ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE
ENGINEERING AND TECHNOLOGY

MODELING REAL-TIME SIMULATION AND CONTROL OF QUADROTOR VEHICLES

M.Sc. THESIS

Haci BARAN

Department of Aeronatical and Astronatical Engineering

Aeronatical Engineering Programme

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE
ENGINEERING AND TECHNOLOGY

MODELING REAL-TIME SIMULATION AND CONTROL OF QUADROTOR VEHICLES

M.Sc. THESIS

Haci BARAN
(511151166)

Department of Aeronatical and Astronatical Engineering

Aeronatical Engineering Programme

Thesis Advisor: Asst. Prof. Dr. İsmail BAYEZİT

JUNE 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

QUADROTOR ARAÇLARININ MODELLENMESİ GERÇEK ZAMANLI SİMÜLASYONU VE KONTROLÜ

YÜKSEK LİSANS TEZİ

Haci BARAN
(511151166)

Uçak Mühendisliği Anabilim Dalı

Uçak Mühendisliği Programı

Tez Danışmanı: Asst. Prof. Dr. İsmail BAYEZİT

HAZİRAN 2019

Haci BARAN, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 511151166 successfully defended the thesis entitled “MODELING REAL-TIME SIMULATION AND CONTROL OF QUADROTOR VEHICLES”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Asst. Prof. Dr. İsmail BAYEZİT** ....
Istanbul Technical University

Jury Members : **Prof. Dr. Fikret Çalışkan** ....
Istanbul Technical University

Asst. Prof. Dr. Muhammed Garip ....
Yıldız Technical University

Asst. Prof. Dr. İsmail Bayezit ....
Istanbul Technical University

Date of Submission : 03 May 2019
Date of Defense : 14 June 2019

To my family,

FOREWORD

Firstly, i would like to thank my thesis supervisor Asst. Prof. Dr. İsmail BAYEZİT for his help and support. Secondly, i would like to thank Model-SİM LAB Group for their support and understanding. Thirdly, i would like to thank PHD student Alper Yükselen for letting me use his Flight Gear quadrotor geometry. Finally, i would like to thank my family and loved ones for their help, support and understanding.

14 june 2019 Haci BARAN
(Research Assistant)

TABLE OF CONTENTS

Page

FOREWORD ix
TABLE OF CONTENTS xi
ABBREVIATIONS xiii
LIST OF FIGURES xv
SUMMARY xix
ÖZET xxi
1. INTRODUCTION 1
  1.1 Literature Review 1
2. PURPOSE OF THE THESIS 3
  2.1 Purpose 3
3. QUADROTOR DESING EXLAMPLES 5
  3.1 Old Quadrotor Helicopter Designs 5
    3.1.1 Brequet richet helicopter 5
    3.1.2 Oemnichen helicopter 6
    3.1.3 Curtis-wright VZ-7 6
  3.2 New Quadrotor Helicopter Designs 7
    3.2.1 Microdrone md4-200 7
    3.2.2 Quattrocopter 8
4. QUADROTOR MODEL 9
  4.1 Quadrotor Model Template Design 9
    4.1.1 Generating different discrete-time trajectories model 9
    4.1.2 Obtaining euler angles model 10
    4.1.3 Obtaining propeller speeds from thrust and torques model 10
      4.1.3.1 Direction cosine matrix 10
    4.1.4 Quadrotor dynamics model 11
      4.1.4.1 Basic conceps for dynamics model 11
      4.1.4.2 Vertical thrust(U1) 11
      4.1.4.3 Roll(U2) 12
      4.1.4.4 Pitch(U3) 12
      4.1.4.5 Yaw(U4) 13
      4.1.4.6 Equations of dynamics system 14
      4.1.4.7 Transfer matrix 16
    4.1.5 Linear model of the quadrotor 16
5. DISCRETE-TIME CONTROLLER 19
  5.1 Discretizing the Signals of Nonlinear and Linear Models 19
  5.2 Discrete Time PID Controller 20
    5.2.1 Discrete time integrals and derivatives of discrete-time PID controller 21

5.2.2 Z-transform for discrete-time PID controller 21
5.2.3 Discrete transfer function of integrals and derivatives for discrete-time PID controller 22
5.2.4 Discrete transfer function of discrete-time PID controller 22
5.2.5 Implementation of digital PID controller 23
5.2.6 Different methods of z-transform 23
5.2.6.1 Zero order hold 23
5.2.6.2 First order hold 24
5.2.6.3 Impulse invariant method 25
5.2.6.4 Matched method 26
5.2.6.5 Bilinear transform method 28
6. QUADROTOR TRAJECTORIES AND THEIR RELATED COMPONENTS 35
6.1 Different Types of Trajectories and Their X Y Z Components 35
6.1.1 Circular region and X Y Z components 35
6.1.2 Rectangular region and X Y Z components 39
6.1.3 Open circular trajectory and X Y Z components 43
6.1.4 Linear movement trajectory and X Y Z components 47
6.1.5 Results for linear model 51
7. FAULT TOLERANT KONTROL SYSTEMS(FTCS) ON THE QUADROTOR 55
7.1 Definition of Fault Tolerant Control Systems 55
7.2 General Approaches to Fault Tolerant Control Systems 55
7.3 Kalman Filter 56
7.4 Fault Detection and Estimation Implementations on the Quadrotor for all Quadrotor Flight Paths 59
7.4.1 Applying sensor fault to linear position components 59
7.4.2 Appying disturbance to linear velocity components and removing it with kalman filter 63
7.4.3 Appying disturbance to euler angles and removing it with kalman filter 67
7.4.4 Appying actuator fault to motor propeller angular speed components .. 70
8. VISULATION 73
8.1 Flight Gear Application On the Quadrotor 73
8.1.1 Definition of flight gear application 73
8.1.2 Flight gear simulink components 73
8.1.3 Flight gear simulink working principles 76
9. CONCLUSION 81
REFERENCES 83
CURRICULUM VITAE 85

ABBREVIATIONS

Ixx	: Moment of inertia around x-axis in body frame
Iyy	: Moment of inertia around y-axis in body frame
Izz	: Moment of inertia around z-axis in body frame
I	: Inertia matrix in body frame
J_{TP}	: total rotational moment of inertia around propeller axis
K_E	: Electric motor constant
R	: Motor resistance
N	: Gear box reduction ratio
X	: Linear position of the quadrotor helicopter along x_E WRT E-frame
$\dot{X}$	: Linear velocity of the quadrotor helicopter along x_E WRT E-frame
$\ddot{X}$	: Linear acceleration of the quadrotor helicopter along x_E WRT E-frame
Y	: Linear position of the quadrotor helicopter along y_E WRT E-frame
$\dot{Y}$	: Linear velocity of the quadrotor helicopter along y_E WRT E-frame
$\ddot{Y}$	: Linear acceleration of the quadrotor helicopter along y_E WRT E-frame
Z	: Linear position of the quadrotor helicopter along z_E WRT E-frame
$\dot{Z}$	: Linear velocity of the quadrotor helicopter along z_E WRT E-frame
$\ddot{Z}$	: Linear acceleration of the quadrotor helicopter along z_E WRT E-frame
ϕ	: Angular position of the quadrotor helicopter along x_2 WRT E-frame (roll)
$\dot{\phi}$	: Angular velocity of the quadrotor helicopter along x_2 WRT E-frame (roll)
$\ddot{\phi}$	: Angular acceleration of the quadrotor helicopter along x_2 WRT E-frame (roll)
θ	: Angular position of the quadrotor helicopter along y_1 WRT E-frame (pitch)
$\dot{\theta}$	: Angular velocity of the quadrotor helicopter along y_1 WRT E-frame (pitch)
$\ddot{\theta}$	: Angular acceleration of the quadrotor helicopter along y_1 WRT E-frame (pitch)
ψ	: Angular position of the quadrotor helicopter along z_E WRT E-frame (psi)
$\dot{\psi}$	: Angular velocity of the quadrotor helicopter along z_E WRT E-frame (psi)
$\ddot{\psi}$	: Angular acceleration of the quadrotor helicopter along z_E WRT E-frame (psi)
U_1	: Vertical thrust in body frame
U_2	: Roll torque in body frame
U_3	: Pitch torque in body frame
U_4	: Yaw Torque in body frame
Ω_1	: Front propeller speed
Ω_2	: Right propeller speed
Ω_3	: Rear propeller speed
Ω_4	: Left propeller speed

LIST OF FIGURES

Page

Figure 3.1 : Breguet-richet gyroplane no 1 6
Figure 3.2 : Oemnichen no 2 6
Figure 3.3 : Curtis-wright vz-7 7
Figure 3.4 : Microdrode md4 - 200 8
Figure 3.5 : Quattrocopter 8
Figure 4.1 : Quadrotor complete model template 9
Figure 4.2 : Quadrotor dynamics 11
Figure 4.3 : Vertical thrust command 12
Figure 4.4 : Roll command 12
Figure 4.5 : Pitch command 13
Figure 4.6 : Yaw command 13
Figure 4.7 : Linear model of the quadrotor 18
Figure 5.1 : Discrete-time PID controller 20
Figure 5.2 : Zero order hold effect on the signals 24
Figure 5.3 : Trapezoidal integral areas 32
Figure 6.1 : Closed circular trajectory 35
Figure 6.2 : closed circular trajectory position components 36
Figure 6.3 : Closed circular trajectory velocity components 36
Figure 6.4 : Closed circular trajectory acceleration components 37
Figure 6.5 : Closed circular trajectory euler angle components 38
Figure 6.6 : Closed circular trajectory euler angle rates components 38
Figure 6.7 : Closed circular trajectory propeller angular speed components 39
Figure 6.8 : Rectangular trajectory 39
Figure 6.9 : Rectangular trajectory position components 40
Figure 6.10 : Rectangular trajectory velocity components 41
Figure 6.11 : Rectangular trajectory euler angle components 41
Figure 6.12 : Rectangular trajectory euler angle rates components 42
Figure 6.13 : Rectangular trajectory propeller angular velocities 42
Figure 6.14 : Open circular trajectory 43
Figure 6.15 : Open circulars trajectory linear position components 44
Figure 6.16 : Open circulars trajectory linear velocity components 44
Figure 6.17 : Open circulars trajectory linear acceleration components 45
Figure 6.18 : Open circulars trajectory euler angle components 45
Figure 6.19 : Open circulars trajectory euler angle rates components 46
Figure 6.20 : Open circulars propeller angular velocity components 46
Figure 6.21 : Linear movement trajectory 47
Figure 6.22 : Linear movement trajectory linear position components 48

Figure 6.23 : Linear movement trajectory linear velocity components 48
Figure 6.24 : Linear movement trajectory linear acceleration components 49
Figure 6.25 : Linear movement trajectory euler angle components 49
Figure 6.26 : Linear position angular acceleration components 50
Figure 6.27 : Linear movement trajectory angular acceleration components 50
Figure 6.28 : Altitude response for linear model of the quadrotor 51
Figure 6.29 : Euler angle response for linear model of the quadrotor 52
Figure 6.30 : Roll(p), Pitch(q) and Yaw(r) rates response for linear model of the quadrotor 52
Figure 6.31 : X, Y and Z response for linear model of the quadrotor 53
Figure 7.1 : Sensor Fault Model 59
Figure 7.2 : Sensor Fault for circular trajectory 60
Figure 7.3 : Sensor fault for rectangular trajectory 60
Figure 7.4 : Sensor fault for linear trajectory 61
Figure 7.5 : Sensor fault for Circular trajectory 61
Figure 7.6 : Sensor Fault for circular trajectory x, y, and z positions 62
Figure 7.7 : Sensor Fault for rectangular trajectory x, y, and z positions 62
Figure 7.8 : Sensor Fault for open linear trajectory x, y, and z positions 63
Figure 7.9 : Sensor Fault for open circular trajectory x, y, and z positions 63
Figure 7.10 : Disturbance Model 64
Figure 7.11 : Kalman filter effect on the disturbance of Vx velocity for rectangular movement 64
Figure 7.12 : Kalman filter effect on the disturbance of Vz for rectangular movement 65
Figure 7.13 : Kalman filter effect on the disturbance of Vx velocity for open linear movement 65
Figure 7.14 : Kalman filter effect on the disturbance of Vz velocity Open Linear movement 66
Figure 7.15 : Kalman filter effect on the disturbance of Vx velocity for open circular movement 66
Figure 7.16 : Kalman filter effect on the disturbance of Vz velocity for open circular movement 67
Figure 7.17 : Kalman filter effect on the disturbance of roll angle for rectangular movement 68
Figure 7.18 : Kalman filter effect on the disturbance of pitch angle for rectangular movement 68
Figure 7.19 : Kalman filter effect on the disturbance of yaw angle for rectangular movement 68
Figure 7.20 : Kalman filter effect on the disturbance of roll angle for open circular movement 69
Figure 7.21 : Kalman filter effect on the disturbance of pitch angle for open circular movement 69
Figure 7.22 : Kalman filter effect on the disturbance of psi angle for open circular movement 70
Figure 7.23 : Actuator Fault Model 70

Figure 7.24: Applying actuator fault to motor propeller speed in rectangular trajectory 71
Figure 7.25: Applying actuator fault to motor propeller speed in open linear trajectory 71
Figure 8.1 : FlightGear run script component 74
Figure 8.2 : Flat earth to LLA component 75
Figure 8.3 : 6DOF animation component 75
Figure 8.4 : runfg.bat file........ 76
Figure 8.5 : scenery file........ 77
Figure 8.6 : Flight simulation file........ 77
Figure 8.7 : Flight simulation file........ 78
Figure 8.8 : Flight simulation file........ 79

MODELING REAL-TIME SIMULATION AND CONTROL OF QUADROTOR VEHICLES

SUMMARY

This thesis includes modeling, real-time simulation and control of the quadrotor. In order to model the quadrotor, matlab simulink blocks are used. In the quadrotor simulink model, trajectory part is formed to provide reference autonomous flight trajectory. After that, reference euler angles are given to the quadrotor for controlling euler angles for a safe flight. Then, by using quadrotor torques and thrust, motor propeller angular speeds are determined. This propeller speeds outputs are used as inputs in order to model dynamic part of the quadrotor. In the dynamic model, linear acceleration, velocity, and position are obtained with the help of corresponding equations. Moreover, euler angles are also determined by using appropriate equations.

Discrete-time PID controller are used to control the quadrotor. In the control part, linear position, euler angles and motor propeller speeds are controlled for the sake of flight. In order to discretize the signals, palse generators and zero order holds are used.

After quadrotor modeling is completed, for visualizing the quadrotor flight trajectory, a simulator called Flight Gear is used. In the Flight Gear simulator part, by using the corresponding airport information, and its' coordinates, airport scenery is downloaded first, then the scenery is loaded to Flight Gear simulator. As the last step to see how the quadrotor fly in the simulator, the simulink model is run and quadrotor simulink model trajectory is observed in the flight area.

Faults can happen in quaddrotor systems. These faults effect the quadrotor system performance and if they are not fixed, they become serious faults and system failures occurs. Disturbance faults, sensor faults and actuator faults can occur inside of the quadrotor system. In the disturbance faults, a disturbance is given to the system and system response becomes noisy. In order to remove the disturbance a Kalman Filter is used. By making fault detection, estimaton and prediction, Kalman Filter eliminates most of the disturbance and the system response turns into an acceptable response. Sensor faults are generated by using a step input for the fault. At the moment of sensor fault, the system response has a jump. However, due to quadrotor controller effect, the system response follows the same direction with a small error. In the actuator faults, step input is used for the actuator fault in the motor propeller speed. When the actuator fault is applied, the propeller speed response jumps. Yet because of the quadrotor controller effect, the speed response goes in the same direction with a small error.

To sum up, in these study, quadrotor modeling in matlab simulink, simulation in Flight Gear and control with discrete time PID controller is explained. Moreover, fault tolerant control systems are implemented to test quadrotor system robustness.

QUADROTOR ARAÇLARININ MODELLENMESİ GERÇEK ZAMANLI SİMÜLASYONU VE KONTROLÜ

ÖZET

Quadrotorlar dört pervaneli insansız hava araçlarıdır. Çok amaçlı olarak kullanılmaktadırlar. Askeri, casusluk, eğitim, araştırma gibi birçok alanda kullanılmaktadır. Quadrotorlar dört motorların pervaneleri sayesinde dikey itki, dönme, yunuslama ve sapma torkları oluşturulur. Dört pervanenin de aynı yönde dönmesiyle dikey itki oluşturulur. sol pervanenin hızı artırılarak (veya azaltılarak) ve sağ pervanenin hızı azaltılarak (veya artırılarak) dönme momenti oluşturulur ve bu moment roll ivmesine sebep olur. arka pervanenin hızı artırılarak (veya azaltılarak) ve ön pervanenin hızı azaltılarak (veya artırılarak) yunuslama momenti oluşturulur ve bu moment pitch (yunuslama) ivmesine sebep olur. ön ve arka pervanenin hızı artırılarak (veya azaltılarak) ve sol ve sağ pervanenin hızı azaltılarak (veya artırılarak) sapma momenti oluşturulur ve bu moment sapma ivmesine sebep olur. Bu dört motorun pervanelerinin dönme gücü sayesinde, quadrotorlar ileri, geri ve dikey hareket edebilirler.

Bu yüksek lisans tezi modelleme, linearizasyon, gerçek zamanlı simülasyon ve kontrol içermektedir. Öncelikle quadrotor simulink modeli oluşturulmuştur. Daha sonra bu simulink modelinin x, y ve z pozisyon bileşenleri, euler açıları ve motor pervanelerinin açısal hızları kesikli zamanlı PID kontrolörle kontrol edilir. Daha sonra bu simulink modeline hata toleranslı kontrol sistemi uygulanır. Son olarak da uçuşu gerçek zamanlı simule etmek için Flight Gear simulatör programı kullanılır.

Qaudrotoru modellemek için matlab simulink blokları kullanılmaktadır. Quadrotor simulink modelinde, quadrotora referans otonom uçuş güzergahı sağlaması için trajectory modeli oluşturulur. Bu modelde x, y ve z pozisyon bileşenleri yardımıyla dairesel, dikdörtgensel ve lineer güzergahlar çizilir. Dairesel güzergahlarda z bileşeninin artırılmasıyla istenen yükseklik elde edilirken x bileşeninin sinüs fonksiyonuyla ve y bileşeninin cosinüs fonksiyonuyla birlikte hareket ettirilerek dairesel bölgeler oluşturulur. Dikdörtgensel bölgeler oluşturmak için önce z bileşeni arttırılır. Sonra x ve y bilişenleriyle dikdörtgensel bölge oluşturulur. Daha sonra z bileşeni tekrar artırılarak x ve y bileşenleri yardımıyla yeni bir dikdörtgensel alan oluşturulur. Lineer güzergahlar oluşturmak önce z bileşeni artırılır daha sonra x ve y bileşenleriyle lineer alanlar oluşturulur.

Euler açılarının kontolü için quadrotora referans euler açıları vermek için ’Obtainin Euler Angles’ isimli bir simulink bloğu oluşturulur. Burada referans ϕ ve θ açıları oluşturulur. ψ açısı ise bu bloğa dışardan verilir. Bu açılara Euler açıları denilmektedir. referans Euler açıları trajectory modelinde oluşturulan referans güzergaha göre şekillenmektedir. Bu referans açılar güvenli bir uçuş için önemlidir.

Daha sonra quadrotor itkisi ve torqları kullanılarak referans açısal motor hızları oluşturulur. Öncelikle referans torklardan referans dikey itki olusturulur.bu itki quadrotorun referans dikey kuvveti olarak da adlandırılır. Daha sonra bu referans dikey

kuvvet ve torkların olusturduğu kontrol inputları ve bir dönüşüm matrisi sayesinde quadrotor referans motorlarının açısal hızları bulunur. Bu motor hızları, dinamik model için giriş değeri oluşturur. Dinamik modelde lineer ivmelenme, hız ve pozisyonlar uygun denklemler kullanılamak elde edilir. Ayrıca euler açıları da gerekli denklemler kullanılarak elde edilir. Öncelikle quadrotor motorlarının açısal hızları elektrksel sabitlerin bulunduğu denklemler yardımıyla bulunur. Daha sonra bulunan bu motor hızlarının bulunduğu denklemler kullanılarak dinamik modeling dikey kuvveti ile dönme, yunuslama ve sapma torkları bulunur. Bulunan dikey itkinin de yer aldığı tezin dördüncü bölümünce açıklanan lineer ivmelenme denklemi sayesinde x, y ve z yönlerindeki lineer ivmelenmeler bulunur. Lineer ivmelenme denkleminin birinci integrali alınırsa lineer hızlar, ikinci integrali alınırsa lineer pozisyonlar bulunur. Dinamik modelde bulunan dönme, yunuslama ve sapma torkları sayesinde euler açıları da elde edilir.

Dinamik modelden sonra dinamik modelden çıkan sürekli sinyalleri kesikli sinyallere dönüştürmek için sürekli sinyalleri kesikli sinyalere dönüştüren bir model oluşturulur. Bu modelde pulse generator isimli simulink bloğu kullanılır. Bu blok kesikli pulse üreterek sinyalleri kesikli hale getirir.$\ddot{\phi}$

Lineer olmayan model elde edildikten sonra lineer model elde edilir. Bunun için küçük açı yaklaşımı kullanılır. Ayrıca ϕ, θ sıfır(0) olarak alınırken ψ açısı sabit olarak alınır. Bu yüzden bunların türevi olan $\dot{\phi}\dot{\theta}$ ve $\dot{\psi}$ sıfır(0) olur Trigonometrik fonksiyonlarda cosinüsler bir(1) değerine yaklaşırken, sinüs fonksiyonları kendi açılarına yaklaşır. Lineer olmayan modeldeki lineer ivmelenme denklemlerinde eliminasyonlar olur. Ayrıca açısal oranlar (p q r) da euler açılarının türevine eşit olur. Açısal momentumlar($\ddot{\phi}\ddot{\theta}\ddot{\psi}$) ise dönme momentinin x, y ve z yönündeki bileşenlerinin bu yönlerdeki atalet momenlerine bölümüne eşittir.

Quadrotoru kontrol etmek kesikli zamanlı PID kontrolör kullanılır. Kontrolöre gelen referans sinyaller ve dinamik modelden gelen sinyallerin birbirinden çıkarılması sonucu oluşan hata kesikli zamanlı PID kontrolör yardımıyla minimize edilir. Kontrolör yardımıyla lineer pozisyonlar, euler açıları ve motor pervanelerinin açısal hızları uçuş güvenliği için kontrol edilir. Referans sinyallerle dinamik modelden gelen sinyallerin cevapları birbirleriyle kıyaslanarak kontrolörün çalışma performası belirlenir.

Quadrotorlarda hata oluşabilir. Oluşan bu hatalar müdahele edilmediği taktirde ciddi hatalara dönüşebilir ve sistem hatalarına sebep olabilir.Hata Toleranslı kontrol sistemleri düzgün çalışan sistemlerde bir arıza meydana geldiğinde o arızayı tolere edip sistemin çalışmasına devam etmesini sağlayan kontrol sistemleridir. Hatayı tespit edip çözüm bulmak için kullanılan yöntemlerden biri kalman filtresidir. Kalman filtresiyle hata estimasyonu yapılıp çözüm bulunarak sistem çalışmasına devam edilir. Kontrolörle de sensör ve akçuator hatalarının düzeltilmesinde etkilidirler. Quadrotorlarda gürültüden kaynaklanan hatalar, sensör hataları ve akçuator hataları oluşabilir. Gürültüden kaynaklanan hatalarda sisteme gürültü verilerek sistem cevabı gürültülü hale getirilir. Daha sonra gürültülü cevaba Kalman Filtresi uygulanarak gürültünün büyük bir kısmı elimine edilir. Simulink modelinde quadrotor dinamik modelinin lineer pozisyon bileşenlerine, lineer hız bileşenlerine ve euler açılarına gürültü uygulanır. Sensör hatalarında hata girişi olarak step kullanılır. Sensör hataları quadrotorun çıkış cevaplarından lineer pozisyon bileşenlerine uygulanır. Dolayısıyla sensör hatasının etkisi quadrotor uçuş güzergahında görülür. Quadrotor sistem cevabında belli bir anda meydana gelen sıçramadan sonra, kesikli zamanlı kontrolör sayesinde sistem cevabı izlemesi gereken yörüngeyi küçük bir hatayla izlemeye devam eder. Akçuator hatalarında motor pervanesinin açısal hızına step girişli hata verilir. Hatadan dolayı hızda belli bir anda sıçrama meydana geldikten sonra, hız cevabı kontrolör sayesinde aynı görüngeyi izlemeye devam eder.

Quadrotor modellemesi tamamlandıktan sonra kuadrotorun uçuşunu görsellemek için flight gear simulatörü kullanılır. Flight Gear quadrotor simulink modeli için olusturulan uçuş güzergahının quadrotor tarafından izlenip izlenmediği gerçek zamanlı görsellemek için kullanılır. Flight Gear matlab simulink uyumlu bir simülatör programıdır. Bu programa simulink modelinden gelen model bilgisi kullanılarak Quadrotor uçuşunun gerçek zamanlı simülasyonu sağlanır. Bunun için öncelikle Flat Earth to LLA bloğu girişindeki Xe girişine modeldeki x ve y pozisyon bağlantıları bağlanır. Z yüksekliği ise bloktaki href girişine bağlanır. Bu blokta quadrotor modelindeki x, y ve z pozisyonları işlenerek enlem, boylam ve yüksekliğe dönüştürülür. Bu blokta oluşturulan enlem, boylam ve yükseklik bilgileri ile quadrotordaki phi, theta ve psi açıları 6DoF Animation bloğuna bağlanır. Flight Gearı çalıştırmak için önce Flat Earth to LLA bloğuna quadrotorunuçacağı bölgenin koordinatları girilir. Daha sonra Generate Run Script bloğuna havalanı bilgileri ile uçağın geometrik bilgisi girilir. Flight Gear çalıştırıldıktan sonra quadrotor için bir trajectory kodu Generate Run Script bloğunda oluşturulur. Simulasyon kısmında görselliği sağlamak için ilgili havaalanının bilgileri ve koordinatları kullanılarak havaalanının senaryosu indirilir. İndirilen senaryo flight gear simulatörüne yüklenir. Flight Gear arayüzünde quadrotor ve uçacağı bölge görülür. Son olarak simulink model çalıştırılır ve uçuş simulatörde görülür.

1. INTRODUCTION

This thesis is about modeling, simulation and control of the quadrotor. Quadrotor is a kind of unmanned aerial vehicle (UAV). These vehicles have four rotors to produce required thrust and torques. Thanks to power of these motors, the quadrotor moves forward, backward and vertically. This vehicle is also called quadrotor helicopter. the helicopter have also four propellers which gererate roll, pitch, yaw torques and vertical thrust depending on their speed rates, and turning directions.

The quadrotors have been used for years. By passing time, the design of these vehicles changed and developped. These UAVs are very effective especially in monitoring, and recording. For example, in military areas, these helicopters are used to observe the enemy actions with the cameras attached to them. Via the cameras, the vehicles record also the movements of the enemies.

1.1 Literature Review

The quadrotor has a deep history. There are a lot of work done about the four rotors helicopter in the past. These studies were combined and they improved the helicopter system. In the below sentenses there are some examples for the studies about the quadrotor.

