

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING



DIPLOMA THESIS

Micro Quadrotor: Design, Modelling,
Identification and Control

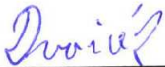
Prague, 2011

Author: Jaromír Dvořák

Prohlášení

Prohlašuji, že jsem svou diplomovou (bakalářskou) práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v příloženém seznamu.

V Praze dne 12.5.2011



podpis

Acknowledgements

First of all, my deepest gratitude belongs to Zdeněk Hurák, for his great support, which enabled me to finish this work. The thesis would not also have been possible without the project contractor and sponsor Kamil Knotek. I would like to thank Pavel Dvořák for the cooperation in hardware development and finally, Marcelo de Lellis, who has made available his support in a number of ways.

This work is dedicated to my dear Katie.

Abstract

The latest developments in MEMS technology, microcontrollers, electrical energy accumulators and motors combined with cheaper costs have fomented a growing number of studies and designs of small UAVs such as quadrotors. This work shows development results of one of such projects and tries to collect its main theoretical and practical aspects, while highlighting the modelling and controller synthesis. The work presents a complete strategy on how to design an advanced, structured flight controller for a quadrotor using an alternative control methods, while trying to document other related aspects of the project, such as quadrotor-specific AHRS algorithm issues, real-time control over wireless link and RPM measurement of BLDC motors by its controllers. Unlike the traditional approach of linearisation around some operating point, the work introduces an advanced singularity-free eigenaxis non-linear controller, capable to handle large-scale maneuvers, such as flip-over.

Abstrakt

Pokrok poslední doby v oblasti MEMS technologie, mikrokontrolérů, akumulátorů a motorů v kombinaci s jejich nižšími cenami způsobil nárůst počtu studií a projektů týkajících se vývoje malých létajících bezpilotních prostředků, jako jsou kvadrotory. Tato práce ukazuje výsledky vývoje jednoho z takových projektů a snaží se představit jeho hlavní teoretické a praktické aspekty, ale hlavně se zaměřuje na modelování a návrh řízení. Práce představuje kompletní strategii jak navrhnout pokročilé, struktorované řízení letu kvadrotoru s použitím alternativních metod řízení. Práce také ukazuje ostatní související aspekty projektu, jako problémy vývoje quadrotor-specific AHRS algoritmu, real-time řízení skrz bezdrátovou linku a měření otáček BLDC motorů jejich regulátorem. Narozdíl od tradičního přístupu linearizace nelineárního modelu kolem operačního bodu, práce představuje pokročilý regulátor rotace pomocí vlastní osy, schopný rozsáhlých manévru, jako je looping.

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Jaromír Dvořák
Study programme: Cybernetics and Robotics
Specialisation: Robotics
Title of Diploma Thesis: Micro Quadrotor: Design, Modeling, Identification and Control

Guidelines:

1. Design and implement a control system for a small indoor quadrotor, which ensures inertial stabilization using inertial measurements (linear accelerations and angular rates).
2. Build a mathematical model and use it for numerical simulation of dynamics. Compare its outputs with laboratory experiments.
3. Implement the simulation model in one of the commonly used environments such as Matlab/Simulink/SimMechanics, Scilab/Scicos or OpenModelica. Document the model perfectly.
4. Design several controllers using both the classical methods (frequency methods, PID) and advanced optimization-based techniques (LQ, Hinf) and compare.

Bibliography/Sources:

- [1] Pounds, P.; Mahony, R.; Gresham, J.: Towards Dynamically Favourable Quad-Rotor Aerial Robots. In Australian Conference on Robotics and Automation, 2004.
- [2] Castillo, P.; Dzul, A.; Lozano, R.: Real-Time Stabilization and Tracking of a Four-Rotor Mini Rotorcraft. IEEE Transactions on Control Systems Technology, Vol. 12, No. 4, pp. 510-516, 2004.
- [3] Bouabdallah, S.; Noth, A.; Siegwart, R.: PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor, In Proc. of Intelligent Robots and Systems (IROS 2004) Vol.3, Oct 2004, pp 2451- 2456.

Diploma Thesis Supervisor: Ing. Zdeněk Hurák, Ph.D.

Valid until: the end of the summer semester of academic year 2011/2012


prof. Ing. Vladimír Mařík, DrSc.
Head of Department




prof. Ing. Boris Šimák, CSc.
Dean

Prague, February 2, 2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Jaromír Dvořák
Studijní program: Kybernetika a robotika (magisterský)
Obor: Robotika
Název tématu: Mikro kvadrotor: návrh, modelování, identifikace a řízení

Pokyny pro vypracování:

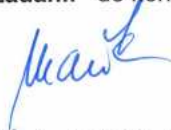
1. Navrhněte a realizujte řídicí systém pro malý čtyřrotorový vrtulník (angl. quadrotor či quadcopter), který zajistí inerciální stabilizaci na základě měření inerciálních úhlových rychlostí a/nebo lineárních zrychlení.
2. Navrhněte matematický model a proveďte numerickou simulaci dynamiky uvažovaného prostředku a srovnání modelu s výsledky identifikačních experimentů.
3. Výsledný model implementujte v některém z běžně používaných modelovacích prostředí (Matlab/Simulink/ SimMechanics, Scilab/Scisos nebo OpenModelica) a dokonale dokumentujte, aby mohl být použit pro výzkum dynamiky letu i návrh řízení dalšími studenty.
4. Pro algoritmický návrh regulátorů použijte jak klasické metody (tvarování frekvenčních charakteristik, PID), tak i pokročilé optimalizační metody (LQ-optimální, Hinf robustní).

Seznam odborné literatury:

- [1] Pounds, P.; Mahony, R.; Gresham, J.: Towards Dynamically Favourable Quad-Rotor Aerial Robots. In Australian Conference on Robotics and Automation, 2004.
- [2] Castillo, P.; Dzul, A.; Lozano, R.: Real-Time Stabilization and Tracking of a Four-Rotor Mini Rotorcraft. IEEE Transactions on Control Systems Technology, Vol. 12, No. 4, pp. 510-516, 2004.
- [3] Bouabdallah, S.; Noth, A.; Siegwart, R.: PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor, In Proc. of Intelligent Robots and Systems (IROS 2004) Vol.3, Oct 2004, pp 2451- 2456.

Vedoucí diplomové práce: Ing. Zdeněk Hurák, Ph.D.

Platnost zadání: do konce letního semestru 2011/2012


prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry




prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 2. 2. 2011

Contents

List of figures	xii
List of symbols	xv
List of tables	xviii
1 Introduction	1
1.1 Motivation and background	1
1.2 Why quad-rotor?	1
1.3 Contributions	3
1.4 Subdivision	4
2 System design, construction and programming	5
2.1 Project status summary	5
2.2 Hardware	6
2.2.1 Quadrotor body and mechanical organisation	8
2.2.2 Actuators	9
2.2.2.1 Selection of BLDC motor	9
2.2.2.2 Design of BLDC controller	10
2.2.3 Power source	12
2.2.4 Sensor mainboard	13
2.2.5 Wireless data link	14
2.2.6 Groundstation circuit	15
2.3 Software	16
2.3.1 Microcontroller firmware	16
2.3.2 PC control software	16
3 Analysis, modelling and controller design	19
3.1 Dynamic Model	19
3.1.1 Rigid-Body Dynamics	20
3.1.2 Actuator Dynamics	21
3.1.2.1 Non-linear Actuator Model	21
3.1.2.2 Linearized Actuator Model	21
3.1.2.3 Input and output non-linear mapping	22
3.1.3 Acting Forces and Moments	23
3.1.3.1 Axial Thrust moments and Drag torque moment	23

3.1.3.2	Gyroscopic Moments from Rotors	24
3.1.3.3	Thrust/lift force	25
3.1.3.4	Earth's Gravity force	25
3.1.3.5	Aerodynamic Forces and Moments	26
3.1.4	Complete Non-Linear Model	26
3.1.4.1	Computer numerical simulation of derived model	27
3.2	Controller design	27
3.2.1	Control Goals	27
3.2.2	Orientation Representation	29
3.2.3	Desired Orientation and Thrust	30
3.2.4	Control of Orientation using Eigenaxis	31
3.2.5	Rotation Controller Synthesis	33
3.2.5.1	Axial Decoupling	33
3.2.5.2	LQ-optimal controller with non-linear overshoot compensator	35
3.2.5.3	Non-linear gyroscopic compensation and thrust control	37
3.2.5.4	Overall control law and actuator linearisation extensions	38
3.2.6	Translation controller design	39
3.2.6.1	Translation damper design	40
3.2.6.2	Position controller design	42
3.2.6.3	Final translation control law	42
3.3	Observer Design	43
3.3.1	Orientation observer design	43
3.3.1.1	The Accelerometer Paradox	44
3.3.1.2	Observer principle	45
3.3.1.3	Vectors as state of the observer	46
3.3.1.4	Quaternion extraction	47
3.3.1.5	Quaternion as state of the observer and gyro bias tracking	49
3.3.1.6	Combined state approach	50
3.3.1.7	Sensor calibration	52
3.3.2	Position observer design	54
4	Identification, implementation, experiments and results	57
4.1	Identification	57
4.1.1	Inertial Parameters	57
4.1.2	Actuator	58
4.2	Implementation and simulation experiments	62
4.2.1	Rotation controller	62
4.2.1.1	LQ-optimal control synthesis	62
4.2.1.2	Coupling with the eigenaxis algorithm	63
4.2.1.3	Gyroscopic compensator	63
4.2.1.4	Overshoot compensator	63
4.2.1.5	Actuator non-linear mapping	65
4.2.2	Target orientation and thrust computation	65
4.2.3	Translation controller	66
4.2.3.1	Translation damper	67

4.2.3.2	Position controller	68
4.2.4	Orientation and position observer	68
4.3	Laboratory experiments	70
5	Conclusion	75
5.1	Future Works	76
	References	79
A	Complete Simulink model	I
B	Simulation plots	IX
C	Appended CD	XIII

List of Figures

1.1	The quadrotor concept.	2
2.1	Project block diagram.	5
2.2	The complete quadrotor assembly.	6
2.3	The hardware parts interconnection block diagram.	7
2.4	Body construction.	8
2.5	Sensor board mounted on the body with elastic bands.	8
2.6	Visualisation of magnetic field inside the BLDC motor.	9
2.7	3D 550 E from PJS.	10
2.8	Block diagram of the general BLDC controller.	11
2.9	Illustrational photo of the controller PCB.	12
2.10	Photo of the main battery.	13
2.11	Assembly of the sensor board.	14
2.12	Simplified timing diagram of the data flow.	14
2.13	Assembly of the groundstation logic board.	15
2.14	Control software block diagram.	16
2.15	Control software visualisation using OpenGL.	17
3.1	Quadrotor's fixed-body reference frame.	19
3.2	Actuator model.	22
3.3	Separate rigid body dynamics	28
3.4	basic model inputs/outputs	28
3.5	Proposed control structure.	29
3.6	Target orientation and thrust block placement in the global structure.	30
3.7	Relation between eigenaxis and body angular velocities.	33
3.8	Rotation controller placement in the global structure.	34
3.9	Single axis dynamic model.	34
3.10	Single axis dynamic model during correction maneuver.	35
3.11	Single axis dynamics with asymptotic tracking and overshoot compensator.	37
3.12	Translation controller placement in the global structure.	39
3.13	Proposed structure of the translation controller.	40
3.14	Orientation observer placement in the global structure.	43
3.15	The 3DOF 2D simplification of a basic quadrotor model.	45
3.16	General diagram of the orientation observer.	46
3.17	Bias tracking principle in a single degree of freedom system.	50
3.18	The AHRS algorithm flowchart.	51

3.19	The AHRS algorithm in pseudo-code.	52
3.20	Visualisation of the ellipsoid fitted to the measured data.	53
3.21	Position observer placement in the global structure.	54
4.1	Quadrotor's airframe and inertial identification scheme.	58
4.2	Scheme of the thrust output function measurement experiment.	59
4.3	Output functions identification of the actuator.	59
4.4	DC characteristics of the actuator.	60
4.5	Identification of the rotor moment of inertia.	60
4.6	Identification of the actuator dynamics.	61
4.7	Decoupled axial dynamics.	62
4.8	Omega magnitude comparison.	63
4.9	Gyroscopic compensator comparison.	64
4.10	Overshoot compensator performance.	64
4.11	Voltages applied to the actuators.	65
4.12	Non-linear actuator extension comparison.	65
4.13	Step response of acceleration commands.	66
4.14	Step response of heading command.	66
4.15	Step response of acceleration commands with the non-linear mapping.	67
4.16	Simple translation damper structure.	68
4.17	Translation controller response with damping in NED frame.	68
4.18	Translation controller response with damping in body frame.	69
4.19	Translation controller performance.	69
4.20	First, wired outdoor flight, 10.4.2010.	70
4.21	Control software running on MS Windows PC.	71
4.22	Eigenaxis error during the flight.	72
4.23	Quaternion elements tracking and error.	73
4.24	Acceleration commands tracking and error.	73
4.25	Heading command tracking and error.	73
A.1	The overall closed-loop diagram.	II
A.2	The quadrotor diagram.	III
A.3	The rotation controller diagram.	IV
A.4	The gyroscopic compensator diagram.	V
A.5	The quaternion logarithm Matlab code.	VI
A.6	The target orientation and thrust computation Matlab code.	VI
A.7	The translation controller diagram.	VII
A.8	The simplified 2D-quadrotor model with rotation controller diagram.	VIII
B.1	Position history.	IX
B.2	Target acceleration history.	X
B.3	Quaternion elements history.	X
B.4	Eigenaxis error history.	X
B.5	Actuator voltage history.	XI
B.6	Actuator angular velocity history.	XI
B.7	Body angular velocity history.	XI

B.8	Body linear velocity history.	XII
B.9	Body acceleration history.	XII
B.10	Accelerometer reading history.	XII

List of Symbols

<i>Symbol</i>	<i>Description</i>	<i>SI unit</i>
Ω	Angular velocity in body frame	rad/s
p	Angular velocity in body frame around X axis	rad/s
q	Angular velocity in body frame around Y axis	rad/s
r	Angular velocity in body frame around Z axis	rad/s
\mathbf{L}	Angular momentum vector in body frame	N·m·s
\mathbf{M}	Total angular moment in body frame	N·m
\mathbf{I}	Inertia tensor of the quadrotor in body frame	kg·m ²
I_x	moment of inertia around X axis	kg·m ²
I_y	moment of inertia around Y axis	kg·m ²
I_z	moment of inertia around Z axis	kg·m ²
\mathbf{V}	Translational velocity in body frame	m/s
u	Translational velocity in body frame, X axis	m/s
v	Translational velocity in body frame, Y axis	m/s
w	Translational velocity in body frame, Z axis	m/s
\mathbf{V}_e	Translational velocity in NED frame	m/s
\mathbf{X}	Position of the quadrotor in NED frame	m
\mathbf{P}	Translational momentum vector in body frame	kg·m/s
\mathbf{F}	Total translational force in body frame	N
m_t	Total mass of the quadrotor	kg
\mathbf{g}	Earth's acceleration vector	m/s ²
u_j	Unbiased voltage applied to the j-th actuator	V
ω_j	Angular velocity of j-th actuator	rad/s
T_j	Thrust output of j-th actuator	N
q_j	Torque output of j-th actuator	N·m
δ_j	Drag torque of j-th actuator	N·m
\mathbf{M}^j	Angular moment of j-th actuator	N·m
L_r	Summed angular momentum of all the actuators	none
T	Total thrust generated by the actuators	N
l_a	distance from each motor to CG	m
ω_0	angular velocity of the operating point	rad/s
u_0	voltage of the operating point	V
I_r	total moment of inertia of the actuator	kg·m ²

K	linear gain of the actuator	V·s/rad
B	bearing damping of the actuator	N·m·s/rad
H	thrust output non-linearity gain	none
F	drag torque output non-linearity gain	none
G_1	actuator voltage to torque constant for stepup	none
G_2	actuator voltage to torque constant for stepdown	none
k	linearised actuator constant: sensitivity	none
l	linearised actuator constant: damping	none
m	linearised actuator constant: thrust change	none
n	linearised actuator constant: drag torque change	none
k_d	Thrust gain correction non-linear mapping constant	none
k_i	Actuator input non-linear mapping constant	none
l_i	Actuator output non-linear mapping constant	none
\mathbf{F}_g	Gravity force	m/s ²
t_{ad}	Translation aerodynamic damping constant	N·s/m
r_{ad}	Rotation aerodynamic damping constant	N·s/rad
\mathbf{Q}	Current orientation state quaternion	none
α	Angle of rotation	rad
$\hat{\mathbf{a}}$	Target acceleration in NED frame	m/s ²
$\hat{\psi}$	Target heading in NED frame	rad
\mathbf{Q}_t	Target tilt quaternion	none
\mathbf{Q}_y	Target yaw quaternion	none
$\hat{\mathbf{Q}}$	Target quaternion	none
$\hat{\mathbf{X}}$	Target position in NED frame	m
\hat{T}	Target thrust in vertical axis	N
\mathbf{Q}_e	Error quaternion	none
\mathbf{r}	Eigenaxis of the error correction maneuver	none
\mathbf{E}	Eigenaxis error vector	rad
\tilde{u}_j	Decoupled axial voltage	V
$\tilde{\omega}_j$	Decoupled axial angular velocity	rad/s
\mathbf{K}_j	LQ-control feedback for j-th axis	none
L_j	Overshoot compensator gain constant for j-th axis	none
Kd_{lat}	Lateral translation damper: gain constant	none
τ_{lat}	Lateral translation damper: time constant	sec
Kp_{lat}	Lateral position controller: proportional term	none
Ki_{lat}	Lateral position controller: integral term	none
Kd_{long}	Longitudinal translation damper: gain constant	none
τ_{long}	Longitudinal translation damper: time constant	sec
Kp_{long}	Longitudinal position controller: proportional term	none
Ki_{long}	Longitudinal position controller: integral term	none
acc	Calibrated accelerometer reading	m/s ²
mag	Calibrated magnetometer reading	none
gyro	Calibrated gyro reading	rad/s
\mathbf{S}_a	Estimated Earth's acceleration vector	m/s ²

S_m	Estimated Earth's magnetic vector	none
S_p	Accelerometer paradox compensator state	none
S_b	Gyro bias estimation vector	rad/s
S_α	Estimated magnetic inclination angle	rad
S_Q	Estimated orientation quaternion	none
Q_c	Measurement error quaternion	none
Q_r	Measurement quaternion	none
K_a	Accelerometer correction weight	none
K_m	Magnetometer correction weight	none
K_b	Bias estimation correction weight	none
K_Q	Quaternion correction weight	none
τ_α	Magnetic inclination lowpass filter time constant	sec
τ_{ac}	Accelerometer paradox compensator time constant	none
K_{ac}	Accelerometer paradox compensator gain constant	none

List of Tables

- 1.1 Difference between the main aerial concepts. 2
- 2.1 Key parameters of the selected BLDC motor. 10
- 2.2 Key parameters of the BLDC controller. 12
- 3.1 Systematic errors of the inertial sensors 44
- 4.1 Basic identified model constants of the body dynamics 58
- 4.2 Basic actuator model identified constants 61
- 4.3 LQ-optimal feedback law for decoupled dynamics 62
- 4.4 Basic actuator model identified constants 67
- 4.5 Orientation observer parameters 70

Chapter 1

Introduction

Goal of the background project was to design, build and control a function sample of a quadrotor. This work presents results of two year intensive project development and tries to collect its main theoretical and practical elements, while focusing on the modelling and controller synthesis. Please note that the development is still in progress, hence some of the key aspects are remained open. Main result of this work is a complete quadrotor model with designed controller.

1.1 Motivation and background

History of this project reaches quite far into the past. Since the growth of aerial industry, control engineering and microprocessors, the area of autonomous flying machines always belonged into my field of interest. When becoming CTU student, I had already designed inertial measurement unit board and want to use it to control a conventional RC model helicopter. Two years ago, I got an offer from VIPRON s.r.o. to *design and realise a low-level control system for autonomous flying robot*. Having the required construction skills, component base and willing to learn, I started to build the new quadrotor for education purposes, using the knowledge base from existing projects together with my own ideas.

1.2 Why quad-rotor?

From one of the very basic field of view, the aerial vehicles can be divided into two groups. The table 1.1 shows some of the main features.

For our approach, there are undeniable advantages of a rotating wing over the fixed wing aircraft, especially for short-range or indoor missions where maneuverability, omnidirectional movement, ability of hovering and obstacle avoidance is critical. The main difference is that the copter is omnidirectional in translation, i.e. it can move up, down, left or right, backward or forward in space independently. Price to pay for this property

Table 1.1: Difference between the main aerial concepts.

	<i>Fixed wing</i>	<i>Rotating wing</i>
<i>Power consumption</i>	lower	higher
<i>Maneuverability</i>	non-omnidirectional	omnidirectional
<i>Range</i>	higher	lower
<i>Vtol</i>	no	yes
<i>Speed</i>	higher	lower
<i>Hover mode</i>	no	yes
<i>Control fault tolerance</i>	not always critical	often deadly

is the need for constant power to the propellers to "keep the device in-the-air", which results to shorter battery life and shorter flight range.



Figure 1.1: The quadrotor concept.

<http://www.rc-airplane-world.com/rc-ufo.html>

There are many rotating-wing aircraft configurations, like the classical helicopter model with tail rotor for yaw control or multi-rotor and co-axial configurations. Quadrotor over the classical helicopter concept is mechanically much simpler and easier to construct. There is no need for complicated rotor head, blades that must change their attack angles, collective pitching, nor tail rotor for yaw stabilisation. There are only 4 (or any even number) of rotating propellers with parallel axis mounted around a rigid body. The propeller doesn't only generate a static thrust, but also a torque, which can be effectively used to control the yaw angle. One half of the propellers are rotating clockwise, the other half counter-clockwise. That's why even number of propellers is needed. Attitude, heading - (*spatial orientation*) and movement - (*spatial translation*) is controlled via the power of propeller motors. Each propeller must be, of course, controllable separately, which makes it a challenging control engineering problem. By using today's brushless synchronous motors with excellent rigidity and lifetime, which actually depends

only on their bearings, the resulting device is, from the mechanical field of view, practically faulty-free, care-free, and can be easily constructed using cheap, available parts. In recent years, the quadrotor aircraft concept has become popular for UAV applications due to their high maneuvering capabilities, making them especially adequate not only for aerial surveillance applications, but also as single units in swarm robotics and collective behaviour studies.

1.3 Contributions

- The project involves design and construction of the function sample, to be used as a basis for further development and education. All the electronic circuitry was therefore designed especially for this project, aiming to utilize the potential of currently available parts and come through the competition with another similar projects. From special features of this project I can mention the custom-made BLDC controllers allowing the motor RPM measurement and faster transient responses, or alternative usage of some commonly available sensors, like optical mouse camera with attached optics for visual motion detection. Finally, the controller was placed outside of the quadrotor body, which requires a special workaround to allow fast real-time control over the wireless link, but enables a comfortable way of designing a controller, easy demonstration, education and much more.
- Apart from an ordinary control problem, where U is input to and Y is a *known* output from a real system we wish to control, the quadrotor control problem is bit more complicated. Usually, the target values for an aircraft are spatial orientation and/or spatial position. Nevertheless, unlike a rotary encoder in an inverted pendulum, there is no solid-state sensor that can provide these variables. In fact, the output vector, Y , is not known. The value of Y can only be *estimated* using a multiple sensors which reads a related values. Apart from vision-based systems [51], that uses some external sensors to determine orientation/position, one of the goals of this work will be to discuss the possibilities of an orientation and position *observers* for the quadrotor concept, making it independent on external workarounds. The highly experimental approach of the quadrotor-specific AHRS unit was synthesized and will be presented.
- Regarding the aircraft as rigid body, the key question is how to describe its orientation. *Euler angles* have been successfully used for control of fixed-wing aircraft as natural representation of errors from straightforward flight. Nevertheless, for an omnidirectional aircraft, non-linearities arise when moving far from the zero orientation state. As an attitude description alternative, quaternions offer a potentially significant advantage in terms of mathematical robustness, not suffering from singularities, but also by enabling a more energy-optimized control action, especially when more aggressive maneuvers are considered. Despite these advantages, they have been relatively neglected in aeronautics. In aerospace field, the problem of

reorienting an aircraft with minimum effort through large angles, without any path constraints, has been addressed in several contexts. A rigid body in any initial orientation can be rotated into a final orientation through a single rotation about a fixed axis, which is called an eigenaxis rotation and constitutes the "shortest" rotation between the two orientations [6]. Unlike the traditional approach of linearisation the non-linear model around some operating point, this work aims to synthesize a non-linear controller that will be operational in *entire orientational space* and thus able to handle very large-scale maneuvers, such as flip-over. The eigenaxis control approach was found to be new in the quadrotor design field, thus a paper about the problematics was sent to *IEEE MSC 2011*.

1.4 Subdivision

The work will begin with a short description of the real quadrotor project construction. In the next chapter, I will derive the complete, non-linear dynamic model of the aircraft, followed by the proposal of a suitable controller structure with separated translation and rotation subsystem. Advantages of eigenaxis control will be examined and advanced rotation controller with various non-linear extensions will be synthesized. Next, I will present a suitable precedent translation controller. Finally, possibilities of orientation/position state observers will be discussed. The last chapter will attempt to join both construction and modelling branches together. The system identification and controller implementation will be presented, various numerical simulations performed and finally, the designed controller will be verified on the real aircraft. Although the sections are presented one after another, the reader should be aware of the fact that encountered needs, limitations and necessary interventions during the system development influenced all the parts, thus the construction, modelling, identification and implementation were done more like in parallel.

Chapter 2

System design, construction and programming

2.1 Project status summary

The project is currently being developed in cooperation with the industrial contractor. The particular project subtask was to design and implement the vestibular part (low level dynamic controller) for our aircraft, represented by the coloured blocks in fig. 2.1. The precedent layer will consist of an *task/mission control* unit, which will pass the commands to the dynamic controller. Mission control unit design belongs to the area of machine perception, image processing and artificial intelligence, rather than control engineering.

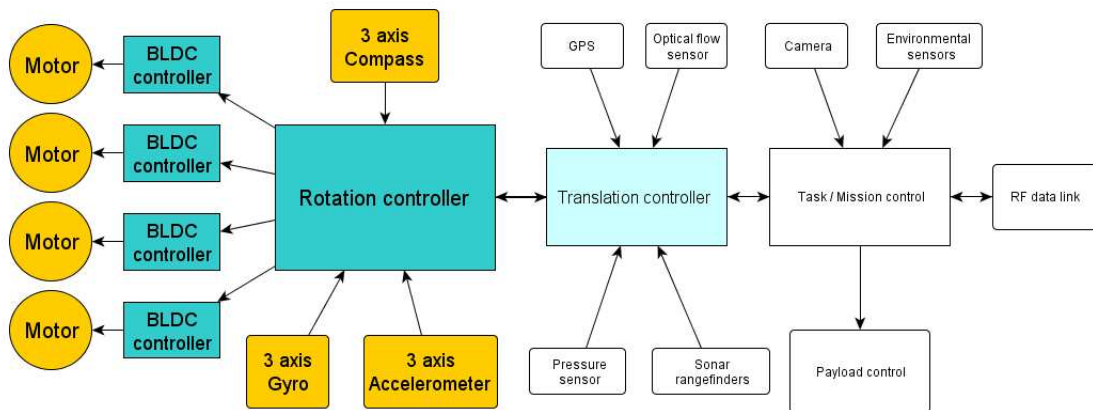


Figure 2.1: Project block diagram.

I have decided to build a function sample of the quadrotor at first, suitable especially for the educational purposes. Hence, personal computer is used in role of real-time controller hardware, instead of placing the controller onboard. This conception enables to do the development, tuning and adjustments in a very comfortable way and won't practically limit the computational power. Nevertheless, there is the need for a fast and low latency data link between the computer and the aircraft that will be able to transfer

the sensor measurements to the PC and motor commands back within one chosen discrete controller time step. There is only a simple sensor board onboard with fast data telemetry. When development is done, the controller will be placed onboard. The remaining part of this chapter will briefly describe the construction of this function sample.



Figure 2.2: The complete quadrotor assembly.

2.2 Hardware

Hardware part of the project consists of the groundstation and the vehicle. The quadrotor represents a combination of various elements, such as aluminium-carbon body, brushless motors, rechargeable battery, motor controllers, microcontrollers (MCU), sensors and aerial interface whereas the groundstation consists of a personal computer and interfacing circuitry. From electrical field of view, heart of both quadrotor and groundstation circuitry are microcontrollers handling the necessary peripherals. The onboard computer queries the sensors, passes the readings through data link, reads back the motor commands and sends them to the motor controllers. Ground microcontroller reads data from the air and from the RC model transmitter, passes them to the PC, waits for the

computation to complete and sends the result back to the aircraft.

