik Platform Mühendislik ve Fen Bilimleri Dergisi*, 3 (2), 1-7. DOI: 10.5505/apjes.2015.46220

Alami, F., Hussian, A., Ajlouni, N., 2020, Design and Implementation of a Multi-Stage PID Controller for Non-inertial Referenced UAV, Electrica, 10.5152/electrica.2020.20004.