Tommaso Bresciani explains the modeling and control of the quadrotor in [12]. This article particularly includes modeling method, controller type and experimental part for the quadrotor modeling. Francesco Sebatino obtains mathematical for the quadrotor dynamics in [13]. After forming the quadrotor dynamic model, this model is linearized and a linear controller is modelled for the linear model. After forming the whole model, the system response is simulated. Bora Erginer focused on the modeling and control of the quadrotor in [14]. Sumaila Musa reviews the most powerful methods for the quadrotor modeling and control in [15]. Ankyda Ji and Kamran Turkoglu explain quadrotor dynamics modeling, integration and corresponding hardwares platforms, state-space modeling for dynamics of the plant, system identification and model verification in [17]. Vladimir Dobrokhodov and Noel Du Toit give information about two essential goals. First of all, their thesis improve a 6DOF flight dynamics model of a multi-copter UAV with high correctness and implementation of the linear attitude controller onboard of an industrial quadcopter. Second of all, it makes stronger the weakly joint efforts mathWorks and the open-source community to build a software setup providing rapid control software prototyping in [18]. Gopalakrishnan Eswar Murthi gives information about non-linear Quadrotor model controller designing. gradient descent method is used to tune the controller in this study [19]. Qasim Muhammed explains modeling of the dynamic system and attitude control of Quadrotor helicopter in [20]. Michael David Schmidt interprets about the ANGEL project(Aerial Network Guided Electronic Lookout) which takes a system engineering approach to the design, development, testing and implementation of a quadrotor [21]. Pritpal Singh illustrates the quadrotor dynamic model and model-autonomous control system. He also explains the complete improved quadrotor description that is used for monitoring goal with real time object detection in [21]. SANG MIN OH gives information about quadrotor helicopter modeling and control in [21].

2. PURPOSE OF THE THESIS

2.1 Purpose

The purpose of this thesis is modeling, real-time simulation and control of a four rotor quadrotor vehicle. In order to determine the control method and mathematical model, the system template model has to be obtained. The aim of modeling is to obtain the complete matlab simulink model of the quadrotor. Modeling the system mathematically is very crucial to have a good model. Many systems' dynamics like electrical, mechanical and biological systems consist of differential equations. In this thesis, by using Newton and Euler equations, the quadrotor model is obtained.

After modeling the quadrotor, a controller is aimed to be designed for the system. For this quadrotor, discrete time PID and PD controllers are purposed to be designed. Quadrotor linear position components, euler angles and motor propeller angular speeds are desired to be controlled with the controller.

After modeling the nonlinear model of the quadrotor, the nonlinear model is aimed to be linearized and this linear model will be controlled with the discrete-time PID controller.

Moreover, fault tolerant control system is purposed to be implemented on the nonlinear model. To illustrate, sensor faults, actuator faults and disturbances are applied to the quadrotor system responses.

Furthermore, in order to provide real-time simulation, Flight Gear simulator will be applied to the quadrotor simulink model.

To sum up, in this thesis, the nonlinear and linear quadrotor systems and controllers will be completed via matlab and simulink. Then, the nonlinear model will be exposed to sensor faults, actuator faults and disturbances. Finally, in order to see how the quadrotor fly, Flight Gear simulator will be implemented.

3. QUADROTOR DESING EXLAMPLES

3.1 Old Quadrotor Helicopter Designs

Human beings have been interested in quadrotor hekicopters from the beginnings of 20. century and this interest has grown up by time untill today. In the beginning, for the purpose of acommodate people's flying desire, quadrotor helicopters are designed with a big size. However, by time, with the effect of improving electronical control systems and the necessity for unmanned aerial vehicles, these helicopters were designed unmanned with a small size.

3.1.1 Brequet richet helicopter

In the beginning of 20. century, a scientist and academician named Charles Richet made a small unmanned helicopter. Despite this small umanned vehicles couldn't fly, one of the student of Charles Richet called Louis Brequet who becomes one of the aviation pioneer was influenced from this little helicopter [1]. In 1906, Louis Brequet and his brother Jacques Brequet started helicopter experiments under Professor Richet's guide.

In 1907, the brothers succeed to make their first helicopter which was able to carry human being. That helicopter had four rotors and was named Gyroplane No.1. The main purpose for this four rotor helicopter was to fly itself under the pilot control.

In this helicopter, while opposite propellers turned in the same direction, the others turned in the opposite direction. That's why the total torque on the quadrotor helicopter became zero. The helicopter's empty weight was 510 kg which consist of 345 kg gross tare weight and 165 kg motor with its' fuel. The brequet brothers found a 68 kg pilot and he was sitting in the center of the helicopter under the motor. With the weight of the pilot, the quadrotor helicopter's net weight was 578 kg.

The brothers firstly tried the helicopter in the summer of 1907. The quadrotor was tied with a rope and its' only action was vertical take off. During the experiment, the helicopter had a 1.5 meter altitude with the pilot.

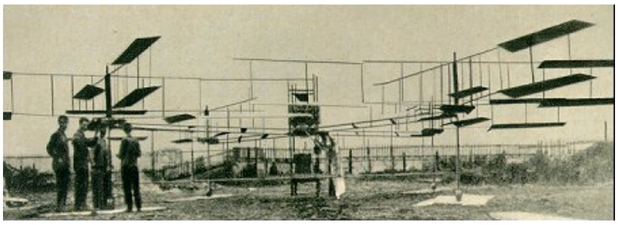


Figure 3.1 : Breguet-richet gyroplane no 1

3.1.2 Oemnichen helicopter

Etienne Oemnichen [2] started the rotorcraft experiments in 1920 and designed six VTOL vehicles. The most popular one was Oemnichen No.2 which had 4 rotors and 8 propellers. When this helicopter flied, it had a 120hp rotorcraft motor. Then, that motor was exchanged with a 180hp motor. That helicopter had 4 propellers at the end of its' arms. the propellers were being bended to change their angles. There was also small propellers on that helicopter. The one mounted on the helicopter nose was being used as rudder to determine the vehicles' direction. The 5 propellers on the horizontal axis were used to balance the helicopter laterally. The last 2 propellers' task were to push the vehicle on the front side. Figure 3.2 indicates an example for Oemnichen No.2.

Oemnichen No 2 made a lot of experimetal flights and reached an acceptable stability and controllability level. This helicopter achieved a record by reaching 360m m range. The helicopter can fly 1 km within 7 minutes and 40 seconds.

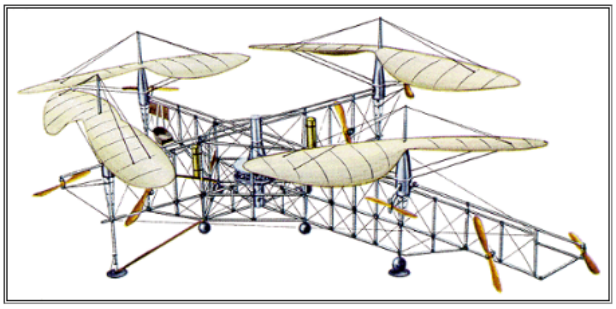


Figure 3.2 : Oemnichen no 2

3.1.3 Curtis-wright VZ-7

Curtis-Wright VZ-7 [3] had a very simple structure. It was designed to help the military with transportation needs. This helicopter had a square shape. The helicopter center structure included pilot chair, flight controllers, fuel and oil tanks and the vehicle's one of the turbin motor. The quadrotor helicopter also had a simple control system. Forward movements were controlled by adjusting each propeller's thrust and yaw angle was controlled via moving wings on the motor exhaust. Figure 3.3 shows the quadrotor while flying.

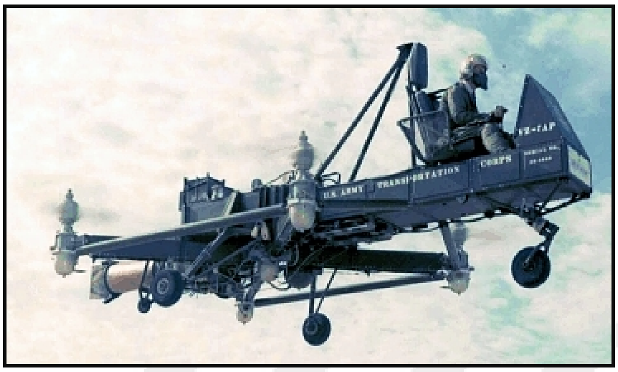


Figure 3.3 : Curtis-wright vz-7

3.2 New Quadrotor Helicopter Designs

Nowadays, the quadrotors are designed as unmanned. The ones which can carry human beings are not designed anymore. The quadrotors are put into unmanned VTOL rotorcrafts. There are different kinds of quadrotor types that are explained below.

3.2.1 Microdrone md4-200

This quadrotor [4] was designed in 2006 firstly. It has accelerometer, gyroscope, air pressure sensor, humidity sensor and temperature sensor. Via these tools this drone provides altitude, latitude and heading. Moreover, it has a GPS module that enable staying in the current state and reference point. Figure 3.4 illustrates an example of Microdrone md4-200. This drone has at least 20 minutes flight time.

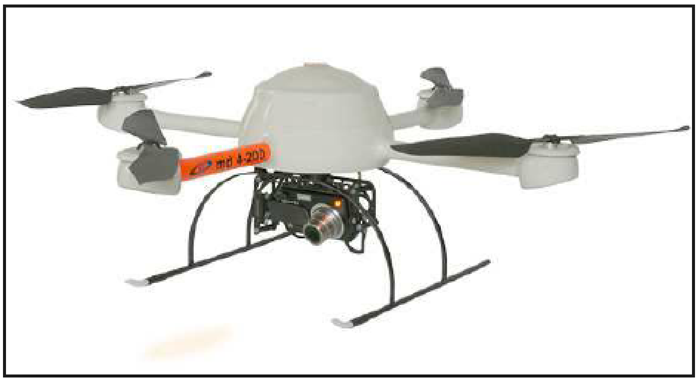


Figure 3.4 : Microdrode md4 - 200

3.2.2 Quattrocopter

Quattrocopter [5] has 550 gram weight and 65 cm length. It can fly autonomously. Moreover, this quadrotor can fly about 20 minutes with 1 km range. With the help of its' GPS, accelerometer and gyroscope, this quadrotor can make position control. Moreover, this drone has air data sensor, IMU, R/C receiver, A/D converter and power amplifier. Figure 3.5 shows an example of Quattrocopter.

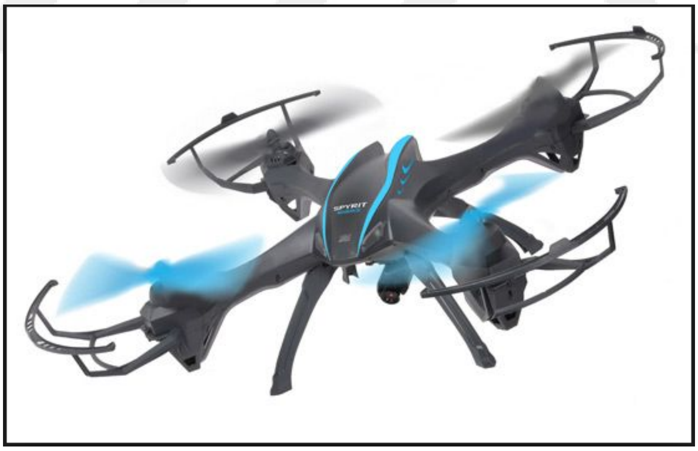


Figure 3.5 : Quattrocopter

4. QUADROTOR MODEL

4.1 Quadrotor Model Template Design

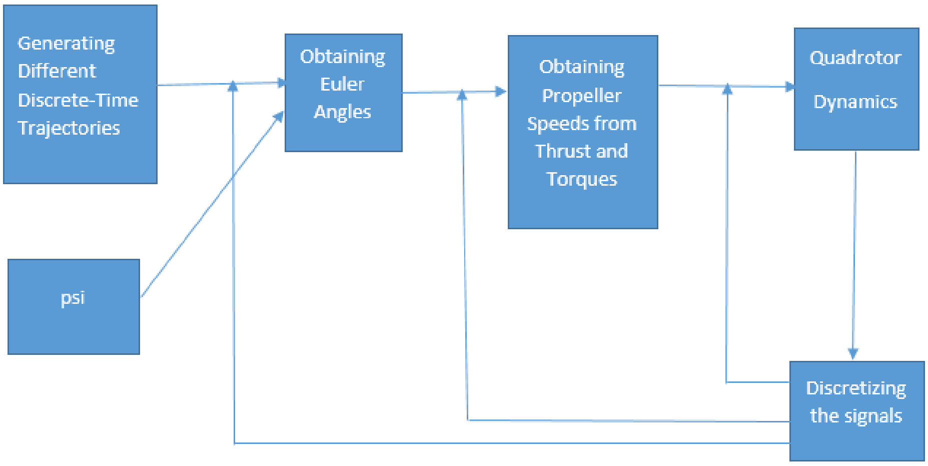

Generating Different Discrete-Time Trajectories
psi
Obtaining Euler Angles
Obtaining Propeller Speeds from Thrust and Torques
Quadrotor Dynamics
Discretizing the signals

Figure 4.1 : Quadrotor complete model template

In the figure 4.1, the quadrotor complete model template is illustrated. From the template, first of all, in the generating different discrete-time trajectories model, a discrete reference autonomous trajectory is formed by determining x, y, and z components for the quadrotor helicopter. In the second model, euler angles are drived. Psi angle is obtained as constant from outside. In the third model, motor propeller speeds are obtained from vertical thrust, roll, pitch, and yaw torques. In the Quadrotor Dynamics model, linear positions, velocities, accelerations are obtained. Moreover, Angular positions, velocities and accelerations are also derived. In the last model named discretizing the signals, the continuous signals coming from dynamics model are discretized for the discrete-time PID controller.

4.1.1 Generating different discrete-time trajectories model

In this model, Quadrotor trajectory is composed. The reference trajectory is formed with the help of reference position components which are standed for xd, yd, and zd. On the other hand, since the trajectory is shaped depending on time, this trajectory provides an autonomous flight for the quadrotor.

4.1.2 Obtaining euler angles model

In this model, reference euler angles are acquired for the quadrotor helicopter. While psi angle is given from outside as a constant, phi and theta angles are derived by using the formulas as follows:

$$\phi = arcsin(\frac{d_x S_\psi - d_y C_\psi}{d_x^2 + d_y^2 + (d_z + g)^2}) \tag{4.1}$$

$$\theta = arctan(\frac{d_x C_\psi - d_y S_\psi}{d_z + g}) \tag{4.2}$$

Where d is the distance from center of the mass to the motors.

4.1.3 Obtaining propeller speeds from thrust and torques model

In this model, vertical thrust, roll, pitch and yaw torques represented with U_1, U_2, U_3, and U_4 respectively are achieved first. Then by using these values, motor propeller angular speeds are acquired with the help of the formula below:

$$\Omega_1^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_3 - \frac{1}{4d}U_4 \tag{4.3}$$

$$\Omega_2^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \tag{4.4}$$

$$\Omega_3^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_3 - \frac{1}{4d}U_4 \tag{4.5}$$

$$\Omega_4^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \tag{4.6}$$

4.1.3.1 Direction cosine matrix

Direction cosine matrix which is used to define the relationship between inertia frame and body frame. By using direction cosine matrix, moving from inertia frame to body frame is performed. This is important because calculations are easier and simulation outputs show better results.

$$R_{x\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}, R_{y\theta} = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}, R_{z\psi} = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{4.7}$$

By multiplaying them, direction cosine matrix is obtained.

$$R_{x\phi}R_{y\theta}R_{z\psi} = R_{\phi\,\theta\,\psi} = \begin{bmatrix} c_{\psi}c_{\theta} & -s_{\psi}c_{\phi}+c_{\psi}s_{\theta}s_{\phi} & s_{\psi}s_{\phi}+c_{\psi}s_{\theta}c_{\phi} \\ s_{\psi}c_{\theta} & c_{\psi}c_{\phi}+s_{\psi}s_{\theta}s_{\phi} & -s_{\psi}s_{\phi}+s_{\psi}s_{\theta}c_{\phi} \\ s_{\theta} & c_{\theta}s_{\phi} & c_{\theta}c_{\phi} \end{bmatrix} \tag{4.8}$$

4.1.4 Quadrotor dynamics model

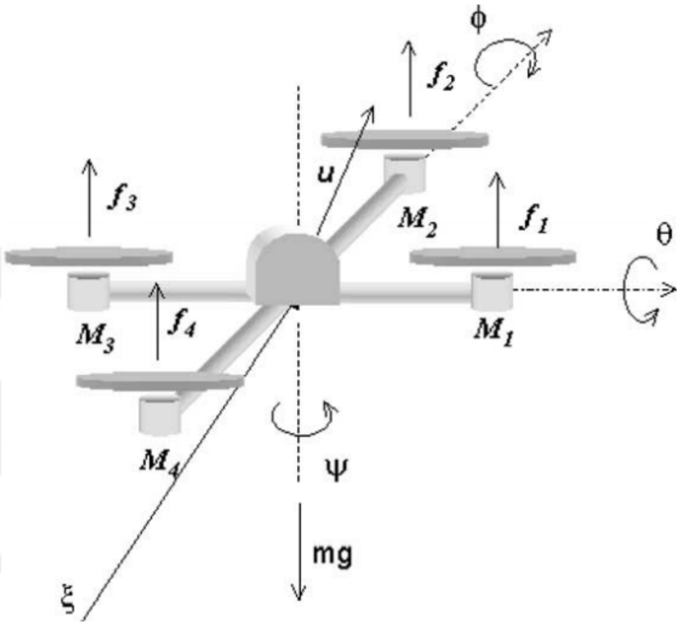


Figure 4.2 : Quadrotor dynamics

In the figure 4.2 quadrotor dynamic forces and moments are shown.

4.1.4.1 Basic conceps for dynamics model

There are four basic quadrotor movements which provide a certain height and attitude for the helicopter. These four movements are descripted as follows:

4.1.4.2 Vertical thrust(U1)

By increasing (or decreasing) all the propeller speeds with the same amount. It generates a vertical force WRT body-fixed frame which increases or decreases the quadrotors' height. If the helicopter has a horizontal position, inertial frame and body-fixed frame vertical directions coincide. Or else, vertical and horizontal accelerations in the inertial frame are generated by the throttle command. Figure 4.3 demonstrate the throttle command on a quadrotor sketch. In this command the propellers speed for each one is equal $\Omega H + \Delta A$.

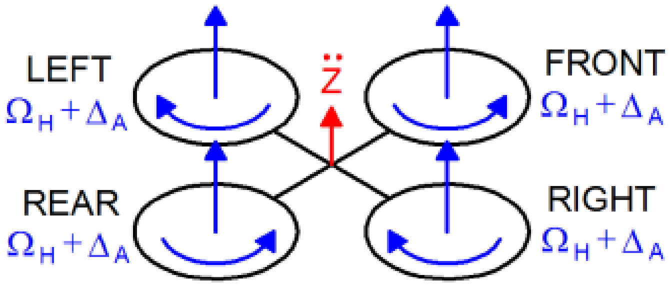


Figure 4.3 : Vertical thrust command

4.1.4.3 Roll(U2)

The roll command is obtained by raising (lowering) the left propeller speed and lowering (or raising) the right propeller. This command causes a body frame torque in x direction that enables the quadrotor turn. The total vertical thrust for roll command is equal to that one for hovering command. That's why this command produces only a roll angle acceleration. Figure 4.4 demonsrates the roll command on a quadrotor sketch.

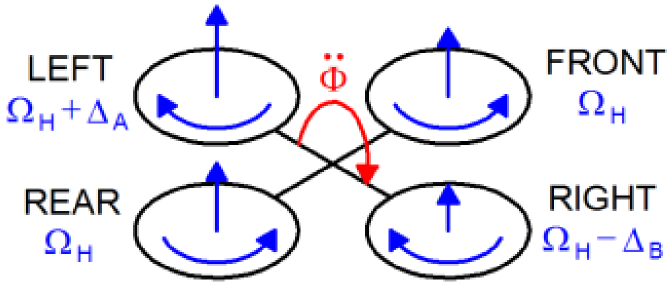


Figure 4.4 : Roll command

4.1.4.4 Pitch(U3)

Pitch command is obtained by raising (or lowering) the rear propeller speed and by lowering (or raising) the front propeller. This command causes a body frame torque in y axis direction that enables a turning movement for the quadrotor helicopter. For pitch command,the oveall vertical thrust is equal to that one in hovering command. Thus, this command generates only a pitch angle acceleration.

4.1.4.5 Yaw(U4)

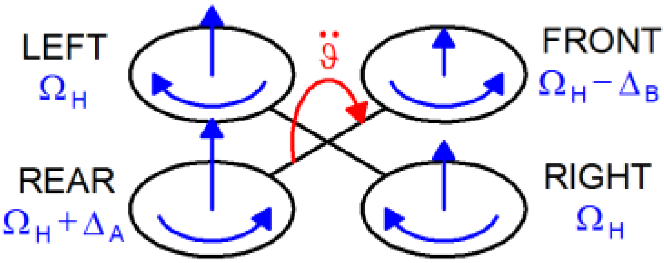


Figure 4.5 : Pitch command

In order to enable yaw command, the front-rear propeller's speed is raised or (lowered) and the left-right propeller's speed is lowered (or raised). This command leads to a body frame torque in z axis direction that provide a turning movement for the quadrotor. By rotating the left-right propellers in clockwise direction and front-rear ones in counterclockwise direction, the yaw movement is provided. That's why, when the overall torque is unbalanced, the quadrotor turns on itself around body frame z axis direction. Figure 4.6 demonstrates the yaw command on a quadrotor helicopter sketch. A and B parameters which are seen in the sketch below, are selected to maintains the vertical thrust in order to keep it stay the same and they can't have too large values. Moreover, for small values of $\Delta A, \Delta B \sim \Delta A$.

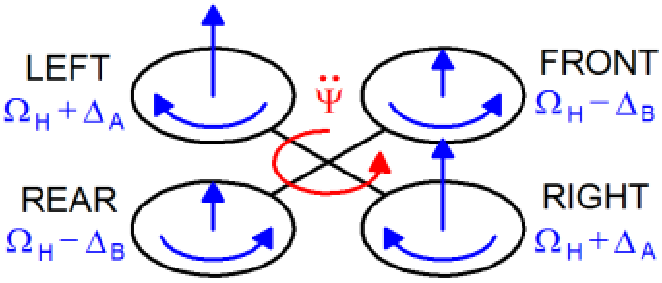


Figure 4.6 : Yaw command

4.1.4.6 Equations of dynamics system

Linear acceleration equations for inertia frame are as follows. By taking their integral, linear speeds are obtained.

$$\dot{u} = (ur - wg) + gsin(\theta) \tag{4.9}$$

$$\dot{v} = (wp - ur) - gcos(\theta)sin(\phi) \tag{4.10}$$

$$\dot{w} = (uq - vp) - gcos(\theta)sin(\phi) + \frac{U_1}{m} \tag{4.11}$$

Angular acceleration equations for inertia frame are as follows. By taking their integrals roll rate (p) pitch rate(q) and yaw rate(r) are found.

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{J_{TP}}{I_{xx}} q\Omega + \frac{U_2}{I_{xx}} \tag{4.12}$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} pr - \frac{J_{TP}}{I_{yy}} p\Omega + \frac{U_3}{I_{yy}} \tag{4.13}$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{U_4}{I_{yy}} \tag{4.14}$$

Motor propeller acceleration equations for inertia frame are as follows. By taking the accelerations' integrals, the propeller angular speeds are obtained. When these propellers start to turn, required thrust, roll, pitch and yaw torques are generated. These thrust and torques provide movement for the quadrotor. With the help of thrust and torques, different kinds of trajectories can be generated according to the propellers' angular speed ratios.

$$\dot{\Omega}_1 = \frac{K_E K_M n N N \Omega_1}{R J_{TP}} - \frac{d}{J_{TP}} (\Omega_1)^2 + \frac{K_M n N v_1}{R J_{TP}} \tag{4.15}$$

$$\dot{\Omega}_2 = \frac{K_E K_M n N N \Omega_2}{R J_{TP}} - \frac{d}{J_{TP}} (\Omega_2)^2 + \frac{K_M n N v_2}{R J_{TP}} \tag{4.16}$$

$$\dot{\Omega}_3 = \frac{K_E K_M n N N \Omega_3}{R J_{TP}} - \frac{d}{J_{TP}} (\Omega_3)^2 + \frac{K_M n N v_3}{R J_{TP}} \tag{4.17}$$

$$\dot{\Omega}_4 = \frac{K_E K_M n N N \Omega_4}{R J_{TP}} - \frac{d}{J_{TP}} (\Omega_4)^2 + \frac{K_M n N v_4}{R J_{TP}} \tag{4.18}$$

Vertical thrust equation for inertia frame is as follows. Vertical thrust enables lift and this lift increases the altitude. The omegas stand for the propellers' angular speeds.

$$U_1 = (\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{4.19}$$

Roll torque equation for inertia frame is as follows. Roll torque provides roll movement for the quadrotor during the flight.

$$U_2 = lb(-\Omega_2^2 + \Omega_4^2) \tag{4.20}$$

Pitch torque equation for inertia frame is as follows. Pitch torque provides pitch movement for the quadrotor.

$$U_3 = lb(-\Omega_1^2 + \Omega_3^2) \tag{4.21}$$

Yaw torque equation for inertia frame is as follows. Yaw torque enables lateral movement and the direction changes.

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{4.22}$$

Overall propeller speed equation for inertia frame is as follows:

$$\Omega = b(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) \tag{4.23}$$

Linear acceleration equations for body frame are as follows. In order to obtain linear positions from the accelerations, their integrals has to be taken twice. Moreover, linear speeds are found by taking the accelerations' integrals once.

$$\ddot{X} = (sin\psi sin\phi + cos\psi sin\theta cos\phi)\frac{U_1}{m} \tag{4.24}$$

$$\ddot{Y} = (-cos\psi sin\phi + sin\psi sin\theta cos\phi)\frac{U_1}{m} \tag{4.25}$$

$$\ddot{Z} = -g + (cos\theta cos\phi)\frac{U_1}{m} \tag{4.26}$$

4.1.4.7 Transfer matrix

The transfer matrix is used to provide connection between the angular velocities in the earth frame and that of in the body – fixed frame.

$$\begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & \dfrac{sin\phi}{cos\theta} & \dfrac{cos\phi}{cos\theta} \end{bmatrix} \tag{4.27}$$

4.1.5 Linear model of the quadrotor

The quadrotor model is linearized near hover position since this position is the most stabilized position.

Equilibrium hover configuration relations are as follows. By using small angle approximation, the following relations are obtained.

$$\phi = \theta = 0 \tag{4.28}$$

$$\psi = \psi_0 \tag{4.29}$$

$$\dot{\phi} = \dot{\theta} = \dot{\psi} = 0 \tag{4.30}$$

Linearization of trigonometric functions are as follows.and Small angle approximation provides that while cosine functions of theta and phi close to 1, sine functions of theta and phi close to the related angle.

$$cos\phi \approx 1 \tag{4.31}$$

$$cos\theta \approx 1 \tag{4.32}$$

$$sin\phi \approx \phi \tag{4.33}$$

$$sin\theta \approx \theta \tag{4.34}$$

Linear momentum equation of motion near hover position is as follows. As seen in the equations, phi and theta angles are seen by themselves as radians after using small angle approximation.

$$\ddot{x} = (\theta cos\psi + \phi sin\psi)\frac{U_1}{m} \tag{4.35}$$

$$\ddot{y} = (\theta sin\psi - \phi cos\psi)\frac{U_1}{m} \tag{4.36}$$

$$\ddot{z} = -mg + U_1 \tag{4.37}$$

Angular rates near hover position is as follows. The following relations are acquired and roll(p), pitch(q) and yaw(r) rates are found depended on roll, pitch and yaw derivatives.

$$p = \dot{\phi} \tag{4.38}$$

$$q = \dot{\theta} \tag{4.39}$$

$$r = \dot{\psi} \tag{4.40}$$

Angular momentum equation near hover position is as follows. The momentum relations are found depended on the roll command derivatives taken in the x, y, and z axis as indicated in the following relations.

$$\ddot{\phi} = \frac{U_{2x}}{I_{xx}} \tag{4.41}$$

$$\ddot{\theta} = \frac{U_{2y}}{I_{yy}} \tag{4.42}$$

$$\ddot{\psi} = \frac{U_{2z}}{I_{zz}} \tag{4.43}$$

Figure 4.7 shows linear model of the quadrotor. The quadrotor model is linearized near hover position.