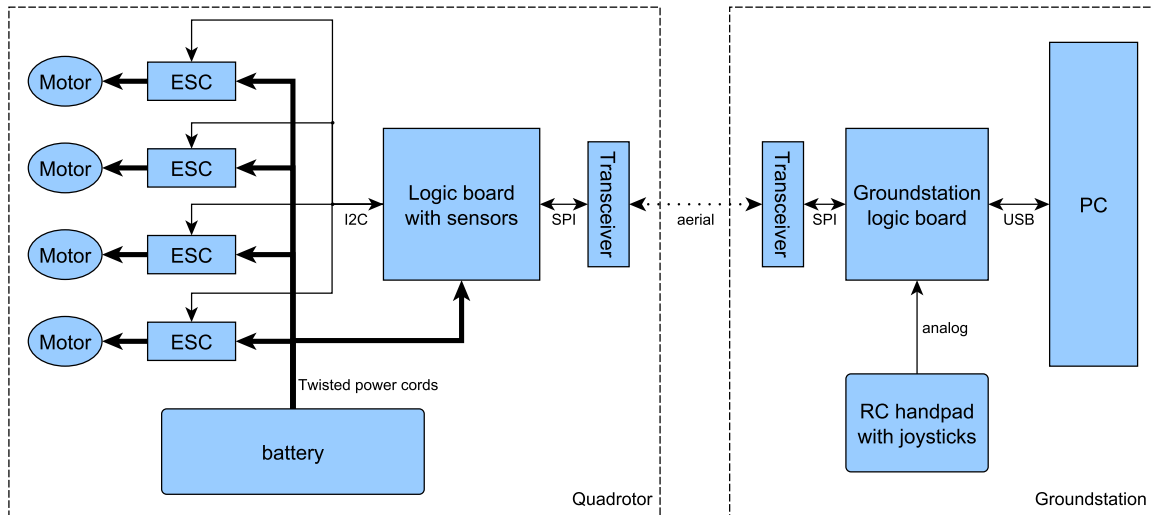


Figure 2.3: The hardware parts interconnection block diagram.

Main components of the groundstation

- RC handpad with two joysticks as human pilot interface
- logic board
- wireless data transceiver
- PC as the real-time dynamic controller

Main components of the aerial vehicle

- aluminium-carbon body
- four BLDC motors with attached propellers
- four BLDC motor controllers (ESC)
- 2.2Ah lithium-ion polymer battery
- logic board with sensors
- wireless data transceiver



(a) Aluminium cross piece.

(b) Aluminium motor mount.

Figure 2.4: Body construction.

2.2.1 Quadrotor body and mechanical organisation

The main body of the quadrotor is made of four carbon fibre pipes with $8mm$ diameter which are glued into the aluminium cross piece in the center. The arms are ended with aluminium mount with a perpendicular hole for motor rod.

Several mechanical problems were encountered during the vehicle construction. Primarily, the the rotating motors with just slightly unbalanced propellers are causing vibrations, which spreads through the body and interferes the accelerometer. For some values of revolutions per seconds ratio, high frequency acceleration shocks can arise in the center cross that are even out of accelerometer's range. Then the accelerometer becomes non-linear, mean value of its readings does not respond to the earth's gravity acceleration and error in orientation estimation will develop. The problem has been temporarily solved by mounting the entire sensor board on the body using elastic bands. The additional carbon pipes were also added as braces aiming to cancel the shocks in the center cross by spreading the vibrations through them. I am planning the further experiments with various other materials and main body construction to cancel the vibrations.

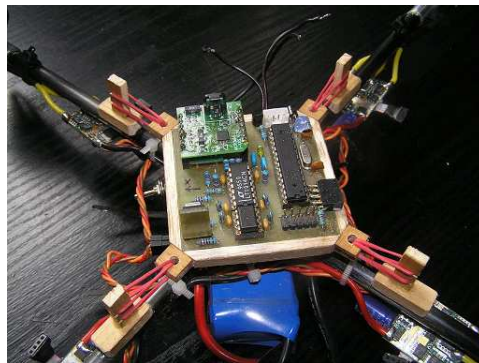


Figure 2.5: Sensor board mounted on the body with elastic bands.

Another problem to solve is that the high current flow to the motors (orientationally, about 12A total is needed to maintain device in the air) causes magnetic fields that interferes the magnetometer. Currently, the device is equipped with twisted wire pairs for high current supply that cancels the parasitic fields, nevertheless some residual error in magnetic readings are still present.

2.2.2 Actuators

From the variety of propeller actuators, suitable type for this project are classical brushed DC motor with mechanic commutator or brushless DC motor. Limitations of brushed DC motors overcome by BLDC motors include lower efficiency and susceptibility of the commutator assembly to mechanical wear and consequent need for servicing, at the cost of potentially less rugged and more complex and expensive control electronics. Brushless DC motors (BLDC motors, BL motors) also known as electronically commutated motors (ECMs, EC motors) are synchronous electric motors powered by direct-current (DC) electricity and having electronic commutation systems, rather than mechanical commutators and brushes. The current-to-torque and voltage-to-speed relationships of BLDC motors are linear, like DC motor. Moreover, classical DC motors usually have less torque at lower speeds, leading to the need of gearbox in front of the propeller. Relationship between the voltage and speed can be approximated by a first order dynamic system.

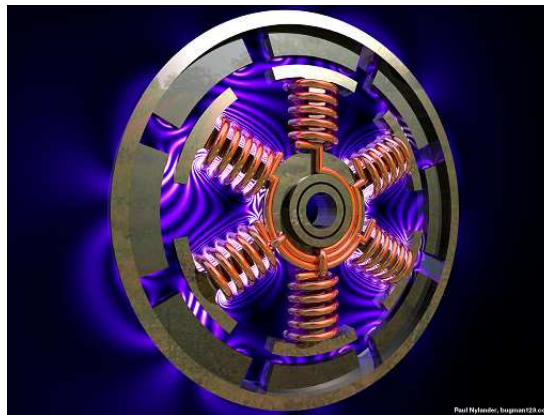


Figure 2.6: Visualisation of magnetic field inside the BLDC motor.

<http://www.bugman123.com/Engineering/index.html>

2.2.2.1 Selection of BLDC motor

To select a suitable motor and propeller combination for the device, one must consider several parameters, from which of the mains are weight, static thrust, efficiency etc. The various motor parameters have been examined. Note that some parameters needs to be cross-compared. For example the efficiency can be outbid by a weight, e.g. more massive

Table 2.1: Key parameters of the selected BLDC motor.

<i>Parameter</i>	<i>Value</i>
Voltage	10.6V (3 Li-Ion cells)
Maximum power	110W
Thrust	750g
Weight	55g
Efficiency	6.76g/W
Relative thrust	13.64g/g
Propeller size	11/4.7

motor with higher efficiency can be replaced by a lighter one with lower efficiency. After extensive seek through, the 3D 550 E [45] from a Czech manufacturer PJS was selected.



Figure 2.7: 3D 550 E from PJS.

<http://www.pjs.cz>

2.2.2.2 Design of BLDC controller

Although BLDC motors are generally synchronous machines, the phase alternating is usually not forced, but a need for electronic commutator appears to achieve the best results and to ensure optimal control. Because the controller must directly drive the phase alternation, the controller needs some means of determining the rotor's orientation/position (relative to the stator coils). Some designs use Hall effect sensors or a rotary encoder to directly measure the rotor position. Others measure the back EMF in the undriven coils to infer the rotor position, eliminating the need for separate Hall effect sensors, and therefore are often called sensorless controllers. They enable the even simpler motor construction, without the need of expensive hall sensors. There are commercially available sensorless BLDC controllers, especially for aircraft RC models. However, I have decided to design my own system because of the interfacing and especially, the possibility for revolutions per second measurement to be used as state variable in the rotation feedback controller. Designing of such realization electronic circuit requires considering cooperation of some delicate components, such as high-current switching transistors, a microcontroller and

simultaneous low-voltage back EMF signal sampling, making the PCB design a challenging task. The main components of the circuit are properly controlled six high powered MOSFET drivers and suitable microcontroller, with integrated peripherals allowing the PWM control and A/D conversion of the back EMF signals. Due to extensive experiences and deep knowledge of the architecture, I have chosen to use the ATmega48 from the AVR family of Atmel corp as a suitable microcontroller [37].

The another reason for the choice of designing the specific BLDC drive, was the fact that most of the conventional BLDC controllers have a different transient responses when accelerating and decelerating. This is an intrinsic property of the PWM driving strategy. Most of the commercially available BLDC controllers avoid this problem by artificially slowing-down the stepup response, by a ramp applied to the PWM signal. Such approach, used in most existing quadrotor designs, results in slowing down the entire actuator dynamics. Such system can therefore be stabilised by an ordinary PID controller, where the information about the motor RPM is not necessary. In my design, an advanced switching strategy is implemented, aiming to equalize the stepup and stepdown response without the need of slowing down the transient response.

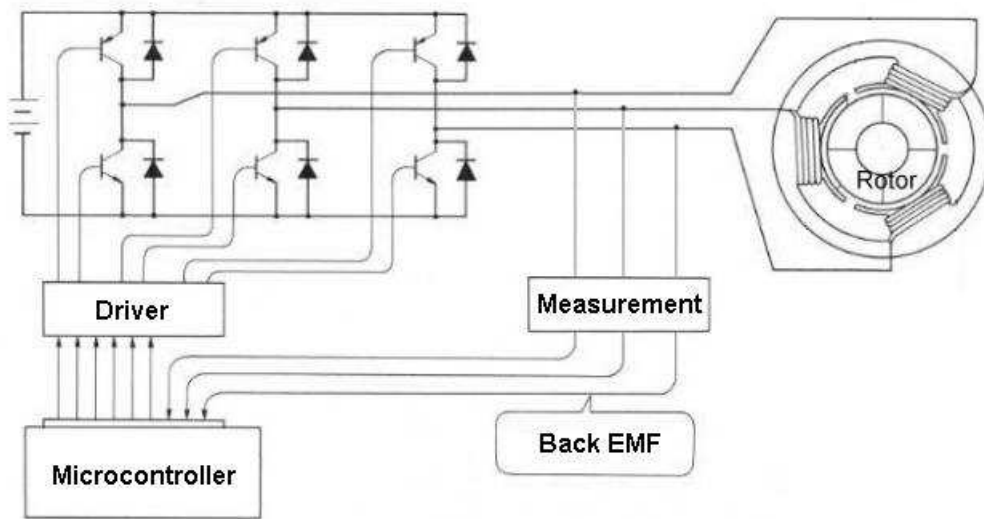


Figure 2.8: Block diagram of the general BLDC controller.

An extensive work has yet been done in the theory of sensorless BLDC control [44], as well as many circuit realisations have been designed [36]. The design is almost entirely an implementation task and the details are out of focus of this work. Only some key resulting parameters of the designed unit will be presented.

Table 2.2: Key parameters of the BLDC controller.

<i>Parameter</i>	<i>Value</i>
Voltage range	tested 7.4 - 14.4 V (2-4 lipol cells)
Maximum current	six 40A N-channel mosfets, tested up to 16A
Interface	Bi-directional, addressed I2C and/or UART.
Operation	The input is target PWM duty cycle and output is actual RPM of the motor. Availability of voltage, current measurement, self-test result, fault detection. Safety motor shut off at communication lost.



Figure 2.9: Illustrational photo of the controller PCB.

<http://www.mikrokoetter.de/>

2.2.3 Power source

The interesting possibility for power source of final version of the device, discussed with the contractor, was a fuel cell or a micro gas turbine. However, such devices is often much more expensive than a electrochemical battery and suitable for larger scale quadrotors. Thus, for development, from the variety of commercially accessible mobile power sources, I have chosen the lithium-ion polymer rechargeable battery.

Lithium-ion batteries are common in consumer electronics and in recent days they also recorded huge entrance as the RC model hobby power sources. They are one of the most popular types of rechargeable battery for portable electronics, with one of the best energy densities, no memory effect, and a slow loss of charge when not in use. Beyond consumer electronics, LIBs are also growing in popularity for military, electric vehicle, and aerospace applications. Research is yielding a stream of improvements to traditional LIB technology, focusing on energy density, durability, cost, and intrinsic safety. The battery was chosen to fulfill the motor supply requirements. It consists of three serially connected cells. The nominal voltage is 10.6V, capacity 2200mAh and weight 180g.



Figure 2.10: Photo of the main battery.

<http://www.rc-modelar.cz/>

2.2.4 Sensor mainboard

Currently used main logic board was originally developed as an autopilot platform for classical micro helicopter. It is based on the ATmega168 microcontroller [37]. The on-board sensors include triaxial accelerometer ADXL335 from Analog Devices [47], triaxial magnetometer module MicroMag 3D from PNI [41] and triple gyro¹ ENC-03 from Murata [46]. The magnetometer is connected to the MCU through SPI interface. The analog sensors (accelerometer and gyros) are connected to the multiplexed 10bit A/D converter of ATmega168 through second-order lowpass filter realized using operating amplifiers. MCU samples each of the six analog signals at about 4kHz and performs a numerical filtering (exponential filter for the accelerometer readings to obtain most recent value and period-averaging for the gyros, suitable for the integration). Simultaneously, the custom-made BLDC controllers, connected through the I2C interface provide information about each motor RPM. The overall sampling frequency is $f_s = 100Hz$. For experimenting, the logic board can also be equipped with a translational sensors, such as sonar rangefinder, optical proximity detector [40] and optical mouse flow sensor [39]. All the readings are collected by onboard microcontroller and sent through RF data link towards a PC, which is used for real-time control. The computer then performs the control scheme according to fig. 3.21 and finally sends the commands back to the aircraft, which are then passed on to four onboard BLDC controllers.

¹Angular rate sensor.

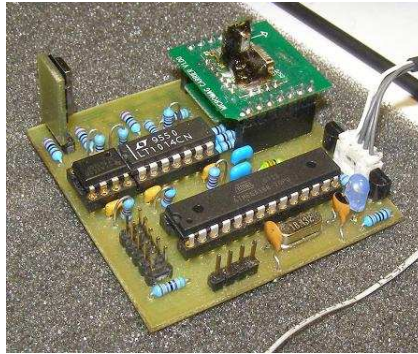


Figure 2.11: Assembly of the sensor board.

2.2.5 Wireless data link

The conception of placing the controller outside the quadrotor body demands a fast and low latency data link between the computer and the aircraft that will enable transfer of the sensor measurements to the PC and motor commands back within one chosen discrete controller time step. It took a lot of time to find such a data link on the market. It has been shown that Bluetooth or Zigbee standards are unsuitable for real-time control, because of their high, unpredictable latency. Nordic NRF24L01 [42] provides a low-level SPI interface for data transfer in 2.4GHz band with data rates up to 2MBPS. However, since there is no encapsulating protocol, an extensive support must be provided from the MCU side, such as channel selection, data loss handling etc. The simplified data throughput timing diagram of the entire system is shown in fig. 2.12.

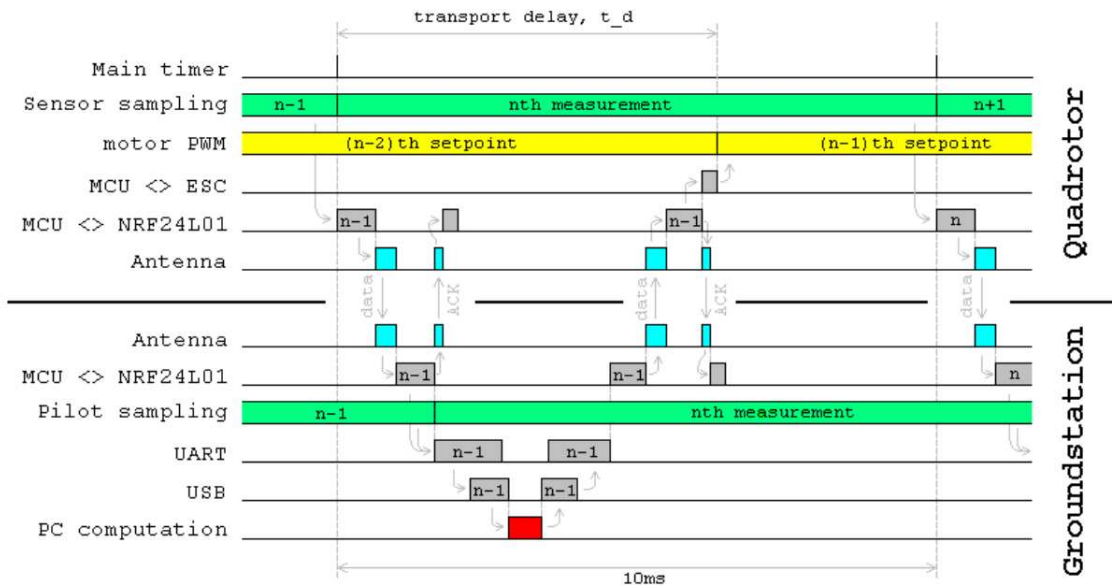


Figure 2.12: Simplified timing diagram of the data flow.

Note that the transport delay, t_d , may vary due to the jitter in USB-to-RS232 converter and PC computation, according to its load, for example. In order to operate correctly, t_d must be always lower than the sampling period.

2.2.6 Groundstation circuit

The groundstation circuit is an interface between the wireless link to the quadrotor, human pilot and the PC performing the dynamical controller. The heart of this circuit is the ATmega644 microcontroller [38]. The main issue here was a problem of finding the appropriate interface at the PC side. At the beginning, a COM port was used due to its intrinsic low latency. Nevertheless, during the development it was found out that the maximum speed of the UART, 115200 baud/sec is not sufficient. Thus, seeking for the higher throughput interface but preserving the low latency, I found out that some of the USB-to-RS232 converters can meet these requirements. Current unit is USB powered and uses the high-baudrate enabled Prolific PL2303 USB-to-RS232 converter.

For the next version, nRF24LU1+ will be used, which features a fully integrated USB 2.0 compliant device controller. Thus, if the throughput and latency requirements will be met, it will eliminate the need for extra microcontroller. RC Handpad can be then connected to the PC with commercialy available interface.

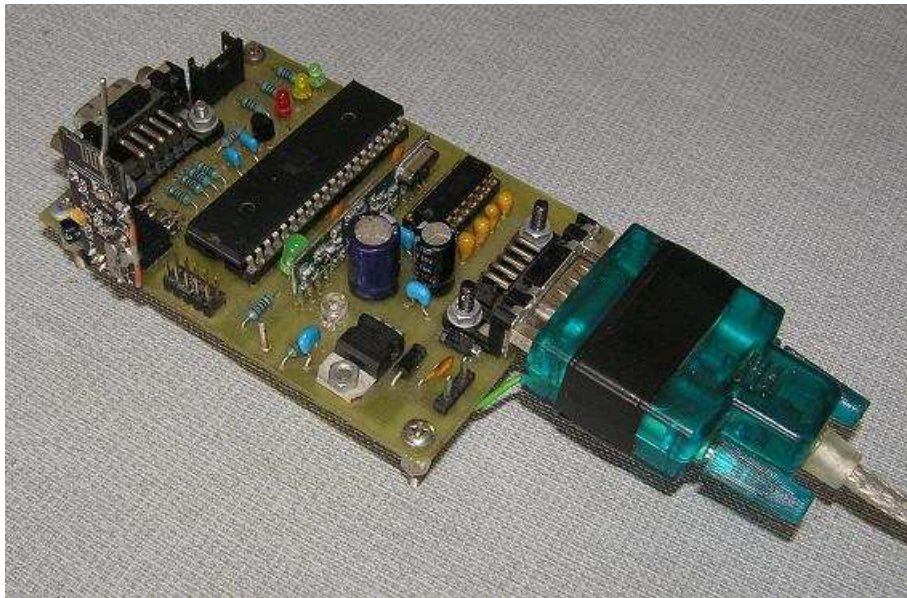


Figure 2.13: Assembly of the groundstation logic board.

2.3 Software

2.3.1 Microcontroller firmware

There are totally 6 microcontrollers involved. All the firmware was designed and implemented especially for this project. The high precision timing demands leads to the need for interrupt driven peripheral service and an extensive workaround for correct program synchronization. The firmware was programmed in C using AVR GCC compiler, with some in-line assembler optimizations and functions.

2.3.2 PC control software

To enable the real-time control, all the algorithms were implemented in C using the GNU GCC compiler. Multi-threaded approach is used to reach the desired performance. The threads use different priorities and various synchronization techniques. The current build runs on MS windows, uses OpenGL and SDL graphical libraries for visualisation and a console windows for user input, constant adjustments etc.

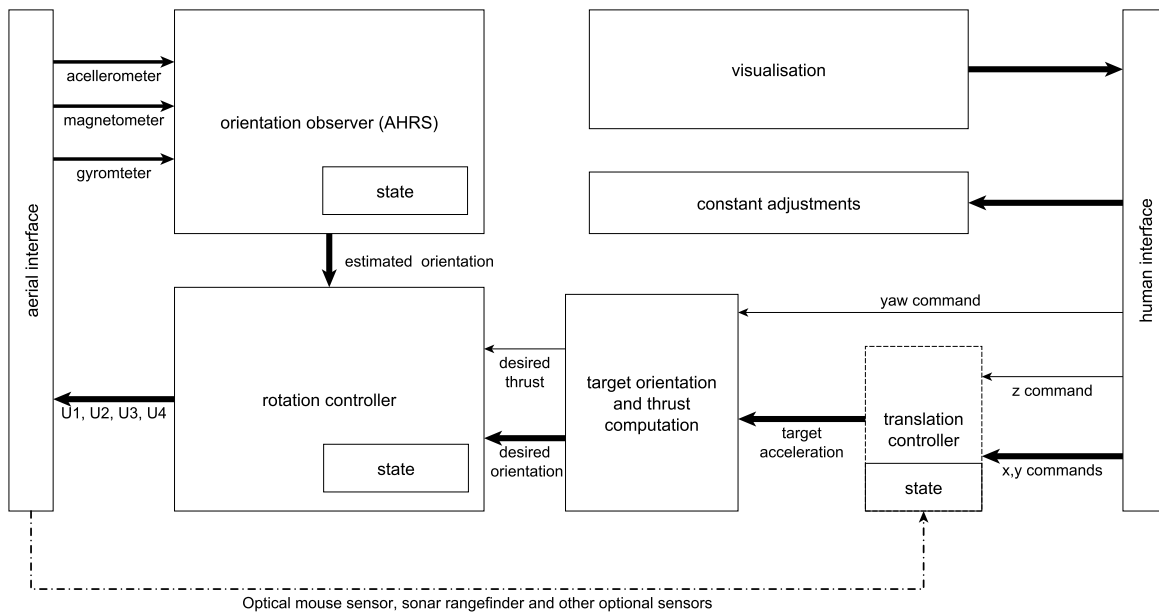


Figure 2.14: Control software block diagram.

Modular design of the software respects the proposed control structure. Apart from the visualisation, input and communication modules, the main components are the orientation observer, translation damper and rotation controller. These modules are approximating continuous designs by rectangle discretization and running on the same sampling frequency, directed by the quadrotor mainboard. The translation controller is reduced to a translation damper. The pilot is placed in the role of position controller in the way that he directs target spatial acceleration.

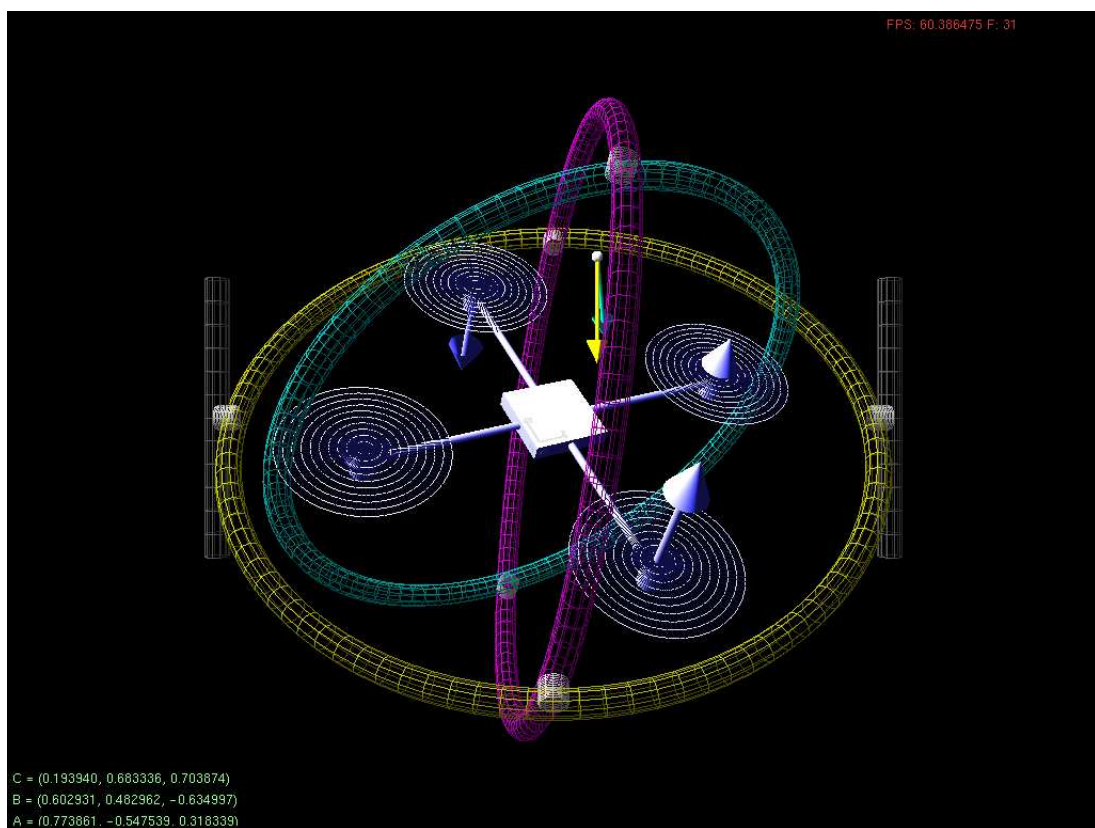


Figure 2.15: Control software visualisation using OpenGL.

Chapter 3

Analysis, modelling and controller design

In this chapter, I will introduce a theoretical view on the quad-rotor dynamics. At first, non-linear model will be developed and a suitable controller structure proposed. Each block in the structure will be then examined and its design strategy will be derived.

3.1 Dynamic Model

A simplified schematics of the quadrotor's body-fixed and inertial frames of reference is shown in fig. 3.1. Both of them are right-handed coordinate systems where the right-hand rule applies for determining the direction of a vector cross-product. For the first, X, Y and Z are its orthogonal axes with correspondent body linear velocity vector $\mathbf{V} = {}^b [u \ v \ w]^T$ and angular rate vector $\mathbf{\Omega} = {}^b [p \ q \ r]^T$. The second one is the NED¹ inertial (navigation) reference frame, with which initially the body-fixed coincides.

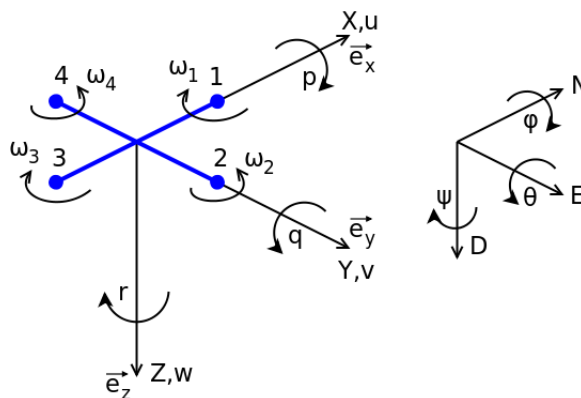


Figure 3.1: Quadrotor's fixed-body reference frame.

¹NED stands for North-East-Down coordinate system.

3.1.1 Rigid-Body Dynamics

In an inertial frame of reference it is known that the torque (moment) is defined as the time derivative of the angular momentum, $\mathbf{M}_n \triangleq \frac{d\mathbf{L}}{dt} = \frac{d}{dt} (\mathbf{I}_n \cdot \boldsymbol{\Omega})$, where \mathbf{I}_n is the body's inertia tensor measured in NED inertial reference frame, a 3x3 matrix. However, to simplify the calculations, \mathbf{M} is rather considered in the body-fixed rotating frame, using principal axes with the origin at the centre of gravity. In this frame, the moment of inertia tensor is constant² (and diagonal), $\mathbf{I} = \text{diag}(I_x, I_y, I_z)$.