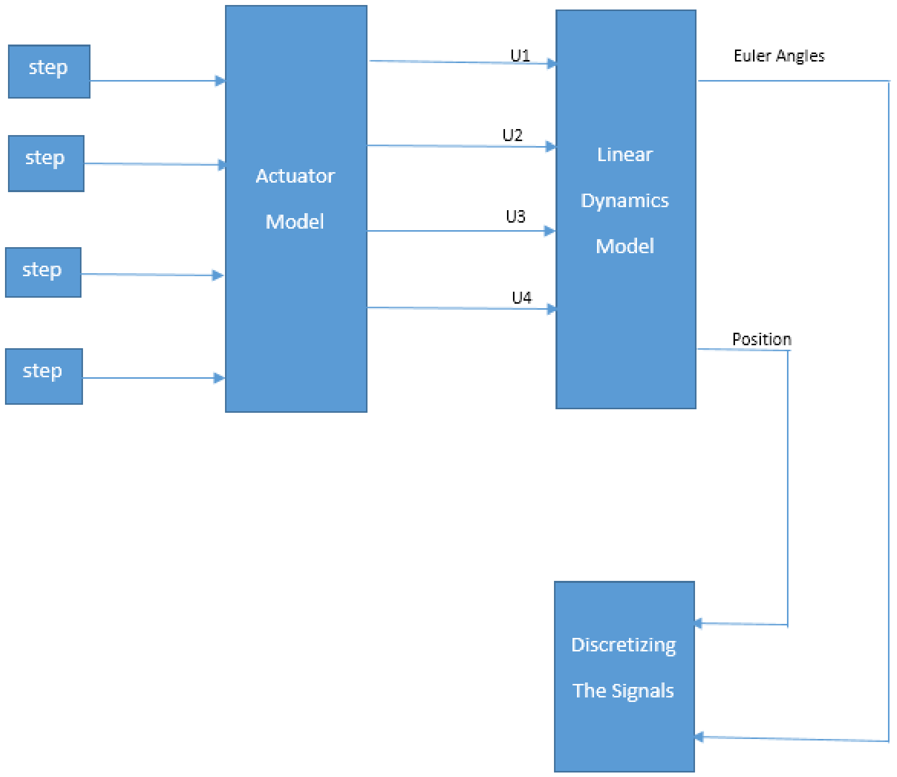


Figure 4.7 : Linear model of the quadrotor

5. DISCRETE-TIME CONTROLLER

5.1 Discretizing the Signals of Nonlinear and Linear Models

Since the discrete-time PID controller is used in this quadrotor model, the quadrotor dynamics model output signals should be discretized. For this purpose, a pulse generator is used. Pulse generators discretize the signals by generating pulses between short time intervals. Discretized outputs of this model is used as feedbacks for the controller.

In order to discretize a system, different methods are applied. To examplify, a continuous controller may be designed first, then the continuous controller is discretized. Or, the continuous plant is discretized first, then the discrete controller is designed. Both of the methods use z transform and give the equivalent discrete systems. There is an example about z transform below:

$$G(s) = \frac{1}{s+1} \tag{5.1}$$

then

$$g(t) = e^{-t} \tag{5.2}$$

By using z transform

$$Z(g[k]) = G(z) \tag{5.3}$$

$$= \frac{Z}{Z - e^{-T}} \tag{5.4}$$

If T = 0.1

then

$$G(z) = \frac{Z}{Z - 0.9048} \tag{5.5}$$

There are five methods to discretize the continuous signals. These methods are zero order hold, first order hold, Impulse, Tustin(bilinear transform) and matched. The method choosen depends on the goal to accomplish.

5.2 Discrete Time PID Controller

The discrete-time closed –loop PID controller has the greatest fame among controller systems. Its' time response and rise time is faster than the continuous-time PID controller. In continuous-time of any digital controller, the stability must be checked. However, digital inputs are used for the computer after using z-transform to convert the system to a digital control system. There are some differences between continuous–time controller and discrete–time controller. Firstly, a continuous–time controller is designed in the s–domain while a discrete–time controller is designed in z–domain. Continuous–time controller can be converted to discrete–time controller by using zero order hold, first order hold, impulse, tustin, and matched approximations. Zero order hold is default and used if the model has both continuous-time and discrete-time signals.

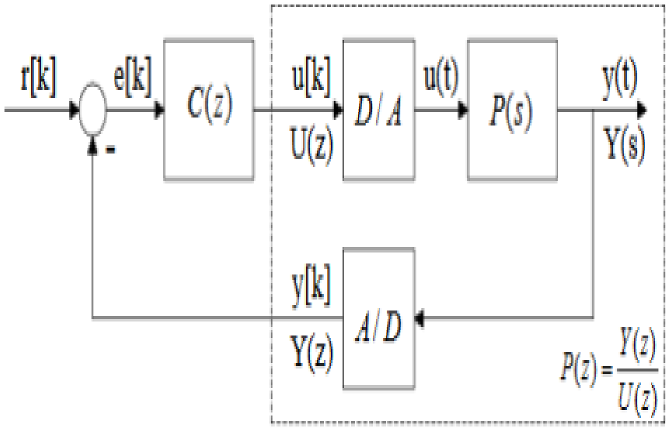


Figure 5.1 : Discrete-time PID controller

In figure 5.1, the measurement from system through is discretized and quantized with a digital to analog converter. The discrete measurements are compared to discrete set points to generate an error term. The error term is fed into a discrete controller which is represented in the Z domain and finally the discrete commands are converted back to the continuous domain with a digital to analog converter. In other word, so as to transform a controller language (digital) to a process language (analog), a digital to analog converter (DAC) is used. A common method to do that just with zero order hold. This takes the last commanded value and just holds it there until a new command is issued. In order to turn the plant output into the discrete-time signal, an analog to digital converter is often used. In other word, For transforming from a process language to a digital controller language, an analag to digital converter (ADC) is used.

5.2.1 Discrete time integrals and derivatives of discrete-time PID controller

Discrete-time derivatimes are as follows:

$$y(k) = \frac{f(k) - f(k-1)}{T_s} \tag{5.6}$$

Discrete-time integrals(trapezoidal) are as follows:

$$\int_0^{kT_s} y(t)dt = y(k) \tag{5.7}$$

$$\approx y(k-1) + \frac{f(k) - f(k-1)}{2} T_s \tag{5.8}$$

5.2.2 Z-transform for discrete-time PID controller

$$X[z] = Zx[k] \tag{5.9}$$

$$= \sum_{k=1}^{\infty} z^{-k} \tag{5.10}$$

Time-shifting property is as follows:

Zx[k-n] = z$^{-n}X[z]$

where Zx[k] = X[z]

5.2.3 Discrete transfer function of integrals and derivatives for discrete-time PID controller

Discrete transfer function of derivative is as follows:

$$y(k) = \frac{f(k) - f(k-1)}{T_s} \tag{5.11}$$

$$y(k) = \frac{F(z) - z^{-1}F(z)}{T_s} \tag{5.12}$$

$$\frac{Y[z]}{F[z]} = \frac{z-1}{zT_s} \tag{5.13}$$

$$y(k) = y(k-1) + \frac{f(k) - f(k-1)}{2}T_s \tag{5.14}$$

$$Y[z](1 - z^{-1}) = \frac{T_s}{2}F[z](1 + z^{-1}) \tag{5.15}$$

$$\frac{Y[z]}{F[z]} = \frac{T_s}{2}\frac{z+1}{z-1} \tag{5.16}$$

5.2.4 Discrete transfer function of discrete-time PID controller

To solve the differential equations for illustrating the output response of the control system in discrete-time PID controller, Z-transform is implemented.

$$\frac{U[Z]}{E[Z]} = K_p + K_i\frac{T_s}{2}\frac{T_s z+1}{z-1} + K_d\frac{z-1}{zT_s} \tag{5.17}$$

$$\frac{U[Z]}{E[Z]} = \frac{K_p(z^2 - z) + K_i\frac{T_s}{2}(z^2 + z) + \frac{K_p}{T_s}(z^2 - 2z + 1)}{z^2 - z} \tag{5.18}$$

$$\frac{U[Z]}{E[Z]} = \frac{(K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s})z^2 + (-K_p + K_i\frac{T_s}{2} - \frac{2K_d}{T_s})z + \frac{K_d}{T_s}}{z^2 - z} \tag{5.19}$$

5.2.5 Implementation of digital PID controller

$$\frac{U[Z]}{E[Z]} = \frac{(K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s})z^2 + (-K_p + K_i\frac{T_s}{2} - \frac{2K_d}{T_s})z + \frac{K_d}{T_s}}{z^2 - z} \tag{5.20}$$

$$\frac{U[Z]}{E[Z]} = \frac{(K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s}) + (-K_p + K_i\frac{T_s}{2} - \frac{2K_d}{T_s})z^{-}1 + \frac{K_d}{T_s}}{1 - z^{-}1} \tag{5.21}$$

$$U[Z] = z^{-}1[k-1] + aE[z]bz^{-}1E[z] + cz^{-}2E[z] \tag{5.22}$$

$$u[k] = u[k-1] + aE[k]bE[k-1] + ce[z] \tag{5.23}$$

$$a = (K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s}) \tag{5.24}$$

$$b = (-K_p + K_i\frac{T_s}{2} - \frac{2K_d}{T_s}) \tag{5.25}$$

$$c = \frac{K_d}{T_s} \tag{5.26}$$

5.2.6 Different methods of z-transform

5.2.6.1 Zero order hold

For the systems where the discrete values are constant between samples, zero order hold method would want to be used so that the staircase input produces an exact match between the discrete and continuous time domain systems or to tie it back to the controller design. It is desired that discrete controller to behave like continuous controller when subjected to the same types of inputs and in the case, the inputs are not impulses but step functions that are generated by a zero order hold. Therefore, it is desired that discretization method to include the effects of that zero order hold as well as discretizing the system.

In a system, discrete signals are sent to zero order hold to convert them to a continuous-time signal. This continuous-time signal is fed into the continuous plant which has continuous output. Finally, the plant output is sampled at some sample time to get discrete output of the plant. With the block diagram represented in the figure 5.2, it is tried to represent the zero order hold, the plant and the sampler all together as a Z domain transfer function.

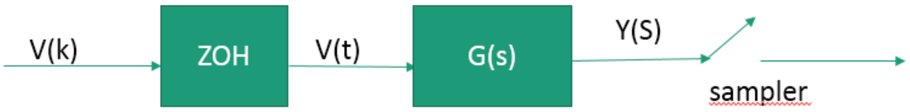


Figure 5.2 : Zero order hold effect on the signals

As seen in the figure 5.2, by using zero order hold, discrete time signal turns into continuous-time signal. At the end of the continuous signal, a sampler is used to discretize the signal for the feedback.

z transform transfer function for zero order hold is as follows:

$$x_1(t) = \int_0^t \delta(t - T - \tau) g_1(\tau) d\tau = g_1(t - T) \tag{5.27}$$

$$X(s) = \frac{1 - e^{-Ts}}{s} G(s) \tag{5.28}$$

$$X(s) = \frac{(1 - e^{-Ts}) G(s)}{s} \tag{5.29}$$

$$G_1(s) = \frac{G(s)}{s} \tag{5.30}$$

$$X(s) = (1 - e^{-Ts}) G_1(s) \tag{5.31}$$

$$Z[g_1(t)] = G_1(z) \tag{5.32}$$

$$Z[x_1(t)] = Z[g_1(t - T)] = z^{-1} G_1(z) \tag{5.33}$$

$$\begin{aligned} X(z) &= Z[G_1(s) - e^{-Ts} G_1(s)] = Z[g_1(t)] - Z[x_1(t)] \\ &= G_1(z) - z^{-Ts} G_1(z) = (1 - z^{-1}) G_1(z) \\ &= Z[X(s)] = (1 - z^{-1}) Z \frac{G(s)}{s} \end{aligned}$$

$$X(z) = Z[X(s)] = (1 - z^{-1}) Z \frac{G(s)}{s} \tag{5.34}$$

5.2.6.2 First order hold

z transform transfer function for first order hold is as follows:

$$h(t) = (1 + \frac{t}{T}) u(t) - \frac{t - T}{T} u(t - T) - u(t - T) \tag{5.35}$$

$$
\begin{aligned}
H(s) &= (\frac{1}{s}+\frac{1}{Ts^2})-\frac{1}{Ts^2}e^{-Ts}-\frac{1}{s}e^{-Ts} \\
&= \frac{1-e^{-Ts}}{s}+\frac{1-e^{-Ts}}{Ts^2}
\end{aligned}
$$

$$
H(s) = (1-e^{-Ts})\frac{Ts+1}{Ts^2} \tag{5.36}
$$

Unit step laplace transform is as follows:

$$
X^*(s) = Z[X(s)] = \frac{1-e^{-Ts}}{s} \tag{5.37}
$$

$$
\begin{aligned}
G_{h1}(s) &= Z[X(s)] = \frac{1-e^{-Ts}}{s} \\
&= (1-e^{-Ts})^2\frac{Ts+1}{Ts^2}
\end{aligned}
$$

$$
G_{h1}(s) = (\frac{1-e^{-Ts}}{s})^2\frac{Ts+1}{T} \tag{5.38}
$$

$$
X(s) = (\frac{1-e^{-Ts}}{s})^2\frac{Ts+1}{Ts^2} \tag{5.39}
$$

$$
X[z] = (1-z^{-1})^2 Z\frac{Ts+1}{Ts^2}G(s) \tag{5.40}
$$

5.2.6.3 Impulse invariant method

Impulse invariant method converts a continuous-time signal into a discrete-time signals by using the transformations below:

$$
H(s) = H(z) \tag{5.41}
$$

$$
\frac{c}{s-b} = \frac{T_s c}{1-az^{-1}} \tag{5.42}
$$

$$
\frac{c}{(s-b)^2} = T_s^2\frac{caz^{-1}}{(1-az^{-1})^2} \tag{5.43}
$$

$$
\frac{c}{(s-b)^3} = T_s^3\frac{caz^{-1}(1+az^{-1})}{2(1-az^{-1})^3} \tag{5.44}
$$

$$
\frac{c}{(s-b)^4} = T_s^4\frac{caz^{-1}(1+az^{-1}+a^2z^{-2})}{6(1-az^{-1})^4} \tag{5.45}
$$

5.2.6.4 Matched method

This techique is also called the pole-zero mapping or pole-zero matching method The working principle of this technique is to match all poles and zeros of the s-plane to the z-plane areas $z = e^{sT}$. The sample interval for this method is $T = \frac{1}{f_s}$

The general formula for this technique is shown below:

$$H(z) = \frac{U(z)}{E(z)} = k\frac{\prod_{k=1}^{m}(z - z_k)}{\prod_{i=1}^{n}(z - p_i)} \tag{5.46}$$

The main subjective for matched method is to convert the continuous domain into the discrete domain. This is called mapping. The matched method is about taking each pole and zero of continuous domain(s) and move them to the correct spot in the discrete domain(z).

The dynamics of Linear Time Invariant system is defined by the locations of its' poles, zeros and DC gain. There are four main steps for matched method :

step 1: mapping each pole/zero to the z-plane
step 2: add zeros at infinity if necessary
step 3: Removing zeros at infinity to make a strictly proper function
step 4: Adjusting the gain

An example about matched method is shown below:

$$G(s) = \frac{3}{s^2 + 3s + 2} = 3\frac{1}{(s+1)(s+2)} \tag{5.47}$$

DC gain is equal to 3

the poles are -1 and -2. Since there are in the left half plane(LHP), the system is stable.

The matched poles and zeros in the z domain for sample time $T = 1$ is found as below:

$$Z = e^{sT}$$

$$then\,for\,s_1 = -1$$

$$z_1 = e^{-1*1} = 0.3679$$

$$for\,s_2 = -2$$

$$z_2 = e^{-2*1} = 0.1353$$

$$So\;G(z) = \frac{1}{(z-0.3679)(z-0.1353)} \tag{5.48}$$

step 2: add zeros at infinity

$$G(s) = \frac{3}{s^2+3s+2} \tag{5.49}$$

there are hidden zeros in the s domain transfer function above. So, z domain transfer function is formed as below:

$$G(z) = \frac{(z-1)(z-1)}{(z-0.3679)(z-0.1353)} \tag{5.50}$$

step 3: making the transfer function strictly proper

Real life systems are casual. The output depends on past or current inputs, not future inputs. Casual systems have equal or fewer zeros than poles Strictly proper systems have only fewer finite zeros than poles. Then discrete-time transfer function becomes

$$G(z) = \frac{(z-1)}{(z-0.3679)(z-0.1353)} \tag{5.51}$$

By removing one of the zeros from the transfer function.

step 4: Fixing the gain

$$DCgain\frac{3}{s^2+3s+2} = DCgain\frac{z+1}{(z-0.3679)(z-1353)}K \tag{5.52}$$

By setting s to 0 and z to 1 the equivalent becomes:

$$\frac{3}{2} = \frac{z+1}{0.6321 * 0.8647} K$$
$$\text{so} \quad K = 0.4099$$

then the discrete-time transfer function becomes:

$$G(z) = \frac{(z+1)0.4099}{(z-0.3679)(z-1353)}$$

5.2.6.5 Bilinear transform method

This method takes advantage of trapezoid areas constituted by average of the selected rectangles used in the forward and backward rectangular rule. This method is known three names which are Bilinear Transform, Tustin Method and Trapezoidal Integration Method. Trapezoidal Integration Method is not as popular as the others.

The mathematical model for the transform is below:

$$u(kT_s) = u((k-1)T_s) + T_s \frac{e(kT_s) + e((k-1)T_s)}{2} \tag{5.53}$$

Discrete equivalent of H(s) is achieved with the help of z transform

$$\begin{aligned} U(z) &= z^{-1}U(z) + \frac{T_s}{2}((E(z) + z^{-1}E(z)) \\ (1 - z^{-1})U(z) &= \frac{T_s}{2}(1 + z^{-1})E(z) \end{aligned}$$

$$\frac{E(z)}{U(z)} = \frac{2}{T_s}\frac{z-1}{z+1} \tag{5.54}$$

Which indicates that the expression below should be implemented in order to turn the continuous signal into the discretize signal.

$$S \leftarrow \frac{2}{T_s}\frac{z-1}{z+1} \tag{5.55}$$

Transform from z plane to w plane is shown as below:

$$z = \frac{w+1}{w-1} \tag{5.56}$$

which proves that

$$w = \frac{z+1}{z-1} \tag{5.57}$$

Which maps the inside of the unit circle in the z plane into the left half of the w plane.

The Inverse Bilinear Transform may be optained as follows:

$$\begin{aligned} S &= \frac{2}{T}\frac{z-1}{z+1} \\ \frac{T}{2}S &= \frac{z-1}{z+1} \\ \frac{T}{2}S &= \frac{z-1}{z+1} \\ \frac{T}{2}Sz + \frac{T}{2}S &= z-1 \\ z(\frac{T}{2}S-1) &= -\frac{T}{2}S-1 \end{aligned}$$

$$z = \frac{1+\frac{T}{2}S}{1-\frac{T}{2}S} \tag{5.58}$$

For s = -1+j and T = 2 as sample time

$$\begin{aligned} z &= \frac{1+\frac{2}{2}S}{1-\frac{2}{2}S} \\ &= \frac{1+\frac{2}{2}(-1+j)}{1-\frac{2}{2}(-1+j)} \end{aligned}$$

$$z = \frac{j}{2+j} \tag{5.59}$$

Moreover the whole system can be mapped(or converted to z domain) By using T = 0.1 seconds as sample time

$$\begin{aligned} G(s) &= \frac{2}{s+1} \\ s &= \frac{2}{0.1}\frac{z-1}{z+1} \\ G(s) &= \frac{2}{20\frac{z-1}{z+1}+1} \end{aligned}$$

$$G(s) = \frac{2z+2}{21z-19} \tag{5.60}$$

the z transform may be converted(mapped) back to the s transform as follows:

$$z = \frac{1+\frac{s}{20}}{1-\frac{s}{20}} \tag{5.61}$$

If the z relation is substituted into the s relation below

$$s = \frac{2}{0.1}\frac{z-1}{z+1}$$

$$G(s) = \frac{2\frac{1+\frac{s}{20}}{1-\frac{s}{20}}+2}{21\frac{1+\frac{s}{20}}{1-\frac{s}{20}}-19} = \frac{2}{s+1} \tag{5.62}$$

The z relation in z transform can be derived as follows:

$$z = e^{sT}$$

$$Then$$

$$G(z) = \frac{1}{z - 0.1} = \frac{1}{e^{sT} - 0.1}$$

e^{sT} is nonlinear. That's why it should be linearized with the help of series below

$$e^{x} = \sum_{n=0}^{\infty} \frac{x^{n}}{n!}$$

$$z = e^{sT} = e^{sT} e^{sT} = \frac{e^{sT}}{e^{-sT}}$$

$$e^{sT} = 1 + \frac{sT}{2} + \frac{sT^{2}}{8} + ...$$

$1 + \frac{sT}{1}$ is the first order approximation for z

z = 1+sT

and a general relation for first order approximation of z is shown as follows:

$$z = \frac{a + bs}{c + ds}$$

A more accurate first order approximation for e^{sT} is derived as below:

$$\frac{e^{sT}}{e^{-sT}} = \frac{1 + \frac{sT}{2} + \frac{sT^{2}}{8} + ...}{1 - \frac{sT}{2} + \frac{sT^{2}}{8} - ...}$$

$$z = \frac{1 + \frac{sT}{2}}{1 - \frac{sT}{2}} \tag{5.63}$$

This is a more accurate first order approximation of e^{sT} which stands for z

Another method is using trapezoidal integration that can be approximated as follows:

$$y(x) = \int_{x_0}^{x_1} f(x)dx$$

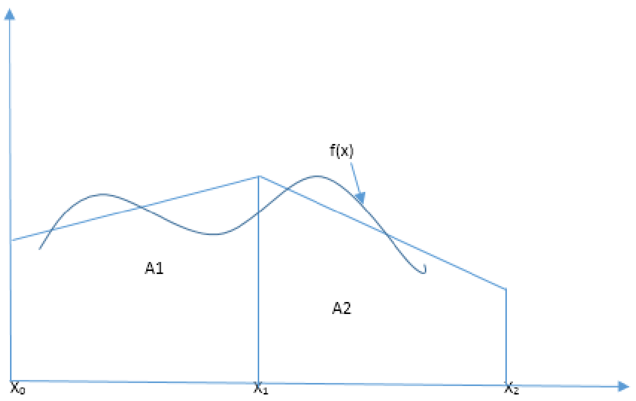


Figure 5.3 : Trapezoidal integral areas

As shown in figure 5.3, A1 and A2 indicates areas.

$$y(x_1)=\int_{x_0}^{x_1} f(x)dx=(x_1-x_2)[\frac{f(x_0)+f(x_1)}{2}]$$
$$y(x_2)\approx(x_2-x_1)[\frac{f(x_1)+f(x_2)}{2}]+y(x_1)$$

If this relation is substituted into a system

$$G(s)=\frac{Y(s)}{X(s)}$$
$$y(t)=\int_0^t x(t)dt$$

By breaking up with sample periods represented with T

$y(t)=\int_0^{kT} x(t)dt$ this relation is still continuous since ka doesn't have to be an integer.

$$y(x_2)\approx(x_2-x_1)[\frac{f(x_1)+f(x_2)}{2}]+y(x_1)$$

by generalizing this relation for any point x$_k$

$$
\begin{aligned}
y_k &\approx \Delta[\frac{f(x_1)+f(x_2)}{2}]+y_{k-1}\\
y(kT) &= \int_{kT-T}^{kT} f(t)dt + \int_{0}^{kT-T} f(t)dt\\
y(kT) &\approx T[\frac{x(kT)+x(kT-T)}{2}]+ykT-T\\
y_k &= y_{k-1}+\frac{T}{2}(x_k+x_{k-1})\\
y_k-y_{k-1} &\approx \frac{T}{2}(x_k+x_{k-1})
\end{aligned}
$$

By taking z transform

$$
\begin{aligned}
Y(Z)(1-z^{-1}) &= \frac{T}{2}X(Z)(1+z^{(}-1))\\
\frac{Y(Z)}{X(Z)} &= \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}\\
\frac{Y(S)}{X(S)} &= \frac{1}{s} = \frac{T}{2}\frac{z+1}{z-1}
\end{aligned}
$$

$$
s \approx \frac{T}{2}\frac{z+1}{z-1} \tag{5.64}
$$

The bilinear transform is an approximation of differential equations using trapezoidal integration.

Warping process may be applied to the bilinear transform in the frequency domain. The relation for warping process of bilinear transform can be derived as follows:

$$
\begin{aligned}
z &= \frac{1+\frac{sT}{2}}{1-\frac{sT}{2}}\\
z &= \frac{1+\frac{jw_nT}{2}}{1-\frac{jw_nT}{2}}
\end{aligned}
$$

By converting back using $z = e^{sT}$

$$
e^{jw_nT} = \frac{1+\frac{jw_nT}{2}}{1-\frac{jw_nT}{2}}
$$

$$W_d = \frac{1}{Tj} ln(\frac{1+\frac{jw_nT}{2}}{1-\frac{jw_nT}{2}}) \tag{5.65}$$

6. QUADROTOR TRAJECTORIES AND THEIR RELATED COMPONENTS

6.1 Different Types of Trajectories and Their X Y Z Components

6.1.1 Circular region and X Y Z components

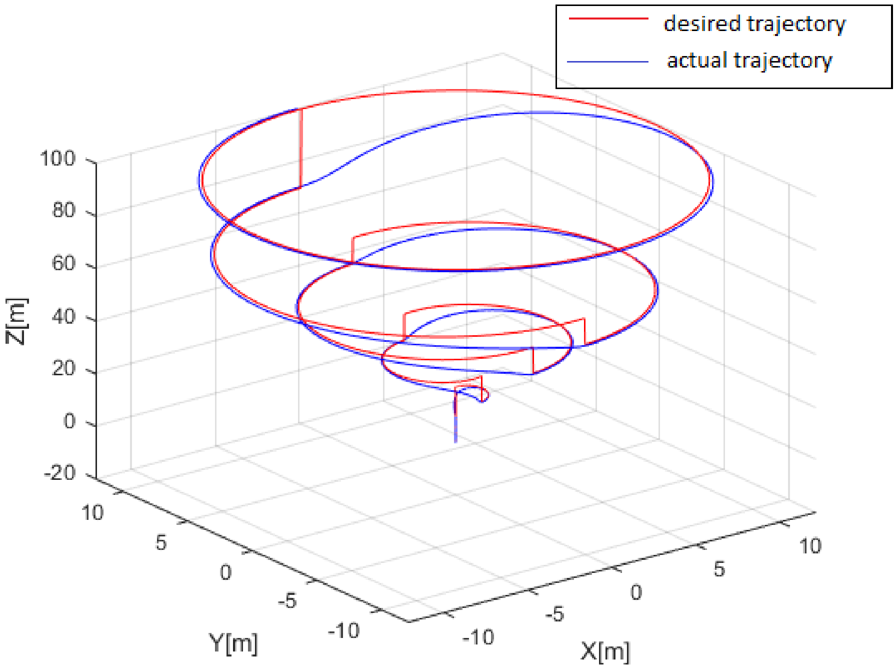


Figure 6.1 : Closed circular trajectory

In the figure 6.1, the quadrotor makes a closed circular region while rising. In this trajectory, the quadrotor is drawing circles and increasing the height at the same time. As it can be observed, desired and actual trajectories are following each other.

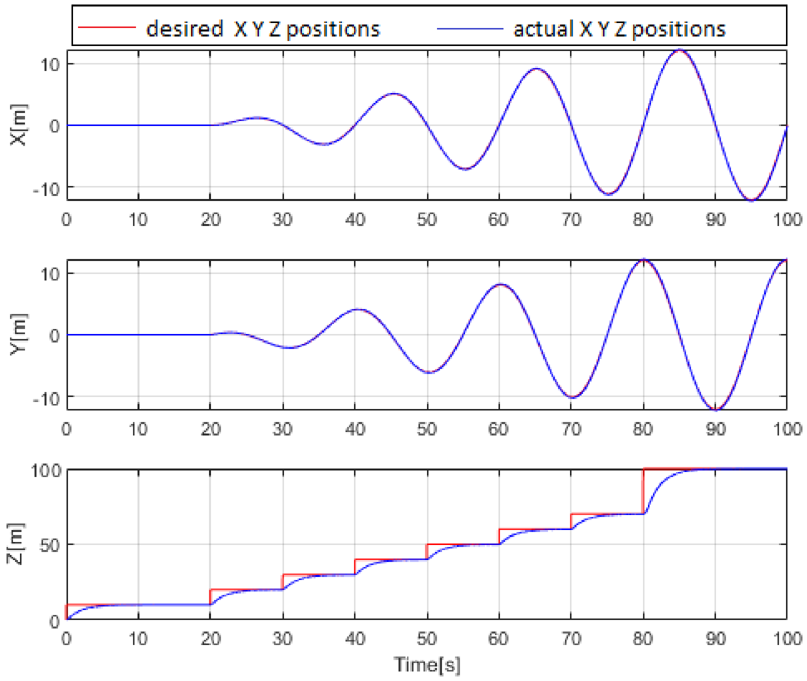


Figure 6.2 : closed circular trajectory position components

In figure 6.2, X Y Z components are shown for the closed circular region trajectory. In this components, while Z component is changes almost linearly, X and Y components are changing with sine and cosine fuctions respectively. As it is seen from the figure, desired and actual position components follow each other.

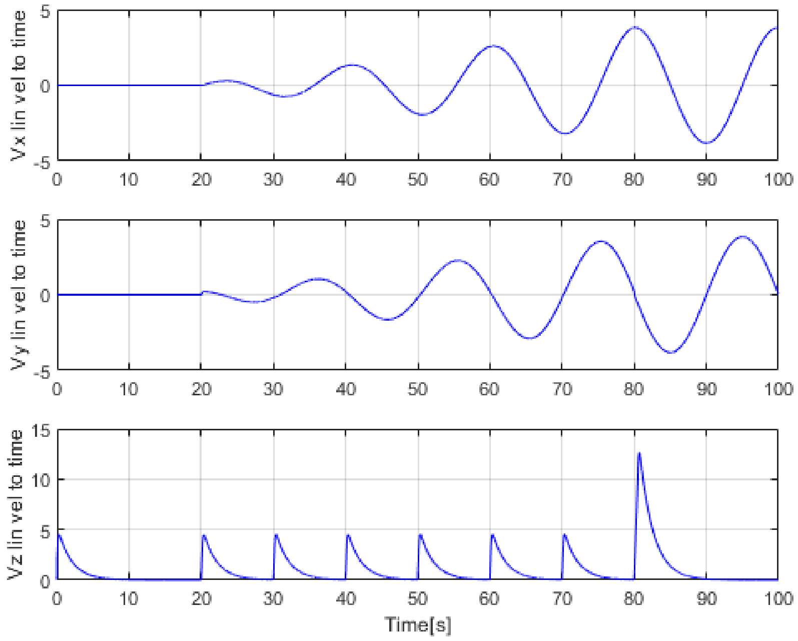


Figure 6.3 : Closed circular trajectory velocity components

In the figure 6.3, linear velocity components Vx Vy Vz are are shown. While Vx and Vy are fluctuating, Vz component increases vertically and decreases exponentially.

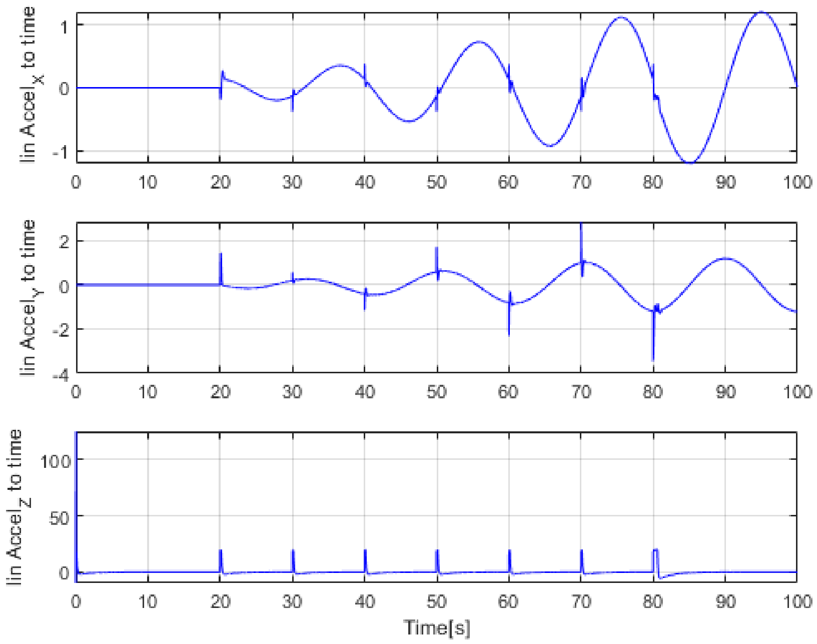


Figure 6.4 : Closed circular trajectory acceleration components

In the figure 6.4, linear acceleration components are illustrated for circular trajectory. Depending on the Vx and Vy linear velocity components, accelerations in x and y directions are fluctuating and jumping on some specific moments. Acceleration in z direction generally stays fix, however, jump on the specific moments.

In figure 6.5, while euler angle components phi and theta are changing with sine and cosine functions respectively, psi component rises first, then stays fix during the flight. As it can be observed, desired and actual components follow each other.

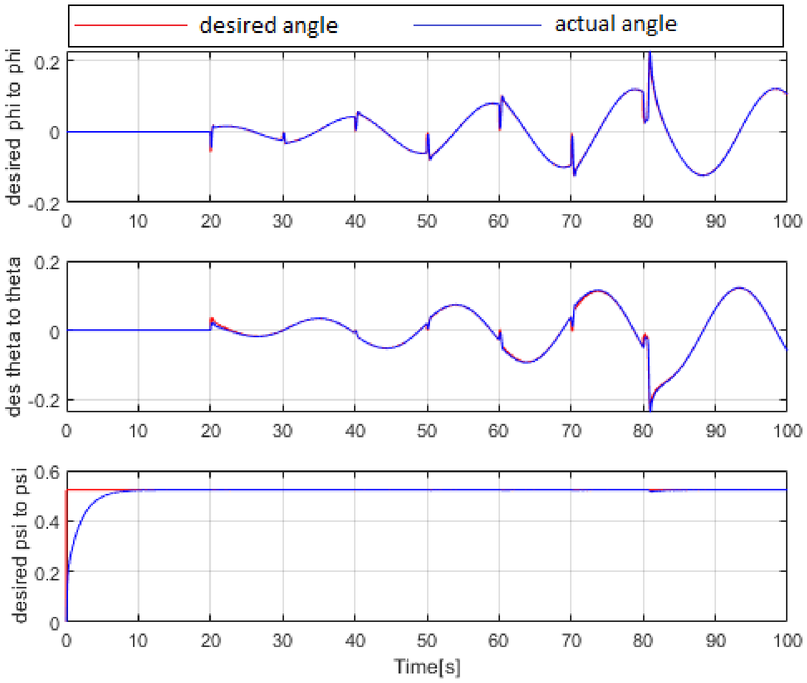


Figure 6.5 : Closed circular trajectory euler angle components

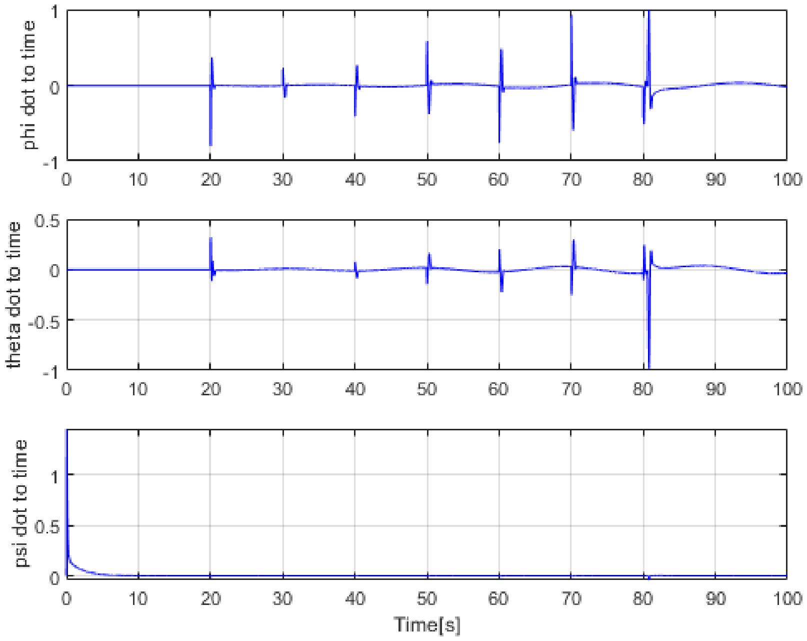


Figure 6.6 : Closed circular trajectory euler angle rates components

In the figure 6.6, the angular acceleration components for circular trajectory is shown. Accelerations in x and y directions stay stable except changing on some particular times. Beside, acceleration in z direction decreases to zero and stay in this value until at the end of the flight.

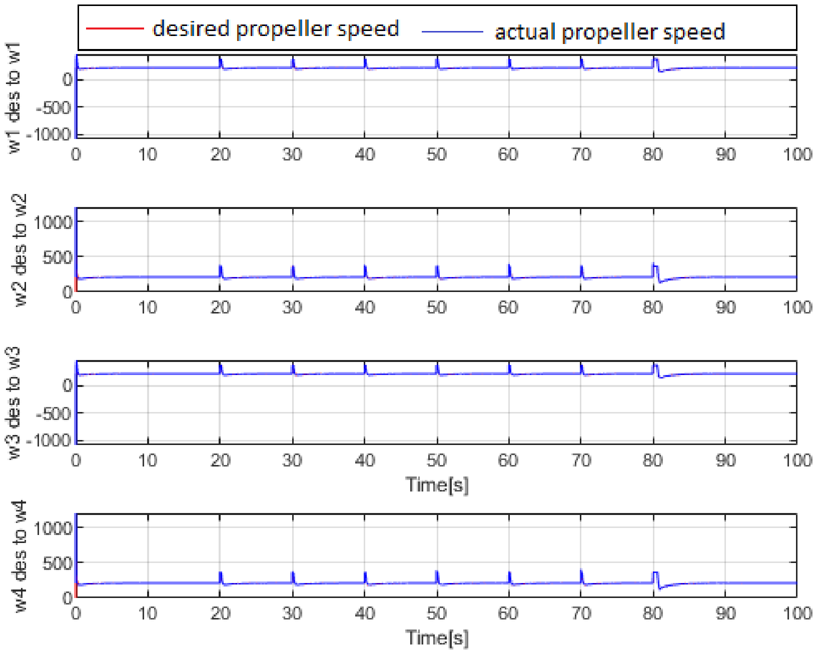


Figure 6.7 : Closed circular trajectory propeller angular speed components

In figure 6.7, propeller speed components are illustrated. All four components rise until a specific value, then remain fix during the flight. As it can be understood, desired and actual propeller speed components follow each other.

6.1.2 Rectangular region and X Y Z components

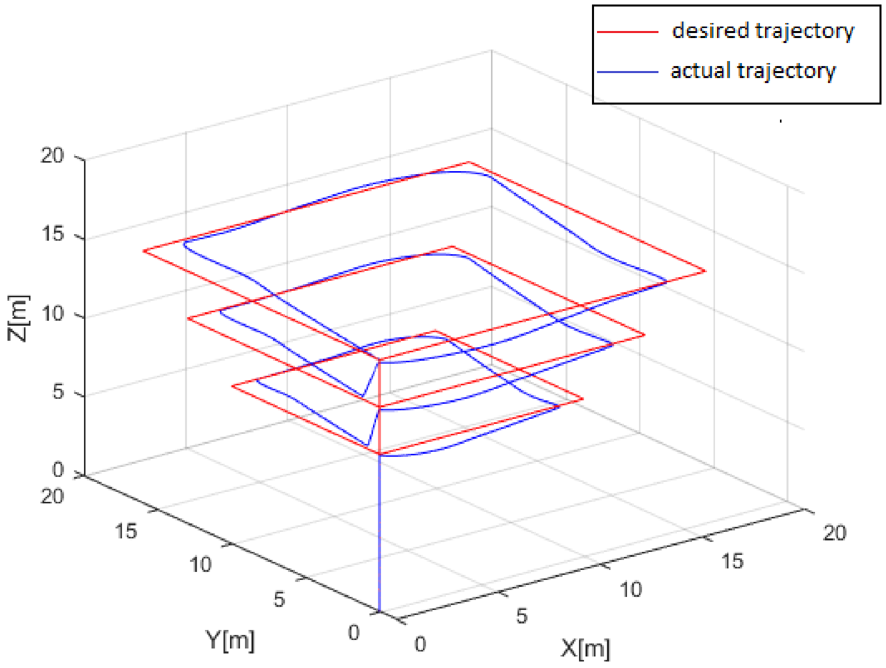


Figure 6.8 : Rectangular trajectory

In figure 6.8, the quadrotor makes rectangular region trajectory. In this trajectory, the quadrotor helicopter is rose first, then makes a rectangular region. after drawing the first rectangular region, the quadrotor increases the height and makes another rectangular region again.

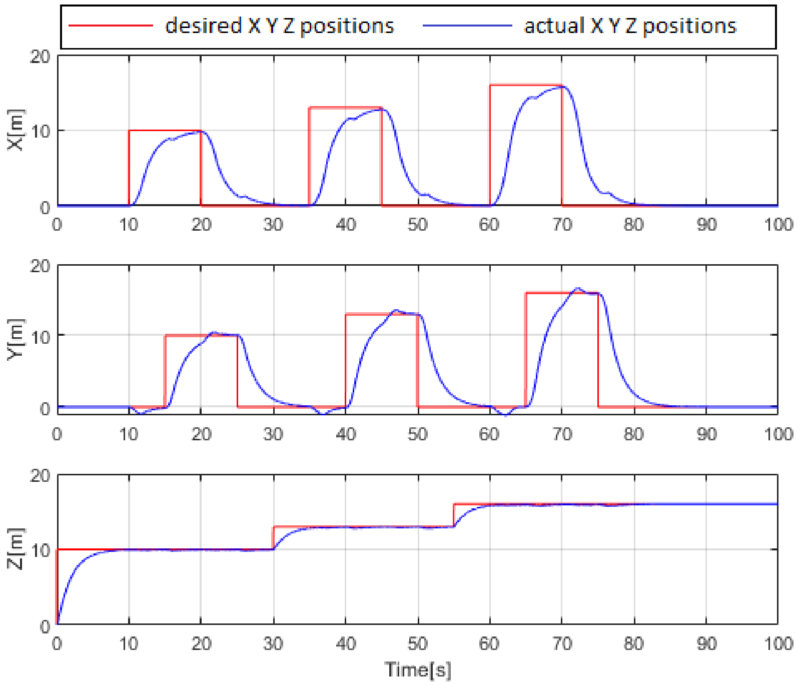


Figure 6.9 : Rectangular trajectory position components

In the figure 6.9, X Y Z components are shown for the rectangular region. In this components, Z components rises first, then stay fix until the first rectangular region is finished. Z component repeats the same movements until the second rectangular region is drawn. X and Y components do not change for a while. Then they draw a rectangular region together. These two components do the same movements until the second rectangular region is completed. As it is seen from the figure, desired and actual components follow each other.

In the figure 6.10, linear velocity components Vx Vy and Vz are illustrated for rectangular movements. As it is observed from the figure, Vz component lowers until reaches almost zero value whereas Vx and Vy components changes between -5 and 5 km/h.

In the figure 6.11, euler angles are demonstrated for rectangular trajectory. As it can be interpreted from the figure, while psi angle is increased first then fixed, phi and theta angles fluctuate between -0.5 and 0.5 which are acceptable values for this flight.

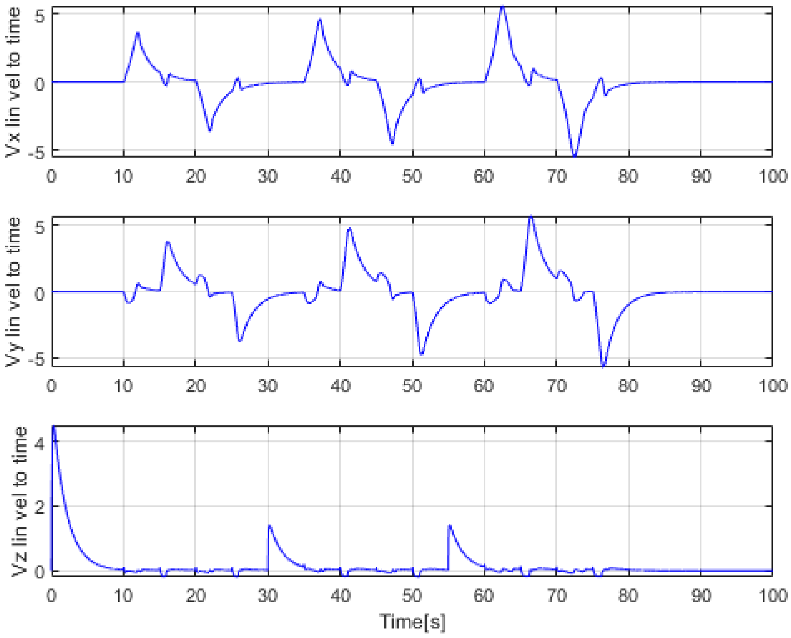


Figure 6.10 : Rectangular trajectory velocity components

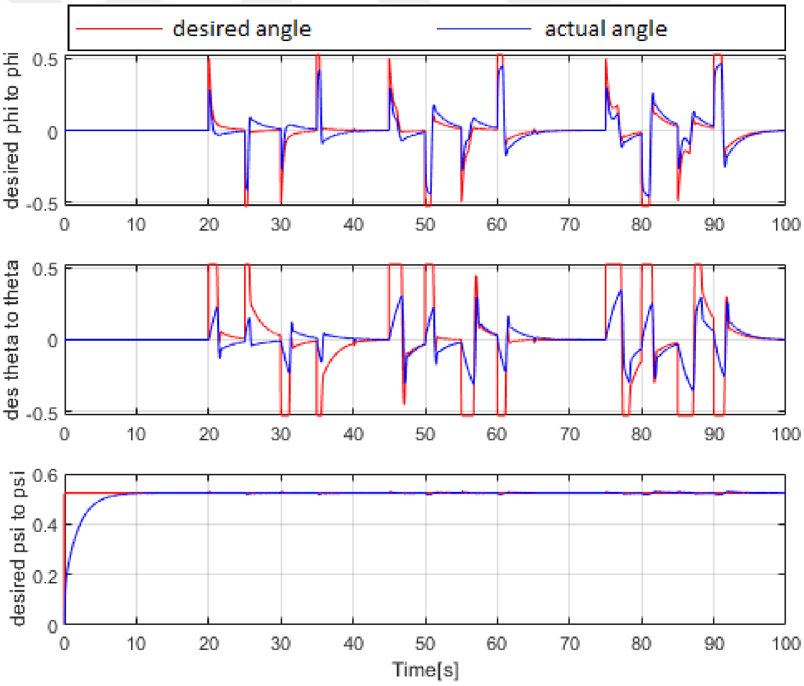


Figure 6.11 : Rectangular trajectory euler angle components

Furthermore, desired and real values of euler angles show similar behaviors during the flight that proves the controller is working properly.

In the figure 6.12, euler angle rates are presented for this linear rectangular movement. During the flight, as understood from the figure, roll and pitch angles' rates are coming

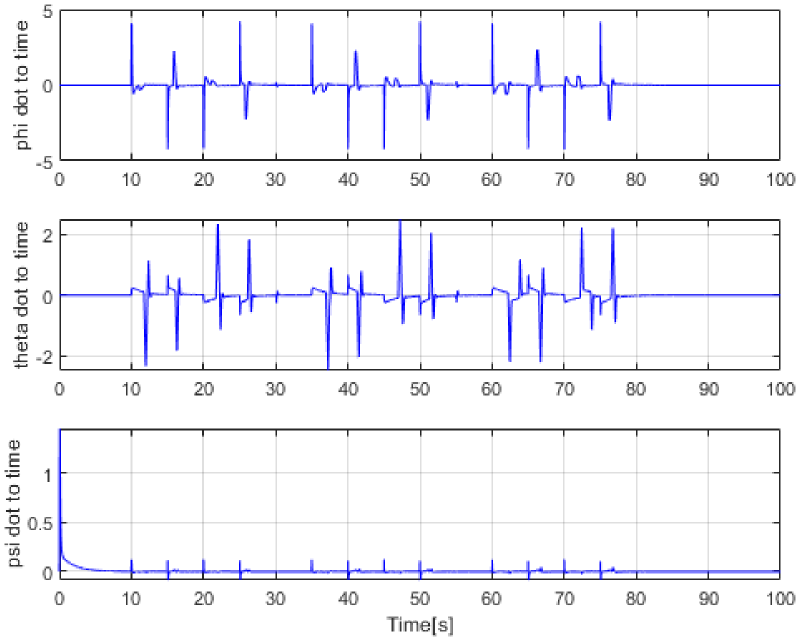


Figure 6.12 : Rectangular trajectory euler angle rates components

and going between -5 and 5 while yaw angle rate decreases sharp and goes to almost zero. Then, psi rate value remain stable just above the zero.

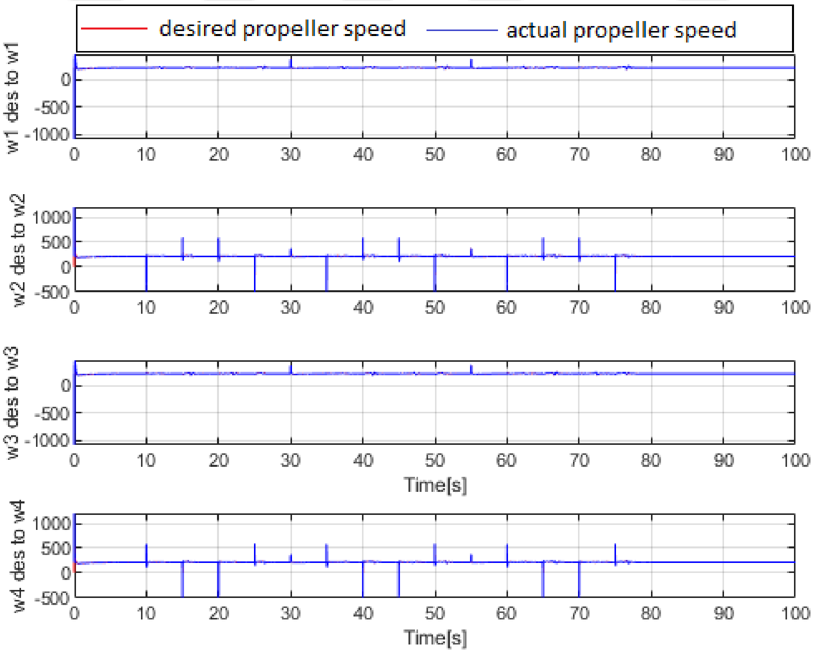


Figure 6.13 : Rectangular trajectory propeller angular velocities

In the figure 6.13, motor propeller angular speeds are demonstrated for this trajectory. From the figure, all speed components have fix values after arriving particular values.

6.1.3 Open circular trajectory and X Y Z components

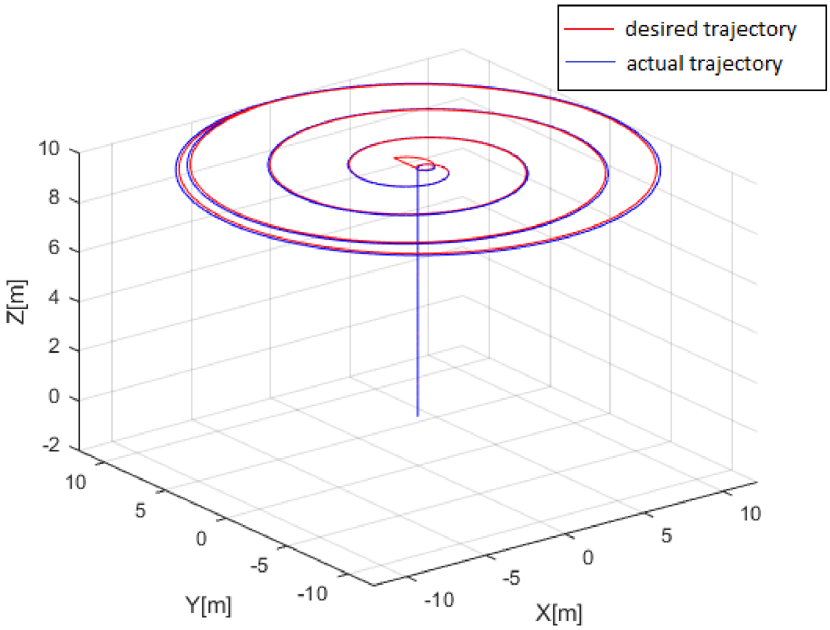


Figure 6.14 : Open circular trajectory

In the figure 6.14, the quadrotor makes open circular regions after arrives a particular height. In this trajectory, the helicopter only rises first, then makes different size of circles. On the other hand, desired and actual trajectories are following each other that indicates the controller has an acceptable working process.

In the figure 6.15, X Y Z components are shown for the open circular region. In these components, Z increases until a specific height, after that, X and Y components make growing circles via sine and cosine functions respectively. Moreover, desired and actual position components follow each other.

In the figure 6.16, velocity components for this flight path are illustrated. From the figure, while Vx and Vy components are waving between -5 and 5, Vz lowers sharply then goes to zero and stay zero during the whole flight.

In the figure 6.17, linear acceleration components are presented for this open circular movements. As seen in the figure, whereas linear accelerations in x and y directions are fluctuating between acceptable intervals, linear acceleration in z direction suddenly decreases then goes to zero and stay fix right over the zero value through the entire flight.