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (3.1)$$

The angular momentum vector L can now be written as $\mathbf{L} = [I_x p \ I_y q \ I_z r]^T$. In a rotating reference frame, the time derivative must be replaced with *time derivative in rotating reference frame*, it yields

$$\mathbf{M} = \left(\frac{d\mathbf{L}}{dt} \right)_{\text{rot}} + \boldsymbol{\Omega} \times \mathbf{L} \iff \mathbf{M} = \mathbf{I} \cdot \dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times (\mathbf{I} \cdot \boldsymbol{\Omega}) \quad (3.2)$$

which is the particular vector form of Euler's equations. By developing the cross-product term its algebraic form is found as the set of equations

$$\begin{aligned} M_x &= I_x \dot{p} + (I_z - I_y) q r \\ M_y &= I_y \dot{q} + (I_x - I_z) r p \\ M_z &= I_z \dot{r} + (I_y - I_x) p q \end{aligned} \quad (3.3)$$

which are also referred to as the *Euler moment equations*. One can note the physical natural sense in these equations: the simultaneous rotation around two axis will generate a torque around a third axis, given that the previous causal two axis don't have the same inertia.

Similarly to the reasoning applied until here to the rotational aspect of the rigid-body dynamics, in the translational case a force is generated, according to Newton's 2nd law, as $\mathbf{F} \triangleq \frac{d\mathbf{P}}{dt} = \frac{d}{dt} (m \cdot \mathbf{V})$, where m is the total mass of the quadrotor in whose center the origin of the aircraft's fixed-body rotating frame is located. Once again turning to the body-fixed rotational frame and defining translational momentum vector $\mathbf{P} = [m_t u \ m_t v \ m_t w]^T$, the calculation is simplified to

$$\mathbf{F} = \left(\frac{d\mathbf{P}}{dt} \right)_{\text{rot}} + \boldsymbol{\Omega} \times \mathbf{P} \iff \mathbf{F} = m \left(\dot{\mathbf{V}} + \boldsymbol{\Omega} \times \mathbf{V} \right) \quad (3.4)$$

By solving the cross-product, the set of *force equations* is obtained

$$\begin{aligned} F_x &= m_t (\dot{u} + q w - v r) \\ F_y &= m_t (\dot{v} + r u - w p) \\ F_z &= m_t (\dot{w} + p v - q u) \end{aligned} \quad (3.5)$$

²Assumption is valid because flexible modes are disregarded and aircraft's mass is constant.

3.1.2 Actuator Dynamics

In this section, a simplified model of a BLDC (brushless DC) motor with attached propeller will be introduced. Please note that this particular model was designed *especially* for the quadrotor control purposes and may not consider all the physical influences. Input to the actuator is unbiased applied voltage u_j and output vector is formed of propeller angular velocity ω_j , axial thrust T_j and drag torque δ_j . By term *unbiased* I mean that the voltage offset constant has been subtracted from (and needs to be added to) the actual motor voltage input, such that the DC gain from voltage u_j to angular velocity ω_j crosses the origin. The voltage offset is caused by non-linearities of the bearing friction and BLDC driving strategy and will not be referred to anymore. Note that the actuator input, u_j , is expressed in volts rather than natural percentage of PWM duty cycle. This choice was made to make the system independent on the battery charge level. The u_j needs to be scaled according to the actual battery voltage and therefore passed to the BLDC controller as PWM duty cycle percentage.

3.1.2.1 Non-linear Actuator Model

To simplify the model, the stress-free motor will be treated as an ordinary, linear first-order system. The fast dynamics caused by magnetic inductance will be neglected. Nevertheless, experiments have shown that a conventional cheap BLDC motor with controller has different dynamics when accelerating and decelerating due to its PWM driving strategy. A special motor controller was developed in order to minimize this effect, however a certain difference between the two poles still remained, described by $f_1(u_j - K\omega_j)$ in the model.

The propeller adds non-linear damping caused by the air friction with the blades, resulting in a drag torque $\delta_j = f_4(\omega_j)$, which [1] demonstrates to be polynomially dependent on ω . This damping makes non-linear not only the steady-state gain, but also the entire dynamics, which is often mistreated. From fig. 3.2, following the Newton's 2nd law $\dot{\omega}_j = (q_M - q_B - \delta_j)/I_r$. Since torque can be transferred from propeller to body through the motor only (air is an independent liquid medium with its own inertia), the output torque equals $q_j = q_M - q_B$ which is again equal to δ_j in steady state.

The model dynamics is described as

$$I_r \dot{\omega}_j = f_1(u_j - K\omega_j) - B\omega_j - f_4(\omega_j) \quad (3.6)$$

with output relations

$$\begin{aligned} q_j &= f_1(u_j - K\omega_j) - B\omega_j = I_r \dot{\omega}_j + \delta_j \\ T_j &= f_3(\omega_j) \end{aligned} \quad (3.7)$$

3.1.2.2 Linearized Actuator Model

To linearize the dynamics, in (3.6) the non-linear sensitivity function caused by the asymmetric up and down step response can be averaged, $f_1(u_j - K\omega_j) = \frac{G_1 + G_2}{2} (u_j - K\omega_j)$. Polynomial relations $f_3(\omega_j)$ and $f_4(\omega_j)$ can be linearized around the operating point ω_0 ,

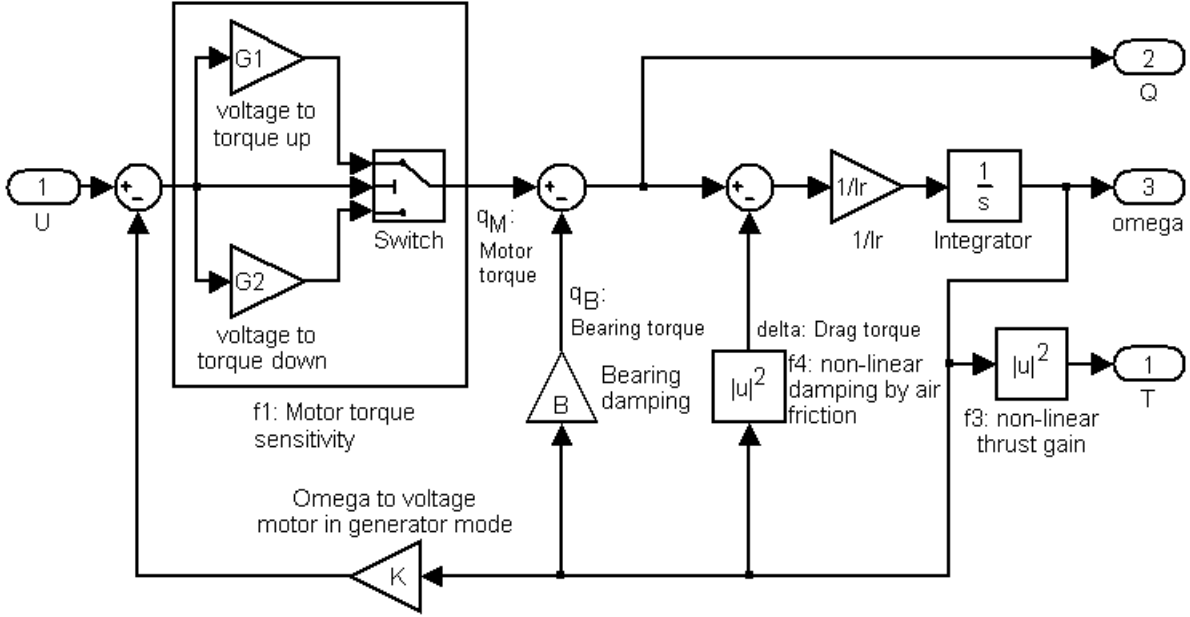


Figure 3.2: Actuator model.

where $T = m_t g = 4f_3(\omega_0)$, the sum of all four propeller thrusts equals the gravity force acting on the body, $T_j = f_3(\omega_j) \equiv n\Delta\omega_j + f_3(\omega_0)$ and $\delta_j = f_4(\omega_j) \equiv m\Delta\omega_j + f_4(\omega_0)$, leading to a linearized dynamics describing the actuator as an ordinary first-order system

$$I_r \dot{\omega}_j = k \left(u_j - \frac{l\omega_0 + f_4(\omega_0)}{k} \right) - (m + l)\Delta\omega_j \quad (3.8)$$

with linearized output relations

$$\begin{aligned} q_j &= I_r \dot{\omega}_j + m\Delta\omega_j + f_4(\omega_0) \\ T_j &= n\Delta\omega_j + f_3(\omega_0) \end{aligned} \quad (3.9)$$

where $k = \frac{G_1 + G_2}{2}$, $l = K \frac{G_1 + G_2}{2} + B$, $m = \dot{f}_4(\omega_0)$ and $n = \dot{f}_3(\omega_0)$. Note that in linear approximation around ω_0 , steady-state drag torque δ_j is proportional to axial thrust T_j , namely $q_j \doteq \frac{m}{n} T_j$, when $\frac{n}{m} = \frac{f_3(\omega_j)}{f_4(\omega_j)}$ or at least $\frac{n}{m} = \frac{f_3(\omega_0)}{f_4(\omega_0)}$.

3.1.2.3 Input and output non-linear mapping

For further extension of the operational range of the linear controller, we will define here a steady-state gains of the non-linear model.

Let's assume that inner polynomial relations, $f_4(\omega_j)$ and $f_3(\omega_j)$, can be approximated by $f_4(\omega_j) \doteq H\omega_j^2$ and $f_3(\omega_j) \doteq F\omega_j^2$. We can then define steady-state gain from u_j to the

ω_j , from the non-linear model (3.6), using the new combined constants, as

$$\omega_j(u_j)_\infty = \frac{\sqrt{l^2 + 4ku_jH} - l}{2H} \quad (3.10)$$

and more, when inserting (3.10) into the output non-linear function $f_3(\omega_j)$, we can obtain the total DC gain from voltage u_j to thrust t_j

$$t_j(u_j)_\infty = f_3(\omega_j(u_j)_\infty) = F \frac{2l^2 + 4ku_jH - 2l\sqrt{l^2 + 4ku_jH}}{4H^2} \quad (3.11)$$

Similarly, for the DC gain from voltage u_j to drag torque δ_j

$$\delta_j(u_j)_\infty = f_4(\omega_j(u_j)_\infty) = H \frac{2l^2 + 4ku_jH - 2l\sqrt{l^2 + 4ku_jH}}{4H^2} \quad (3.12)$$

It can be shown later, for the particular identified actuator constants, that both (3.11) and (3.12) in the entire voltage operating range, can be *well* approximated by a simple parabolic function, such that $t_j(u_j)_\infty \doteq \alpha u_j^2$ and $\delta_j(u_j)_\infty \doteq \beta u_j^2$. This result suggest to create an inverse non-linearity (i.e. square root), which will be then inserted in front of the non-linear actuator.

3.1.3 Acting Forces and Moments

Hereby all those forces and moments which act on the model derived in the previous section shall be addressed.

3.1.3.1 Axial Thrust moments and Drag torque moment

It is known that, as the blades of the propellers rotate in the air, they "push" the air into a specific direction, in this case downwards, thus producing the thrust/lift force, however not without being affected by the reaction of the air flow onto them, what we call the *drag torque*. Once the propellers' axis of rotation is assumed perfectly aligned with the Z -axis of the aircraft body-fixed frame, the drag torque does not affect the other axis. In fact, this is the main mechanism to cause the quadrotor to yaw. Also, the difference of the thrust force pushing on the same axis generates moment only around that axis, which enables the tilt control. On the contrary to gyroscopic moments, these forces are always present when angular speed of some of the rotors is nonzero. The forces are outputs of Actuator dynamics, derived in the previous section.

The difference in thrust produced by the propellers in the same axis define a moment around that axis. Also, for the Z -axis only, the drag thrust reaction δ_j caused by the air friction with the blades applies, hence

$$\begin{aligned} M_x^T &= l_a (T_4 - T_2) \\ M_y^T &= l_a (T_1 - T_3) \\ M_z^T &= \delta_1 - \delta_2 + \delta_3 - \delta_4 \end{aligned} \quad (3.13)$$

where l_a is the lever length of each of the quadrotor's arms, assumed to be the same for all of them.

As shown in the previous section, the linearized drag torque δ_j can be treated as linearly dependent on T_j , when $nf_4(\omega_0) = mf_3(\omega_0)$, in such a way that all thrusts can be mapped into moments through a matrix

$$\begin{bmatrix} T \\ M_x^T \\ M_y^T \\ M_z^T \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -l_a & 0 & l_a \\ -l_a & 0 & l_a & 0 \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \quad (3.14)$$

where $c = \frac{m}{n}$. The matrix is invertible, thus the axial moments are uniquely determining the applied thrusts by the propellers. This is an important relation, since it reflects the fact that number of inputs is equal to number of controllable degrees of freedom. Quad-rotor has 4 inputs, representing the motor voltages, and simultaneously only 4 of the total 6 degrees of freedom can be independently controlled. I've chosen $\mathbf{Y}_{control} = [x \ y \ z \ \psi]$ as a final control goal.

3.1.3.2 Gyroscopic Moments from Rotors

The four rotors induce gyroscopic moments on the aircraft due to their angular velocity, ω_j , being an additional mechanism which causes the quadrotor to yaw. Moreover, their gyroscopic effect affects also, although relatively less intense, the other two axes. Considering the spin direction of rotor $j = 1$, the gyroscopic torque resulting from the interaction of the rotor with the rotating aircraft and acted on the generic rotor $j = 1 \dots 4$ is given, similarly to (3.2), by

$$\mathbf{M}^j = \frac{d\mathbf{L}^j}{dt} + \boldsymbol{\Omega} \times \mathbf{L}^j = \mathbf{I}^j \cdot \dot{\boldsymbol{\omega}}^j + \boldsymbol{\Omega} \times (\mathbf{I}^j \cdot \boldsymbol{\omega}^j) \quad (3.15)$$

where \mathbf{I}^j is the gyroscopic inertia, namely that of the rotating part of the rotor. By solving the cross-product it comes

$$\mathbf{M}^j = \begin{bmatrix} I_x^j \dot{\omega}_x^j \\ I_y^j \dot{\omega}_y^j \\ I_z^j \dot{\omega}_z^j \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_x^j \omega_x^j \\ I_y^j \omega_y^j \\ I_z^j \omega_z^j \end{bmatrix} = \begin{bmatrix} I_x^j \dot{\omega}_x^j + I_z^j \omega_z^j q - I_y^j \omega_y^j r \\ I_y^j \dot{\omega}_y^j + I_x^j \omega_x^j r - I_z^j \omega_z^j p \\ I_z^j \dot{\omega}_z^j + I_y^j \omega_y^j p - I_x^j \omega_x^j q \end{bmatrix} \quad (3.16)$$

However, the direction of ω_j coincides with the Z -axis of the aircraft whereas all its other components are zero, therefore equation 3.16 is simplified and the set of 3 algebraic equations expressing the gyroscopic torques acted on a rotor $j = 1 \dots 4$ is

$$\begin{aligned} M_x^j &= I_j \omega_j q \\ M_y^j &= -I_j \omega_j p \\ M_z^j &= I_j \dot{\omega}_j \end{aligned} \quad (3.17)$$

Now for taking into consideration all rotors, all angular speeds need to be summed up according to their respective sign (direction) and assuming that the gyroscopic mass of each rotor is the same, I_r , we define

$$L_R = I_r \sum_{j=1}^4 \omega_j = \omega_1 - \omega_2 + \omega_3 - \omega_4 \quad (3.18)$$

Considering the reaction torques on the aircraft's body, hence inverting the sign of the set (3.17), the total rotor gyroscopic moment exerted on the aircraft's body is

$$\begin{aligned} M_x^R &= -I_r \omega_R q = -L_R q \\ M_y^R &= I_r \omega_R p = L_R p \\ M_z^R &= -I_r \dot{\omega}_R = -\dot{L}_R \end{aligned} \quad (3.19)$$

3.1.3.3 Thrust/lift force

The thrust force is generated by the rotation of the propellers through the viscous air and is used to maintain the aircraft in the air. This force is always aligned with the body-fixed Z -axis, thus the components on the other axis are zero. Considering again each rotor $j = 1 \dots 4$, according to [?] the thrust can be modelled, as acted on the aircraft, as

$$T_z = - \sum_{j=1}^4 T_j = -T \quad (3.20)$$

The effects of the thrust generated by the rotors through their attached propellers is directly calculated in the body-fixed frame.

$${}^b\vec{F}_T = [0 \quad 0 \quad -T]^T \quad (3.21)$$

3.1.3.4 Earth's Gravity force

The Earth's gravitational field around the quadrotor causes its weight force to act upon it on its mass center. This is modeled in the inertial (NED) frame again simply by Newton's 2nd law as

$${}^n\vec{F}_g = [0 \quad 0 \quad mg]^T \quad (3.22)$$

where $g = 9,81 m/s^2$ is the absolute value of Earth's gravity acceleration. However this acting force needs to be considered in the body-fixed frame, therefore the need for a rotation matrix appears. This is the first time when I need to consider a rotation representation in our model. Let's write coordinate transform in matrix-form for this moment

$${}^b\vec{F}_g = {}^bR {}^n\vec{F}_g \quad (3.23)$$

3.1.3.5 Aerodynamic Forces and Moments

By term *Aerodynamic Forces and Moments* I mean the forces caused by air friction of the quad-rotor body due to its translational/rotational movements. The separate actuator model was designed previously. However, it works in a static air environment, meaning that the overall model is valid only when assuming that the air flow through the propellers is much faster than translational speed of the quadrotor body. Adding the body movements into actuator dynamics will result in much more complicated model and thus is disregarded.

It was shown in further controller (and orientation observer) design that the even though the aerodynamic forces exerted upon the quadrotor are very small due low translational speeds, they cannot be completely disregarded. For simplification, we will consider the air friction as a simple, first order, linear damping caused by negative feedback from translational speed in body frame, u, v, w to acting force

$${}^bF_{\mathbf{a}} = -\text{diag}(\mathbf{t}_{\text{ad}})\mathbf{V} \quad (3.24)$$

and similarly, from p, q, r to acting moment

$${}^bM_{\mathbf{a}} = -\text{diag}(\mathbf{r}_{\text{ad}})\mathbf{\Omega} \quad (3.25)$$

3.1.4 Complete Non-Linear Model

Having already assessed the forces and moments acting on the quadrotor, its non-linear model is obtained by applying (3.19) and (3.13) into the left side of (3.3). The model assumes $I_x \doteq I_y$, so that the $p q$ product of M_z in (3.3) is not considered. From (3.7), the gyroscopic torque M_z^R is already contained in the actuator output q_j . After rearranging the terms and isolating the angular accelerations, the *moment equations* are obtained

$$\begin{cases} \dot{p} = \frac{l_a}{I_x} (T_4 - T_2) + \frac{g_x(\omega, \mathbf{\Omega})}{I_x} - \frac{\mathbf{r}_{\text{ad}p}}{I_x} p \\ \dot{q} = \frac{l_a}{I_y} (T_1 - T_3) + \frac{g_y(\omega, \mathbf{\Omega})}{I_y} - \frac{\mathbf{r}_{\text{ad}q}}{I_y} q \\ \dot{r} = \frac{1}{I_z} (q_1 - q_2 + q_3 - q_4) - \frac{\mathbf{r}_{\text{ad}r}}{I_z} r \end{cases} \quad (3.26)$$

where $g_x(\omega, \mathbf{\Omega})$ and $g_y(\omega, \mathbf{\Omega})$ are gyroscopic torques defined as

$$\begin{aligned} g_x(\omega, \mathbf{\Omega}) &= (I_y - I_z) q r - L_R q \\ g_y(\omega, \mathbf{\Omega}) &= -(I_x - I_z) r p + L_R p \end{aligned} \quad (3.27)$$

whereas by inserting the set of equations (??) into the left side of (3.5) and isolating the

translational accelerations, the *force equations* arise

$$\begin{cases} \dot{u} = -q w + v r + \frac{{}^bF_{g_x}}{m_t} - \frac{\mathbf{t}_{ad_u}}{m_t} u \\ \dot{v} = -r u + w p + \frac{{}^bF_{g_y}}{m_t} - \frac{\mathbf{t}_{ad_v}}{m_t} v \\ \dot{w} = -p v + q u - \frac{T}{m_T} + \frac{{}^bF_{g_z}}{m_t} - \frac{\mathbf{t}_{ad_w}}{m_t} w \end{cases} \quad (3.28)$$

Similar quadrotor model can be found in [2].

3.1.4.1 Computer numerical simulation of derived model

The overall model has been implemented in Simulink as a single block with four inputs u_1, u_2, u_3, u_4 and several outputs from which of the mains are position, orientation and motor angular speed vector ω . Up to now, we tried to avoid the need for particular orientation representation in the above equations. This is due to a non-straightforward and non-intuitive relation of angular rates to orientation describing variables, which, in general, could not be obtained by simple integration of angular rates, on the contrary to linear rates. This non-linear relation makes the optimal problem in some meaning of a rigid-body reorientation very challenging and has been recalled several times, especially in aerospace field.

For our approach, I created first a *rigid-body dynamics* block that takes the acting forces and moments in body frame, uses the body mass m_t and body moment of inertia \mathbf{I} as parameter and performs all the necessary calculations, including body-related gyroscopic effects. This makes the resulting diagram more straightforward and first member of (3.27) and first two members of (3.28) doesn't have to be considered, as they're part of rigid-body dynamics, computed by the block itself. The block has its internal state of position, rates and orientation, which are aswell its outputs. The block uses quaternion as orientation representation. Reasoning for this choice will be explained later. Overall model diagram is presented in the appendix.

3.2 Controller design

3.2.1 Control Goals

There are in total 10 outputs³, $\mathbf{Y} = [\phi \ \theta \ \psi \ x \ y \ z \ \omega_1 \ \omega_2 \ \omega_3 \ \omega_4]$, that can be naturally measured or estimated from physical sensor readings, and 4 inputs, $\mathbf{U} = [u_1 \ u_2 \ u_3 \ u_4]$, representing the voltage (PWM duty cycle) level for each of the actuators.

The goal is to reach a target spatial orientation and/or position. As the model has less inputs than outputs, not all of the latter can be controlled independently, i.e. the

³Disconsidering their time derivatives and integrals.

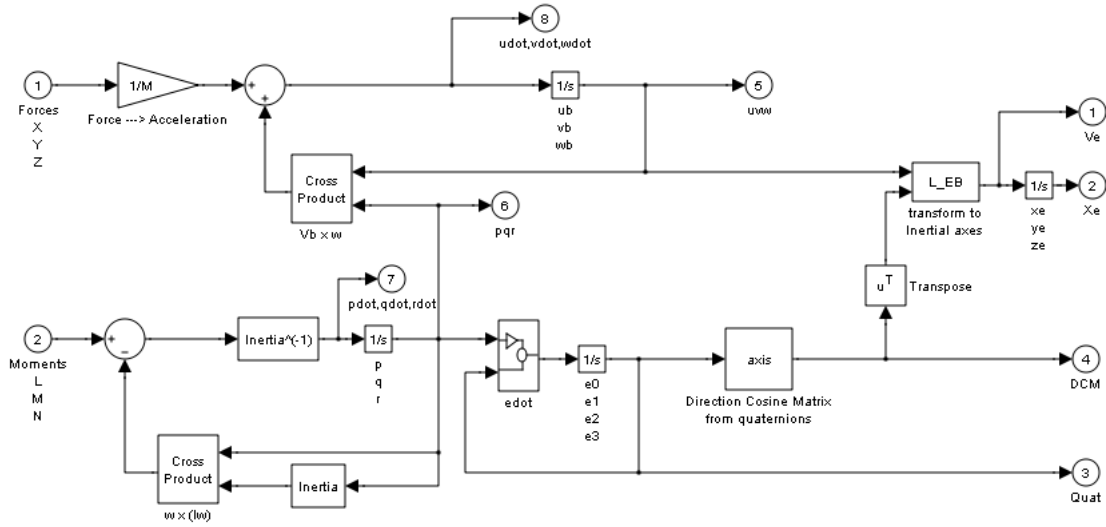


Figure 3.3: Separate rigid body dynamics

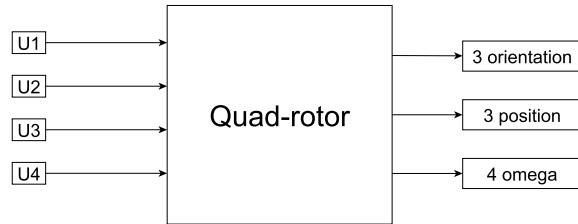


Figure 3.4: basic model inputs/outputs

device is not holonomic. In gravity-free environment, the non-holonomic idea can be brought to all parts: the device can only move, or generate thrust T along Z -axis (2-wheel robot analogy). In a NED frame, the generated thrust vector ${}^n\mathbf{T}$ can be separated into horizontal and vertical components as ${}^n\mathbf{T} = [{}^n\mathbf{T}_h \quad {}^nT_v]$. The horizontal one ${}^n\mathbf{T}_h$ is a vector that accelerates the aircraft in horizontal (North-East) plane whereas the vertical, nT_v , is a scalar that must be equal to Earth’s gravity magnitude in order to maintain the altitude. The simplified 2D-diagram is shown in fig.3.15. However, the direction of ${}^n\mathbf{T}_h$ depends directly on 2 degrees of freedom describing the attitude, namely the *Euler angles* ϕ and θ , meaning that these outputs can not be controlled independently. Since the position has bigger priority than the orientation itself, the target control vector is reduced to $\mathbf{Y}_{control} = [x \quad y \quad z \quad \psi]$.

Respecting the relationship between the rotation and translation subsystem, the control structure proposed for the overall system is shown in fig. 3.21.

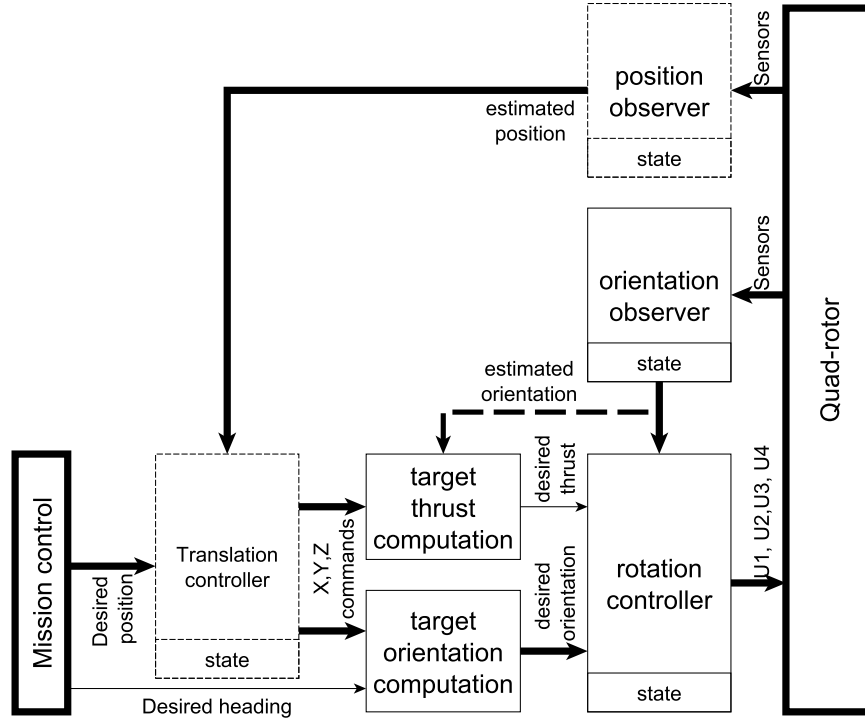


Figure 3.5: Proposed control structure.

3.2.2 Orientation Representation

There are several methods for describing spatial orientation [15]. The most common one is by the *Euler angles* ϕ , θ and ψ , directly used, in aeronautics, as proportional error inputs to the controllers. However, ϕ , θ and ψ represent consecutive rotations applied to the rigid body, thus the controllers should correctly act in the same order, stabilising the axes respectively. Nevertheless, in an orientational state far from the operating point, during simultaneous stabilization of all the axes based on *Euler angles* error, significant errors arise due to non-linearities and even singularities as the angle change does not correspond to angular velocity.

Each spatial rotation can be expressed using the eigenaxis vector and rotation angle [5] [16]. Rotation quaternions are effectively used for such purpose. Nevertheless, they add a certain redundancy (4 scalar numbers to describe 3 degrees of freedom) in opposite to *Euler angles*. Rotation quaternions have unitary norm and their eigenaxis r is a unit vector also. A special non-commutative multiplication operation is defined for the quaternions, denoted with the \otimes operator [5]. Multiplying two or more rotation quaternions produces another rotation quaternion that represents the two consecutive rotations performed on the coordinate system. The noncommutativity of the \otimes operator reflects the fact that a rigid body is in different orientation state after two consecutive rotations, depending on their order. A conjugate quaternion \mathbf{Q}^{-1} represents a backward rotation. Relations between the quaternions, angle α of rotation and eigenaxis vector \mathbf{r}

are defined as

$$\mathbf{Q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) \mathbf{r} \end{bmatrix} \quad (3.29)$$

$$\ln \mathbf{Q} = \frac{\mathbf{r}\alpha}{2} \quad (3.30)$$

3.2.3 Desired Orientation and Thrust

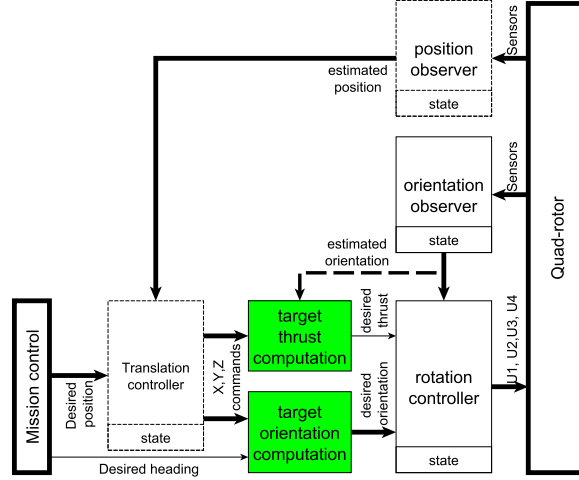


Figure 3.6: Target orientation and thrust block placement in the global structure.