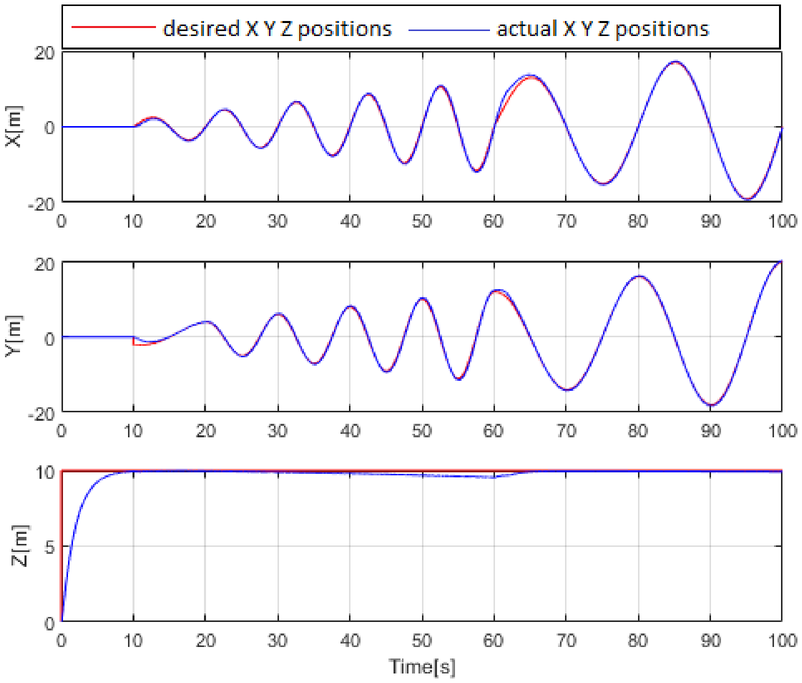


Figure 6.15 : Open circulars trajectory linear position components

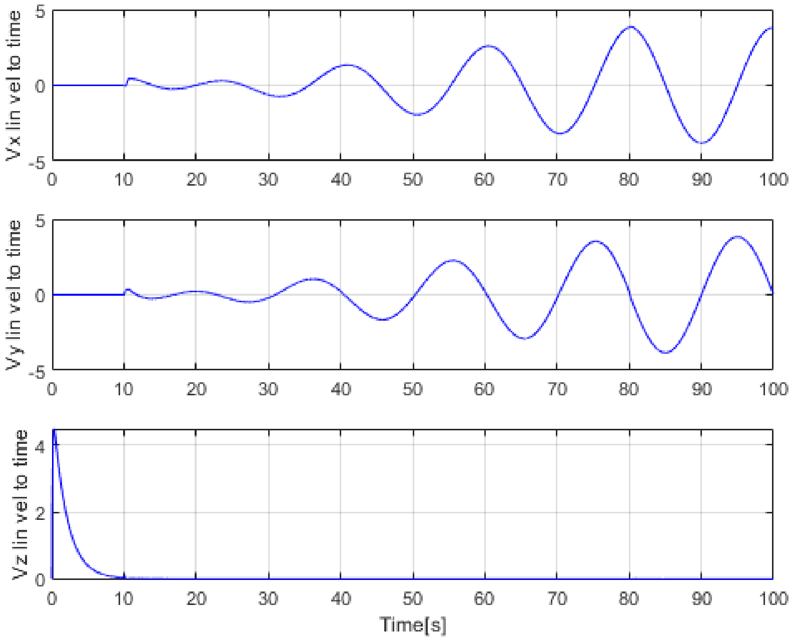


Figure 6.16 : Open circulars trajectory linear velocity components

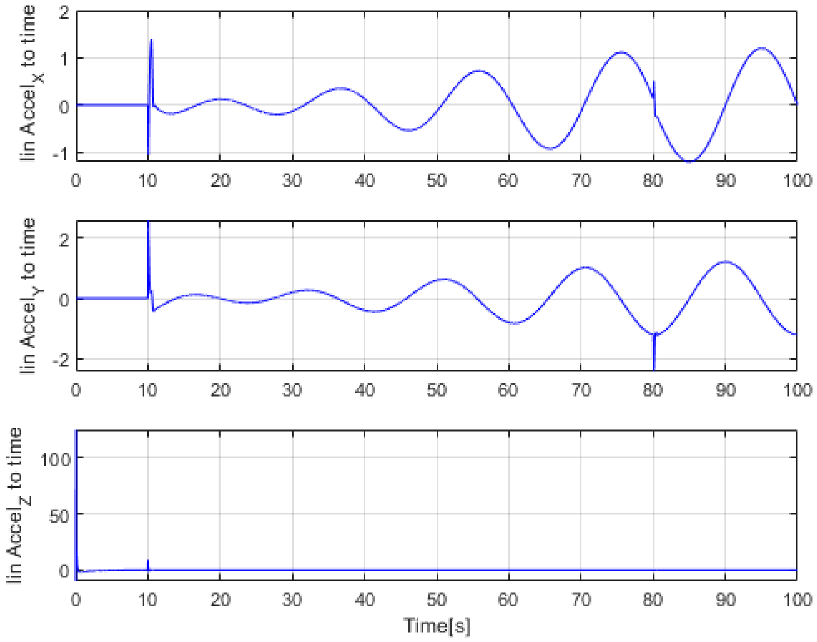


Figure 6.17 : Open circulars trajectory linear acceleration components

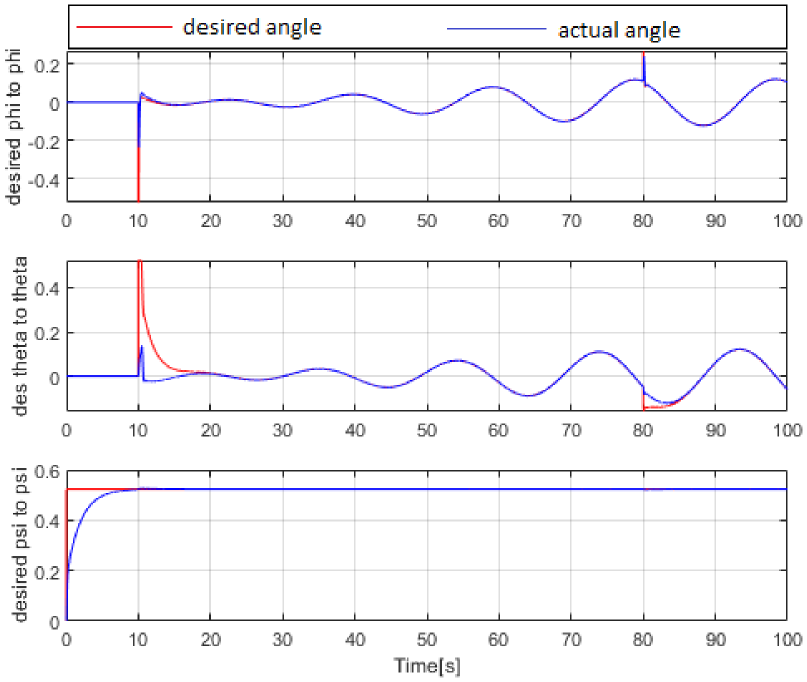


Figure 6.18 : Open circulars trajectory euler angle components

In the figure 6.18, the euler angle components for this flight trajectory are indicated. while phi and theta angles have small waves during the complete flight, psi angle rises fast in the first place, then remain constant as soon as reaches a particular value through the complete flight.

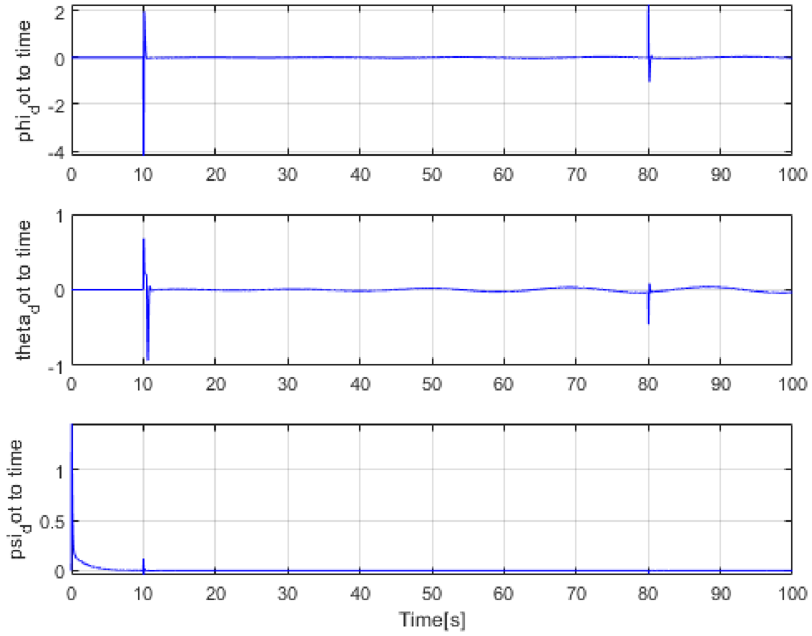


Figure 6.19 : Open circulars trajectory euler angle rates components

In the figure 6.19, euler angle rates are seen for this flight trajectory. almost all angle rates stay zero through the entire flight.

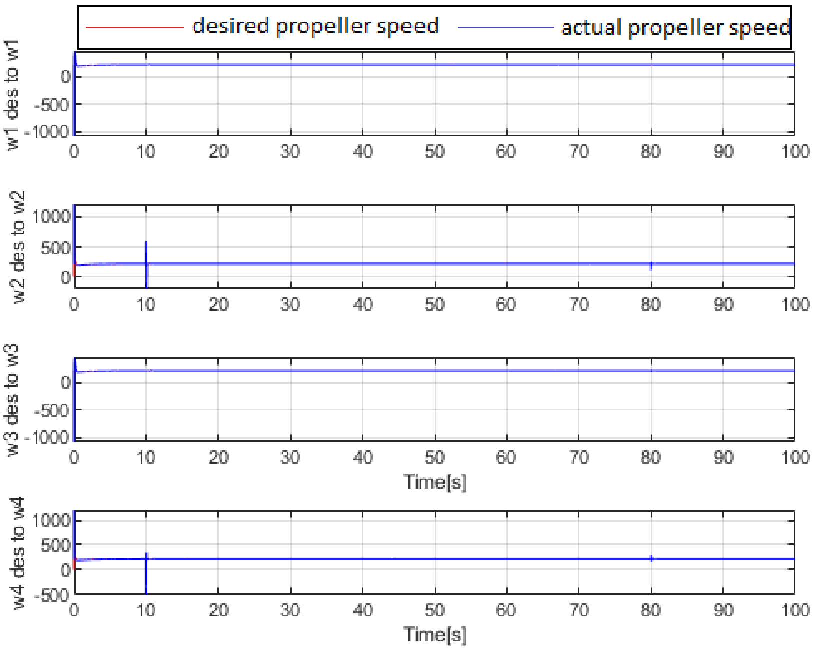


Figure 6.20 : Open circulars propeller angular velocity components

In the figure 6.20, motor propeller angular speeds are illustrated for this flight path. As understood from the figure, each propeller speed has a constant value until end of the flight.

6.1.4 Linear movement trajectory and X Y Z components

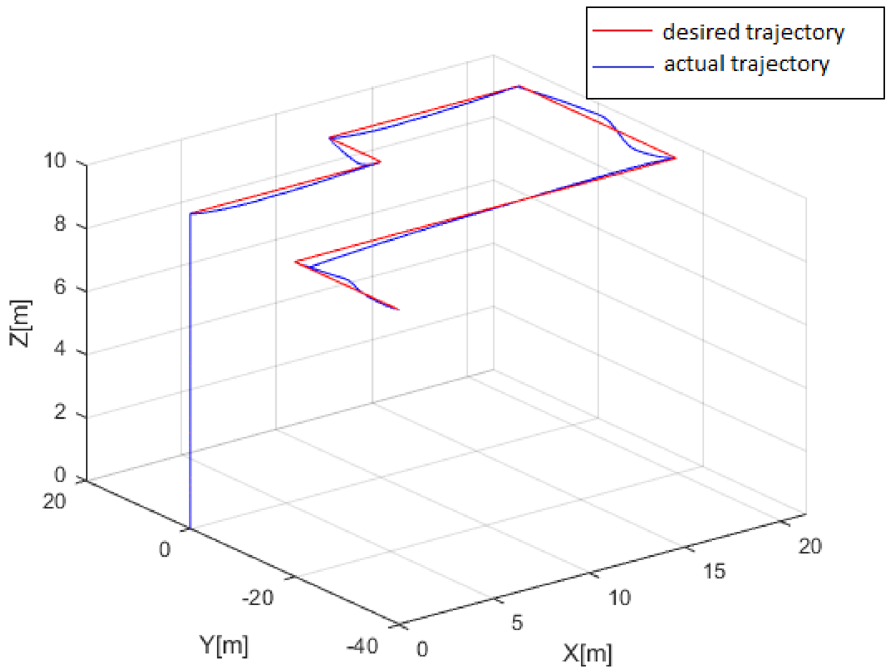


Figure 6.21 : Linear movement trajectory

In the figure 6.21, the quadrotor helicopter makes a vertical movement until a specific altitude. Then, it moves linearly in X and Y directions by a fix altitude. Besides, from the figure, desired and real flight paths have the same directions which means the controller is working properly.

In the figure 6.22, X Y Z components are illustrated for the linear movements components. After Z component reaches a specific altitude by performing a vertical increase, X and Y components draw a linear open region together. On the other hand, in all position components, both desired and actual components move in the same directions that verifies the controller is working correctly.

In the figure 6.23, Linear velocity components Vx Vy and Vz are seen for this flight type. As seen from the figure, while Vx and Vy have sharp waves, Vz decreases suddenly first then remain constant near zero through the whole flight.

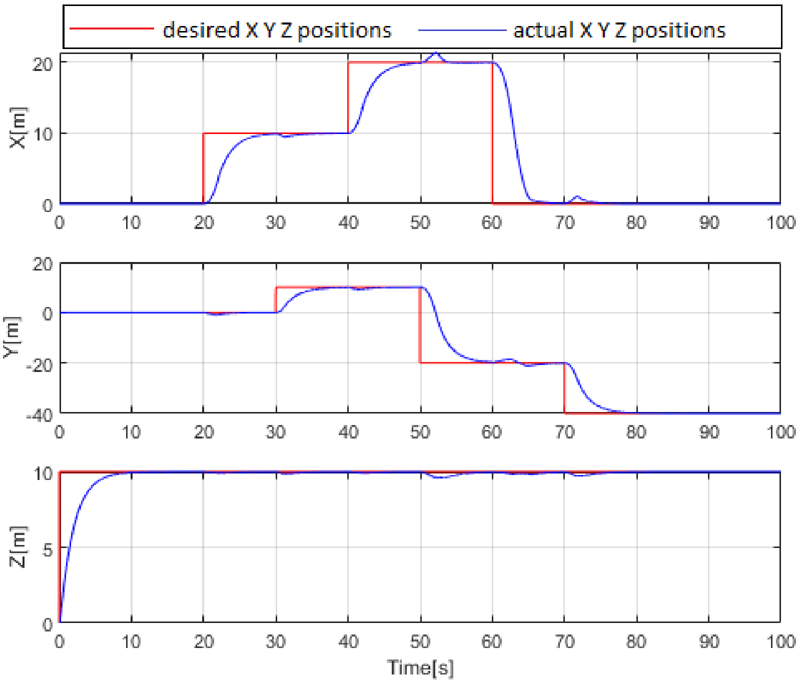


Figure 6.22 : Linear movement trajectory linear position components

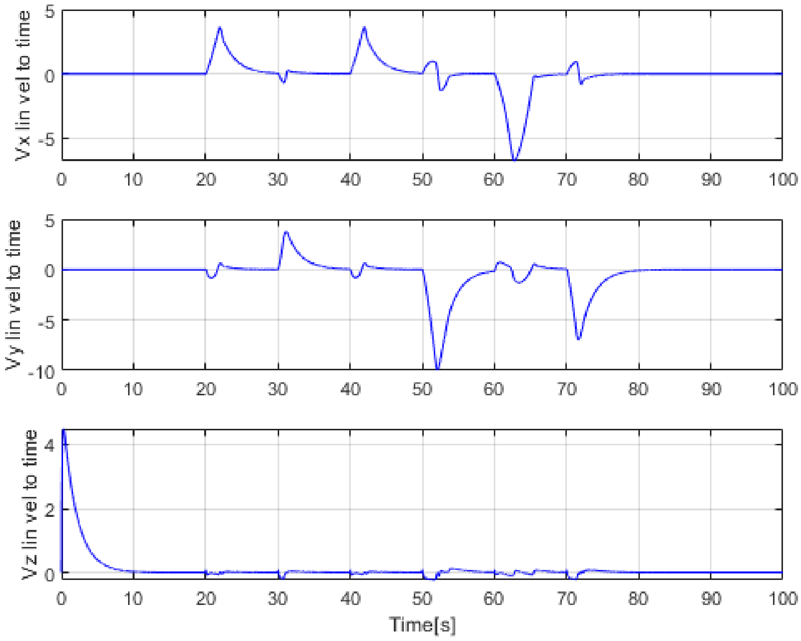


Figure 6.23 : Linear movement trajectory linear velocity components

In the figure 6.24, linear acceleration components are shown for this quadrotor trajectory. As it can be understood from the figure, whereas the accelerations in x and y directions are changing by fluctuating, the acceleration in z direction vertically goes down and remain fix near the zero.

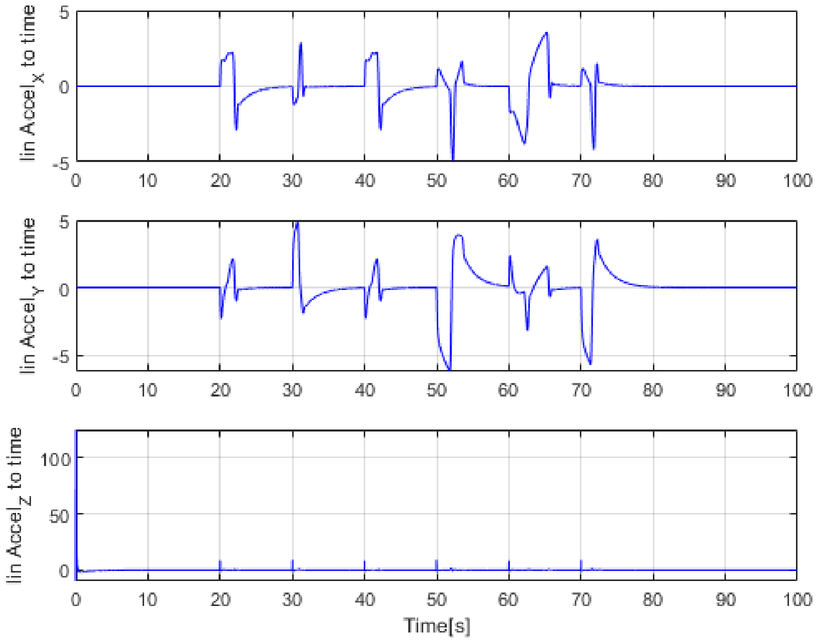


Figure 6.24 : Linear movement trajectory linear acceleration components

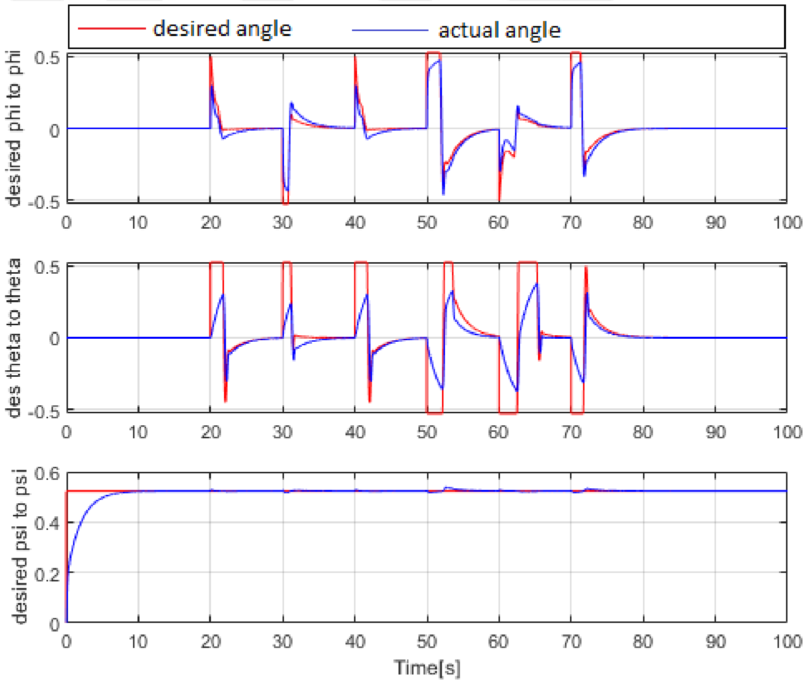


Figure 6.25 : Linear movement trajectory euler angle components

In the figure 6.25, euler angles are presented for this flight path. As interpreted in the figure, phi angle is rising first then stay constant after a while, whereas phi and theta angles go up and down sharply between acceptable intervals through the whole flight.

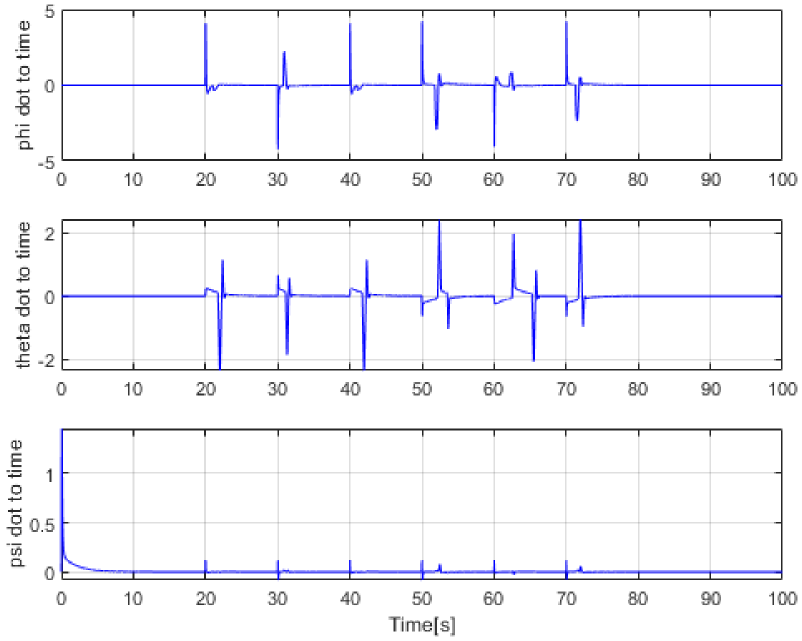


Figure 6.26 : Linear position angular acceleration components

In the figure 6.26, euler angle rates are shown for the trajectory. According to the figure, euler rates in x and y directions has jumps on specific moment, however, in z directions euler angle rates goes to zero fastly.

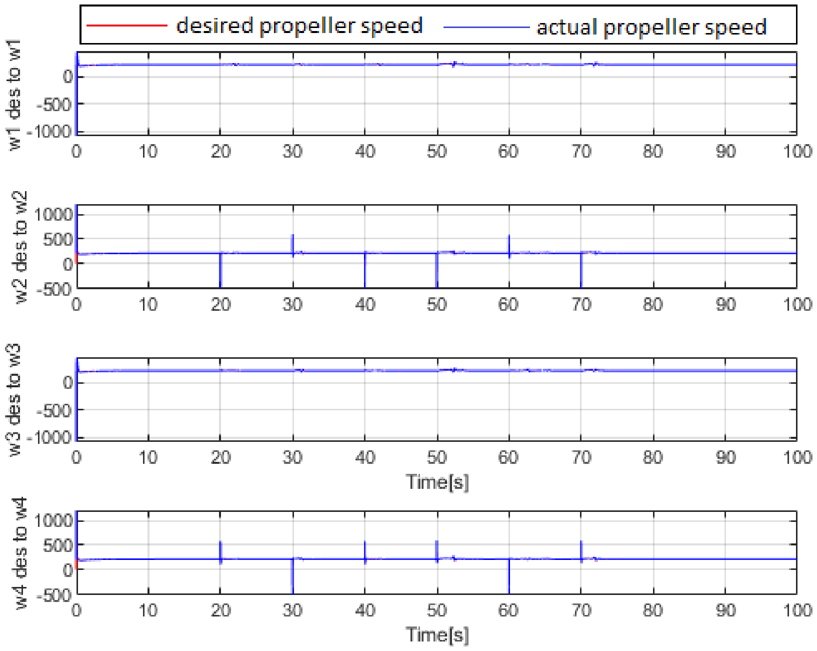


Figure 6.27 : Linear movement trajectory angular acceleration components

In the figure 6.27, motor propeller angular speeds are illustrated for the quadrotor helicopter trajectory. All motor speed components have a constant value through the entire flight.

6.1.5 Results for linear model

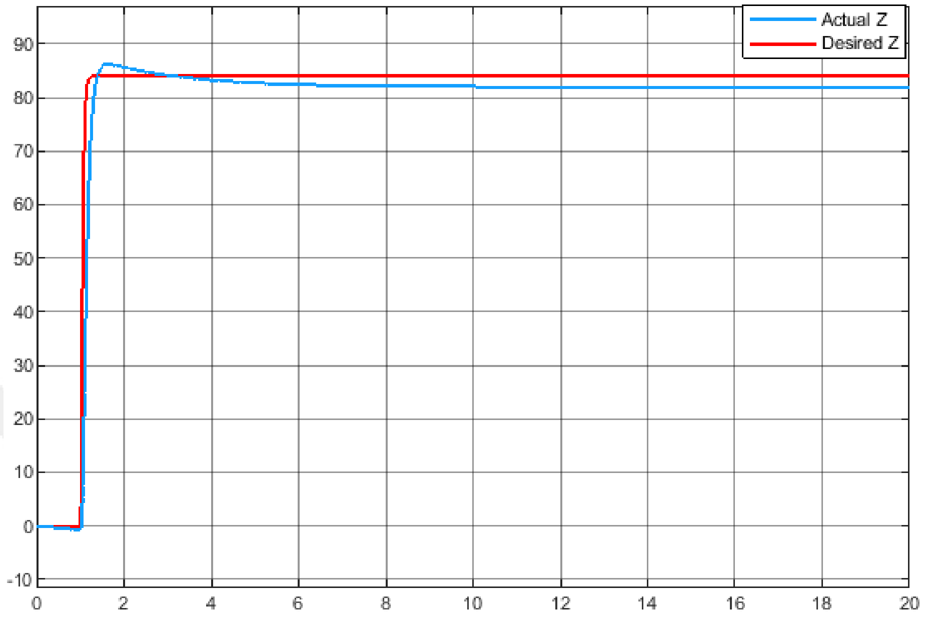


Figure 6.28 : Altitude response for linear model of the quadrotor

Figure 6.28 illustrates the altitude response of the quadrotor. As seen, the altitude response looks linear and stay fixes near hover position.

Figure 6.29 shows the euler angle response for the quadrotor linear model. As expected, since the model is linear, roll, pitch and yaw angles remain zero during the whole flight.

Figure 6.30 indicates roll, pitch and yaw rate responses for the quadrotor linear model. As observed in the figure, roll, pitch and yaw rates values remains zero during the whole flight.

In the figure 6.31, x, y and z linear position components are shown. Since the model is linearized in the hover position, while the altitude remains fixes, x and y component values are zero through the entire flight.

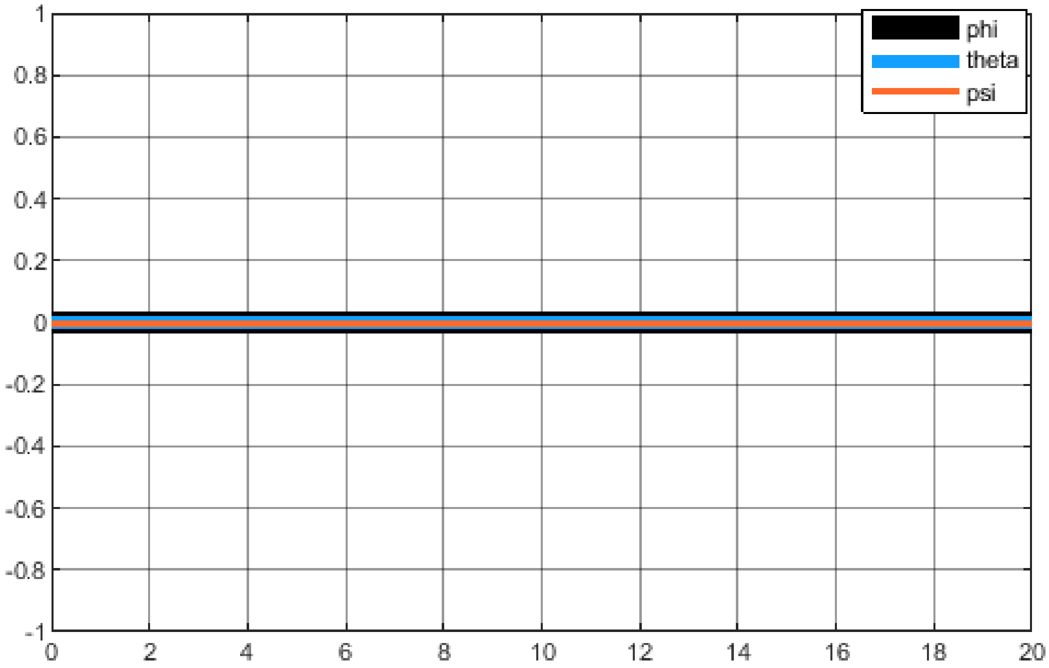


Figure 6.29 : Euler angle response for linear model of the quadrotor

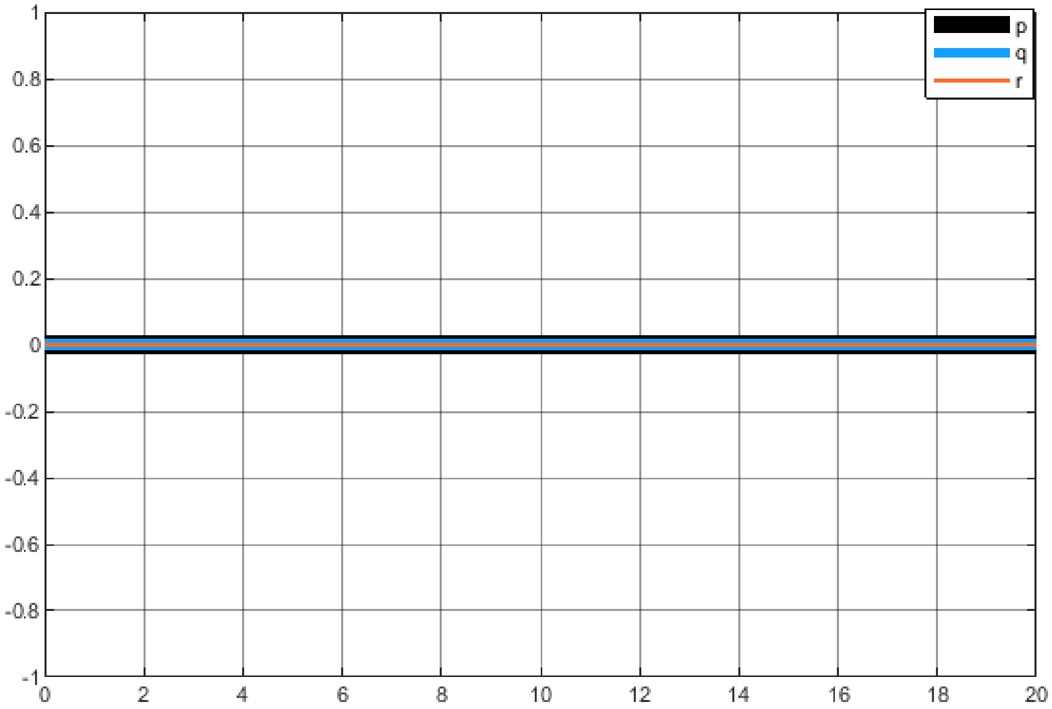


Figure 6.30 : Roll(p), Pitch(q) and Yaw(r) rates response for linear model of the quadrotor

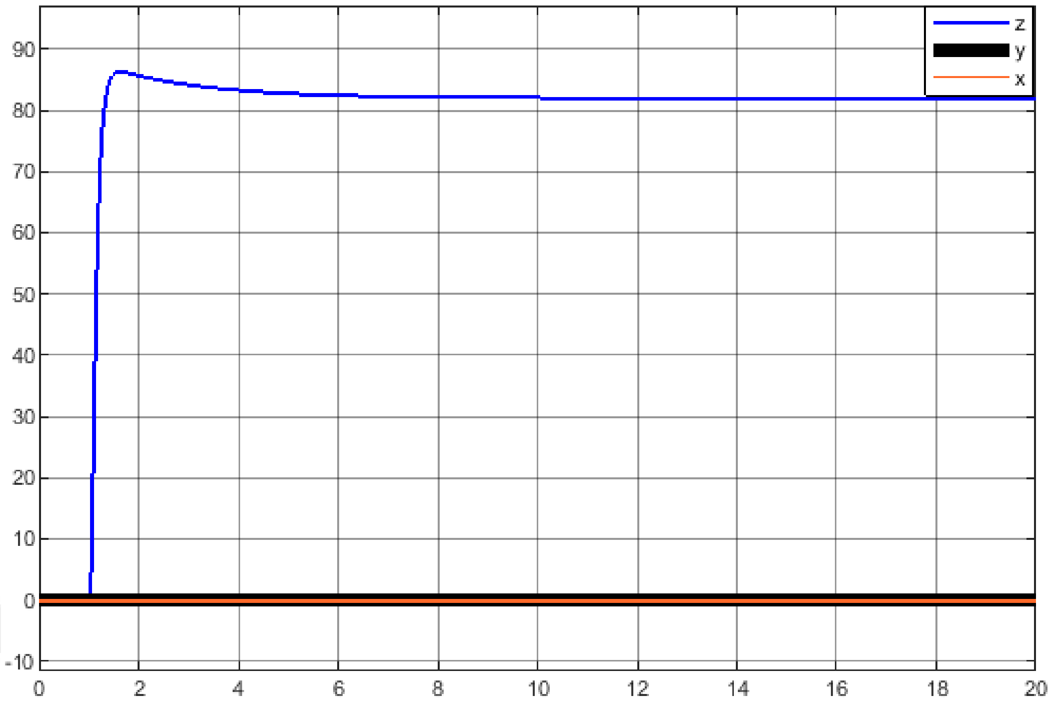


Figure 6.31 : X, Y and Z response for linear model of the quadrotor

YÖK

7. FAULT TOLERANT KONTROL SYSTEMS(FTCS) ON THE QUADROTOR

It is so important that quadrotor systems survive in case of any impact coming from outside or when faults occur inside of the system. To examplify, when sensor faults or actuator faults, the controllers should continue to work in order to provide a safe flight. Fault detection and estimation is so crucial in these faulty cases.

7.1 Definition of Fault Tolerant Control Systems

Vehicles working with automated systems, sometimes gives faults. Since they have autonomous systems. This typical faults bring failures and serious damages. In order to prevent this, fault tolerant control systems are designed. This systems make the vehicle continues to work despite of these kinds of faults. The goal of fault tolerant systems is to hinder that small faults grows and turn into hazardous ones which lead to system failures.

7.2 General Approaches to Fault Tolerant Control Systems

Existing efforts in FTCS pattern can be categorized into two main approaches as passive and active depending on how the redundancy is being utilized. In a passive approach, the sensible system constituent failures are thought to be known a priori, and the control system consider all these failure modes in the design stage. the control system will stay fixed during the whole system operation Once it is designed. The control system should still be able to maintain the design performance, even if in the case of component malfunctions. In other words, in passive FTCS, the person must guarantee that the control system works in the case of all possible system operating scenarios that have been considered at the design stage, involving potential constituent failures. But it is not possible to say anything about the system behavior in the presence of unexpected failures.

In passive FTCS, since maintaining the system stability in the case of various component failures from the performing viewpoint, the patterned controller must be conservative. It is very hard for a passive FTCS to be optimal from the performance point of view alone, from typical relationships between the optimality and the robustness.

Contrary, an active FTCS reacts to the system constituent failures actively by correctly reconfiguring its control actions so that the system stability can still be acceptable. This approach relies mostly on a real-time fault detection scheme for the most up-to-date knowledge about the status of the system and operating conditions of its' constituents to achieve a successful control system reconfiguration.

7.3 Kalman Filter

Kalman Filter is an implementation used in dynamic systems that make predictions about system states by using former, input, and output information of the system. Like other filters, kalman filters has filtering features, However it has ability about predicting unmeasured features of the system.

Kalman Filter filters the faults by working recursive and real-time. By modeling physical characteristics of the system, this filter optimizes the faults according to mathematical prediction of next state.

In kalman filters, model prediction is compared with observation. The difference between them is scaled with a multiplier called Kalman gain. Then, a feedback is applied to the model in order to enhance next predictions. By a high Kalman gain, the filter observations is followed more closely. By a low Kalman gain, model predictions of the filter is followed more closely. This method tries to generate closer predictions based on the model predictions.

At each time step, kalman filter generates the predictions of real unknown values with their uncertainties. When result of the next estimation is observed, these uncertainties concentrate on predictions which have less uncertainties and the predictions are updated.

State equation of kalman filter is as follows:

$$x_k = Ax_{k_1} + Bu_k + w_{k-1} \tag{7.1}$$

In the equation above, each x_k value is found by adding control signal u_k and former process noise w_{k-1} to its' former value x_{k-1} and with a designed linear combination. Mostly control signal u_k is not used.

Output equation of Kalman Filter is as follows:

$$Z_k = Hx_k + v_k \tag{7.2}$$

In the equation above, it can understood that Any measurement value z_k consists of linear combination of signal current value x_k and measurement disturbance v_k.

Process noise w_{k-1} and measurement noise v_k are independed from each other.

A,B and H stand for matrices. For most of the signal problem, they are only numeric values. Moreover, for the simplicity, even if their values changes, they can be assumed as constants.

After adding the Kalman Filter to the designed model, required parameters and initial values has to be determined.

There are two equation team Prediction/time update team is as follows:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{7.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{7.4}$$

Correction/Measurement Update team is as follows:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{7.5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{7.6}$$

$$P_k = (1 - K_k H)P_k^- \tag{7.7}$$

In these equations above, finding R and Q matrices are so important.

After gathering all required data needed the process can be started. It has to be known that former predictions are used as inputs for current state.

$\hat{x}_k^-$ is the previous prediction. Which means, it is the raw input before updating the measurement fixing. $\hat{P}_k^-$ is the previous fault covariant. These two $\hat{x}_k^-$ and $\hat{P}_k^-$ values are used as prior values for the next correction/measurement update stage. Which means these two values are given as input data for the next stage.

Prediction process is as follows:

1. predict the previous state

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{7.8}$$

2. predict the fault covariant

$$P_k^- = AP_{k-1}A^T + Q \tag{7.9}$$

Correction process is as follows:

1. Estimate the kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{7.10}$$

2. Update the prediction instead of Z_k measurement

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{7.11}$$

3. Update the fault covariant

$$P_k = (1 - K_k H)P_k^- \tag{7.12}$$

In the measurement equations, $\hat{x}_k$ which is the value of x at moment k is found. This is the value needs to be found. Moreover, P_k values also need to be found. These two values are estimated for the next stage. Kalman gain (K_k) is not measured since it is required for the next stage.

7.4 Fault Detection and Estimation Implementations on the Quadrotor for all Quadrotor Flight Paths

In this study, fault detection and estimation implementations have been performed by applying sensor faults, and actuator faults. By subtracting faulty response from the original response, fault detection and estimation are performed. Moreover, some disturbance is applied to the linear velocity components, and euler angles. Then a kalman filter is implemented to remove disturbance from faulty response. After that, by subtracting faulty response from original response, fault detection and estimation are performed. Finally, by subtracting Kalman filter output from the faulty response, Kalman Filter impact on these faults can be understood.

7.4.1 Applying sensor fault to linear position components

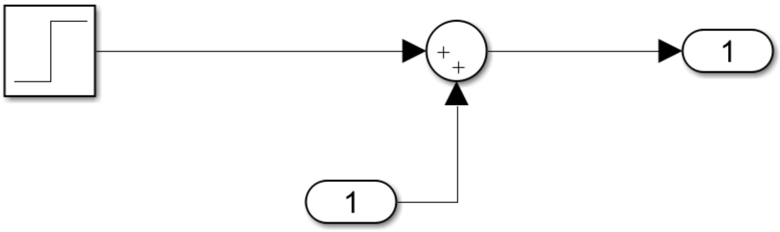


Figure 7.1 : Sensor Fault Model

In the figure 7.1 sensor fault is seen. As interpreted, a step input is used for the sensor fault.

In figure 7.2, 7.3, 7.4, and 7.5 sensor faults are implemented to the different trajectories by applying step input for the fault. As seen in these figures, with the effect of sensor fault applied, a breakdown occurs in shapes. However, the actual trajectory paths follows the desired ones with a small error through the whole flight. That’s why, it is proved that the controller is keep working in spite of sensor faults’ negative effect on the quadrotor trajectories.

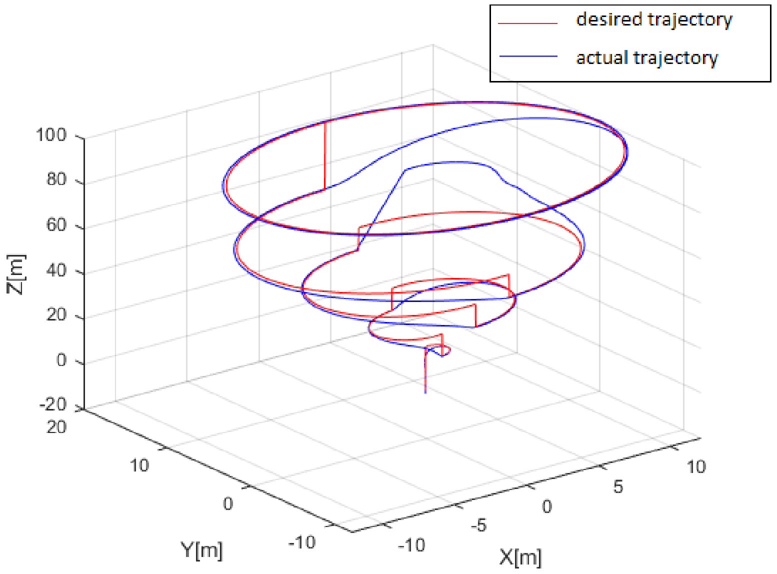


Figure 7.2 : Sensor Fault for circular trajectory

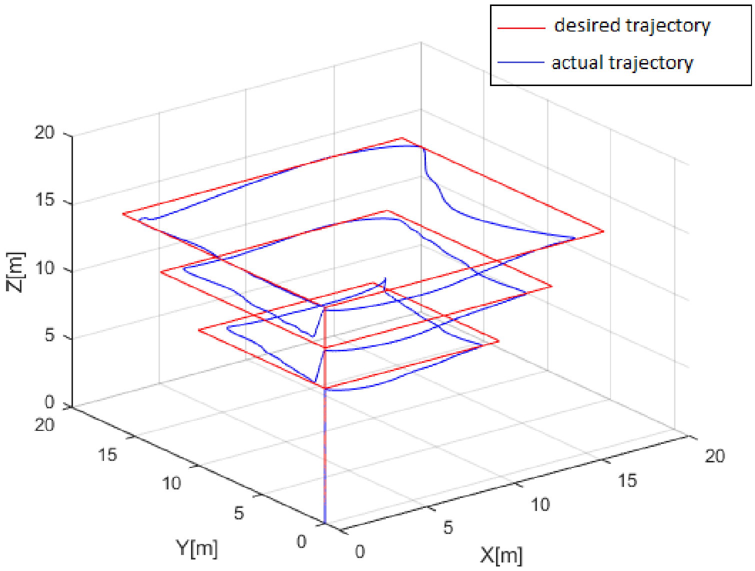


Figure 7.3 : Sensor fault for rectangular trajectory

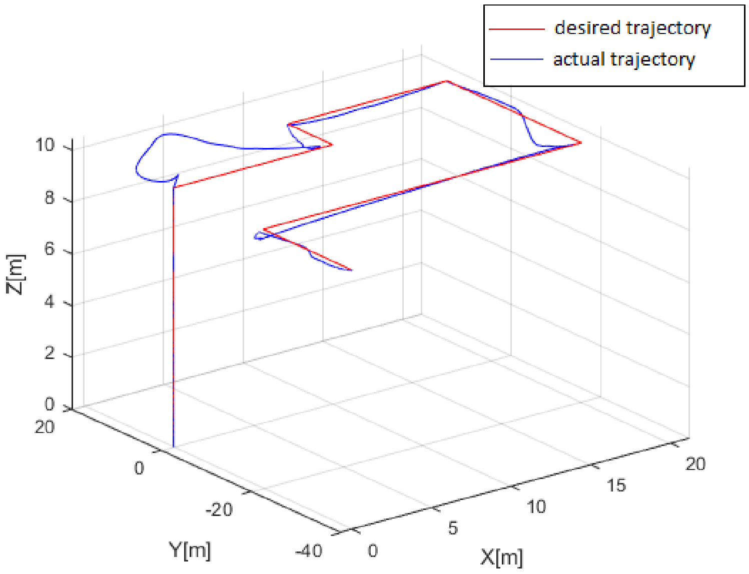


Figure 7.4 : Sensor fault for linear trajectory

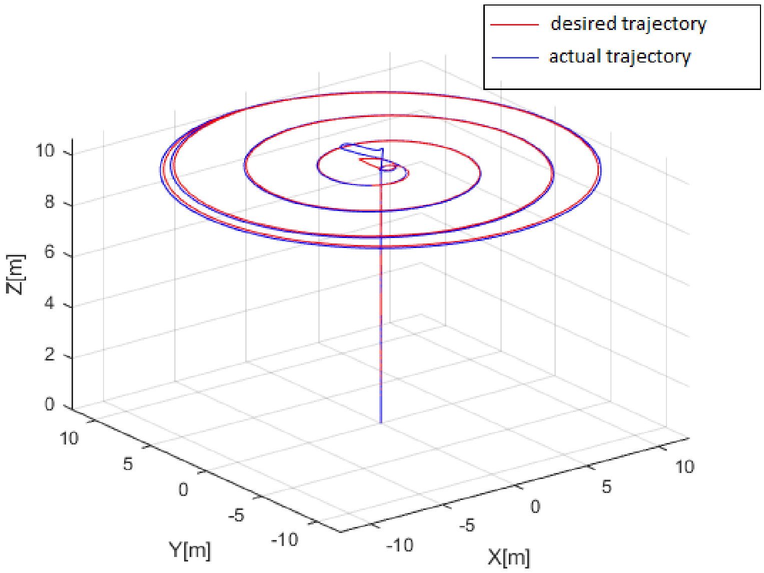


Figure 7.5 : Sensor fault for Circular trajectory

In figure 7.6, 7.7, 7.8, and 7.9, a sensor fault is applied to all linear position components. After having a jump on each figure, faulty actual responses follows the desired ones due to impact of the controller on the linear position components. However sensor fault is not completely fixed as observed in the figures.

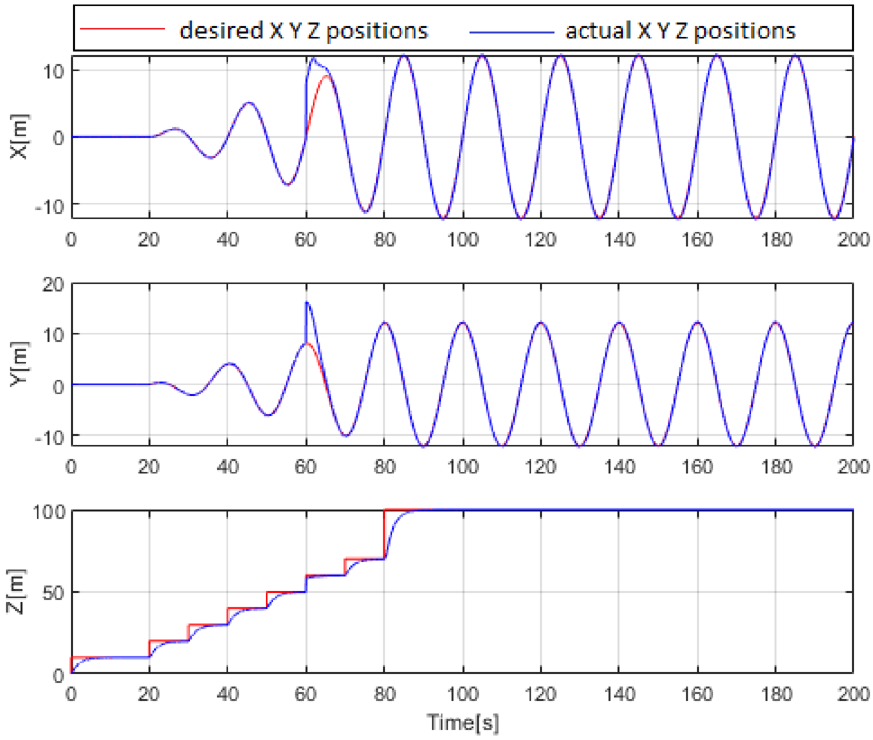


Figure 7.6 : Sensor Fault for circular trajectory x, y, and z positions

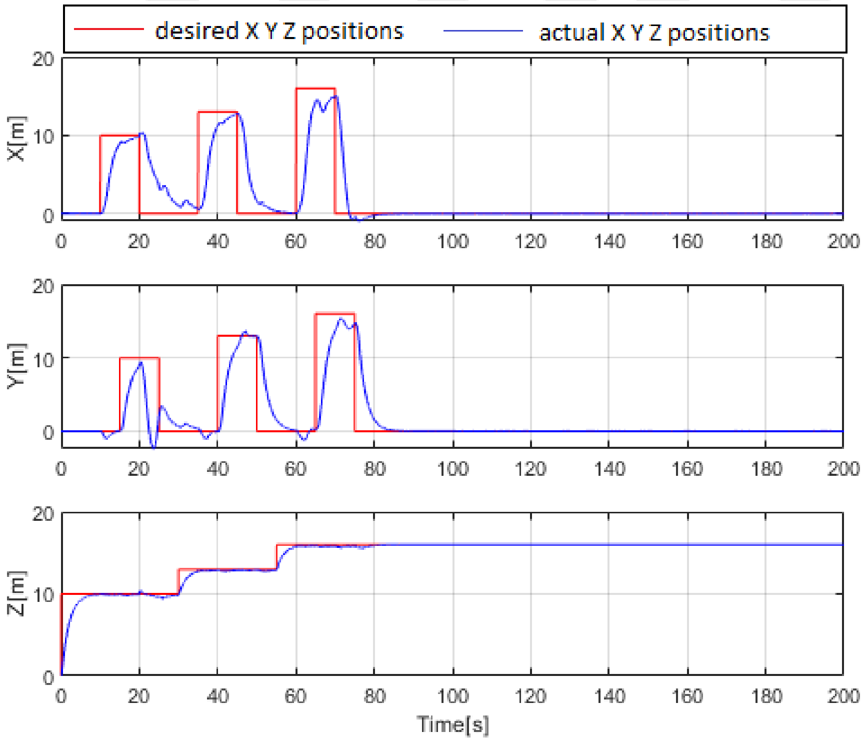


Figure 7.7 : Sensor Fault for rectangular trajectory x, y, and z positions

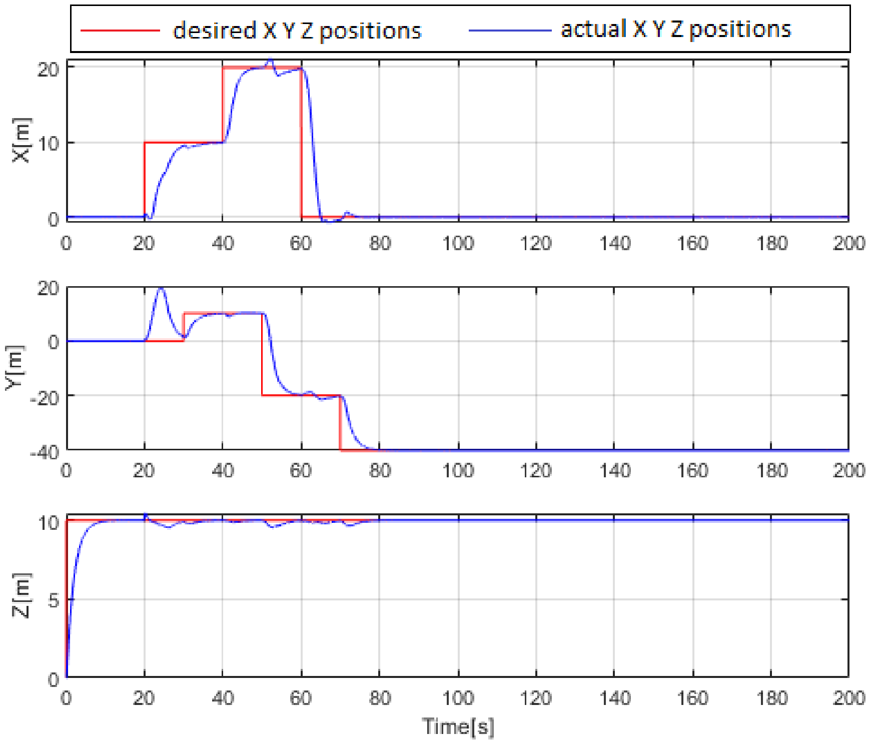


Figure 7.8 : Sensor Fault for open linear trajectory x, y, and z positions

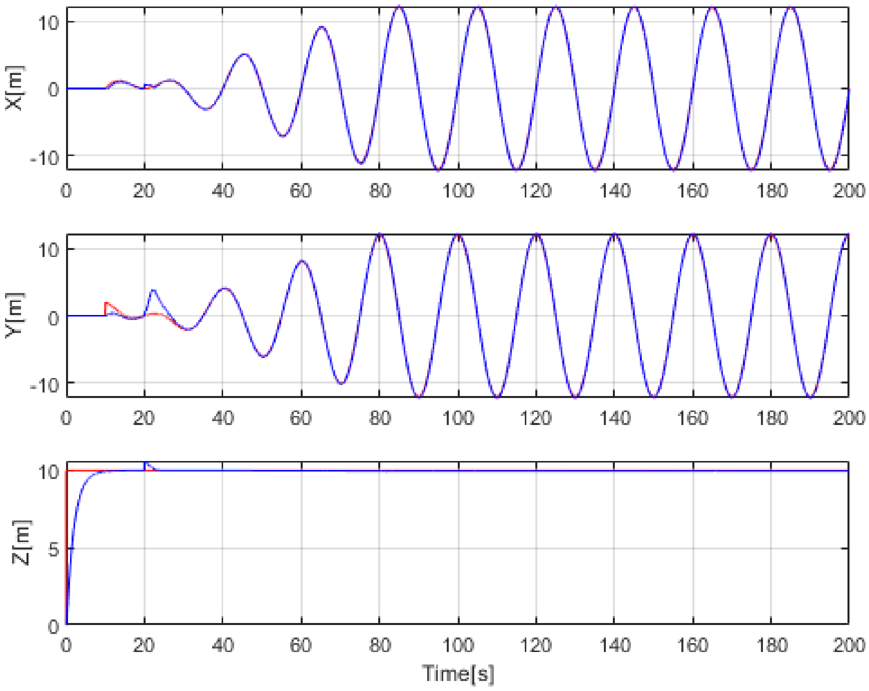


Figure 7.9 : Sensor Fault for open circular trajectory x, y, and z positions

7.4.2 Appying disturbance to linear velocity components and removing it with kalman filter

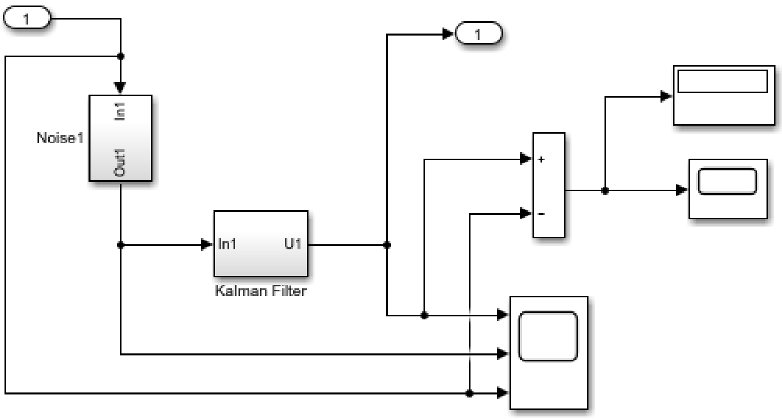


Figure 7.10 : Disturbance Model

In the figure 7.10, adding noise to the model and removing it with Kalman Filter is illustrated. As understood, Kalman Filter is added after the noise.