Regarding the control structure proposed in fig. 3.21 and obeying the priority dynamics of the rotation over the translation subsystem, first the target spatial acceleration and yaw command need to be transformed into a target absolute orientation $\hat{\mathbf{Q}}$ and a thrust \hat{T} in the body frame. Note that this block represents a simple mapping function with no dynamics designed so that the body acceleration in target orientation and thrust state follows *target acceleration* when disregarding the translation aerodynamic damping forces. Considering g as the magnitude of Earth's gravity acceleration, for the tilt we first introduce a vector $\hat{\mathbf{a}} = [\hat{a}_x \ \hat{a}_y \ g + \hat{a}_z]^T$ representing target linear acceleration in NED frame, and then compute the angle of tilt orientation α as that between $\hat{\mathbf{a}}$ and the unitary vector pointing up

$$\alpha = \arccos \frac{\hat{\mathbf{a}}}{|\hat{\mathbf{a}}|} [0 \ 0 \ 1] \quad (3.31)$$

The resulting tilt quaternion \mathbf{Q}_t is then formed as a rotation with eigenaxis composed of vector cross product

$$\mathbf{Q}_t = \left[\begin{array}{c} \cos(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) \frac{\hat{\mathbf{a}}}{|\hat{\mathbf{a}}|} \times [0 \ 0 \ 1]^T \end{array} \right] \quad (3.32)$$

The yaw rotation is simply defined around the Down-axis in the NED frame

$$\mathbf{Q}_y = \left[\begin{array}{ccc} \cos(\frac{\hat{\psi}}{2}) & 0 & 0 \\ 0 & 0 & \sin(\frac{\hat{\psi}}{2}) \end{array} \right]^T \quad (3.33)$$

Finally, $\hat{\mathbf{Q}}$ can be composed by multiplying these two quaternions. Note that this step can be executed in two ways. In (3.34a) first the tilt is applied, thus preserving the absolute accelerations in the NED frame, whereas in (3.34b) the heading is applied first, so that the acceleration commands become relative to heading and hence the aircraft behaves like a manned helicopter as perceived by the translation controller.

$$\hat{\mathbf{Q}} = \mathbf{Q}_y \otimes \mathbf{Q}_t \quad (3.34a)$$

$$\hat{\mathbf{Q}}' = \mathbf{Q}_t \otimes \mathbf{Q}_y \quad (3.34b)$$

Using the total quadrotor mass m_t , the desired sum of the propellers thrust magnitude in the body frame is simply defined as

$$\hat{T} = m_t |\hat{\mathbf{a}}| \quad (3.35)$$

Another way to compute \hat{T} in order to maintain the target thrust in the NED frame Z-axis during the entire correction maneuver could be done by dynamically mapping the desired vertical thrust into the body frame

$$\hat{T}' = m_t \frac{g + \hat{a}_z}{\cos(\beta)} \quad (3.36)$$

where β is the angle between ${}^n[0 \ 0 \ -1]^T$ and the unitary vector pointing down in body-frame.

3.2.4 Control of Orientation using Eigenaxis

Although Billimoria and Wie in [9] showed that the eigenaxis correction maneuver is not time-optimal in general, the maneuver still represents the "shortest" path from one orientation state to another, in natural sense. Each reorientation maneuver can be performed along a single axis and specified angle. This axis is called Euler axis, or Eigen axis, as in view of rotation matrices. Billimoria and Wie showed that target orientation state can be reached faster by deflecting the rigid body from natural eigenaxis rotation first, such that other actuators may help accelerating the rotation and then, when reaching the goal state, revert the deflection back, all using the bang-bang control. Nevertheless,

this technique requires constant saturation of all the actuators, which claims 300% the energy, with gain less than 10% of time, in comparison with simple eigenaxis rotation, which makes it unsuitable for quadrotor approach. Eigenaxis rotation also enables much aggressive maneuvers, like looping, as there are no singularities and no implicit need for a linearisation as all the orientation states are equal.

Assuming we have the orientation estimation subsystem onboard the flying device, which outputs the actual absolute orientation state based on the sensor readings and our target orientation state, error correction maneuver can be achieved by driving the body angular rates proportionally to eigenaxis of error correction maneuver.

Proof: Let's define the error quaternion \mathbf{Q}_e such as

$$\mathbf{Q}_e = \mathbf{Q}^{-1} \otimes \hat{\mathbf{Q}} \quad (3.37)$$

where \mathbf{Q} is the current orientation state quaternion and $\hat{\mathbf{Q}}$ is our target state one. \mathbf{Q}_e is left-invariant with respect to \mathbf{Q} and stands for a rotation needed to be performed in order to reach state $\hat{\mathbf{Q}}$ from state \mathbf{Q} , namely $\hat{\mathbf{Q}} = \mathbf{Q} \otimes \mathbf{Q}_e$. Both \mathbf{Q}_e and $-\mathbf{Q}_e$ lead to the target state, since $\hat{\mathbf{Q}}$ represents the same state as $-\hat{\mathbf{Q}}$. Therefore, aiming at minimizing the rotation angle, we select $\mathbf{Q}_e = \min_{\alpha}(\mathbf{Q}_e, -\mathbf{Q}_e)$. Based on the knowledge of rigid-body angular velocities, an orientation quaternion propagation through time is defined as

$$\dot{\mathbf{Q}} = W(\Omega)\mathbf{Q} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & -r & q \\ q & r & 0 & -p \\ r & -q & p & 0 \end{bmatrix} \mathbf{Q} \quad (3.38)$$

The state \mathbf{Q} must be *renormalized* in order to preserve the unitary norm. Time propagation of \mathbf{Q} can be also rewritten as multiplication by another quaternion formed out of the angular velocities, $\mathbf{Q}_{\Omega} = [0 \ p \ q \ r]^T$, such as the time derivation of the elements corresponds to infinitesimal small rotations of \mathbf{Q} by \mathbf{Q}_{Ω}

$$\dot{\mathbf{Q}} = \frac{1}{2} \mathbf{Q} \otimes [0 \ p \ q \ r]^T \quad (3.39)$$

Assuming $\Omega = c\tilde{\Omega}$ where $\tilde{\Omega}$ is unitary and constant over the time interval $[0, T]$ and c is time function denoting actual magnitude of rotation (eigenaxis angular velocity), the integral of (3.38) starting from the quaternion \mathbf{Q} and preserving $|\hat{\mathbf{Q}}| = 1$ during maneuver yields

$$\hat{\mathbf{Q}} = d e^{W(\tilde{\Omega})I} \mathbf{Q}, \quad I = \int_0^T c(\tau) d\tau \quad (3.40)$$

where d is a scalar used to preserve the unitary norm. Such operation can therefore be viewed as a multiplication of \mathbf{Q} by a constant matrix. Moreover, using the quaternion algebra, the matrix multiplication can be substituted by a quaternion multiplication, $\hat{\mathbf{Q}} = \mathbf{Q} \otimes \mathbf{Q}_r$. Similarly, when integrating (3.39), the operation is again equivalent to a multiplication by the same unitary quaternion \mathbf{Q}_r

$$\hat{\mathbf{Q}} = \mathbf{Q} \otimes d [1 \ pI \ qI \ rI]^T = \mathbf{Q} \otimes \mathbf{Q}_r, \quad I = \int_0^T c(\tau) d\tau \quad (3.41)$$

thus, let's define *maneuver* quaternion Q_r , representing the operation needed to reach state \hat{Q} from Q , as

$$Q_r = [a \quad b\tilde{\Omega}I]^T, \quad I \leq 2\pi \quad (3.42)$$

where $a = \cos(\frac{\alpha}{2})$ and b is a scalar used to preserve the unitary norm, $|Q_r| = 1$. Since the total angle of rotation must be $\alpha = I$, substituting *maneuver* quaternion Q_r by *error* quaternion Q_e in (3.37) and then comparing with (3.29) and (3.30) leads to

$$r\alpha = 2 \ln(Q^{-1} \otimes \hat{Q}) = \tilde{\Omega} \int_0^T c(\tau) d\tau \quad (3.43)$$

meaning that the error rotation correction can be achieved by performing the maneuver which drives the angular velocity Ω proportionally to eigenaxis r , namely $\Omega(t) = c(t)r$ with the time constraint $I = \alpha$. \square

Such constraint is satisfied by each controller-plant system that uses $r\alpha$ as error input and *shows equal dynamic responses* for all three axes disregarding the gyroscopic effects, while fulfilling asymptotic tracking. The proof for P-control is shown in [8].

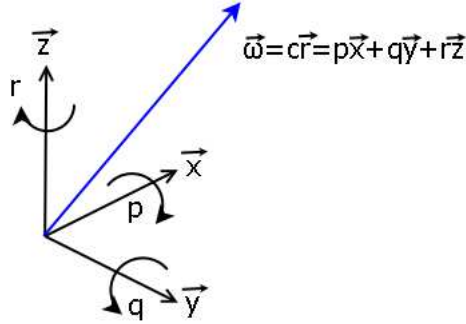


Figure 3.7: Relation between eigenaxis and body angular velocities.

Similar proof could be done for eigenaxis of the rotating matrix R . Moreover, the $r\alpha$ vector can be used as a feedback in existing decoupled axial controllers instead of ϕ , θ and ψ with significantly less errors, when moving far from zero orientation state, because the proportional term is integral of the derivative term (angular velocity). The eigenaxis rotation represents the *shortest* possible correction maneuver (quaternion interpolation) [6]. The only restriction is the constant $\tilde{\Omega}$ during the correction maneuver, requiring equally adjusted dynamics for all axes. Nevertheless, if the feedback controller recalculates the eigenaxis error at each time step, the errors are still much more acceptable during large maneuvers in comparison with *Euler angles*.

3.2.5 Rotation Controller Synthesis

3.2.5.1 Axial Decoupling

Using the superposition principle on the acting thrust and drag torque on each axis in the linearized actuator model from (3.8) and (3.9), the separate decoupled axial dynamics is

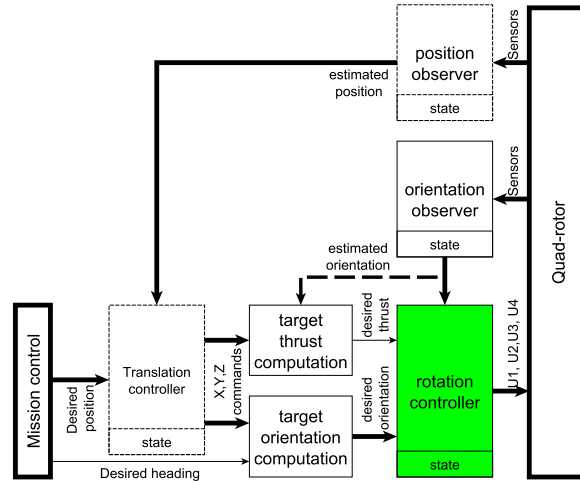


Figure 3.8: Rotation controller placement in the global structure.

shown in fig. 3.9.

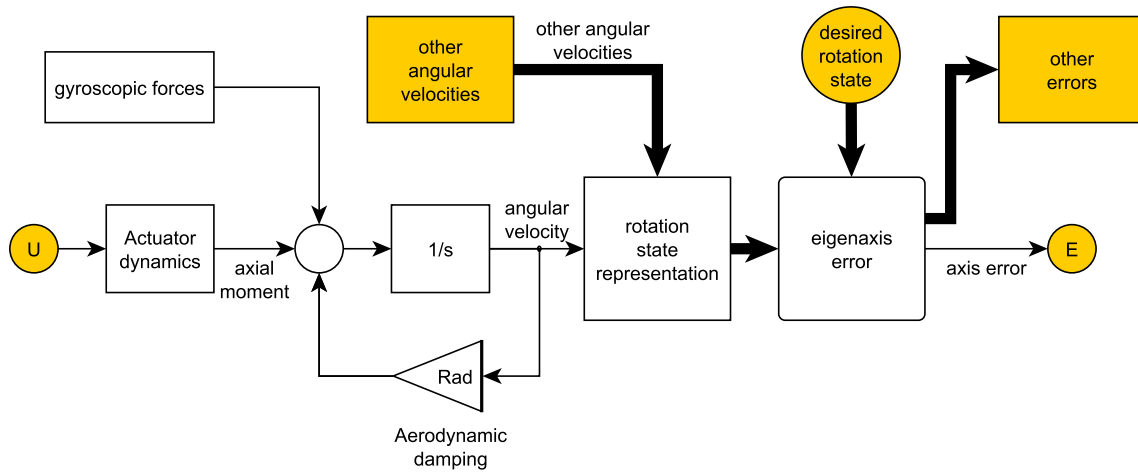


Figure 3.9: Single axis dynamic model.

During the correction maneuver, assuming the equal step responses for all axes from (3.43), the eigenaxis error $[E_x \ E_y \ E_z]^T = 2 \ln(\mathbf{Q} \otimes \hat{\mathbf{Q}}^{-1})$ can be treated as a simple integral of the angular velocity. The diagram shown in fig. 3.9 thus is reduced to the one in fig. 3.10.

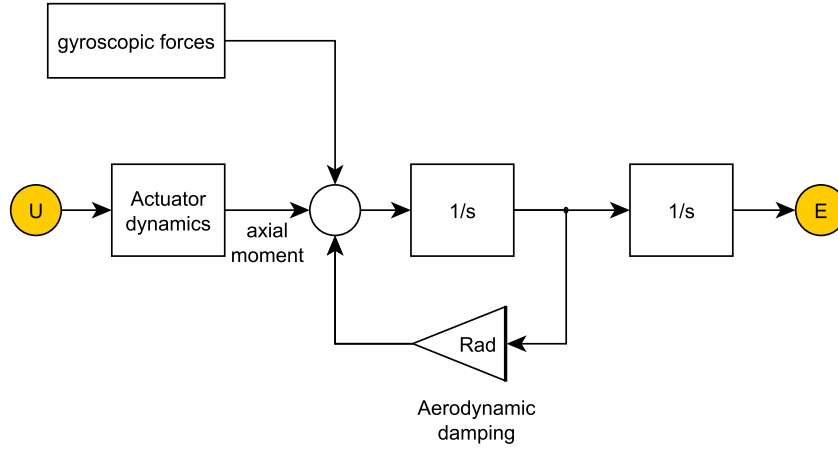


Figure 3.10: Single axis dynamic model during correction maneuver.

$$\begin{aligned}
 \ddot{E}_x &= \frac{l_a}{I_x} (T_4 - T_2) + \frac{r_{adp}}{I_x} \dot{E}_x + \frac{g_x(\omega, \mathbf{\Omega})}{I_x} \\
 \ddot{E}_y &= \frac{l_a}{I_y} (T_1 - T_3) + \frac{r_{adq}}{I_y} \dot{E}_y + \frac{g_y(\omega, \mathbf{\Omega})}{I_y} \\
 \ddot{E}_z &= \frac{1}{I_z} (q_1 - q_2 + q_3 - q_4) + \frac{r_{adr}}{I_z} \dot{E}_z
 \end{aligned} \tag{3.44}$$

3.2.5.2 LQ-optimal controller with non-linear overshoot compensator

Since all the system state variables are available and simultaneously, the precise model is known, the possibility of the LQ control puts up. The hierarchical controller design will not be performed, since the LQ control offers an optimal solution in view of the energy efficiency, simplifies the tuning process and thus enables setting of equal dynamic responses for all three inertial axes. Moreover, on the contrary to similar projects, a simple PID controller failed to stabilise the system, since it does not consider the essential motor angular velocity feedback, which shows to be required due to faster actuator transient responses.

Hereby the conventional LQ-optimal control design will be presented for decoupled axial dynamics augmented with additional integrator to ensure asymptotic tracking. A special non-linear extension will be introduced to compensate the overshoot. Controller will be then equipped with open-loop control to compensate the gyroscopic torques $g_x(\omega, \mathbf{\Omega})$ and $g_y(\omega, \mathbf{\Omega})$ and open-loop thrust control. Linearised decoupled error dynamics on the

X and Y axes can be written in matrix-form, composed of (3.8), (3.9) and (3.44) as

$$\dot{\mathbf{x}}_j = \begin{bmatrix} -\frac{l+m}{I_r} & 0 & 0 \\ \frac{nl_a}{I_j} & \frac{r_{adj}}{I_j} & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x}_j + \begin{bmatrix} \frac{k}{I_r} \\ 0 \\ 0 \end{bmatrix} (\tilde{u}_j) \quad (3.45)$$

$$E_j = [0 \ 0 \ 1] \mathbf{x}_j$$

where $j = \{x, y\}$ is the axis index, $\tilde{u}_x = (u_4 - u_2)$ and $\tilde{u}_y = (u_1 - u_3)$. Similarly, for the vertical axis

$$\dot{\mathbf{x}}_z = \begin{bmatrix} -\frac{l+m}{I_r} & 0 & 0 \\ -\frac{l}{I_z} & \frac{r_{adr}}{I_z} & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x}_z + \begin{bmatrix} \frac{k}{I_r} \\ \frac{k}{I_z} \\ 0 \end{bmatrix} (\tilde{u}_z) \quad (3.46)$$

$$E_z = [0 \ 0 \ 1] \mathbf{x}_z$$

where $\tilde{u}_z = (u_1 - u_2 + u_3 - u_4)$. The LQ-optimal design control with augmented integrator is defined with state feedback vector

$${}_L \tilde{u}_j = -\mathbf{K}_j [\tilde{\omega}_j \ \Omega_j \ E_j \ \int_0^t E_j(\tau) d\tau]^T \quad (3.47)$$

where $j = \{x, y, z\}$ is the axis index. Similarly like the axis voltage \tilde{u}_j , using the superposition principle again, the ω' rotor angular velocity vector can be mapped to the axial omega $\tilde{\omega}$ through a matrix

$$\begin{bmatrix} \tilde{\omega}_x \\ \tilde{\omega}_y \\ \tilde{\omega}_z \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix} \omega' \quad (3.48)$$

which also satisfies the linearized difference actuator model around the operating point, as only the differences remains in $\tilde{\omega}$. The LQ-optimal controller was tuned by careful adjusting of penalisation matrices Q and R . The tuning was an iterative process, which included extensive simulation experiments and real system tests. It was shown that tradeoff between the desired asymptotic tracking behavior versus acceptable *overshoot* is not satisfactory. Generally speaking, control of four integrators connected in serial can be accompanied with an unavoidable overshoot. I-control in our case is not explicitly necessary, however, required to compensate the steady-state errors, like unequal characteristics of all the actuators or displacement of center of gravity from the inertial axes. In our case, the overshoot is caused by the topmost integrator, intended for I-control, which tends to increase its value even when the underlying PD-control could stabilise the system by itself. After the control error E_j reaches zero, the accumulated state inside the integrator then creates the overshoot. Since the topmost integrator is part of the controller, and in our case, pure software implemented, a possibility of some augmented non-linear compensation arises.

Several such techniques have been developed in the literature. As for an example we can name *reset-control* [20] which is, in general, a simple algorithm that "resets" the integrator state as the control error value crosses zero. Nevertheless, this technique might cause here unwanted effects, since the state of the integrator might be "correct" at

zero cross of E_j , compensating some steady-state errors. It has been observed that the fast dynamics can be handled effectively by underlying PD control, whereas the steady state error might develop more likely when the system is in steady state. Such train of thoughts leads to create an overshoot compensator that would "decrease" the input to the integrator when the control error is changing rapidly, i.e. when Ω_j is far from zero. The proposed algorithm is shown in fig. 3.11.

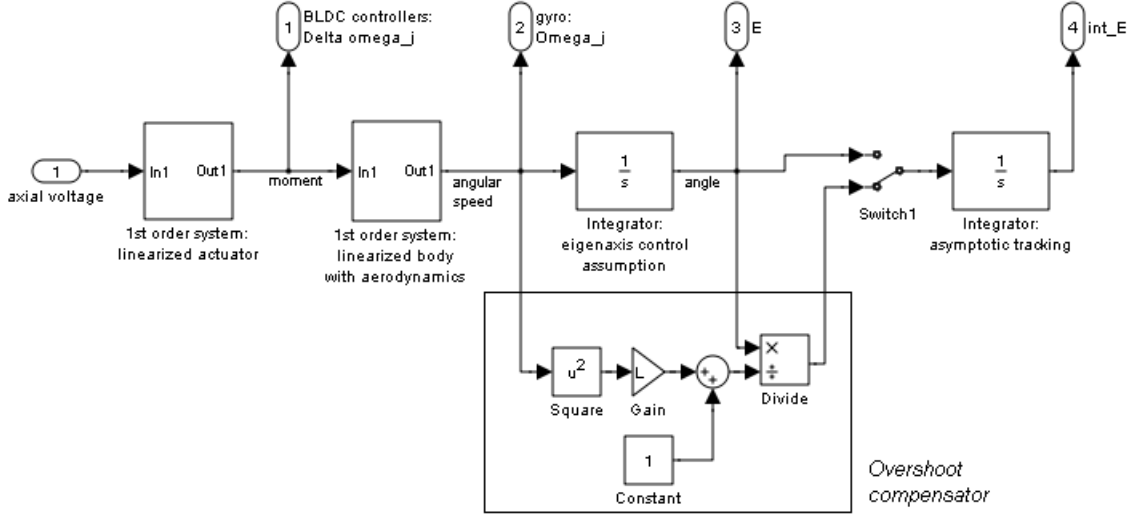


Figure 3.11: Single axis dynamics with asymptotic tracking and overshoot compensator.

where L_j is a hand-tuned constant for j -th axis. The (3.47) therefore changes to

$${}_L \tilde{u}'_j = -\mathbf{K}_j \left[\tilde{\omega}_j \quad \Omega_j \quad E_j \quad \int_0^t \frac{E_j(\tau)}{1+L_j \Omega_j(\tau)^2} d\tau \right]^T \quad (3.49)$$

This is not quite a standard procedure and should be examined using non-linear dynamic system theory to exclude the cause of potential system instability. However, both simulations and real system experiments have proven that it reduces overshoot to less than 30% while preserving suitable asymptotic tracking behavior.

3.2.5.3 Non-linear gyroscopic compensation and thrust control

The gyroscopic torque compensator must be an open-loop controller, since actual gyroscopic torques acting on the body frame cannot be distinguished from the torques used to accelerate the angular velocity. Since $g_x(\omega, \Omega)$ and $g_y(\omega, \Omega)$ can be analytically computed from sensor readings ω, Ω and the steady-state gain of the linearized actuator is known, we can define

$$\begin{aligned} G \tilde{u}_x &= -\frac{m+l}{n k l_a} (P g_x(\omega, \Omega) + D \dot{g}_x(\omega, \Omega)) \\ G \tilde{u}_y &= -\frac{m+l}{n k l_a} (P g_y(\omega, \Omega) + D \dot{g}_y(\omega, \Omega)) \end{aligned} \quad (3.50)$$

where P and D are hand-tuned constants for the open-loop control. Similarly, for the total thrust control, since there is not a straightforward way to distinguish generated thrust force from Earth's gravity vector, open-loop control is proposed as a linearized DC gain of the actuator, such that

$${}_T\tilde{u}_x = \frac{\omega_0 l + f_4(\omega_0)}{4 k f_3(\omega_0)} \hat{T} \quad (3.51)$$

The operating point, $u_0 = \frac{l\omega_0 + f_4(\omega_0)}{k}$, was already considered and is a part of ${}_T\tilde{u}$, since ω_0 is defined as the rotor speed when the sum of all thrusts equals the aircraft's weight $m_t g$.

3.2.5.4 Overall control law and actuator linearisation extensions

Using the superposition principle, the overall, linear control law is then defined

$$\begin{bmatrix} u'_1 \\ u'_2 \\ u'_3 \\ u'_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1/2 & 1/4 \\ 1 & 1/2 & 0 & -1/4 \\ 1 & 0 & 1/2 & 1/4 \\ 1 & -1/2 & 0 & -1/4 \end{bmatrix} \begin{bmatrix} {}_T\tilde{u}' \\ {}_L\tilde{u}'_x + {}_G\tilde{u}_x \\ {}_L\tilde{u}'_y + {}_G\tilde{u}_y \\ {}_L\tilde{u}'_z \end{bmatrix} \quad (3.52)$$

To obtain the complete and yet functioning *linear* controller, we need to furthermore assign the computed voltage $u_j = u'_j$, open-loop thrust voltage ${}_T\tilde{u}' = {}_T\tilde{u}$ and motor angular velocity, $\omega'_j = \omega_j$, in (3.48). These variables have been purposely separated for a special, non-linear mapping technique.

To extend the operational range of the designed controller, I will attempt to create a non-linear input and output mapping functions, which will be used to linearize the non-linear actuator on-the-fly. When designing such non-linear extension for an already-designed controller, one must consider at first, that it should have unitary derivation at the operating point of the controller. Since we have previously stated in (3.11), (3.12) that the non-linearity of the steady state gain from u_j to t_j and from u_j to δ_j can be approximated by a parabolic function with zero offset, this is the only and sufficient condition. Having $\frac{d\sqrt{u'}}{du'} = \frac{1}{2\sqrt{u'}}$, let's define a constant k_i which satisfies the non-linearity unitary derivation $\frac{k_i}{2\sqrt{u'}} = 1$ where its output equals to the actuator operating point $k_i\sqrt{u'} = u_0$. The *input non-linear mapping* function then yields

$$u_j = k_i\sqrt{u'_j} \quad (3.53)$$

where $k_i = \sqrt{2u_0}$ obtained as a solution of the two constraints, u_j is the actual input to non-linear model and u'_j is the *action* value computed by the controller. The secondary effect we need to correct is the evident shift of DC gain, since $u_0 \neq k_i\sqrt{u_0}$. Therefore ${}_T\tilde{u}'$ from (3.51) needs to be scaled. To preserve the unitary thrust gain at operating point u_0 we define constant k_d such that $u_0 = k_i\sqrt{k_d u_0}$. ${}_T\tilde{u}$ is then mapped to ${}_T\tilde{u}'$ simply as

$${}_T\tilde{u}' = k_d {}_T\tilde{u} \quad (3.54)$$

where $k_d = \frac{u_0}{k_i^2} = \frac{1}{2}$.

The ω_j output is one of the state variables used as feedback of the linearised controller. The controller "assumes" that this variable is proportional to angular acceleration. Nevertheless, accepting the assumption that generated thrust t_j is related to angular velocity ω_j non-linearly, the aim for *output linearisation* arises. Once again, the non-linear mapping function should have unitary derivation at the operating point of the controller. Since we're assuming that simply $f_4(\omega_j) \doteq H\omega_j^2$, this is the only and sufficient condition.

Having $\frac{d\omega^2}{d\omega} = 2\omega$, let's define a constant l_i such that $2l_i\omega_0 = 1$. The *output non-linear mapping* function then yields

$$\omega'_j = l_i\sqrt{\omega_j} \quad (3.55)$$

where $l_i = \frac{1}{2\omega_0}$, ω_j is the actual output of the non-linear model and ω'_j is the state variable used by the controller.

Finally, the non linear extended controller is composed of (3.48), (3.49), (3.50), (3.51), (3.52), (3.53), (3.54) and (3.55). One must consider that input and output mapping non-linearities, gyroscopic compensator and overshoot compensator are not the only non-linear extensions inside the controller. The eigenaxis algorithm, $[E_x \ E_y \ E_z]^T = 2 \ln(\mathbf{Q} \otimes \hat{\mathbf{Q}}^{-1})$, is non-linear from its nature. In fact, I do not have mathematical support for validation of most of the extensions inside the system. The non-linear controller was designed to control a non-linear system, in order to operate in entire rotational space and does not have to rely on any operating points. The controller was extensively tested in simulations and can fly the real system on a very good level.

3.2.6 Translation controller design

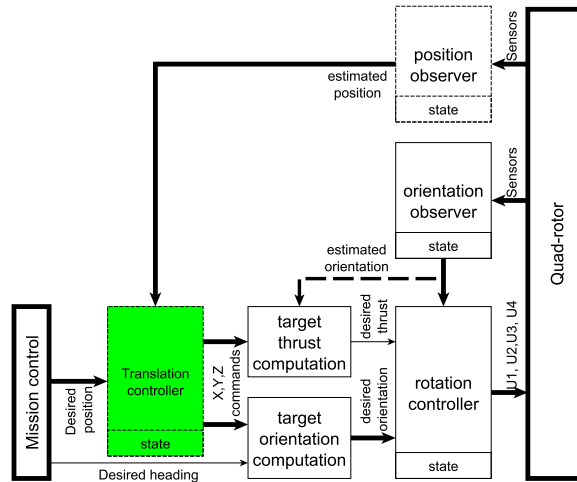


Figure 3.12: Translation controller placement in the global structure.

The translation controller represents the topmost layer of the introduced control structure. In the generalised point of view, actual and target position are its inputs and the

target acceleration vector, $\hat{\mathbf{a}}$, is its output. Both input and output are expressed in NED coordinates. The underlying *target orientation and thrust* computation blocks ensures the universal mapping of the $\hat{\mathbf{a}}$ into the target orientation (when disregarding the translation aerodynamic damping). This non-linear conversion is valid for all $\hat{\mathbf{a}}$ vector directions (e.g. quadrotor upside down), does not work with any operating points, complies with the goal of synthesizing a globally applicable controller and therefore can be even used to perform large scale maneuvers. However, thr decoupling cannot be treated similarly like in the rotation controller, since the axial dynamic response here *depends on the actual orientation state*. Some NED maneuvers involves the rotating (left/right), some does not (up/down) and mainly, all of them depends on initial orientation and body velocity. An exact modelling could be done by linearising the underlying translational dynamic model around the horizontal hovering state, from $\hat{\mathbf{a}}$ to the position. Such linearisation might result in pretty complicated model since the body "rotating" must be incorporated, when considering lateral motion. Nevertheless, simulations have shown that the control of position can be generally synthesized using a decoupled $PD + \dot{D}$ or $PID + \dot{D}$ approach, for each axis of the NED coordinate system.

One must remember that an explicit knowledge of NED position is difficult to obtain. Moreover, the complete position observer has not been designed. Nevertheless, since the onboard accelerometer reads the body acceleration, which can be used directly for translational motion damping, the idea was to split the translation controller into *position controller* and *translation damper*. Translation damper will perform the $D + \dot{D}$ control, based on the accelerometer readings whereas the position controller the P or PI control based on some kind of absolute spatial position readings, like GPS, barrometer, sonar etc. This structure will be designed hierarchically.

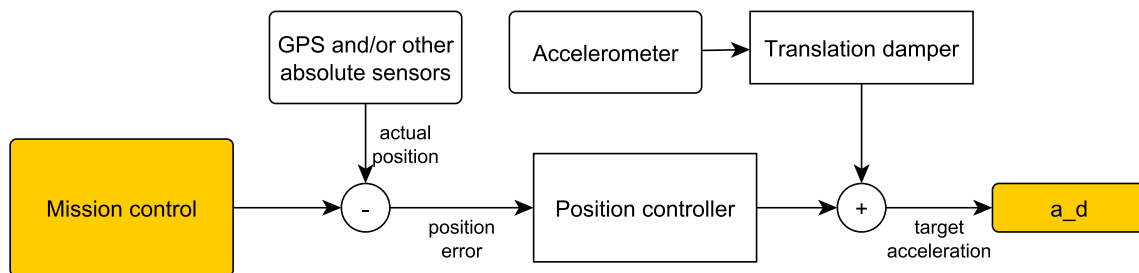


Figure 3.13: Proposed structure of the translation controller.

3.2.6.1 Translation damper design

Since the $\hat{\mathbf{a}}$ vector is expressed in the NED frame, acceleration readings to fed back needs to be also transferred into the NED frame coordinates. However, the rigid body dynamics shown in fig. 3.3 performs the transfer from body to NED coordinaties in *velocity* stage. In between the body translational acceleration, $\dot{\mathbf{V}} = [\dot{u} \ \dot{v} \ \dot{w}]$ and time derivation of the NED velocity vector $\dot{\mathbf{V}}_e$ is a time-varying coordinate transform matrix. The NED acceleration *cannot* be therefore obtained by simply rotating the $\dot{\mathbf{V}}$ vector into NED

frame. Such conversion would require integrating the body acceleration to obtain body velocity \mathbf{V} first, then perform the coordinate transform and then derivate back to obtain the NED acceleration $\dot{\mathbf{V}}_e$. One can reflect the physical natural sense in this statement. If the quadrotor body is moving with constant velocity and then a rotation occurs, the acceleration in NED frame will be generated, even without the change of acceleration vector in body frame.

The best results were measured when using the $\dot{\mathbf{V}}_e$ vector for direct damping in NED frame. However, since the conversion would require additional controller states, as for the integrators of the NED acceleration, it may result in potential causes of instability. To avoid this problem, I propose the translation damping to be performed in body frame coordinates as for an alternative solution. The $D + \dot{D}$ control approach requires the integration of $\dot{\mathbf{V}}$ vector, since it already represents a second time derivation of the position. To avoid the potential drift when integrating the accelerometer readings, and similarly considering the influence of aerodynamic damping of the translational motion, a different approach is suggested. The simulations have shown that $D + \dot{D}$ control can be replaced with a first-order lowpass filter, where the tuned parameters would be the time constant and gain, instead of separate gains for D and \dot{D} constraints. Let's then define the *body translational damper*, as

$$\begin{aligned}\mathcal{L}\{\mathbf{B}_d a\}_x &= \frac{Kd_{lat}}{s\tau_{lat} + 1} \mathcal{L}\{\dot{\mathbf{V}}\}_x \\ \mathcal{L}\{\mathbf{B}_d a\}_y &= \frac{Kd_{lat}}{s\tau_{lat} + 1} \mathcal{L}\{\dot{\mathbf{V}}\}_y \\ \mathcal{L}\{\mathbf{B}_d a\}_z &= \frac{Kd_{long}}{s\tau_{long} + 1} \mathcal{L}\{\dot{\mathbf{V}}\}_z\end{aligned}\tag{3.56}$$

where Kd_{lat} , Kd_{long} , τ_{lat} and τ_{long} are the damper parameters for lateral and longitudinal motions. Direct usage of \mathbf{acc} reading for damping is impossible due to *the Accelerometer Paradox*. However, the *the Accelerometer Paradox* also results in natural lowpass filtering of the \mathbf{acc} vector direction caused by the aerodynamic forces and therefore could possibly aid the damping as well, which is a subject of further examination. The body acceleration, $\dot{\mathbf{V}}$ vector is obtained from (3.71) and depends on correct orientation estimation Q . Note that the ${}^B_d a_n$ vector needs to be transferred again into the NED frame.

The current approach might seem unnecessarily complicated: passing the computed body-frame damping through a coordinate transform, then converting it into a target quaternion and finally convert it back into a body-frame eigenaxis errors. One may object that since the damping has yet been computed in the body frame, it could be used directly inside the feedback of rotation controller, the lateral part as the angle deviations and the longitudinal as thrust damping, as the rotation controller yet works directly in the body frame. Such alternative approach was extensively examined and was found to be unsuitable due to the following reasons.

- Since both translation damper and position controller outputs are joined into a single direction vector $\hat{\mathbf{a}}$, the tilt limitations can be easily achieved by applying saturation limit to this vector. For example, a simple condition can be stated to

prevent the quadrotor to cross the tilt limit of 90 degrees (upside down), or limit the overall tilt deflection to certain suitable value. Such limitations couldnot be achieved when feeding back the damping directly into orientation controller output.

- The separate damping from the vertical acceleration in body frame to the thrust \hat{T} performs much worse than the current approach. Passing the computed damping as a part of the $\hat{\mathbf{a}}$ vector to the *target orientation and thrust computation* blocks helps the damping response not only by modifying the thrust, but also by rotating the entire quadrotor body in parallel, or "against" the arised acceleration. The two approaches were compared in simulations.
- The additional element inside the rotation controller could disturb the eigenaxis algorithm by violating the need for equal axial dynamic response and could even lead to system destabilisation.

3.2.6.2 Position controller design

The topmost position controller can then be accomplished using the *PI* control strategy with an input of the absolute spatial sensors with low update rates (e.g. GPS) and the target position value obtained from the mission control unit.

$$\begin{aligned}
 {}^N_p \hat{a}_x &= Kp_{lat}(\hat{X}_x - X_x) + Ki_{lat} \int_0^T \hat{X}_x(\tau) - X_x(\tau) d\tau \\
 {}^N_p \hat{a}_y &= Kp_{lat}(\hat{X}_y - X_y) + Ki_{lat} \int_0^T \hat{X}_y(\tau) - X_y(\tau) d\tau \\
 {}^N_p \hat{a}_z &= Kp_{long}(\hat{X}_z - X_z) + Ki_{long} \int_0^T \hat{X}_z(\tau) - X_z(\tau) d\tau
 \end{aligned} \tag{3.57}$$

where Kp_{lat} , Kp_{long} , Ki_{lat} and Ki_{long} are the controller parameters for lateral and longitudinal motions. To prevent the possible overshoot caused by the I-control, a compensator can be engaged, similar like the one shown in fig. 3.11.

3.2.6.3 Final translation control law

The overall law is obtained by combining the results of translation damper and position controller, according to the fig. 3.13. The output of the translation damper must be coordinate transferred from body to NED.

$$\hat{\mathbf{a}} = {}^N_p \hat{\mathbf{a}} + \mathbf{Q}^{-1} {}^B_d \hat{\mathbf{a}} \mathbf{Q} \tag{3.58}$$

Note that all the constants involved inside this controller were obtained from either linearizing the underlying translational motion model and then using the classical controller synthesis method in frequency and time domain, or experimentally adjusted by an iterative process, involving the simulation and laboratory experiments.

3.3 Observer Design

3.3.1 Orientation observer design

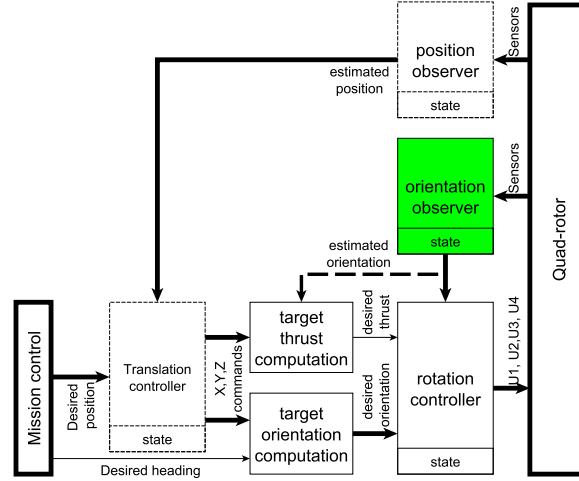


Figure 3.14: Orientation observer placement in the global structure.

The problem of determining the actual orientation state, often using inertial sensors only, has been again recalled many times in aerospace field [10], [11], [19]. However, quadrotor-specific approach has not been found in the available literature. The solid-state unit providing the orientation state is often abbreviated the Attitude Heading Reference System (AHRS). The approach presented here will be highly experimental. The principles will be more explained using a natural language rather than mathematical derivations. The algorithm proved to be functional and suitable for the quadrotor control usage. Nevertheless, the mathematical validation of the approach remains open.

Latest progress in MEMS technology (Micro-Electro-Mechanical Systems) enabled to manufacture single chip sensors that can be used to avoid the need of high precision mechanical gyroscopes. A common type of modern AHRS unit, suitable and affordable for this project, is fully electronical and uses inertial and magnetic sensors to determine the absolute spatial rotation. Such unit can be constructed to work stand-alone as a black box with no evident inputs. It determines the absolute rotation using accelerometers, magnetometers and gyrometers. The key difference between an IMU (inertial measurement unit) and an AHRS is the addition of an on-board processing system in an AHRS which provides solved attitude and heading solutions versus an IMU which just delivers sensor data to an additional device that solves the attitude solution. AHRS differ from traditional inertial navigation systems by attempting to estimate only attitude (i.e. roll, pitch, yaw a.k.a heading) states, rather than attitude, position and velocity as is the case with an INS.

The orientation state could be theoretically determined only from the accelerometer and magnetometer, because of the readings are two linearly independent vectors in NED frame, which is enough to uniquely determine the absolute orientation. Similarly or by

Table 3.1: Systematic errors of the inertial sensors

<i>sensor</i>	<i>used to measure</i>	<i>systematic error</i>
accelerometer	earth's acceleration - tilt	parasitic acceleration and vibrations
magnetometer	earth's magnetic field - heading	parasitic magnetic field
gyro	angular rates	drift error

integrating the gyro readings. Nevertheless, all the sensors have some unavoidable systematic errors. For a reasonable rotation determination, we need to use the measurements from all of them. A form of non-linear estimation such as a Kalman filter is typically used to compute the solution from these multiple sources. The method is often recalled *fusion algorithm*. In the language of control engineering, the correct name is rather orientation estimator or orientation *observer* - an algorithm, that works with model of actual system and tries to adjust its state according to the sensor measurements that are somehow related to the state of the real system.

3.3.1.1 The Accelerometer Paradox

As shown in table 3.1, accelerometer is used to measure Earth's acceleration vector. However, accelerometer also reads the translational acceleration in body frame, $\mathbf{acc} = \dot{\mathbf{V}} - \mathbf{g}$ where V is the velocity vector in body frame. Since the correction of orientation estimation relies only on a two, linearly independent spatial vectors *acc* and *mag*, the superposition of linear acceleration $\dot{\mathbf{V}}$ to *acc* will result in estimation error, if not corrected. Most of the universal, yet modern AHRS algorithms often does not incorporate a model related to translational movements of the unit carrier. This phenomena is often simply solved by applying less weight on accelerometer estimation correction stage, requiring more precise gyro sensors to avoid the estimation drift. In higher level aerospace field, the observer often contains a complete model, including the rigid body motion and is generally able to estimate the translational movements and can therefore distinguish between the two componets of accelerometer readings.

During late development of this project, an orientation estimation error caused by translational movements of the quadrotor becomes evident and a call for joining the orientation and position observer equipped with the motion model arises. Nevertheless, a special phenomena, called *the Accelerometer Paradox* has been encountered. For explanation, a simplified motion model of the quadrotor in a flat surface with three degree of freedom is shown in 3.15.

The accelerometer reads the difference of acceleration from a free fall. One must consider that the NED coordinate system is not an inertial system, since there is the Earth's acceleration present. The inertial coordinate system on Earth is attached to a

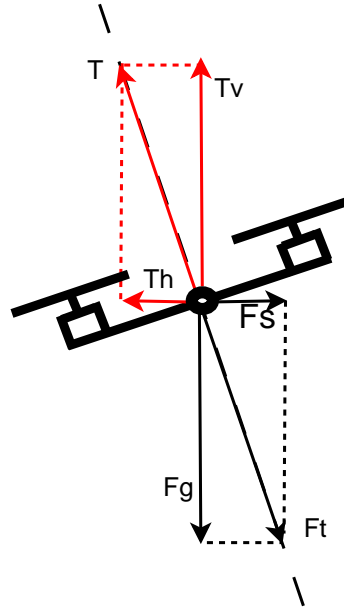


Figure 3.15: The 3DOF 2D simplification of a basic quadrotor model.

body free falling towards the centre of the Earth. Accelerometer placed on the desk will read the acceleration vector pointing up, as the table is generating a force that inhibits the accelerometer to fall. Such force can be viewed to as similar to the one that would accelerate the carrier with accelerometer to the side. For more detailed explanation, please refer to [21]. Thus, for the quadrotor case, the accelerometer reading $\mathbf{acc} = \frac{-\mathbf{F}_s - \mathbf{F}_g}{m_t} = \frac{\mathbf{T}_h + \mathbf{T}_v}{m_t} = \frac{\mathbf{T}}{m_t}$. Since accelerometer is attached to the quadrotor body and thrust vector T is always aligned with the vertical axis, the accelerometer will read a nonzero value only in the Z-axis. Other two parts of the \mathbf{acc} vector will be zero. The similar result is obtained from (3.28) when subtracting the Earth's acceleration $\mathbf{acc} = \dot{\mathbf{V}} - \mathbf{g}$ and neglecting the gyroscopic and disregarding the aerodynamic forces. However, the aerodynamics forces are actually the ones that makes the accelerometer useful again in the orientation estimation. Aerodynamics forces reflects natural resistance of translational movement by air friction of the body. After the time constant given as $\frac{1}{t_{ad}}$, the resistance of air friction becomes as high that the accelerating stops and the velocity becomes constant. After that time, we can expect the accelerometer to be measuring only the gravity again $\mathbf{acc} = -\mathbf{F}_g m_t$. Thus, to include the translational model into orientation observer, a complete and accurate aerodynamics model would be needed. Due to non-availability of some key values, such as wind speed and need for a complex modelling, an alternative solution to correct *the Accelerometer Paradox* will be proposed.

3.3.1.2 Observer principle

The general discrete observer has two main stages: *prediction* and *correction*. Both of them are modifying the internal state, that represents the current spatial orientation in our case. Since the gyro readings correspond to *time change* of the orientation, they will

most likely act in the *prediction* stage, whereas accelerometer and magnetometer readings are related to the state directly, they need to be used in *correction* stage. The general diagram of the discrete-time orientation observer is shown in fig. 3.16.

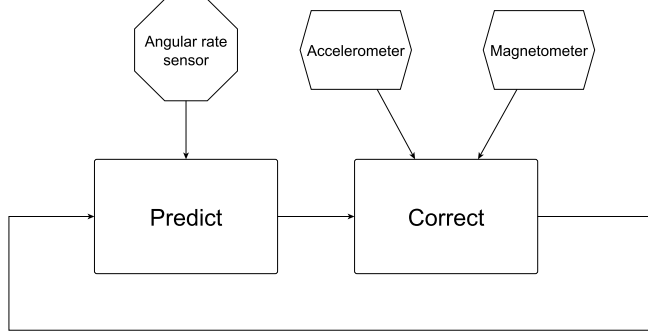


Figure 3.16: General diagram of the orientation observer.

The key question is the choice of spatial orientation representation for the state of the observer. Commonly used *Euler angles*, ϕ , θ and ψ , are related highly non-linearly to all the sensor readings. Moreover, the relation varies according to the actual orientation state. Such approach thus needs on-the-fly linearisation in the correction stage, requires time demanding computation of a Jacobian matrix at each time step and generally leads to explicit need for the *extended Kalman filter* [13], [14]. An alternative method to this standard, but computationally very demanding technique was described in [18].

3.3.1.3 Vectors as state of the observer

For the first part of the observer state S , let's define estimated magnetic vector, S_m , and acceleration vector, S_a . The *prediction* stage will therefore consist of time-varying time propagation of these three dimensional spatial vectors. The propagation can be also viewed as an infinitesimally small rotation by a rotational matrix formed out of the angular velocities obtained from gyro sensors

$$\begin{aligned} \mathbf{S}'_a &= \mathbf{S}_a + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \mathbf{S}_a dt \\ \mathbf{S}'_m &= \mathbf{S}_m + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \mathbf{S}_m dt \end{aligned} \quad (3.59)$$

where dt , using rectangular discretization, is the time interval elapsed since the last prediction. The *correction* stage for the vector states is then a simple state injection with weight K_a and K_m defined as

$$\begin{aligned} \mathbf{S}_a &= \mathbf{S}_a + (\mathbf{acc} - \mathbf{S}'_a) K'_a dt \\ \mathbf{S}_m &= \mathbf{S}_m + (\mathbf{mag} - \mathbf{S}'_m) K_m dt \end{aligned} \quad (3.60)$$

where *acc* and *mag* are unbiased accelerometer and magnetometer readings. The observer now predicts both vectors using angular rates measured by gyro and corrects the developing drift by actual accelerometer and magnetometer readings. Constants K'_a and K'_m represents the correction weight i.e. how much we "trust" to the accelerometer and magnetometer in relative to the gyro, and similarly it is related to cutoff frequency of lowpass filter for the accelerometer and magnetometer.

Here I will present a special strategy on how to compensate *the Accelerometer Paradox*. It was stated above that aerodynamic forces after certain time constant causes the aircraft to stop accelerating and furthermore accelerometer readings are not disturbed by the translational acceleration. It is known that a *change* of either direction or magnitude of the thrust generated by the propeller can be viewed to as a step applied to the aerodynamics. After its time constant, $\frac{1}{t_{ad}}$, system will enter into steady state and accelerometer can be used to aid the orientation estimation again. The main idea is to create a function, $f_{ac}(\Omega, \hat{T})$, that estimates the actual *change* of the thrust vector. Output of this function will be connected to a first order lowpass filter, with a time constant τ_{ac} reflecting the constant of the aerodynamics, $\frac{1}{t_{ad}}$. Accelerometer correction weight, K_a , will be then adjusted dynamically, indirect proportionally to the output of this filter. Let's define a new state variable, S_p , representing the state of this filter. The *Accelerometer Paradox* compensator can be then described as

$$\begin{aligned} \dot{\mathbf{S}}_p &= (f_{ac}(\Omega, \hat{T}) - \mathbf{S}_p) \frac{dt}{\tau_{ac}} \\ K'_a &= \frac{K_a}{1 + K_{ac} \mathbf{S}_p} \end{aligned} \quad (3.61)$$

where K_{ac} is the gain of the compensator. The expression of $f_{ac}(\Omega, \hat{T})$ function is not yet clearly stated currently being experimented with. A simplest possibility is to use a squared magnitude of the Ω vector, $f_{ac}(\Omega, \hat{T}) = |\Omega|^2$, assuming that the thrust vector changes when device is rotating. One can object that the thrust vector change can be obtained analytically. However, such computation relies mainly on the correct orientation estimation. Since this is a part of the orientation observer, an unwanted feedback can arise and destabilise the system. A special care must be taken mainly when using the gyro bias tracking together with the *Accelerometer Paradox* compensator. Generally, each additional state in such highly nonlinear and coupled system, formed out of a model, observer and controller might cause an instability. For that reason, it is intended to keep the amount of necessary states at minimum.

3.3.1.4 Quaternion extraction

The controller requires orientation state in the form of a quaternion. Several techniques were introduced in the literature on how to form a quaternion from accelerometer and magnetometer readings. The most common method involves solving inverse relation of quaternion rotation for the vectors, a set of non-linear equations. This not very easy to implement and moreover, reference magnetometer and accelerometer readings, are required, which might not be invariant during entire flight period. Here I will present a more analytical method, without the need of any reference measurements and easy to

implement. Since both vectors are linearly independent, they are uniquely determining the spatial orientation and a complete DCM matrix can be obtained simply by applying vector operations for the states, as

$$\mathbf{R} = \begin{bmatrix} \frac{\tilde{\mathbf{S}}_{\mathbf{a}} \times \mathbf{S}_{\mathbf{m}}}{|\tilde{\mathbf{S}}_{\mathbf{a}} \times \mathbf{S}_{\mathbf{m}}|} & \frac{(\tilde{\mathbf{S}}_{\mathbf{a}} \times \mathbf{S}_{\mathbf{m}}) \times \tilde{\mathbf{S}}_{\mathbf{a}}}{|(\tilde{\mathbf{S}}_{\mathbf{a}} \times \mathbf{S}_{\mathbf{m}}) \times \tilde{\mathbf{S}}_{\mathbf{a}}|} & -\frac{\tilde{\mathbf{S}}_{\mathbf{a}}}{|\tilde{\mathbf{S}}_{\mathbf{a}}|} \end{bmatrix} \quad (3.62)$$

Assuming $\tilde{\mathbf{S}}_{\mathbf{a}} = \mathbf{S}_{\mathbf{a}}$, acceleration vector is used to determine the tilt (2 degrees of freedom) and magnetic vector to determine heading (remaining single degree of freedom). Nevertheless, the method has been criticized, since magnetometer can provide a valuable information of tilt as well, which is not used here, in opposite of the method described in [19]. To improve this problem and simultaneously avoid the need of reference magnetometer and accelerometer measurements, a special algorithm, called the *angle filter* is proposed. Basically, it is a single order lowpass filter with state of an angle between magnetic and acceleration vector, which modifies the acceleration vector so that the angle in a plane defined by Earth's acceleration and magnetic vector tends to remain constant. Let's define a new state variable for our observer, S_a , representing the angle between acceleration and magnetic vectors. Algorithm can be then defined as a set of differential equations

$$\begin{aligned} \dot{S}_\alpha &= \left(-a \cos \left(\frac{\mathbf{S}_{\mathbf{a}} \bullet \mathbf{S}_{\mathbf{m}}}{|\mathbf{S}_{\mathbf{a}}| |\mathbf{S}_{\mathbf{m}}|} \right) - S_\alpha \right) \frac{1}{\tau_\alpha} \\ \tilde{\mathbf{S}}_{\mathbf{a}} &= \frac{\mathbf{S}_{\mathbf{m}}}{|\mathbf{S}_{\mathbf{m}}|} \cos(S_\alpha) + \frac{(\mathbf{S}_{\mathbf{a}} \times \mathbf{S}_{\mathbf{m}}) \times \mathbf{S}_{\mathbf{m}}}{|(\mathbf{S}_{\mathbf{a}} \times \mathbf{S}_{\mathbf{m}}) \times \mathbf{S}_{\mathbf{m}}|} \sin(S_\alpha) \end{aligned} \quad (3.63)$$

where $\tilde{\mathbf{S}}_{\mathbf{a}}$ is the "angle filtered" acceleration vector and τ_α is the time constant for the lowpass angle filter. When inserting $\tilde{\mathbf{S}}_{\mathbf{a}}$ from (3.63) into (3.62), the tilt is not determined only by accelerometer reading, but also depends on a change of angle between the magnetic and acceleration readings. Sudden change of estimated acceleration vector direction, S_a not accompanied by a sudden change of estimated magnetic vector direction, S_m won't invoke a sudden change of a orientation estimation in R . In this situation, the orientation estimation will follow the change in S_a exponentially with time constant τ_α , preserving the S_α angle in North-Down plane. Therefore, setting of τ_α should reflect the expected time constant of the real magnetic inclination change. Note that the noise power and bandwidth of the accelerometer is supposed to be much higher than noise power and bandwidth of the magnetometer. The final task is then to convert the DCM matrix \mathbf{R} into the form of a rotation quaternion. This generally involves finding an eigenaxis of the \mathbf{R} matrix corresponding to eigenvalue 1. Various algorithms have been designed to avoid the need of numerical computations, using the special properties of the DCM matrix. A good, singularity-free algorithm was designed by Klumpp [12]. It computes the quaternion analytically as function of \mathbf{R} using only linear algebra and program flow conditions. Let's then simply define our result quaternion estimation Q_r as

$$\mathbf{Q}_r = \text{klumpp}(\mathbf{R}) \quad (3.64)$$

3.3.1.5 Quaternion as state of the observer and gyro bias tracking

In the above section, both *prediction* and *correction* stages were applied on the three dimensional vectors. The other possibility is a direct usage of a quaternion widely used in the existing AHRS designs [19]. Let's define S_Q as observer state. For *prediction*, time propagation of a quaternion was already described in (3.38):

$$\mathbf{S}'_Q = S_Q + \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & -r & q \\ q & r & 0 & -p \\ r & -q & p & 0 \end{bmatrix} \mathbf{S}_Q dt \quad (3.65)$$

The *correction* stage is often performed as a state injection, or in sensor readings space using on-the-fly linearisation by forward-inverse transformation of the reference magnetometer and accelerometer readings [19]. I have chosen to use a state injection of a *measurement* quaternion computed directly from sensor readings. Such quaternion is easily obtained by an algorithm developed above. After substituting the unbiased sensor readings directly into the set of equations (3.62), (3.63) and (3.64) namely $\tilde{\mathbf{S}}_a = acc$ and $\mathbf{S}_m = mag$, Q_r is then the *measurement* quaternion. Since both state and measurement are available in the same representation, a constant and diagonal gain can be applied in state injection for the quaternion elements. Such technique is used widely in the existing AHRS algorithms.