In this section, some disturbance is applied to the linear velocity components Vx and Vz in the first place. Then, a Kalman Filter is applied to the faulty responses so that the velocity components have acceptable responses. At the end, fault diagnosis and Kalman filter effect is determined by subtracting faulty response from the original response and Kalman Filter response.

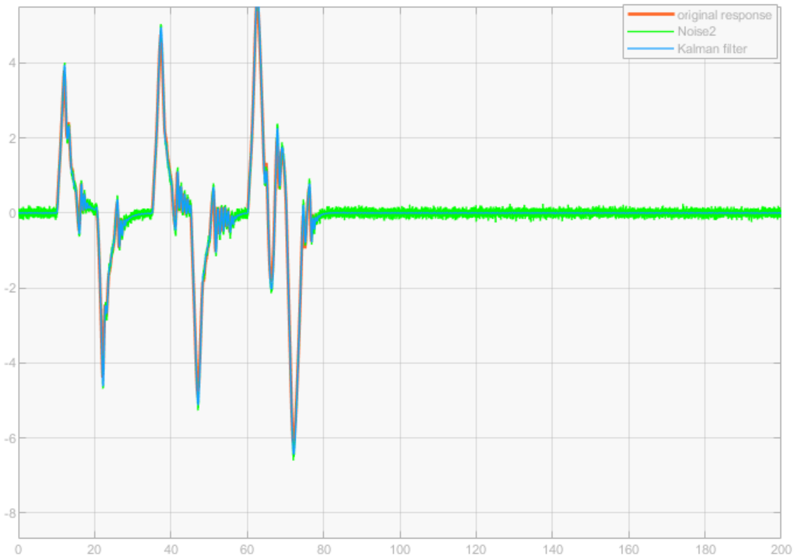


Figure 7.11 : Kalman filter effect on the disturbance of Vx velocity for rectangular movement

In the figure 7.11, some disturbance is applied to the linear velocity component Vx and a kalman response is used so that this component has an appropriate response. As seen from the figure, the red line is original response and there is no disturbance on it.

However, the green line has disturbance. The blue line shows the kalman impact on the disturbance. As seen on the blue line, most of the disturbance is removed from the faulty response which proves Kalman Filter turn the faulty response into the acceptable response.

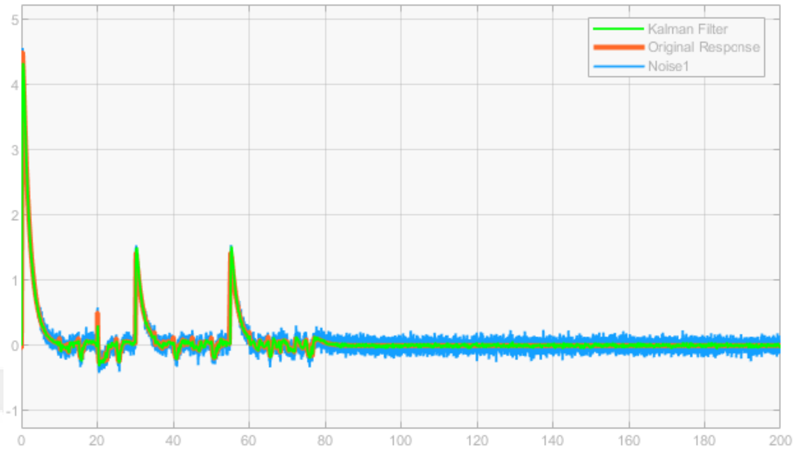


Figure 7.12 : Kalman filter effect on the disturbance of Vz for rectangular movement

In the figure 7.12, a Kalman Filter is implemented to the linear velocity component Vz of open linear trajectory in order to remove the disturbance from the component. Original response is presented with red line while noisy response and Kalman Filter response are shown with blue line and green line respectively. As observed from the figure, most of the disturbance is eliminated from the faulty response.

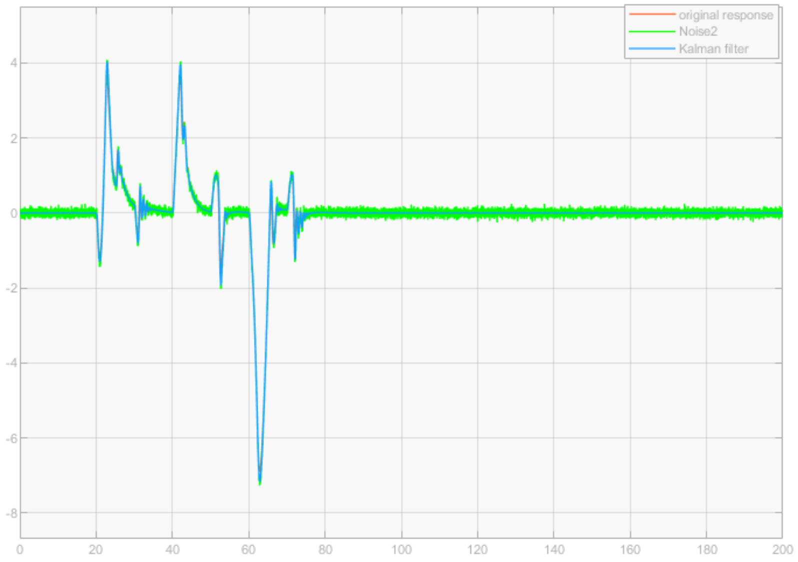


Figure 7.13 : Kalman filter effect on the disturbance of Vx velocity for open linear movement

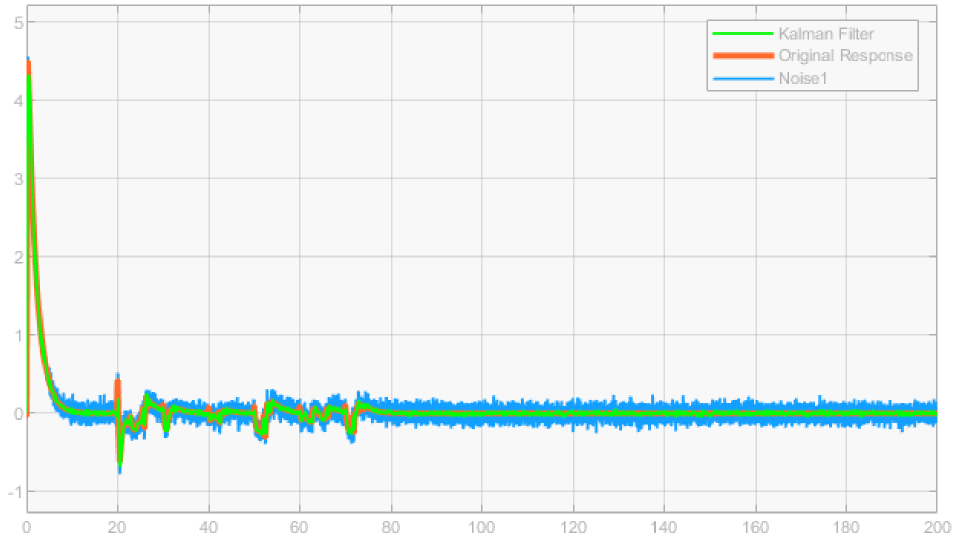


Figure 7.14 : Kalman filter effect on the disturbance of Vz velocity Open Linear movement

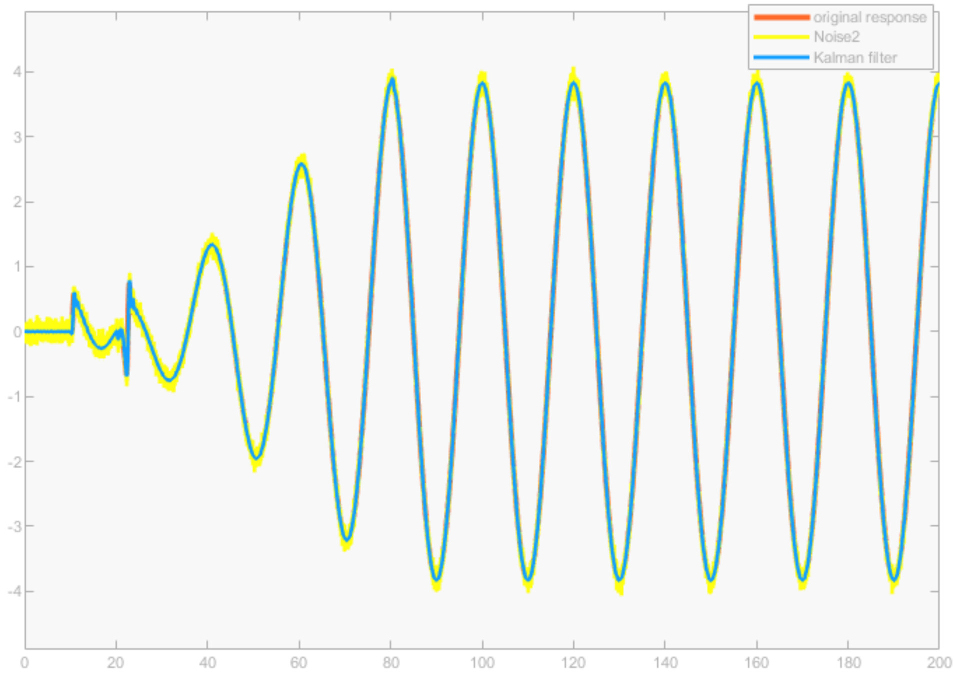


Figure 7.15 : Kalman filter effect on the disturbance of Vx velocity for open circular movement

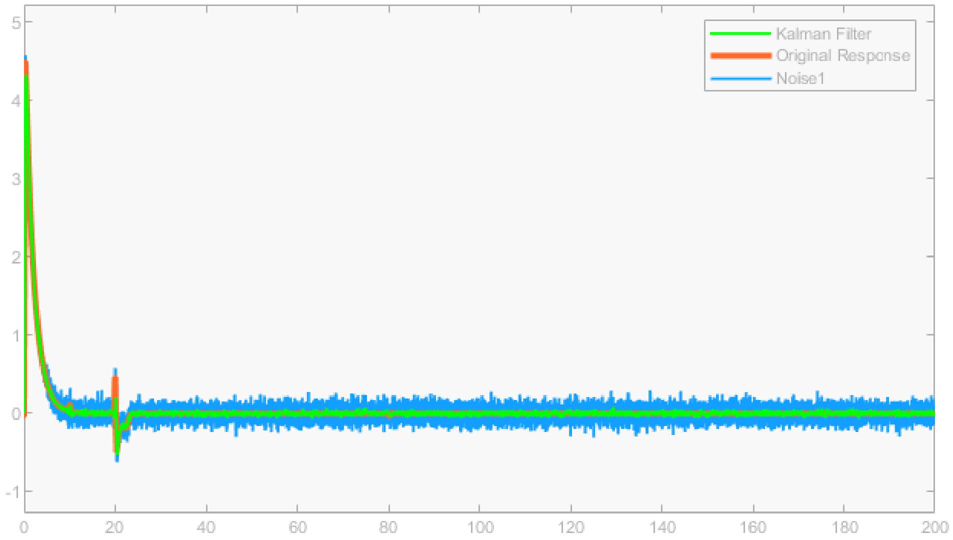


Figure 7.16 : Kalman filter effect on the disturbance of Vz velocity for open circular movement

In the figure 7.13, and 7.14, the same process is implemented as performed in figure 7.15 and 7.16. Some disturbance is used first, then a Kalman Filter is applied in order to remove the disturbance. At the end, Kalman Filter response shows the same behavior as the original one and most of the disturbance is eliminated.

7.4.3 Appying disturbance to euler angles and removing it with kalman filter

In the figure 7.17, 7.18 and 7.19, some disturbance is implemented to roll, pitch and yaw angles responses. red lines show the noisy responses whereas blue lines represents Kalman Filter responses. As detected from the figures, faults are detected via red lines first, then by using Kalman Filter, these noises are fixed and appropriate responses are obtained at the end.

In the figure 7.20, disturbance is added to original yellow line. As a result, original response becomes noisy response represented with red line. In order to hinder that this fault becomes a serious failure, a Kalman Filter is applied to the faulty one. With the effect of Kalman Filter, most of the disturbance is eliminated from the red line and an acceptable response is obtained. This acceptable response is illustrated with blue line.

In the figure 7.21 and 7.22, similar processes are implemented and with Kalman Filter good results are observed in the system responses.

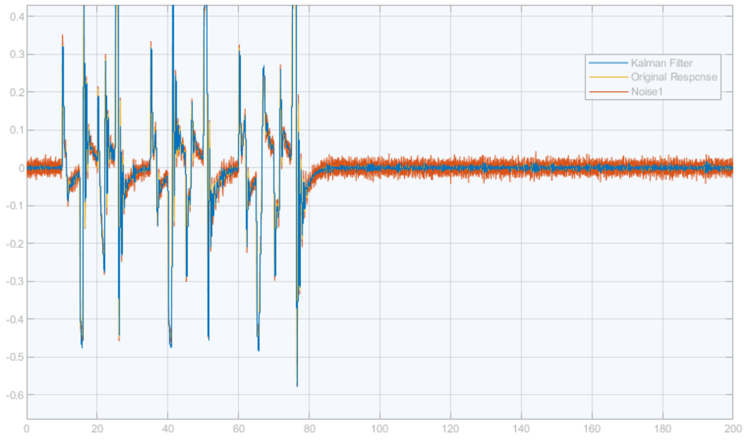


Figure 7.17 : Kalman filter effect on the disturbance of roll angle for rectangular movement

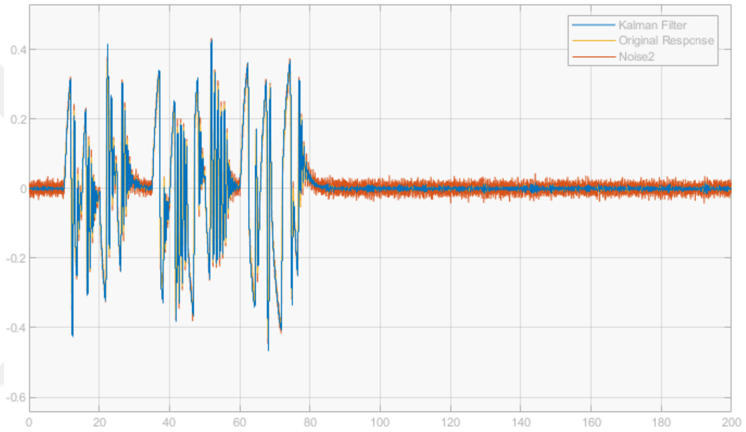


Figure 7.18 : Kalman filter effect on the disturbance of pitch angle for rectangular movement

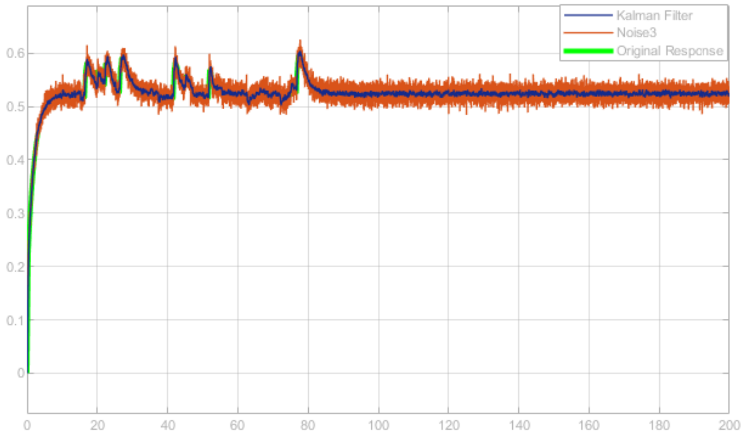


Figure 7.19 : Kalman filter effect on the disturbance of yaw angle for rectangular movement

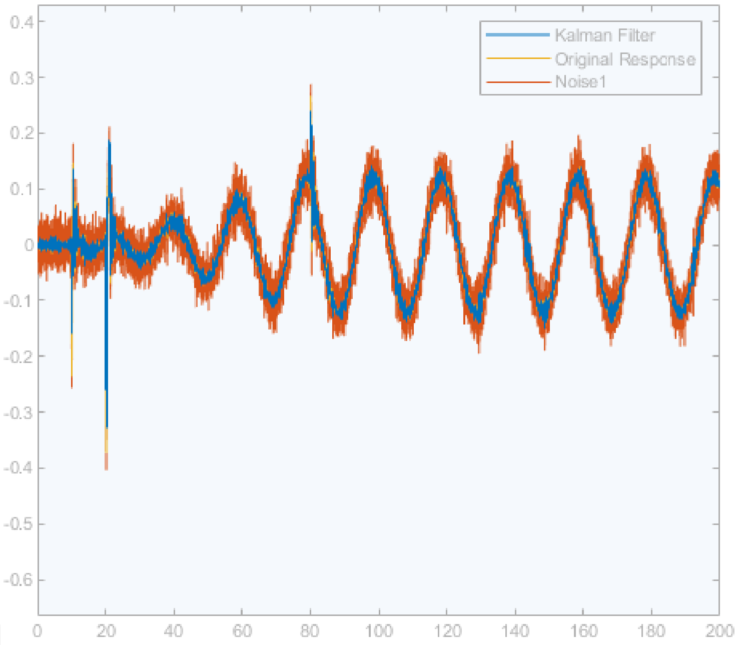


Figure 7.20 : Kalman filter effect on the disturbance of roll angle for open circular movement

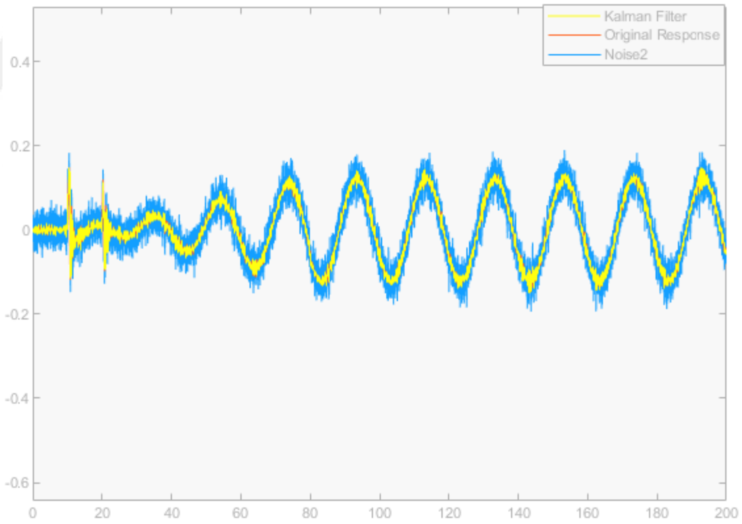


Figure 7.21 : Kalman filter effect on the disturbance of pitch angle for open circular movement

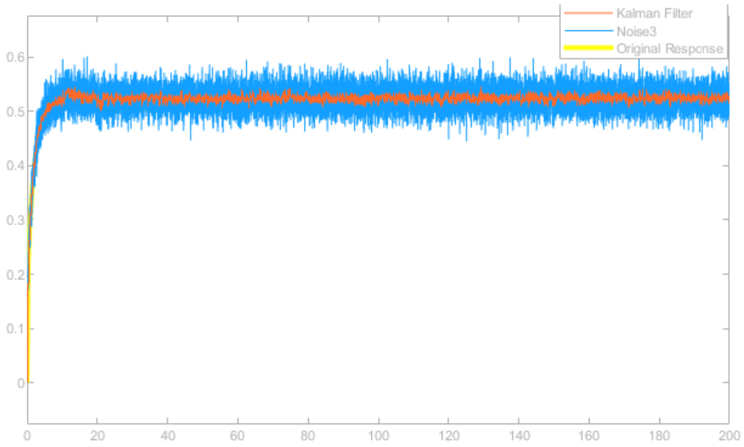


Figure 7.22 : Kalman filter effect on the disturbance of psi angle for open circular movement

7.4.4 Appying actuator fault to motor propeller angular speed components

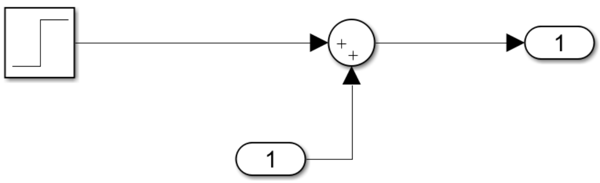


Figure 7.23 : Actuator Fault Model

In the figure 7.23 actuator fault is seen. As interpreted, a step input is used for the sensor fault.

In the figure 7.24, and 7.25, first, motor propellers are exposed to actuator faults at the 20. second by using a step response as the fault input. Since these faults are effective on the propellers in 20. second, a big jump is seen on that moment. Yet, the propeller speed response goes back to normal behavior due to the controller effect.

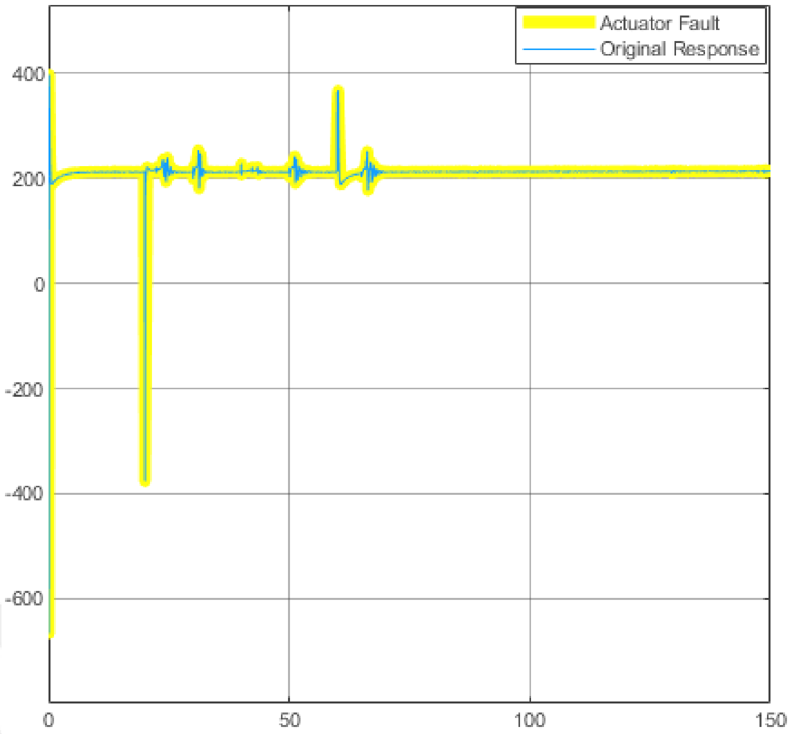


Figure 7.24 : Applying actuator fault to motor propeller speed in rectangular trajectory

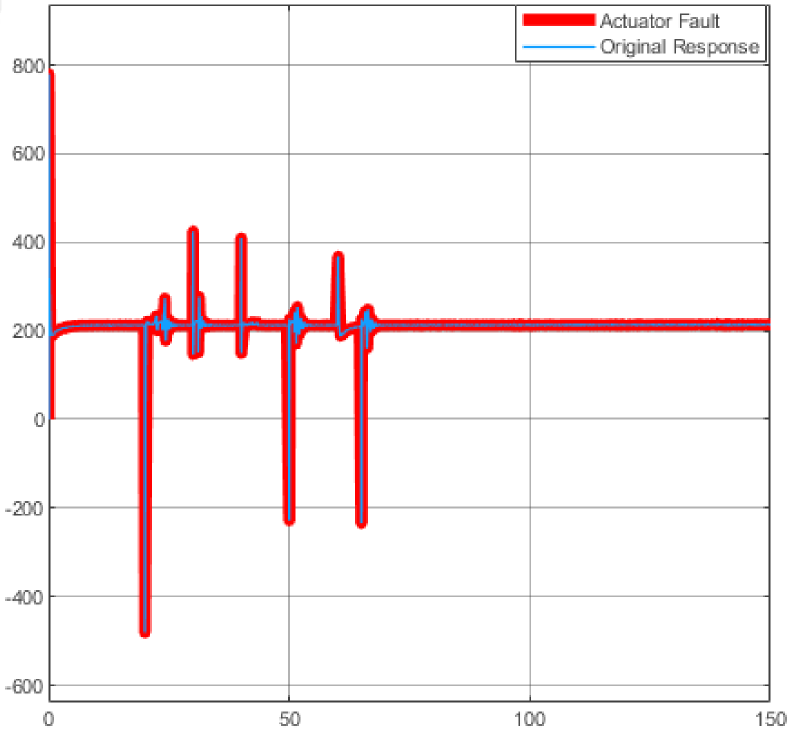


Figure 7.25 : Applying actuator fault to motor propeller speed in open linear trajectory

8. VISULATION

8.1 Flight Gear Application On the Quadrotor

8.1.1 Definition of flight gear application

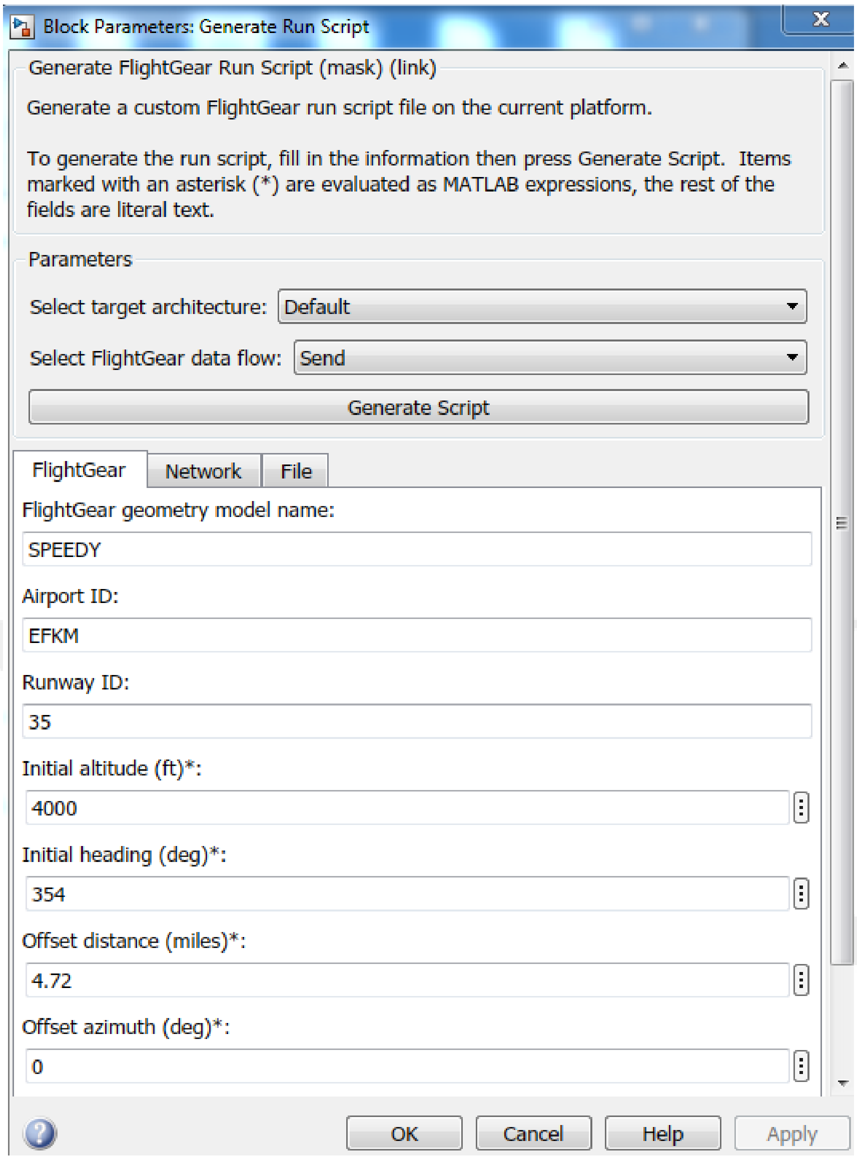
Flight gear simulator is a software application that provides the advantage of seeing and understanding whether our flying model flies properly or not. In this thesis, Flight Gear is implemented to the quadrotor simulink model. The determined quadrotor simulink trajectory is implemented to the Flight Gear simulator. This trajectory is observed in the Flight Gear interface. And also, this simulator enables almost real scenes from different airlines.