Nevertheless, one must consider a fact that a rotation quaternions have unitary norm, requiring the re-normalisation after each time step. Applying correction directly to the quaternion elements may thus result in some unwanted effects. When the orientation difference between Q_r and S_Q is high, the S_Q elements might be close to zero after the correction. The time change of S_Q even after re-normalization then does not correspond to a closest path in the orientational spherical space, i.e. S_Q does not change "smoothly". Re-normalization may even fail, when all the elements of S_Q are zero. To avoid this problem, a technique known as *quaternion interpolation* is proposed here. At first, we define an error quaternion \mathbf{Q}_c that represents rotation needed to perform from state S_Q to state Q_r , such as

$$\mathbf{Q}_c = \mathbf{S}_Q^{-1} \otimes \mathbf{Q}_r \quad (3.66)$$

the angle of rotation of \mathbf{Q}_c is then scaled by a constant K_Q .

$$\mathbf{Q}_i = \left[\cos(K_Q \text{acos}(E_0) dt) \quad \sin(K_Q \text{acos}(E_0) dt) \frac{[Q_{c1} \quad Q_{c2} \quad Q_{c3}]}{\| [Q_{c1} \quad Q_{c2} \quad Q_{c3}] \|} \right]^T \quad (3.67)$$

And finally, state \mathbf{S}_Q is then multiplied by the scaled quaternion

$$\mathbf{S}_Q = \mathbf{S}_Q' \otimes \mathbf{Q}_i \quad (3.68)$$

The \mathbf{S}_Q therefore follows \mathbf{Q}_r exponentially, but in the rotational space. On the contrary to a simple linear correction applied to the quaternion elements, quaternion interpolation always follows the *closest* path from \mathbf{Q}_r to \mathbf{S}_Q with the angle of error changing

exponentially. Moreover, since the error quaternion \mathbf{Q}_c represents the error in our current estimated state, we can extract the eigenaxis of \mathbf{Q}_c and use it to track the bias of the gyro sensors. If the deviation in our current estimation, represented by the angle of \mathbf{Q}_c , tends to stay high for a certain period of time, it is most likely caused due to an error in the *prediction* stage, i.e. there is an offset in gyro readings. Let's then define a gyro bias state, \mathbf{S}_b . We can adjust the bias simply by integrating the eigenaxis multiplied by the rotation angle of \mathbf{Q}_c

$$\mathbf{S}_b = \int_0^T K_b \ln(\mathbf{Q}_c(\tau)) dt \quad (3.69)$$

where K_b is the bias state injection weight, and the angular rates can be finally extracted from the gyro readings as

$$\Omega = \text{gyro} - \mathbf{S}_b \quad (3.70)$$

The principle of bias tracking in a 1DOF system is shown in fig. 3.17. Note that K_b must be much smaller than K_a for the bias estimator to work.

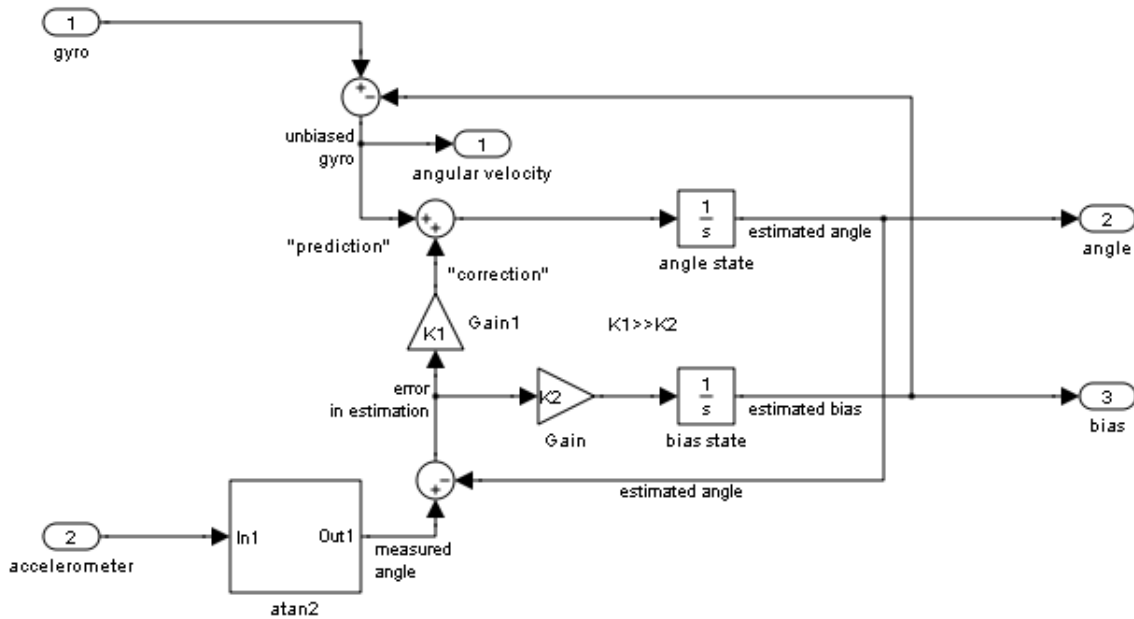


Figure 3.17: Bias tracking principle in a single degree of freedom system.

3.3.1.6 Combined state approach

A two independent AHRS approaches were described above. Advantages and disadvantages of them follows.

- The usage of two linearly independent vectors as the observer state and output extraction of a quaternion can be modified to correct the accelerometer paradox,

but cannot generally track gyro bias, since the vectors are representing only two degrees of freedom in view of rotation description. Moreover, the measurement noise can be expected to be a gaussian white noise superponed to both magnetometer and accelerometer readings. The linear, constant and diagonal state injection of such sensor readings will filter out the noise similarly like a first order lowpass filter.

- The usage of a quaternion as the observer state allows an easy gyro bias tracking. However, correcting the accelerometer paradox then is not a straightforward task. Moreover, the quaternion correction stage requires converting both accelerometer and magnetometer readings into the quaternion first. Such conversion is highly non-linear and high frequency noise with high amplitude may result in even higher errors in the obtained measurement quaternion Q_r .

Thus, aiming to combine the advantages and to diminish the disadvantages of both these attitudes, a combined state approach is presented. Both the vectors and the quaternion are held as the observer state. Both of them are predicted and corrected. The sensor readings are used to correct only the vector states, to enable the possibility of accelerometer paradox compensation and to filter out the higher frequency measurement error. Quaternion is then extracted from the vectors and used as a measurement for the quaternion correction stage, enabling the gyro bias tracking. In that way, a "second-order" observer is created, combining the advantages of both attitudes. Note that this is a highly-experimental approach and the algorithm convergence should be examined. however, best results have been recorded using it in comparison with traditional AHRS algorithms. Constant tuning procedure for this algorithm has not yet been derived.

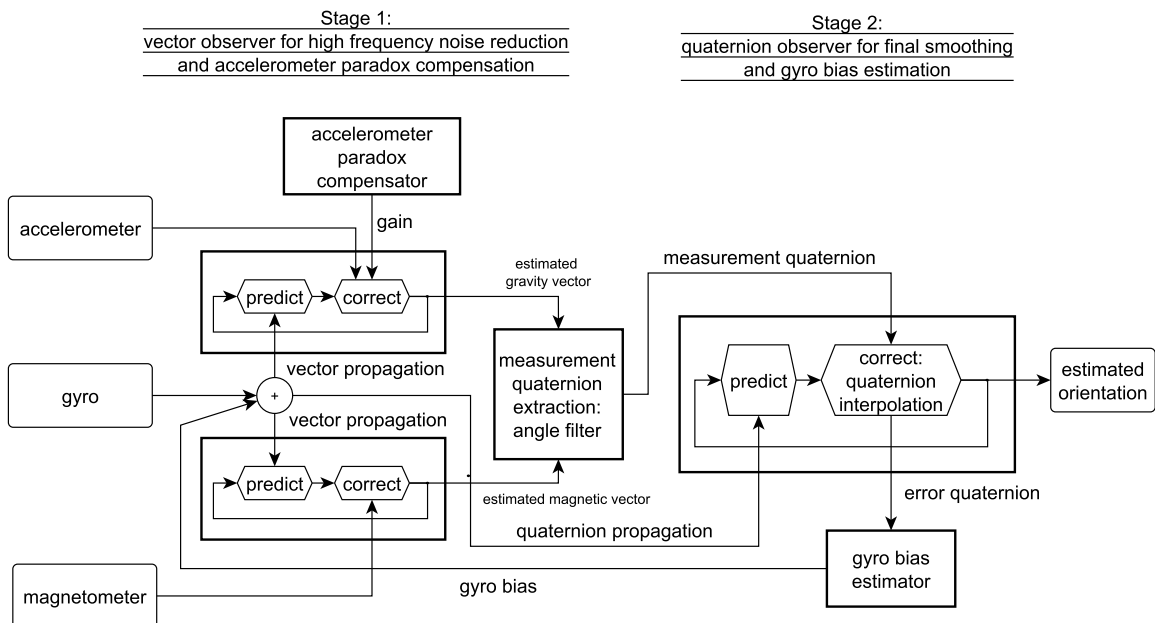


Figure 3.18: The AHRS algorithm flowchart.

```

function AHRS_Update (vector acc, vector mag, vector gyro, structure S, parameter dt)

%Ubias the gyro readings to obtain the angular velocities (Omega)
Omega = gyro - S.b;

%apply the vector prediction to accelerometer state
s.a = vector_predict(S.a, Omega, dt);
%apply the vector prediction to magnetometer state
S.m = vector_predict(S.m, Omega, dt);
%apply the quaternion prediction to orientation state
S.q = quat_predict(S.q, Omega, dt);

%apply lowpass filter on the estimated thrust vector change
S.O = S.O + (f_ac(Omega) - S.O) * dt/TAU_AERODYNAMICS);
%compute the target correction weight
k = K_a / (1 + S.O*K_AERODYNAMICS);

%apply correction to the accelerometer state
S.a = S.a + (acc[i] - S.a[i]) * k * dt);
%apply correction to the magnetometer state
S.m = S.m + (mag[i] - S.m[i]) * K_m * dt);

%extract the measurement quaternion from accelerometer and magnetometer state
Qr, S.alpha = accmag2quat_filt(S.a, S.m, S.alpha, dt);

%compute the error quaternion (from measurement to state)
E = quat_multiply(quat_conjugate(state->x), Qr); // E = x^-1 * Qr

%update the bias estimation
S.b = S.b + quat_log(E, axis) * K_b * dt);

%apply the correction (Quaternion interpolation) to the orientation state
S.Q = quat_multiply(S.Q, quat_scale(E, K_Q*dt)); //x = x * quat_scale(E,K_Q)

end function

```

Figure 3.19: The AHRS algorithm in pseudo-code.

3.3.1.7 Sensor calibration

Up to now, we supposed that **acc**, **mag** and **gyro** sensor reading vectors are scaled in real units and does not suffer from offset (bias) errors. To obtain these values, all three sensors, aiding the orientation estimation (accelerometer, magnetometer and gyro) needs to be calibrated. A proces of linear calibration, in general, involves finding two parameters for each vector component: an *offset* and a *scale*. The method of calibration for the onboard sensors is described below.

- The cheap gyro sensor offsets often vary with the temperature change and moreover, a misplaced offset setting is critical to the orientation estimation. This is the reason for insisting on automated tracking of the gyro bias values by the developed AHRS algorithm, even during the flight. The initial value for the offsets are set during the pre-flight initialization procedure, supposing that the device is standing by. On the contrary, the scale for a gyro reading is a parameter that often does not change rapidly with time. Thus, it is being set as a constant determined using

external calibration tool, or as a result of the computation involving the datasheet specifications, gain of analog circuitry and the A/D converter.

- Both magnetometer and accelerometer calibration procedure is a special human-aided process implemented in the software. The procedure involves placing (rotating) the quadrotor body to most possible states of orientation. Both magnetometer and accelerometer data are sampled and then passed to a special ellipsoid fitting algorithm, which performs solving a set of overloaded linear equations. The algorithm is realised using Lagrange multipliers, which ensures finding least square solution of the six unknown variables, from which three of them are main axes of the ellipsoid and remaining three are its center coordinates. Main axes are then used as *scale* parameters whereas the center coordinates are used as *offsets* for each sensor. Please note that a superposition of linear acceleration to the accelerometer readings can disturb the calibration process of the accelerometer. At least six samples are required to solve the set of six equations with six unknown variables. The details can be found in [24], [25], [26]. The fitting algorithm was simplified by disregarding the coefficients of cross-variable multiplicants in the ellipsoid quadric equation, which is used in [26] to compensate also the perpendicular axes misalignment and cross influencing. The simplified algorithm thus assumes that main axes of the ellipsoid are parallel to the inertial axes and the simplification therefore yields to finding eigenvalues of a 3x3 matrix only, which can be computed analytically by solving roots of its third order characteristic polynomial.

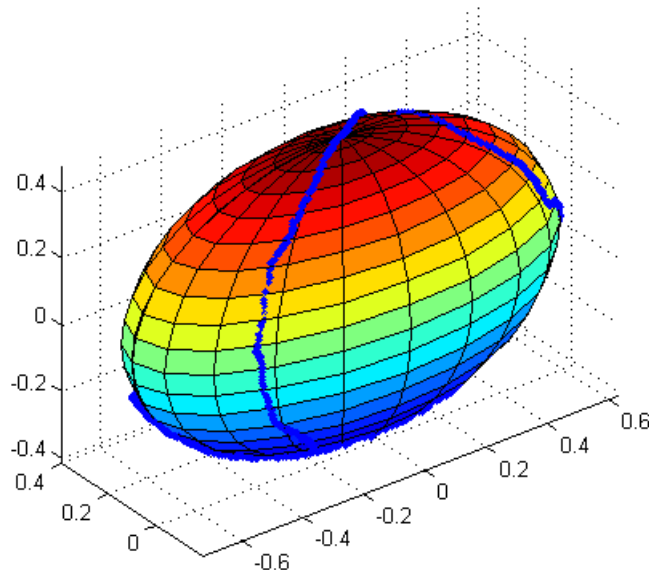


Figure 3.20: Visualisation of the ellipsoid fitted to the measured data.

3.3.2 Position observer design

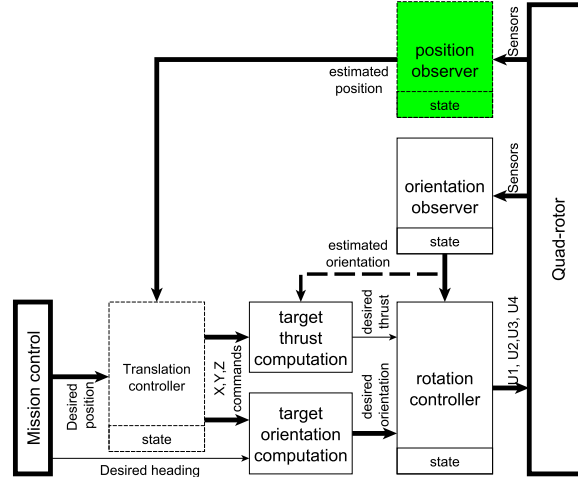


Figure 3.21: Position observer placement in the global structure.

Just like for the orientation, a similar problem arises for the spatial position determination. A simple, stand-alone, absolute and global, spatial position sensor, suitable for translation control does not exist. GPS is the one what comes into consideration, but due to some limitations, like low accuracy, long time period between measurements and the need for constant GPS signal, it cannot be used independently as a spatial position reference for translation control, especially in a micro UAV. Also another, satellite independent physical sensors could be used to aid in determining the position, like barometric pressure sensor, which could provide valuable information of altitude, or sonar rangefinder to measure distance to the ground, useful for example when performing landing procedure. An advanced form of vision-based navigation, radar, omnidirectional sonar or lidar sensors also comes into consideration, especially for obstacle avoidance. For experimenting, current device is equipped with optical flux sensor, a device being used in personal computer optical mice. Generally a camera with onboard DSP processor that makes the 2D cross-correlation of two consecutive frames and tries to determine image movement. I have attached an board camera optics to this device to focus on the ground pattern and currently experimenting with it. There is also onboard sonar rangefinder for constant altitude hold. Measurement of these sensors must be corrected according to the orientation.

Since the orientation state comes as output of the orientation observer, we can subtract the gravitational vector from the accelerometer reading and use it for translation prediction. The goal is then to design a fusion algorithm, similarly like for orientation, which would combine the acceleration reading and some of the absolute sensors to estimate the position. As shown in fig. 2.1, the task of translation controller (and position observer) are mostly out of focus of this work. Only a simple relation will be stated here as to be used for *translational motion damping*.

For the translation damper, an essential value is the body acceleration. Since the

magnitude of Earth's gravity is known, we can obtain the values of \dot{u} , \dot{v} and \dot{w} by simply subtracting the g expressed in the body frame from accelerometer reading. The g can be translated into body frame coordinates by quaternion vector rotation,

$$\dot{\mathbf{V}} = \mathbf{acc} + \mathbf{Q} \mathbf{g} \mathbf{Q}^{-1} \quad (3.71)$$

The result, however, relies on an accurate orientation estimation, Q . Now, since \dot{V} is known, comes into consideration to use it as an aid for the orientation observer, because the orientation observer needs only the \mathbf{g} part of accelerometer reading in body frame, which equals $g = \dot{\mathbf{V}} - \mathbf{acc}$. Subtracting the \dot{V} vector from accelerometer reading would permanently solve *The Accelerometer paradox*. Nevertheless, the \dot{V} is again computed using the orientation estimation Q . Such algebraic loop would thus unavoidably result in system destabilisation. The solution for this problem might be found in considering the translational model (damping by aerodynamic forces), and joining the *orientation* and *position* observer subsystems together, into a sort of a basic *inertial navigation system*.

Chapter 4

Identification, implementation, experiments and results

The last chapter will attempt to join the theoretical and construction branches of the project. The model constants will be identified, previously designed control structure will be synthesized and numerical simulations will be performed. Finally, the controller will be tested on the real quadrotor.

4.1 Identification

The function sample of the quadrotor underwent extensive measurements and experiments to determine the constants required by the numerical model. The main procedures will be generally described here.

4.1.1 Inertial Parameters

For the sake of simplifying the identification process and yet seeking a good approximation of the parameters, the quadrotor's inertial structure was regarded as two perpendicular rods corresponding to the arms and X, Y axes of the aircraft, with one point-mass for the rotor at each edge, distant l_a from its rotation axis, as depicted in figure 4.1.

All the aircraft's mass excluding the rotors' is assumed to be homogeneously distributed inside the sphere of radius R , centered in the origin of the axes. Knowing that the moment of inertia of a solid sphere around an axis σ is given by ${}^s I_\sigma = \frac{2}{5} m_s R^2$ whereas for a point-mass, representing the actuator, distant l_a from the rotation axis is given by ${}^r I_\sigma = m_r l_a^2$, and having the sphere mass as $m_s = m - 4m_r$, the moment of inertia around axes X and Y , due to their symmetry, is then easily calculated as

$$I_x = I_y = {}^2 I_x + {}^4 I_x + {}^s I_x = {}^1 I_y + {}^3 I_y + {}^s I_y = 2 (m_r l_a^2) + \frac{2}{5} m_s R^2 \quad (4.1)$$

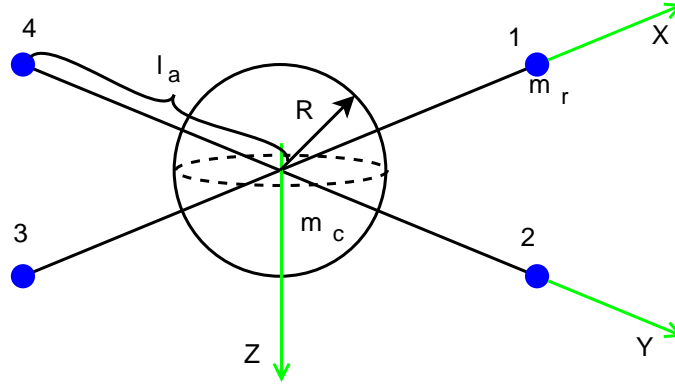


Figure 4.1: Quadrotor's airframe and inertial identification scheme.

Table 4.1: Basic identified model constants of the body dynamics

<i>constant</i>	<i>value</i>	<i>description</i>
m_t	0.694kg	total mass of the body
I_x	$5.87 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$	moment of inertia around X-axis
I_y	$5.87 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$	moment of inertia around Y-axis
I_z	$10.73 \cdot 10^{-3} \text{ kg}\cdot\text{m}^2$	moment of inertia around Z-axis
l_a	0.18m	distance from each motor to CG
t_{ad}	0.5	translation aerodynamic damping
r_{ad}	0	rotation aerodynamic damping

For the Z -axis the four rotors need to be considered, thus

$$I_z = \sum_{j=1}^4 {}^j I_z + {}^s I_z = \sum_{j=1}^4 (m_r l_a^2) + \frac{2}{5} m_s R^2 \quad (4.2)$$

The aerodynamic constants, t_{ad} and r_{ad} are generally hard to identify. Rotational negative feedback, r_{ad} , was decided to be disregarded due to low angular speeds of the quadrotor body. Nevertheless, the translational negative feedback, t_{ad} , is a key for the orientation observer to work in order to handle with *the Accelerometer paradox*. Observations lead to the assigned value. However, these variables needs an additional attention to be properly identified.

4.1.2 Actuator

The actuator constants were identified using various performed experiments with a *single actuator*, assuming that all of them are equal. The angular velocity, ω_j , was available by direct measurement, as being a part of the sensor measurement transmitted to the groundstation.

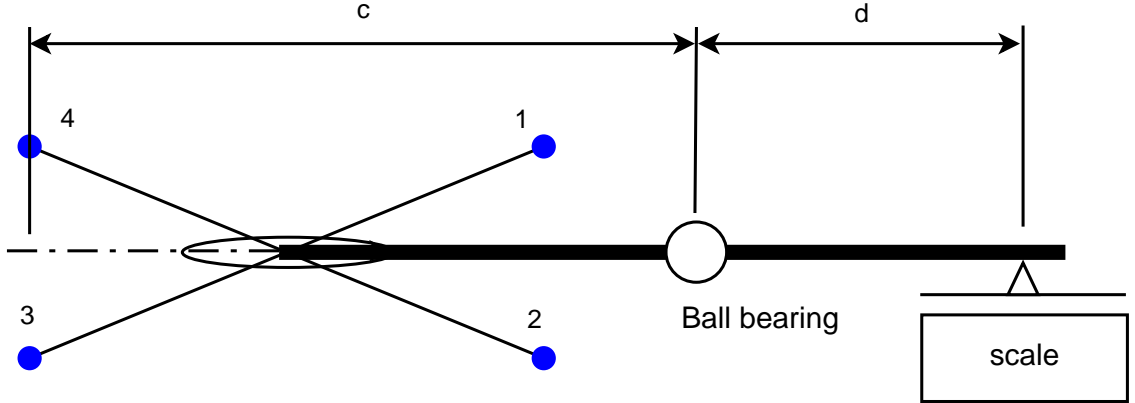


Figure 4.2: Scheme of the thrust output function measurement experiment.

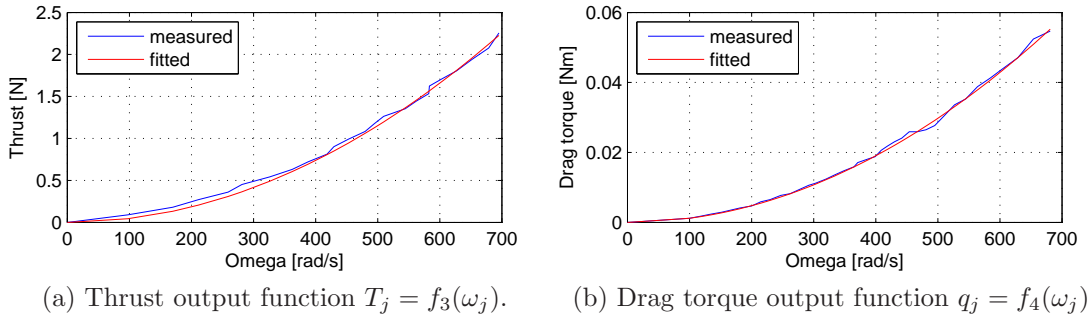


Figure 4.3: Output functions identification of the actuator.

The actuator output, $f_3(\omega)$ and $f_4(\omega)$ functions were identified using a digital scale. The aircraft was mounted on arm of a lever with one degree of freedom, as shown in fig. 4.2, where c is distance from the lever axis to the actuator and d is the scale arm length in corresponding experiment. The lever axis needed to be perpendicular to the Z-axis in order to measure $f_3(\omega)$ and parallel for identifying $f_4(\omega)$. The other arm was put on the scale. The weight variation was observed while driving the angular speed ω_j of a single actuator. The mass difference vector \mathbf{v} was then scaled, for $f_3(\omega) = \frac{g\mathbf{c}\mathbf{v}}{d}$ and for $f_4(\omega) = g\mathbf{d}\mathbf{v}$. The results were obtained by polynomial (parabolic) fitting and is shown in fig. 4.3. Operating point, u_0 and ω_0 were then easily computed as parameters where the total thrust equals the quadrotor mass times Earth's acceleration. Note that all actuators are assumed to be equal.

The constant describing the DC gain of the actuator, K , was obtained from the measurement of the DC characteristics of the actuator, either from the stress-free motor response and then verified on the one of motor with attached propeller shown in fig. 4.4. The stress-free motor characteristic can be well approximated by a linear function $\omega_j = Ku_j$, hence it verifies the assumption in deriving the actuator model. The bearing damping, B , was found difficult to identify as the bearing torque is very small in com-

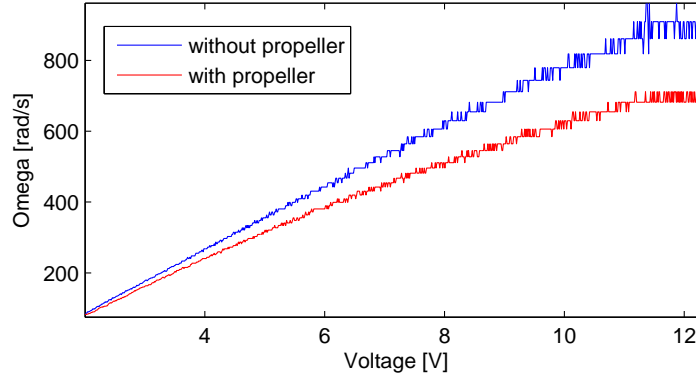
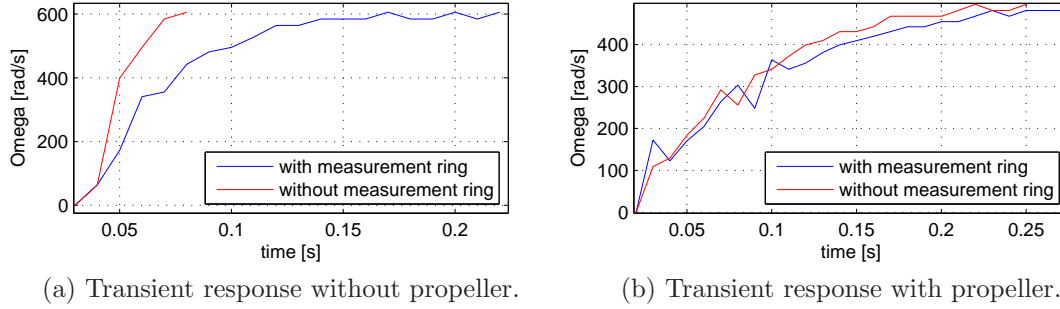


Figure 4.4: DC characteristics of the actuator.



(a) Transient response without propeller.

(b) Transient response with propeller.

Figure 4.5: Identification of the rotor moment of inertia.

parison with the other acting torques. Thus, this constant will be disregarded with the expectation that the contribution of bearing damping is included in the combination of identified K , G_1 and G_2 constants.

A special procedure was performed to identify the rotor moment of inertia using the measurements either of the stress-free motor transient response and of the one of motor with attached propeller. Rotor moment of inertia, I_r , was estimated from change of the transient response of the actuator with mounted reference measurement ring. Using known parameters of the ring, i.e. mass m_g , inner radius d_i and outer radius d_o , the ring moment of inertia is given as $I_g = \frac{m_g}{2} (d_i^2 + d_o^2)$. The rotor moment of inertia is then in relation

$$I_r \frac{1}{\tau_a} = (I_r + I_g) \frac{1}{\tau_b} \quad (4.3)$$

where τ_a and τ_b are time constants of transient response without and with measurement ring, respectively. Note that separate moments of inertia for the propeller and for the motor can be measured in this way and then used to verify the results and improve the precision of the identification process. Step responses are shown in 4.5.

The constants regarding the dynamics, G_1 and G_2 can be determined from the stress-

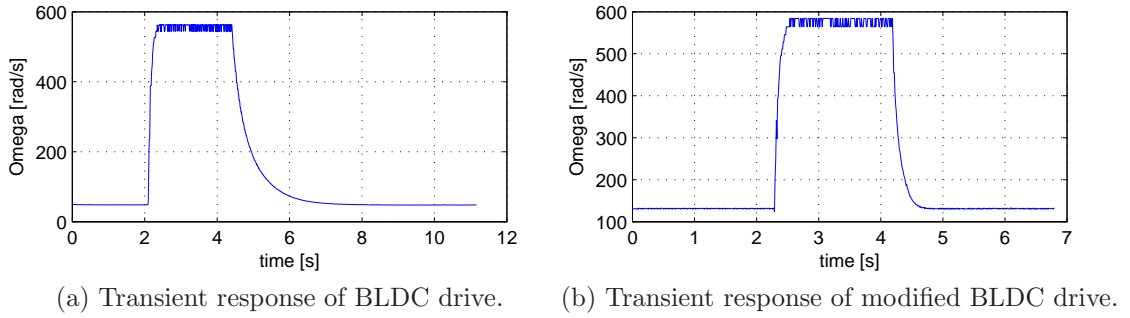


Figure 4.6: Identification of the actuator dynamics.

Table 4.2: Basic actuator model identified constants

<i>constant</i>	<i>value</i>	<i>description</i>
ω_0	608rad/s	angular velocity of the operating point
u_0	9.5V	voltage of the operating point
I_r	$2 \cdot 10^{-4}\text{kg}\cdot\text{m}^2$	total moment of inertia of the actuator
$f_3(\omega_0)$	1.7024N	thrust output of the operating point
$f_4(\omega_0)$	0.0431N·m	drag torque of the operating point
K	0.015319V·s/rad	linear gain of the actuator
B	0N·m·s/rad	bearing damping of the actuator
H	0.000345^2	thrust output non-linearity gain
F	0.002146^2	drag torque output non-linearity gain
k	0.235	linearised actuator constant: sensitivity
l	0.0036	linearised actuator constant: damping
m	0.0002	linearised actuator constant: thrust change
n	0.0063	linearised actuator constant: drag torque change

free rotor transient characteristics. It can promptly be seen that the rotor seems to have a faster response when increasing than when decreasing its speed. This non-linearity is an intrinsic characteristic of the BLDC motor drive and, despite the efforts in trying to redesign the PWM control strategy, the respective first-order time-constants could not be made to be exactly equal. Note that when using the stress-free response for identification, not only the non-linear damping is removed but also the moment of inertia must be changed in the equations, as the propeller is not present. However, to compensate the incidental and disregarded non-linear influences, the values of $G1$ and $G2$ were afterwards fine tuned in the non-linear model simulation to fit the stressed response of the real actuator with propeller around the operating point as close as possible. The gain constant, K , can be also fine tuned in that way. Overall step responses with attached propeller are shown in 4.6.

Table 4.3: LQ-optimal feedback law for decoupled dynamics

	$\Delta\omega_j$	Ω_j	E_j	$\int_0^t E_j(\tau)d\tau$
K_{xy}	0.0386	5.3000	31.9117	-1.7321
K_z	-0.0144	4.0177	31.8421	-1.7321
K_{xy}'	0.0372	4.8363	25.0955	-54.7723
K_z'	-0.0149	3.9442	23.0665	-54.7723

4.2 Implementation and simulation experiments

4.2.1 Rotation controller

Hereby, the complete rotation controller of the main structure from fig. 3.21, as derived in the previous chapter, will be synthesized. At first, the LQ-optimal state feedback controller \mathbf{K} will be tuned for the decoupled axial dynamics. Then implemented and tested in numerical simulations to control orientation of a full 6DOF quadrotor model, with the eigenaxis algorithm coupling. The proposed controller extensions and compensators will be treated. Please note that the orientation observer is disconsidered in here, hence assumed that the true orientation state of the quadrotor is known.

4.2.1.1 LQ-optimal control synthesis

The LQ-optimal controller of the decoupled axial dynamic systems, described in (

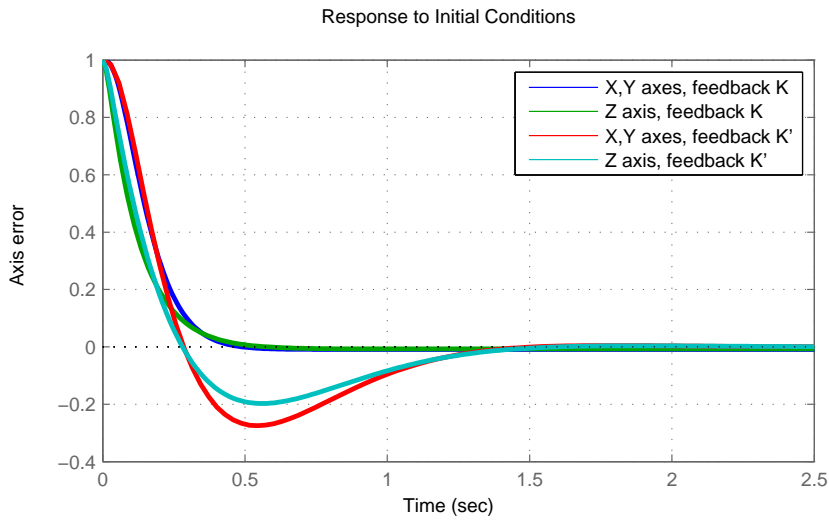


Figure 4.7: Decoupled axial dynamics.

4.2.1.2 Coupling with the eigenaxis algorithm

The eigenaxis axial decoupling computation was used as the error input for the three separate axial controllers. The rotation controller structure was implemented in Simulink with all the specified extensions. The potential advantage of eigenaxis control is shown in fig. 4.8. A step input was applied to the closed-loop system target quaternion, formed out of the quadrotor and rotation controller, namely a command to simultaneously change the tilt around X axis from $\hat{\phi} = -\pi/8$ to $\hat{\phi} = +\pi/8$ and the heading angle from $\hat{\psi} = -\pi/2$ to $\hat{\psi} = +\pi/2$ and the magnitude of $\boldsymbol{\Omega}$ was observed. For comparison, the exact same maneuver was performed replacing the \mathbf{E} vector by the *Euler angles* error. The results have proven that even close to zero orientation states, eigenaxis control offers the same maneuver with less energy claim. The difference rises quickly when moving further from the zero orientation state. Note that the equal rising edge is caused by saturation of actuator inputs.

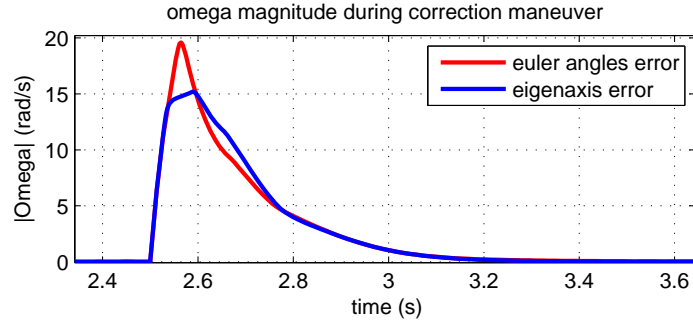


Figure 4.8: Omega magnitude comparison.

4.2.1.3 Gyroscopic compensator

During aggressive maneuvers, the disturbance caused by gyroscopic effects arises. Fig. 4.9 shows a comparison between compensated and uncompensated maneuvers, with applied step in target quaternion representing a change in tilt from $\hat{\theta} = -\pi/4$ to $\hat{\theta} = +\pi/4$ and the heading angle from $\hat{\psi} = -\pi/2$ to $\hat{\psi} = +\pi/2$. The parasitic excitation of E_y is well damped by the compensator, nevertheless the other axes might be negatively affected. After all, the total omega magnitude during correction maneuver was always observed smaller with the compensator, proving that the gyroscopic compensator reduces the action energy claim. The gyroscopic effects will be more evident with different system constants, e.g. with more massive propellers. The compensator constant were iteratively tuned to $P = 2$ and $D = 0$.

4.2.1.4 Overshoot compensator

To avoid the overshoot while preserving the asymptotic tracking behavior, current controller has been augmented with the overshoot compensator according to fig. 3.11. The L_j parameter was iteratively tuned to $L_j = 1$ for all axes. Fig 4.10 compares the controller

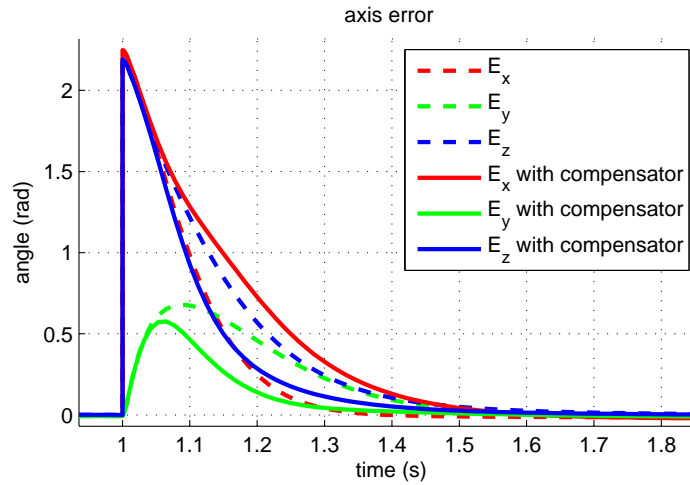


Figure 4.9: Gyroscopic compensator comparison.

performance with and without the compensator on a single rotation axis. At $t = 5$ s, a step of approx. 0.9 rad was applied to the target orientation about that axis. Simultaneously, a steady state error was simulated, so that a parasitic torque was applied to the quadrotor body in the same axis of magnitude 1 Nm at $t = 10$ s, which could reflect putting a certain additional mass on the end of one of the quadrotor arms during the flight. Note that the extended controller compensates the steady-state error at almost same speed, while reducing the overshoot. However, steady-state error compensation performance goes down with angular velocity Ω rising, e.g. when a steady-state error appears during a large maneuver. For illustration, also the computed voltages u_j applied to the actuators is shown in fig. 4.11 to show the steady-state error compensation.

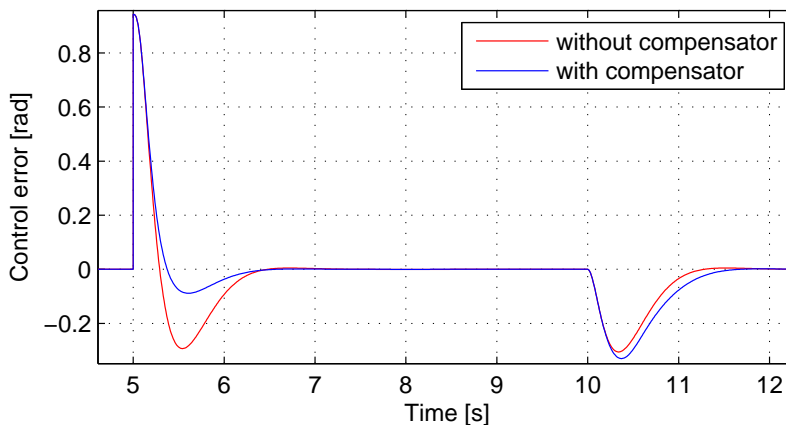


Figure 4.10: Overshoot compensator performance.

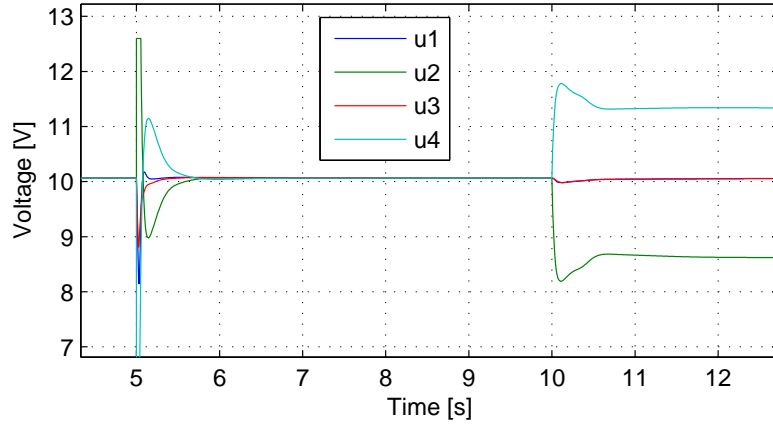


Figure 4.11: Voltages applied to the actuators.

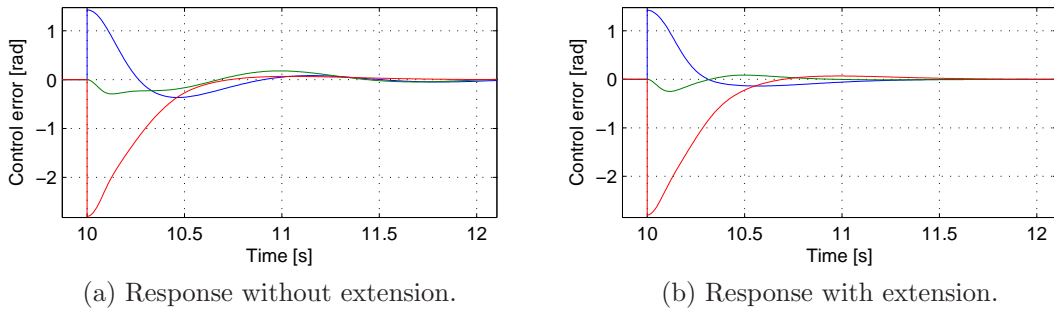


Figure 4.12: Non-linear actuator extension comparison.

4.2.1.5 Actuator non-linear mapping

The actuator non-linear controller mapping extensions, defined in (3.53), (3.54) and (3.55) were implemented to extend the operating range of the controller. The mapping should increase controller performance when acting far from the operating point, ω_0 . Fig. 4.12 shows that apart from correcting the DC gain, the dynamic response is also improved. The DC voltage setpoint for the actuators has been set to approx. $u_0/2$ and the control error was observed. Although the dynamic response is not changed much in the simulations, the improvement on the real system was recorded to be essential, mainly due to errors in motor angular velocity readings and disconsidered effects of the BLDC drive.

4.2.2 Target orientation and thrust computation

Target orientation and thrust computation is a simple mapping relation from the $\hat{\mathbf{a}}$ to target quaternion and thrust vector, with no state variables or dynamic relations. Nevertheless, few plots will be shown to highlight some aspects of the designed controller. The step response from zero input to $\hat{\mathbf{a}} = [4 \quad -2 \quad 0]$ and target heading $\hat{\psi} = \pi/2$ at time

$t = 1$ is shown in fig. 4.13 and fig. 4.14. For illustration, translational aerodynamic damping forces were temporarily removed, otherwise the actual acceleration would exponentially descend to zero. Step in target heading, $\hat{\psi}$, was applied in order to show the ability of independent simultaneous control. The acceleration suffers from steady state errors and, moreover, the quadrotor ascends. This is due to the non-linear thrust gain function of the actuator.

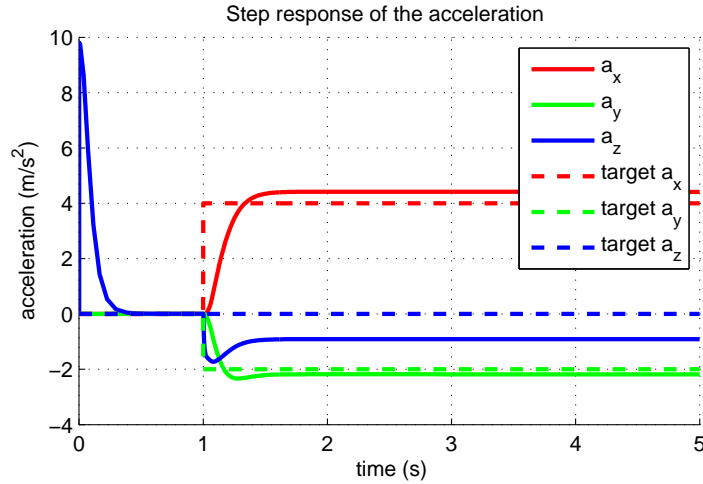


Figure 4.13: Step response of acceleration commands.

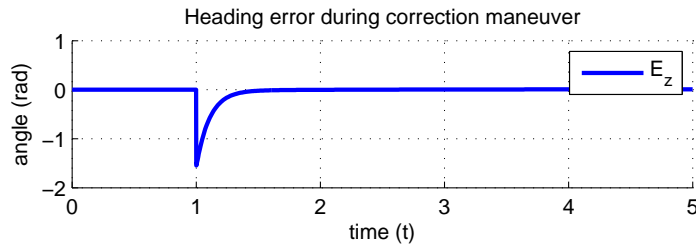


Figure 4.14: Step response of heading command.

The actuator mapping extensions compensates the non-linear thrust gain and therefore the steady-state error is minimized as shown in fig. 4.16. The residual deviation reflects only the error in approximation of (3.11) by a parabolic function. However, the acceleration control is still a kind of open-loop control, as it disregards the aerodynamics forces and the need of additional position sensors arises for the translation controller.

4.2.3 Translation controller

The translation controller takes place in the topmost layer of the proposed structure. Due to high-complexity of the underlying system, both translation damper and position controller were adjusted simultaneously using the rules for general PID tuning. The constants are contained in the table below. Integral control was omitted for this moment.

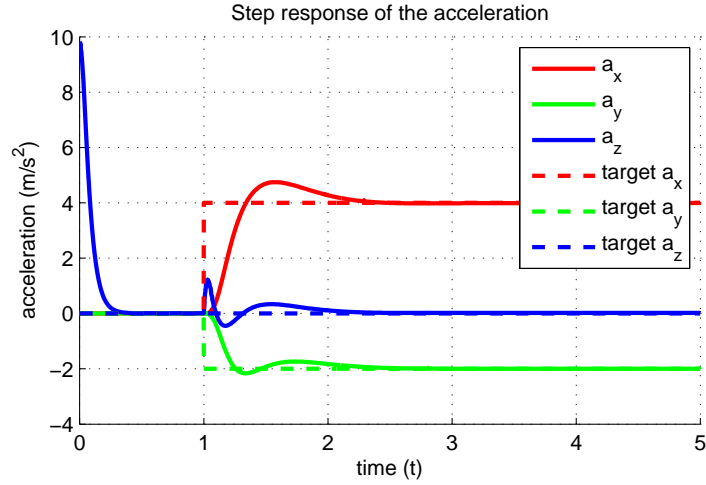


Figure 4.15: Step response of acceleration commands with the non-linear mapping.

Table 4.4: Basic actuator model identified constants

<i>constant</i>	<i>value</i>	<i>description</i>
Kd_{lat}	15	Lateral translation damper: gain constant
Kd_{long}	15	Longitudinal translation damper: gain constant
τ_{lat}	15s	Lateral translation damper: time constant
τ_{long}	6s	Longitudinal translation damper: time constant
Kp_{lat}	3	Lateral position controller: proportional term
Kp_{long}	5	Longitudinal position controller: proportional term
Ki_{lat}	0	Lateral position controller: integral term
Ki_{long}	0	Lateral position controller: integral term

4.2.3.1 Translation damper

As have been allready discussed, best results were encountered when using the NED frame acceleration directly for the damping. However, this variable is difficult to obtain, as a conversion from the onboard accelerometer readings requires additional states (accelerometer integration) which might destabilise the system. As an alternative approach, damping in body-frame was proposed. Fig. 4.17 and fig. 4.17 shows the response of the position command from zero to $\hat{\mathbf{X}} = [10 \ 0 \ 1]$, explained as to go 10 meters north and one meter down. The more evident altitude error, when performing body-frame damping, shown in fig. 4.18, reflect the undamped acceleration, which is generated just by rotating the quadrotor body with nonzero body velocities. Note that body-frame damping result must be coordinate transferred into the NED frame before feeding as part of the $\hat{\mathbf{a}}$ variable. The correct operation of the damper also depends on correct estimation of body acceleration \hat{V} .

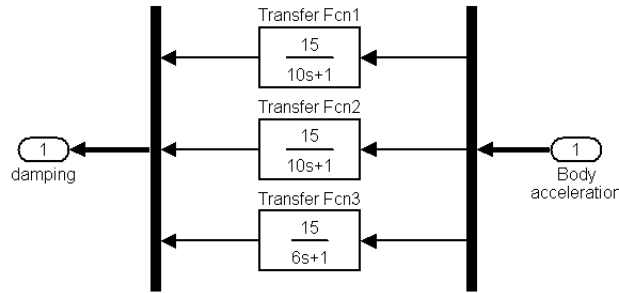


Figure 4.16: Simple translation damper structure.

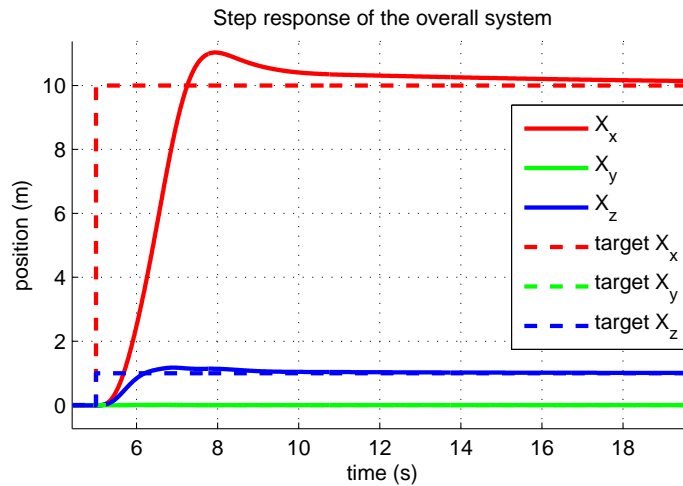


Figure 4.17: Translation controller response with damping in NED frame.

4.2.3.2 Position controller

The position controller needs explicit knowledge of the actual position. The real flight-cappable design should involve cooperation with the upper level structure of task/mission control or at least, characteristics of the position sensors, like GPS, computer vision, etc. In the numerical simulations, exact position was assumed to be known and therefore, proposed PI control can take place. Fig. 4.19 shows the response to various position commands. Cross-influencing between the axes, depending on the command history can be seen in comparison of fig. 4.18 and fig. 4.19. It was shown that a certain overshoot when using PI, or even only P approach is unavoidable, thus calling for some kind of more advanced control approach or for the allready discussed non-linear compensation.

4.2.4 Orientation and position observer

As the design of the proposed orientation observer is highly experimental. A consistent tuning process was not yet created and thus will be ommited. On the contrary to lab-

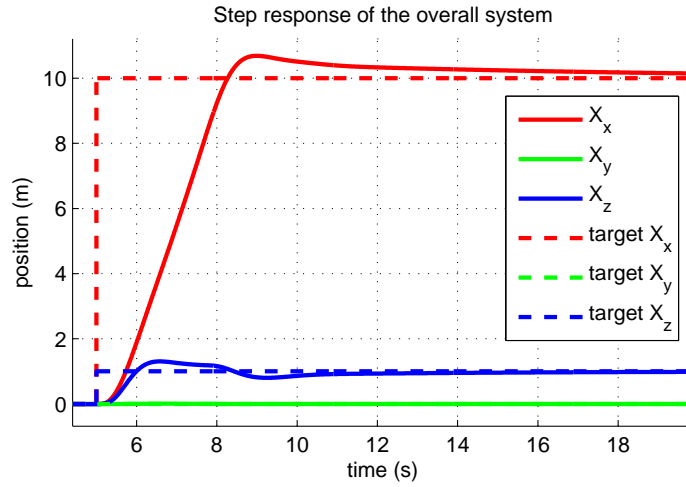


Figure 4.18: Translation controller response with damping in body frame.

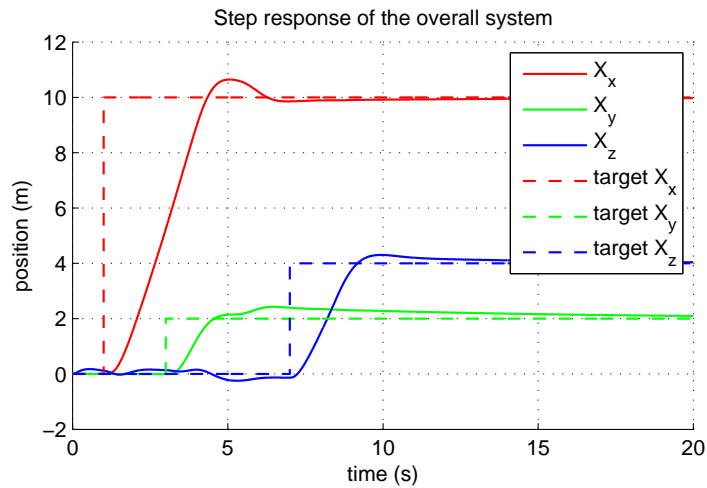


Figure 4.19: Translation controller performance.

oratory experiments, presented simulations assumed both true position and orientation states to be known, as the focus of this work was mainly the controller design. The constant used in the current AHRS subsystem are presented below.

The translation observer was reduced to estimate the body-frame acceleration using orientation estimation and accelerometer reading to (3.71). Similarly like the position controller, the full observer design involves cooperation with the upper level structure of task/mission control and additional sensors, where the two units might closely cooperate.

Table 4.5: Orientation observer parameters

<i>constant</i>	<i>value</i>	<i>description</i>
K_a	5.0	Accelerometer correction weight
K_m	2.5	Magnetometer correction weight
K_b	1.0	Bias estimation correction weight
K_Q	3.3	Quaternion correction weight
τ_α	0.5s	Magnetic inclination lowpass filter time constant
τ_{ac}	2.3s	Accelerometer paradox compensator time constant
K_{ac}	15.0	Accelerometer paradox compensator gain constant

4.3 Laboratory experiments

The designed structured controller was implemented into the software and a lot of real system experiments were performed during the project development. Note that apart from the identification, also model verification and controller tuning involves both simulation and laboratory experiments. All non-linear extensions were also verified on the real system in that way. The translation controller was reduced to translation damper and a human pilot was placed in role of position controller using RC handpad with two joysticks, one to control acceleration in horizontal plane (North-East) and the second for heading and vertical acceleration.



Figure 4.20: First, wired outdoor flight, 10.4.2010.

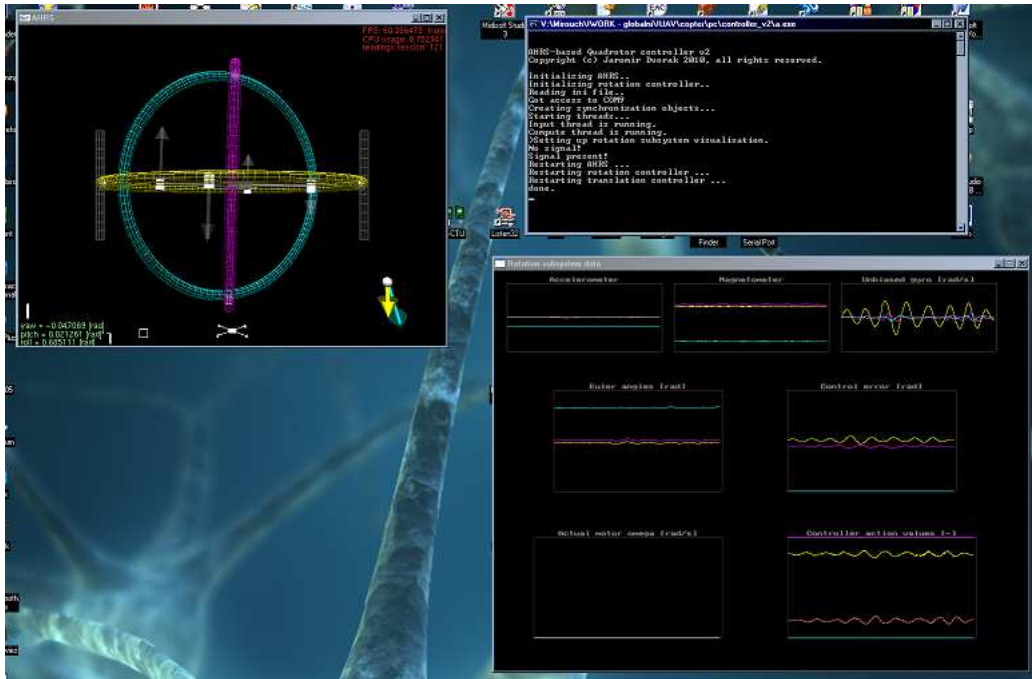


Figure 4.21: Control software running on MS Windows PC.

In the initial phases of the rotation controller verification and tuning, the quadrotor body was mounted on a ball beared lever with one axis perpendicular to one of the quadrotor arms, reducing the system degrees of freedom from six to one. Only two of the four propellers were active during this test. Such experiment was useful to verify the controller applicability and proved the advantages of a state feedback control including the motor angular velocities over the simple PID approach. The PID controller failed to stabilize the system due to improved transient responses of the actuators in comparison with another similar projects, because the motor angular velocities feedback was not present.

Nevertheless, because of the high system complexity and coupling, the flight capable quadrotor controller tuning could not be verified using the lever experiment, mainly due to the lever axis displacement from the center of gravity. Therefore the plots presented below were recorded during the real flight. However, a mechanical tilt and heading platform could be constructed and used for rotation controller tuning and education, as the translational degrees of freedom would be locked.

Fig. 4.22 shows the eigenaxis control error during the real flight. Note that I-control was turned off for the vertical axis during this experiment, hence there is a steady-state error present in the E_z . However, fig. 4.22 is not very illustrational, as the pilot commands are not known. Generally, showing the actual (current) and the target (setpoint) values is not as straightforward as when using the common Euler angles controller, because of the nature of the eigenaxis algorithm. To show the orientation tracking, the actual and target quaternion elements are shown in 4.23.