8.1.2 Flight gear simulink components

In figure 8.1, Flight Gear Run Script component is illustrated. In this component, as seen, model name selected for simulator and necessary airport information like airport ID, Runway ID, Initial altitude, initial heading and offset distance are included. By using these quadrotor geometry and airport information, the exact place of the quadrotor in the Flight Gear is determined before the flight. After necessary places are filled, Generate Script button should be pressed. This button will generate a flight Gear script for the simulator.

In figure 8.2, Flat Earth to LLA component is illustrated. This block takes linear position component X, Y and Z from the simulink model and estimates appropriate latitude, longitude and altitude from flat Earth position.

In figure 8.3, 6Dof animation component of flight gear is presented. This component takes latitude, longitude, altitude from from Flat Earth to LLA component as the first input and roll, pitch, yaw angles from the model outputs as the second input. Animation component drives position and attitude values to the Flight Gear Simulator vehicle.



Generate FlightGear Run Script (mask) (link)

Generate a custom FlightGear run script file on the current platform.

To generate the run script, fill in the information then press Generate Script. Items marked with an asterisk (*) are evaluated as MATLAB expressions, the rest of the fields are literal text.

Parameters

Select target architecture: Default

Select FlightGear data flow: Send

FlightGear geometry model name: SPEEDY

Airport ID: EFKM

Runway ID: 35

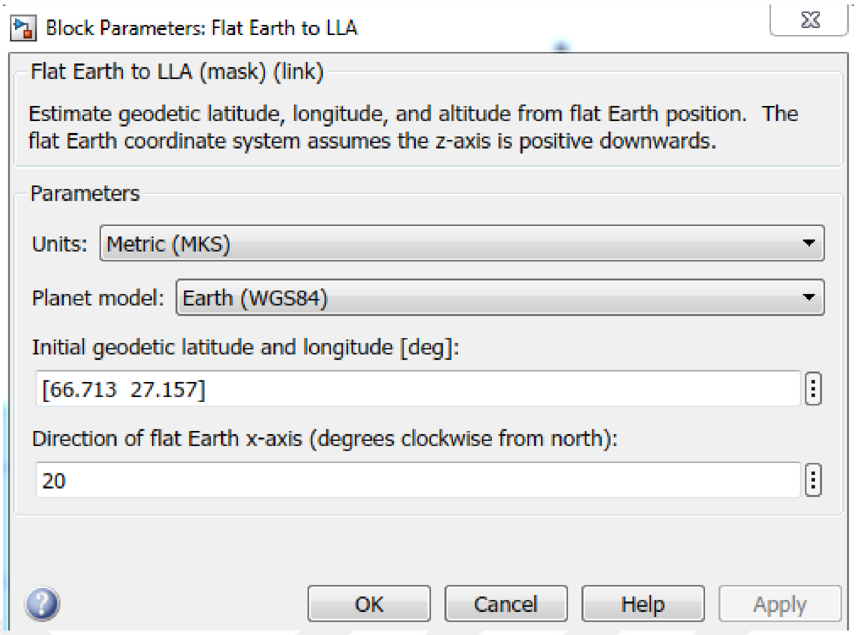
Initial altitude (ft)*: 4000

Initial heading (deg)*: 354

Offset distance (miles)*: 4.72

Offset azimuth (deg)*: 0

Figure 8.1 : FlightGear run script component



Flat Earth to LLA (mask) (link)

Estimate geodetic latitude, longitude, and altitude from flat Earth position. The flat Earth coordinate system assumes the z-axis is positive downwards.

Parameters

Units: Metric (MKS)

Planet model: Earth (WGS84)

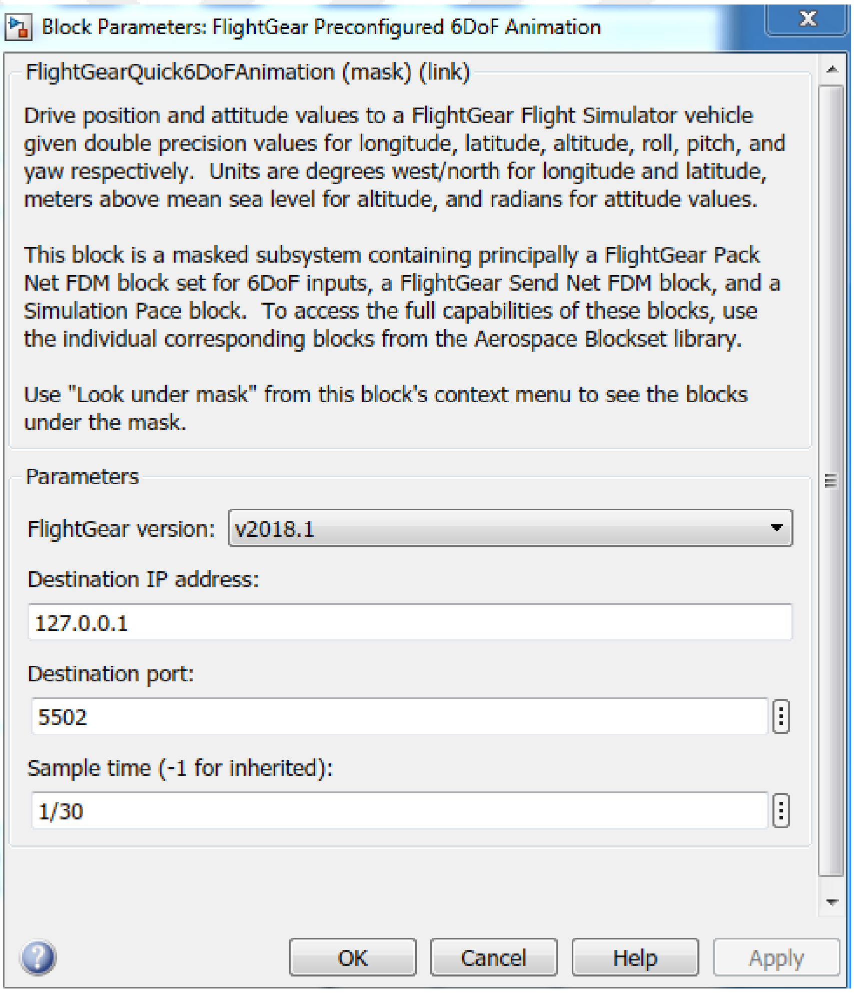
Initial geodetic latitude and longitude [deg]:

[66.713 27.157]

Direction of flat Earth x-axis (degrees clockwise from north):

20

Figure 8.2 : Flat earth to LLA component



FlightGearQuick6DoFAnimation (mask) (link)

Drive position and attitude values to a FlightGear Flight Simulator vehicle given double precision values for longitude, latitude, altitude, roll, pitch, and yaw respectively. Units are degrees west/north for longitude and latitude, meters above mean sea level for altitude, and radians for attitude values.

This block is a masked subsystem containing principally a FlightGear Pack Net FDM block set for 6DoF inputs, a FlightGear Send Net FDM block, and a Simulation Pace block. To access the full capabilities of these blocks, use the individual corresponding blocks from the Aerospace Blockset library.

Use "Look under mask" from this block's context menu to see the blocks under the mask.

Parameters

FlightGear version: v2018.1

Destination IP address:

127.0.0.1

Destination port:

5502

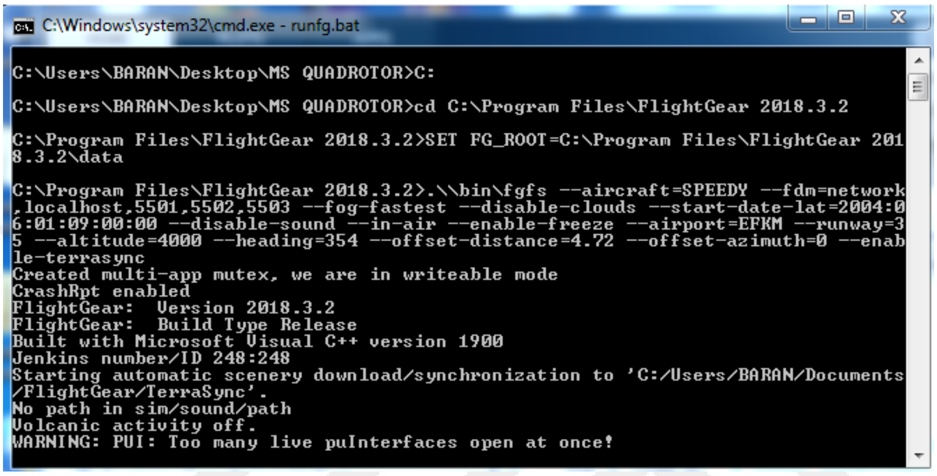
Sample time (-1 for inherited):

1/30

Figure 8.3 : 6DOF animation component

8.1.3 Flight gear simulink working principles

In order to run Flight Gear Simulator, the expression of dos('runfg.bat &') should be written to matlab command window. After that as seen in the following figure 8.4, Flight Gear code will be generated.



```
C:\Users\BARAN\Desktop\MS QUADROTOR>C:

C:\Users\BARAN\Desktop\MS QUADROTOR>cd C:\Program Files\FlightGear 2018.3.2

C:\Program Files\FlightGear 2018.3.2>SET FG_ROOT=C:\Program Files\FlightGear 201
8.3.2\data

C:\Program Files\FlightGear 2018.3.2>.\\bin\fgfs --aircraft=SPEEDY --fdm=network
,localhost,5501,5502,5503 --fog-fastest --disable-clouds --start-date-lat=2004:0
6:01:09:00:00 --disable-sound --in-air --enable-freeze --airport=EFKM --runway=3
5 --altitude=4000 --heading=354 --offset-distance=4.72 --offset-azimuth=0 --enab
le-terrasync
Created multi-app mutex, we are in writeable mode
CrashRpt enabled
FlightGear:  Version 2018.3.2
FlightGear:  Build Type Release
Built with Microsoft Visual C++ version 1900
Jenkins number/ID 248:248
Starting automatic scenery download/synchronization to 'C:/Users/BARAN/Documents
/FlightGear/TerraSync'.
No path in sim/sound/path
Volcanic activity off.
WARNING: PUI: Too many live puInterfaces open at once!
```

Figure 8.4 : runfg.bat file

After Flight Gear code is generated, the airport scenery is downloaded and loaded as in the following figure 8.5.

As the following step, the flight area is seen as observed in figure 8.6, 8.7, 8.8. The are seen in the related figures are determined with the help of Flight Gear blocks. In this area, the flight path quadrotor simulink model is observed. This flight includes the determined maneuvers of the quadrotor discrete trajectory model.

FlightGear
File View Location Autopilot Environment Equipment AI Multiplayer Debug Help
FlightGear
Open Source Flight Simulator
RESCUE
QZ 673
http://www.flightgear.org

downloading scenery

Figure 8.5 : scenery file

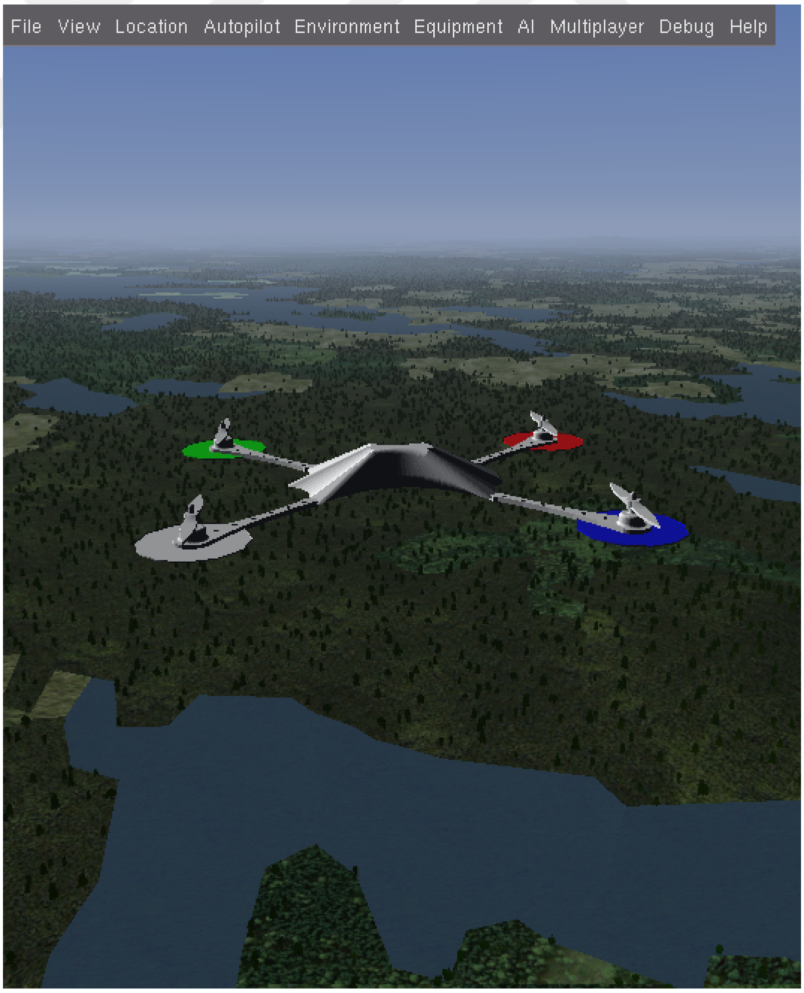

File View Location Autopilot Environment Equipment AI Multiplayer Debug Help

Figure 8.6 : Flight simulation file

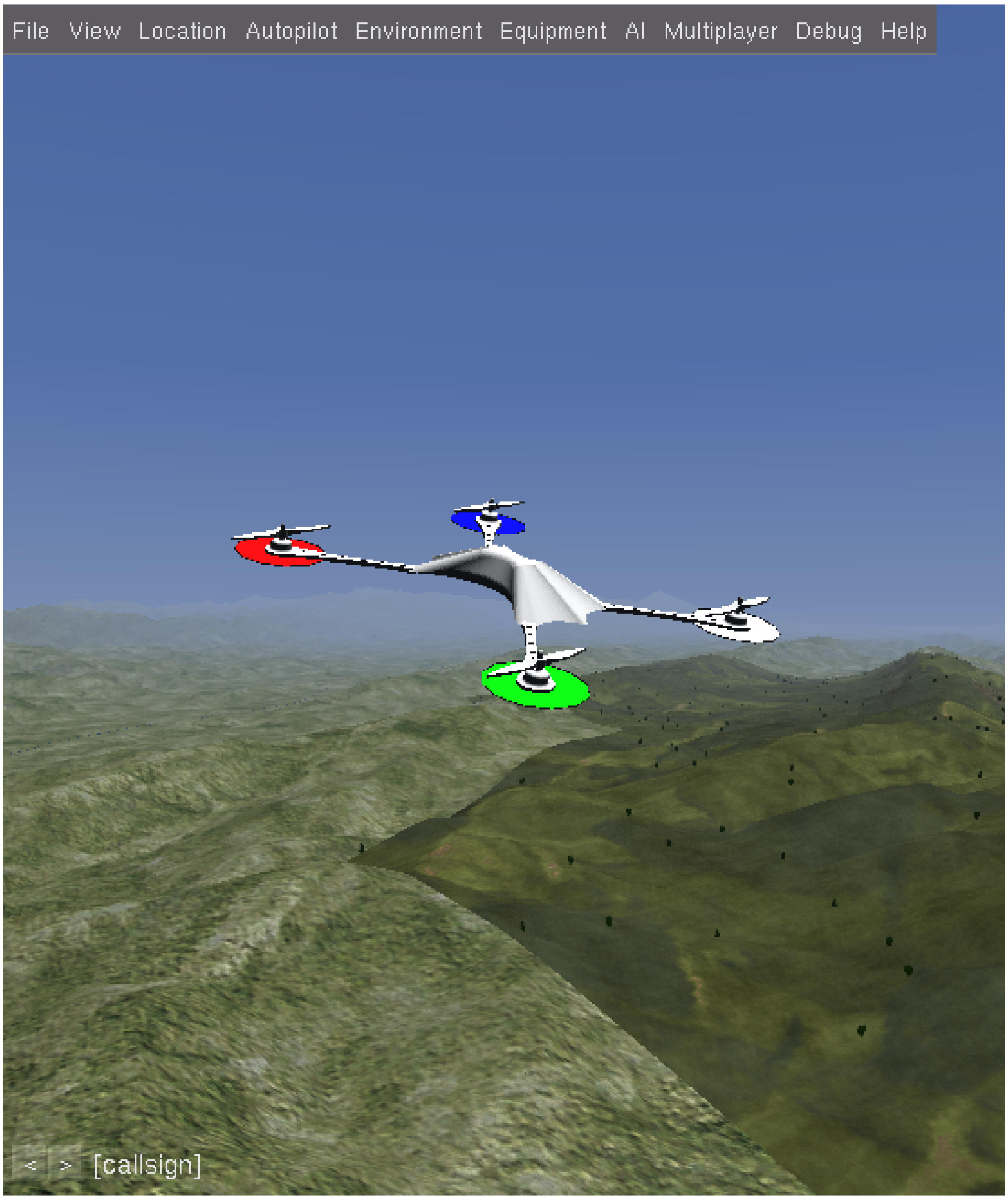


Figure 8.7 : Flight simulation file

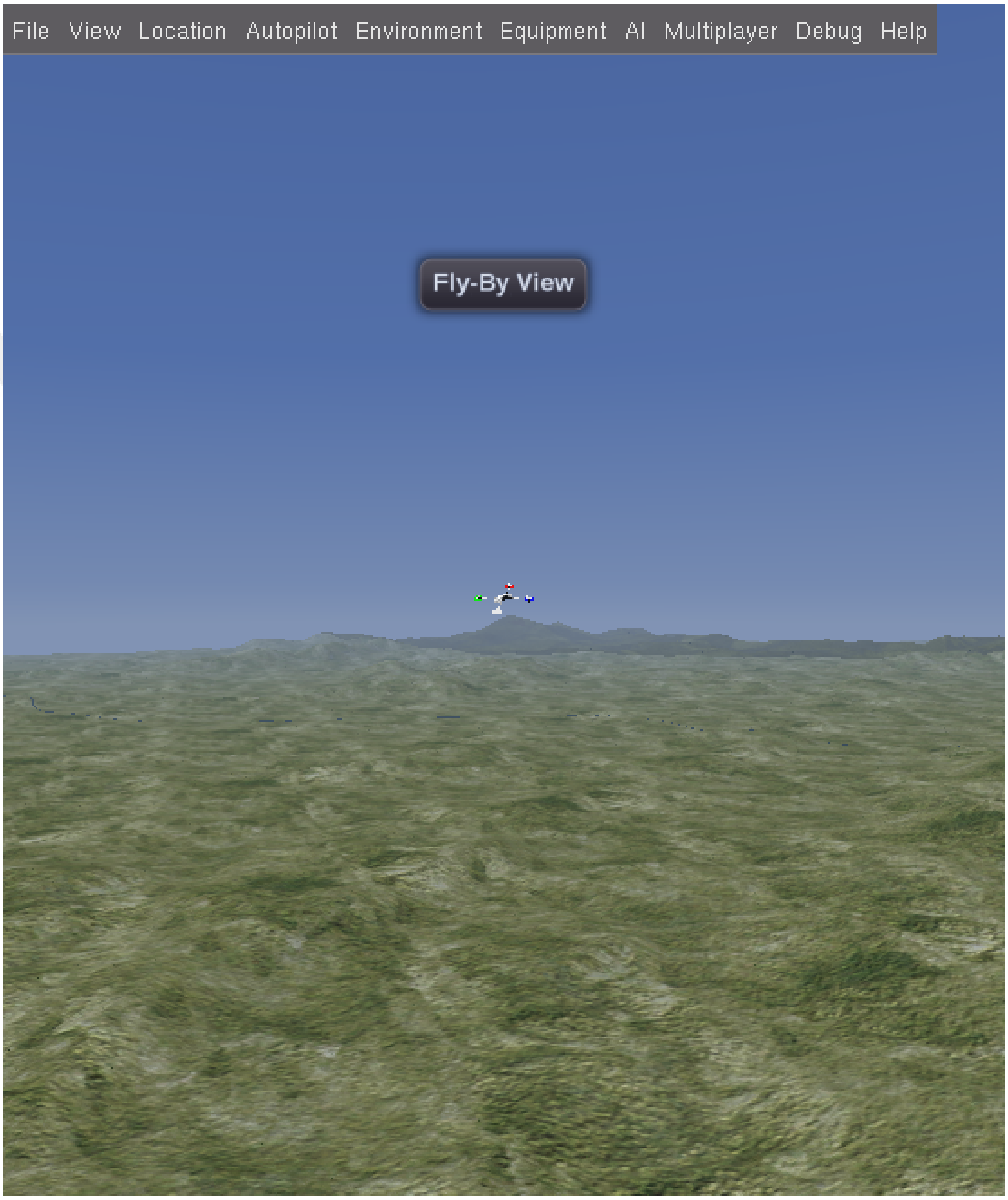


Figure 8.8 : Flight simulation file

9. CONCLUSION

In this thesis, the aim is to design a whole quadrotor model in simulink together with the linear model, real-time simulation and fault tolerant control systems. First of all, a complete quadrotor simulink model is constituted. In this model, with the help of matlab function block, different types of trajectories are achieved and controlled with discrete-time PID controller. In the dynamic model of the quadrotor, linear position, velocity and acceleration are achieved. Moreover, euler angles, motor propeller angular speeds, thrust, and roll, pitch, yaw torques are also performed in this model. In addition, the quadrotor simulink model is linearized and the results are taken. Furthermore, fault tolerant control system is applied to the nonlinear simulink model. Disturbance, sensor and actuator faults are implemented to this model. While sensor faults are added to different kinds of trajectories, actuator faults are added to motor propeller angular speeds. Sensor and actuator faults are applied at a specific moment and fixed with the help of discrete-time PID controller. The faulty responces are compared with the original ones. The faults occured because of disturbance are fixed with the help of Kalman Filter. By adding noise to the related system response, a fault is generated. Then a Kalman Filter is added to the faulty response. Kalman Filter removes most of the disturbance and the system response has an acceptable result. Finally, in order to perform real-time simulation, Flight Gear blocks are used. First of all, position and euler angles simulink model components are connected to the Flight Gear blocks then by running the Flight Gear, the flight area scenery is downloaded and this area is seen in the Flight Gear interface. After running the quadrotor simulink model, the flight simulation is started in the Flight Gear interface.

REFERENCES

[1] **Brequet-Richet Gyroplane No 1**, `http://www.aviastar.org/helicopters_eng/breguet_gyro.php`, date of access: 23.04.2019.

[2] **Oemnichen Helicopter**, `http://www.aviastar.org/helicopters_eng/oemichen.php`, date of access: 23.04.2019.

[3] **Curtis Wright VZ - 7**, `https://en.wikipedia.org/wiki/Curtiss-Wright_VZ-7`, date of access: 23.04.2019.

[4] **Microdrone md4-200**, `https://www.researchgate.net/figure/The-microdrone-md4-200_fig1_263409166`, date of access: 23.04.2019.

[5] **Quattcop**,`https://shop.hepf.com/Multi-Copter/Komplett-Set/SPYRIT-ADVANCE-QUATTROCOPTER-mit-beweglicher-Kamera::14448.html`, date of access: 23.04.2019.

[6] **DiscretePID**, `http://portal.ku.edu.tr/~cbasdogan/Courses/Robotics/projects/Discrete_PID.pdf`, date of access: 12.05.2019.

[7] **DiscreteControl**,`https://www.youtube.com/user/ControlLectures`, date of access: 18.05.2019.

[8] **DiscreteZOHFOH**, `http://web.cecs.pdx.edu/tymerski/ece452/Chapter3.pdf`, date of access: 12.05.2019.

[9] **DiscreteImpulse**, `http://homes.et.aau.dk/yang/DE5/DC/mm1.pdf`, date of access: 12.05.2019.

[10] **DiscreteMatched**, `https://en.wikipedia.org/wiki/Impulse_invariance`, date of access: 12.05.2019.

[11] **DiscreteAll**`http://www.adjutojunior.com.br/controle_processos/slides_chapter8_DISCRETIZATION_OF_CONTINUOS_SYSTEMS.pdf`, date of access:12.05.2019.

[12] **Bresciani, T.** (2008). Modelling, identification and control of a quadrotor helicopter, *MSc Theses*.

[13] **Sabatino, F.**, (2015), Quadrotor control: modeling, nonlinearcontrol design, and simulation.

[14] **Bora Erginer, H.**, (2007), Quadrotor VTOL Aracının Modellenmesi ve Kontrolü.

[15] **Musa, S.** (2018). Techniques for Quadcopter modeling and Design

[16] **El-Sharif, Ibrahim A and Hareb, Fathi O and Zerek, Amer R** (2014). Design of Discrete time PID controller

[17] **Ji, Ankyda and Turkoglu, Kamran** (2015). Development of a Low-Cost Experimental Quadcopter Testbed Using an Arduino Controller and Software

[18] **Fum, Wei Zhong** (2015). Implementation of Simulink controller design on Iris+ quadrotor:

[19] **Gopalakrishnan, Eswarmurthi**, (2016), Quadcopter flight mechanics model and control algorithms:

[20] **QASIM, MUHAMMAD**, (2013), Attitude control for a quadrotor helicopter:

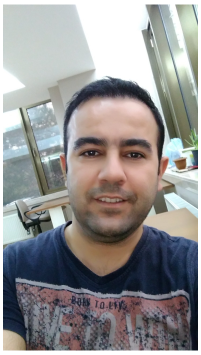
[21] **Schmidt, Michael David**, (2011), Simulation and control of a quadrotor unmanned aerial vehicle.

[22] **Singh, Pritpal**, (2015), Development of Unmanned Aerial Vehicle (Quadcopter) With Real-Time Object Tracking

[23] **Oh, Sang Min**, (2012), Modeling and Control of a Quad-rotor Helicopter

[24] **Jiang, Jin**, (2005), Fault-tolerant control systems—an introductory overview journal=, volume=31, number=1, pages=161–174, year=2005

CURRICULUM VITAE



Name Surname: Haci Baran

Place and Date of Birth: Van / 01.09.1989

E-Mail: baranh16@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2014, Melikşah University, Faculty of Engineering and Architecture, Electrical and Electronical Engineering