The deviation in quaternion elements in steady state is not greater than 0.01, which

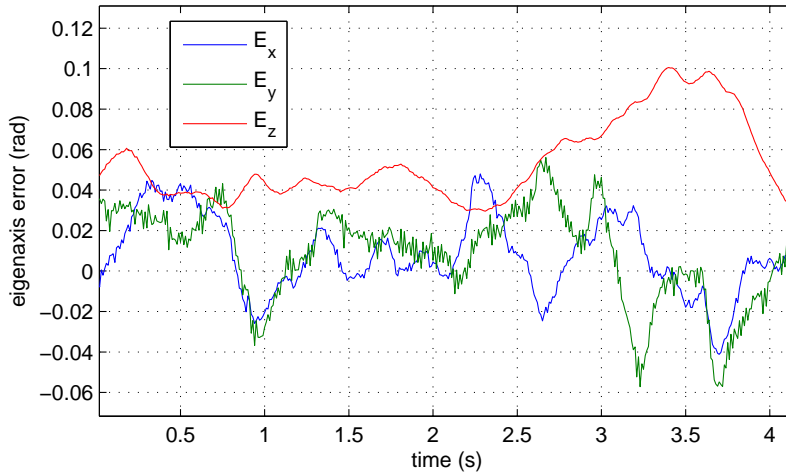


Figure 4.22: Eigenaxis error during the flight.

matches less than one degree orientation error angle magnitude. For even better illustration, the actual state quaternion was converted back to actual heading and expected actual acceleration. Fig. 4.24 and fig. 4.25 shows how well the designed controller tracks the target values during real flight. I-control was turned off during this experiment, using the K feedback. Therefore a slight steady-state errors were met due to slow asymptotical tracking of K feedback and open-loop thrust control. Please note that high frequency noise is caused mainly by error in orientation estimation and the *actual acceleration* in fig. 4.24 does not reflect the real acceleration in body frame $\dot{\mathbf{V}}$, but is an estimated, expected acceleration computed from the current orientation state.

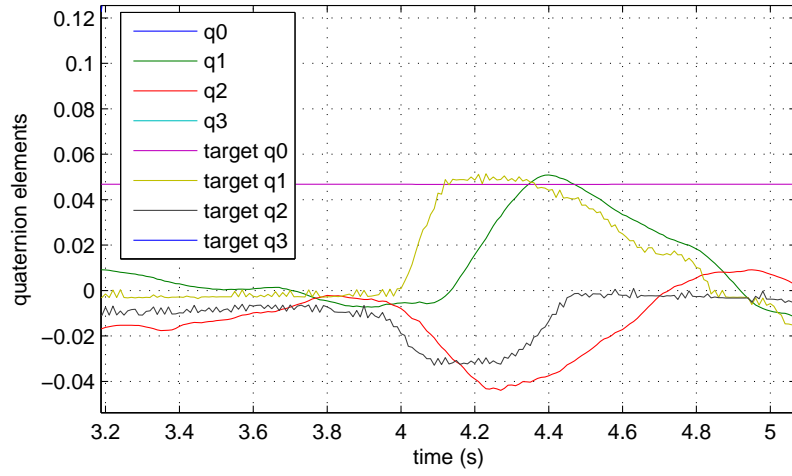


Figure 4.23: Quaternion elements tracking and error.

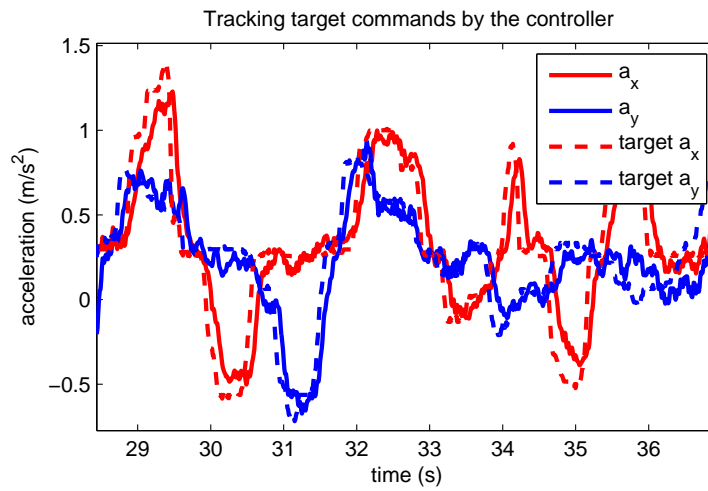


Figure 4.24: Acceleration commands tracking and error.

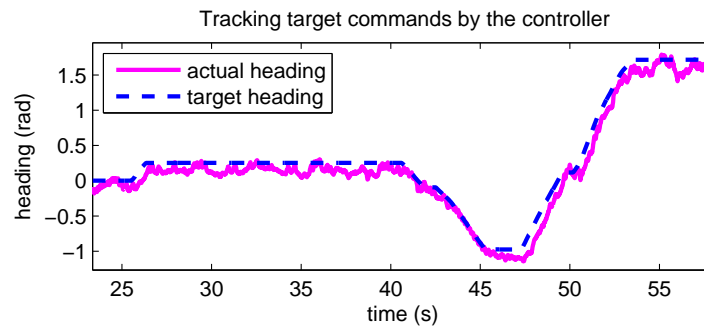


Figure 4.25: Heading command tracking and error.

Chapter 5

Conclusion

This work presented a complete strategy on how to design an advanced, complete structured flight controller based on inertial sensors for a quadrotor using an alternative control methods, while trying to document other related aspects of the overall project. At first, a brief summary of the vehicle construction and programming was presented, enumerating the special features, like real-time control over the wireless link and BLDC motor RPM measurement. Then the full, 6DOF non-linear quadrotor model was derived, trying to include most of the known physical influences, together with quadrotor-specific non-linear actuator model. The model uses quaternions as orientation state representation. The quaternion formulation allowed simple transformations from multiple coordinate systems and does not suffer from any singularities in the rotating space. Main steps of the model constants identification were documented and all constants for the current device are presented. The model was analyzed and a suitable control structure was introduced. Therefore, an alternative method was suggested for orientation control, using the eigenaxis extraction of error quaternion. It was shown that decoupled axial control is possible in an entire orientational space using eigenaxis error, without any singularities. The axis-decoupled LQ-optimal controller was used to drive the actuator voltages. Designed rotation controller was finally augmented with various compensators and non-linear enhancements to improve its performance. The main aspects of the controller were examined and verified in simulations. Design of a translation controller, divided into position controller and translation damper was proposed and the main issues when using the inertial sensors to aid translation control were examined. Finally, the problems of orientation estimation using the inertial sensors were discussed and a highly-experimental quadrotor-specific orientation observer (AHRS) was introduced.

Rotation controller, translation damper and rotation observer were implemented as well in Simulink as in the C code for real-time quadrotor control. Currently, the device is ready to take-off, equipped with all necessary peripherals, ready to experiment with and make measurements on. The controller was successfully verified on a real quadrotor. Test flights have proven that the quadrotor is very stable, able to hover in the air while requiring only slight commands from the pilot to correct the position drift. However, the system performance relies on the onboard AHRS orientation estimation accuracy, which is still an aspect to be improved. The problem of *the Accelerometer Paradox* also needs additional attention together with the convergence of the experimental AHRS algorithm

validation. The AHRS model is not included in the simulations, hence all the presented simulation results are assuming that true orientation state is known.

5.1 Future Works

- Examination of flipping and another large maneuver capabilities of the real quadrotor with synthesized controller. Although the flip-over maneuver was successfully simulated with current controller, it has not yet been tested on the real quadrotor.
- Examination of the model and controller characteristics in frequency domain. The entire design was done in time-domain to avoid the linearisation and allowing to use various non-linear extensions. Nevertheless, the frequency responses should also be treated, for example to exclude the possible flexible modes excitation of the quadrotor body.
- Better formalization and convergency validation of the presented experimental AHRS algorithm. Verification of closed-loop system stability with the orientation observer. This algorithm is used in the current real quadrotor with empirically tuned constants and performs very well together with the controller, however, a tuning technique should be derived.
- More investigations in order to improve the problems regarding *the Accelerometer Paradox*. The problems of orientation estimation and translation damping could be solved by creating a complete INS system.
- Examination of usability of the model-independent eigenaxis controller, described in [7].
- Realization of a position observer with advanced data fusion algorithm from multiple sensors.
- Building of a new quadrotor sample version, with onboard controller.

Bibliography

- [1] P.Pounds, R.Mahony, J.Gresham, P.Corke, J.Roberts: "Towards dynamically-favourable quad-rotor aerial robots", *Australian Conference on Robotics and Automation*, page 10, 2004
- [2] M.J.Stepaniak: "A Quadrotor Sensor Platform", PhD thesis, Ohio University, November 2008
- [3] C.Balas: "Modelling and linear control of a quadrotor", Cranfield University, MSc Thesis, 2006-2007
- [4] S.Bouabdallah, P.Murrieri, R.Siegwart, "Design and Control of an Indoor Micro Quadrotor", Swiss Federal Institute of Technology
- [5] E.Stingu, F.Lewis, "Design and Implementation of a Structured Flight Controller for a 6DoF Quadrotor Using Quaternions", *17th Mediterranean Conference on Control and Automation*, Makedonia Palace, Thessaloniki, Greece, June24-26, 2009
- [6] A.Fleming, P.Sekhavat, I. M. Ross: "Minimum-Time Reorientation of an Asymmetric Rigid Body", *AIAA Guidance, Navigation and Control Conference and Exhibit*, 18-21 August 2008, Honolulu, Hawaii
- [7] J.Lawton, R.W.Beard: "Model Independent Eigenaxis Maneuvers using Quaternion Feedback", Dept. Electrical and Computer Eng. Brigham Young University
- [8] D.P.Han, Q.Weil and Z.X. Li: "Kinematic control of free rigid bodies using dual quaternions", *International Journal of Automation and Computing*, 2008, 5(3), 319-324
- [9] K.D.Bilimoria, B.Wie: "Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft", *Journal of Guidance, Control and Dynamics*, Vol.16, No.3, May-June 1993
- [10] Xiaoying Kong: "Inertial Navigation System Algorithms for Low Cost IMU", Dept. of Mechanical and Mechatronic Engineering, The University of Sydney, August 27, 2000
- [11] J.A.Rios, E.White: "Fusion Filter Algorithm Enhancements For a MEMS GPS/IMU", Crossbow Technology, Inc.
- [12] Itzhack Y.Bar-Itzhack: "New Method for Extracting the Quaternion from a Rotation Matrix", *Journal of Guidance, Control and Dynamics* Vol.18, No.4, 1995

- [13] G.Welch and G.Bishop: "An Introduction to the Kalman Filter", Department of Computer Science, University of North Carolina at Chapel Hill, July 24, 2006
- [14] robotika.cz: "Měření rychlosti", <http://robotika.cz/guide/filtering/en>
- [15] wikipedia: "Rotation representation", http://en.wikipedia.org/wiki/Rotation_representation
- [16] L.Vicci: "Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation", Department of Computer Science, University of North Carolina at Chapel Hill, July 24,2006
- [17] Ben H. Cantrell: "Adaptive Low Pass Filter", *US Patent 3,889,108*, June 10,1975
- [18] P.Batista, C.Silvestre, P.Oliveira: "Vector-Based Attitude Filter for Space Navigation", *Intelligent Robot Systems* DOI 10.1007/s10846-010-9528-2
- [19] M.Řezáč and Z.Hurák: "Low-cost inertial estimation unit based on extended Kalman filtering", Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic
- [20] Banos, A. Vidal, A.: "Design of PI+CI Reset Compensators for second order plants", *IEEE International Symposium on Industrial Electronics*, 2007
- [21] Paul G. Savage: "What Do Accelerometers Measure?", Strapdown Associates, Inc. May 8, 2005
- [22] Jay Farrell: "Aided Navigation: GPS with High Rate Sensors", April 2008
- [23] L.Klauske, T.Lorenz, N.Colberg, M.Janke: "DSP-Copter - A Quadrotor Helicopter Controlled by a Digital Signal Processor"
- [24] A. Fitzgibbon, M. Pilu, R. B. Fisher: "Direct Least Square Fitting of Ellipses", *Tern Analysis and Machine Intelligence*, VOL. 21, NO. 5, MAY 1999
- [25] Q.Li, J.G.Griths: "Least Squares Ellipsoid Specific Fitting", Department of Computer Science, University of Hull, Hull, HU67RX, UK
- [26] X.Zhang, L.Gao: "A Novel Auto-calibration Method of the Vector Magnetometer", *The Ninth International Conference on Electronic Measurement Instruments*, ICEMI2009
- [27] Brian C. Barnes: "Win32 API: průvodce vývojáře", *UNIS publishing*, Vol.1 and Vol.2, 1997
- [28] Pavel Herout: "Učebnice jazyka C", *Kopp*, 1994
- [29] Pavel Herout: "Učebnice jazyka C 2.díl", *Kopp*, 1995
- [30] Miroslav Švorek, Karel Richta: "Připojování periferií k PC", *GRADA*, 1996
- [31] Vladimír Váňa: "Mikrokontroléry Atmel AVR: vývojové prostředí", *BEN*, 2003

- [32] Vladimír Váňa: "Mikrokontroléry Atmel AVR: popis procesorů a instrukční soubor", *BEN*, 2003
- [33] Vladimír Váňa: "Mikrokontroléry Atmel AVR: Assembler", *BEN*, 2003
- [34] Vladimír Váňa: "Mikrokontroléry Atmel AVR: Programování v jazyce C", *BEN*, 2006
- [35] ST: "BLDC MOTOR START ROUTINE FOR THE ST72141 MICROCONTROLLER", Application Note, <http://www.st.com>, 2005
- [36] Atmel: "AVR444: Sensorless control of 3-phase brushless DC motors", Application Note, <http://www.atmel.com> 2006
- [37] Atmel: "ATMega168 microcontroller", datasheet, <http://www.atmel.com>
- [38] Atmel: "ATMega644P microcontroller", datasheet, <http://www.atmel.com>
- [39] Avago: "ADNS-2610 optical mouse sensor", datasheet, <http://www.avagotech.com>
- [40] Sharp: "gp2y0a21yk0f IR proximity detector", datasheet, <http://www.sharp-world.com>
- [41] PNI: "MicroMag 3-Axis Magnetic Sensor Module", datasheet, <http://www.pnicorp.com>
- [42] Nordic: "NRF24L01 wireless transceiver", datasheet, <http://www.nordicsemi.com>
- [43] Prolific: "PL-2303 USB TO RS-232 BRIDGE CONTROLLER", datasheet, <http://www.prolific.com>
- [44] L.Zaplatílek: "Řízení bezkartáčových (BLDC) motorů", Bakalářská práce, Univerzita Pardubice, Fakulta elektrotechniky a informatiky, 2009
- [45] PJS: "3D 550 E", datasheet, <http://www.pjs.cz>
- [46] Murata: "Piezoelectric Vibrating Gyroscopes ENC-03M (GYROSTAR)", datasheet, <http://www.murata.com>
- [47] Analog devices: "ADXL335 Accelerometer", datasheet, <http://www.analog.com>
- [48] Autopilot: existing, open-source UAV project, <http://autopilot.sourceforge.net/>
- [49] Mikrokopter: existing, open-source quadrotor project, <http://www.mikrokopter.de>
- [50] Arducopter: existing, open-source quadrotor project, <http://code.google.com/p/arducopter/wiki/ArduCopter>
- [51] Flying Machine Arena: existing quadrotor project, http://www.idsc.ethz.ch/Research_DAndrea/FMA

Appendix A

Complete Simulink model

The implemented Simulink model diagrams will be presented here. Each of the structured blocks is shown. Also, a simplified 2D-structure of the quadrotor model with controller is presented, with one degree of freedom in rotation and two degrees of freedom in translation.

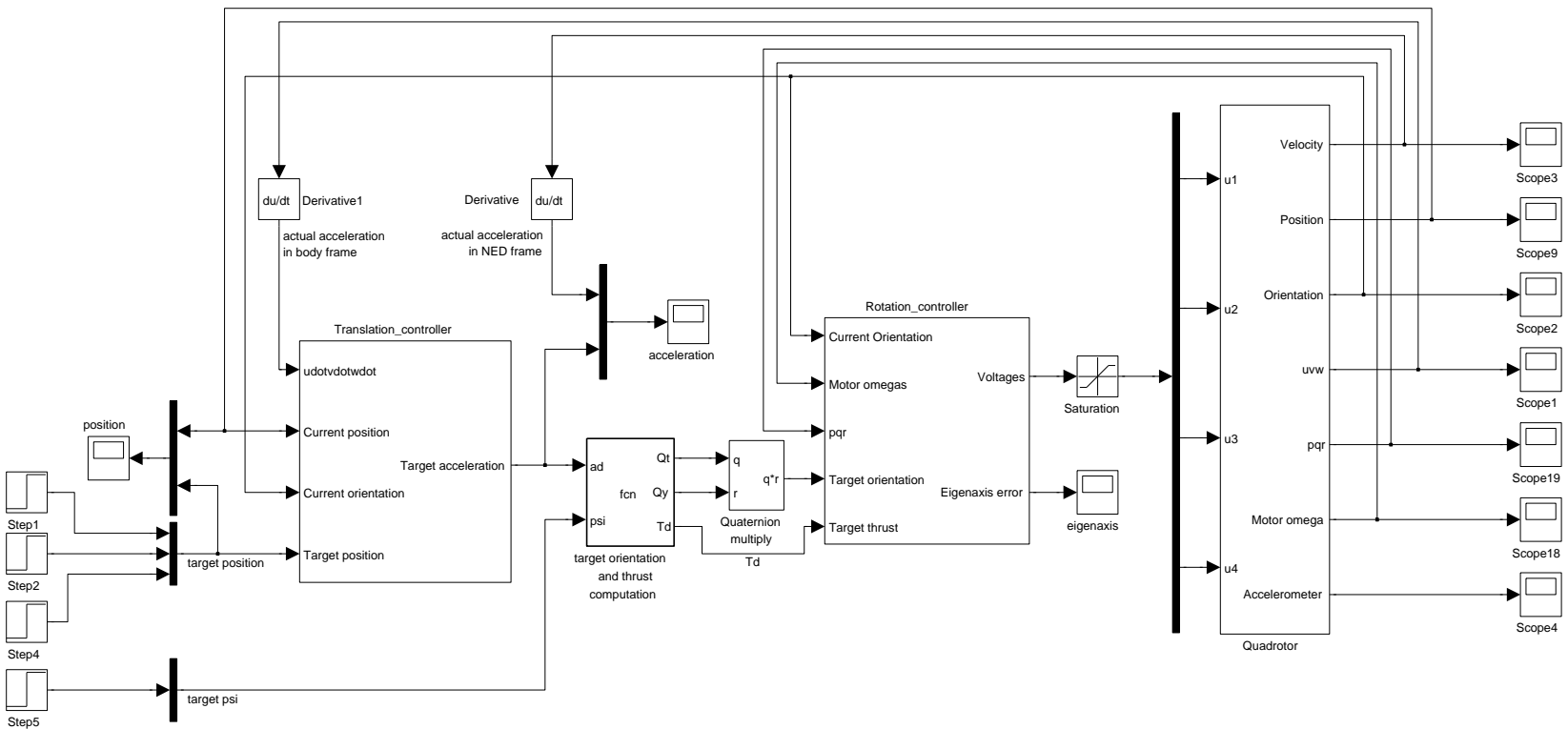
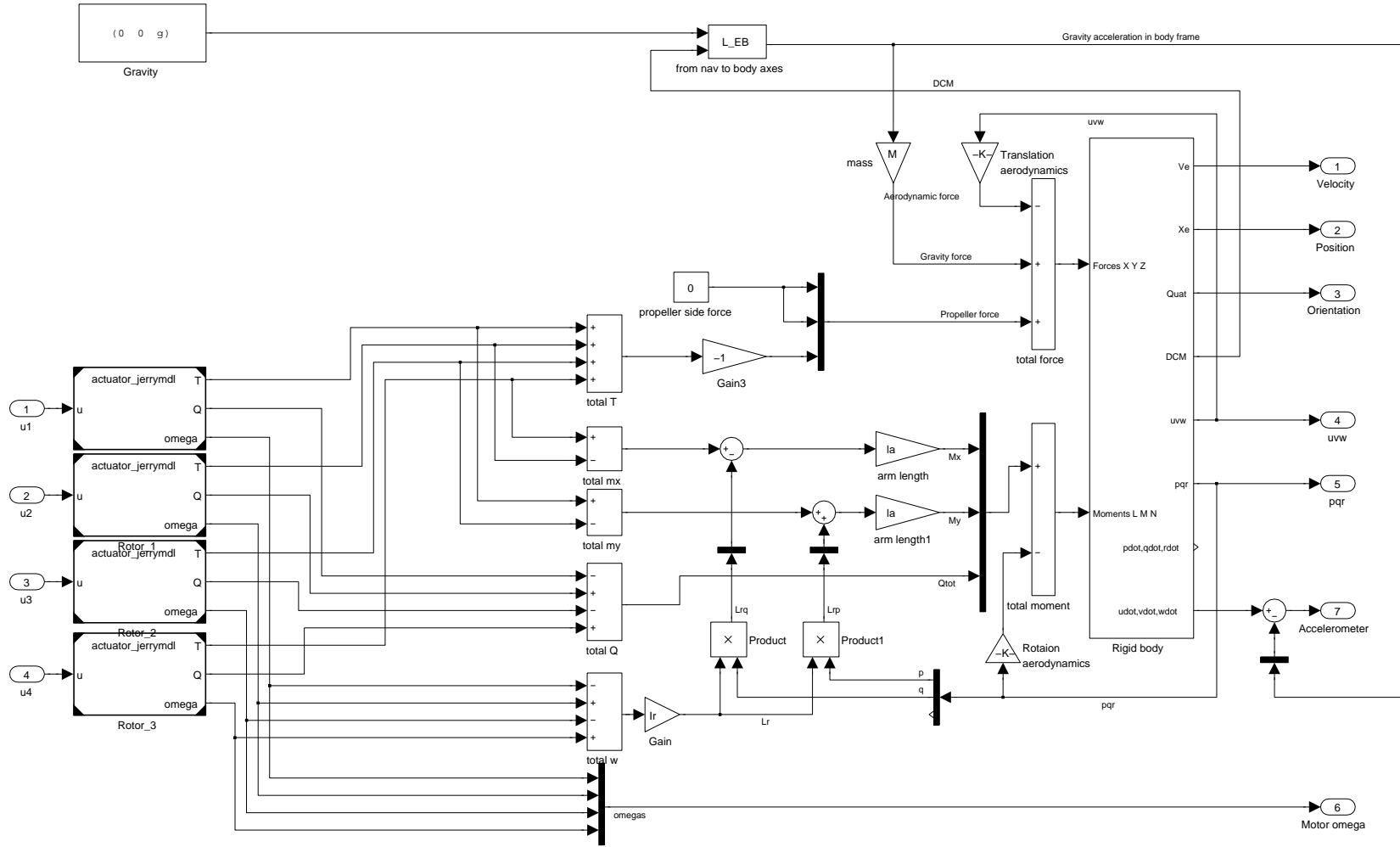


Figure A.1: The overall closed-loop diagram.

Figure A.2: The quadrotor diagram.



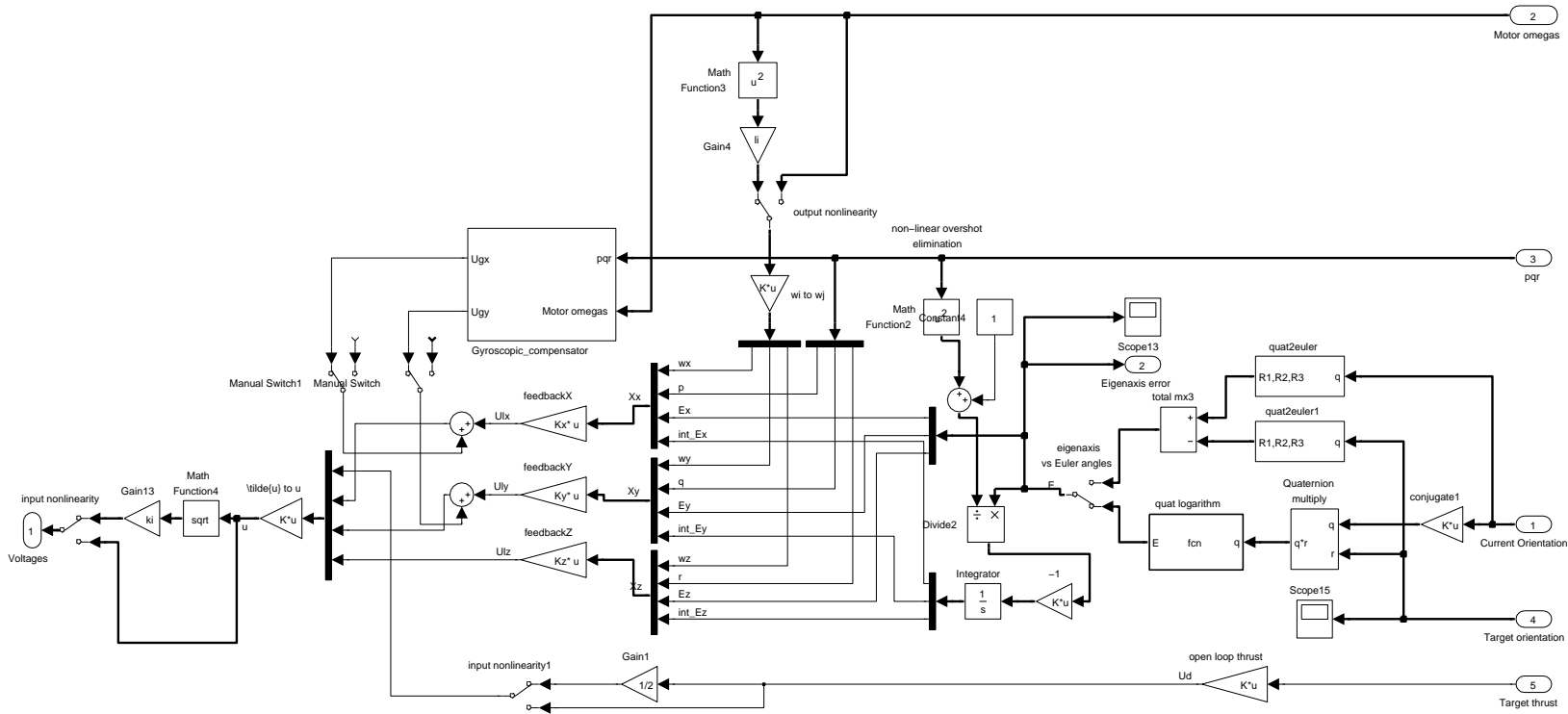


Figure A.3: The rotation controller diagram.

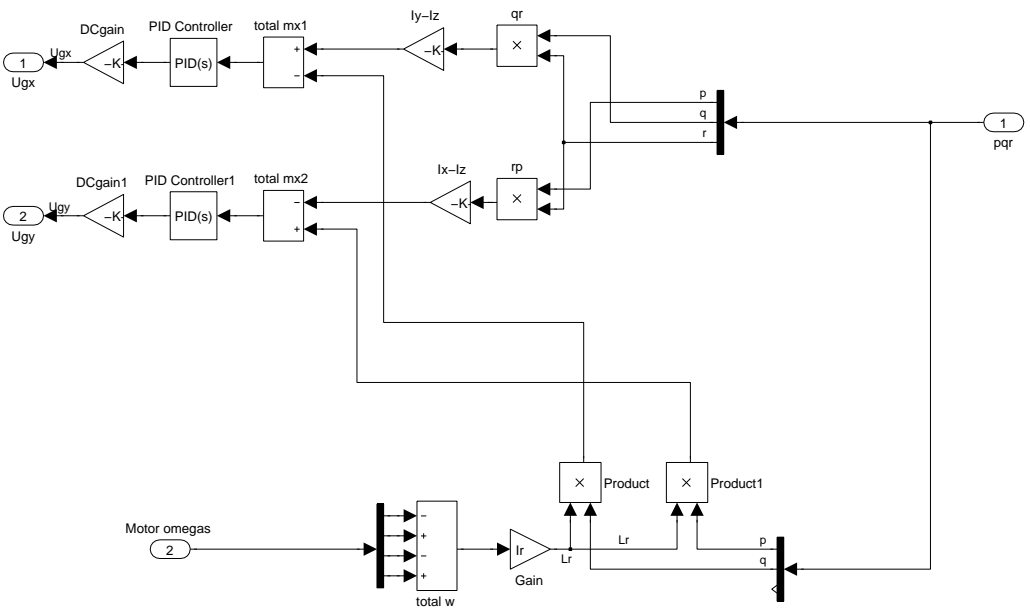


Figure A.4: The gyroscopic compensator diagram.

```

function E = fcn(q)
%#eml

e = q / norm(q);

if(e(1) < 0)
    e = -e;
end

angle = 2*acos(e(1));
size=norm([e(2); e(3); e(4)]);

if(size == 0)
    size = 1;
end

E = -angle * [ e(2); e(3); e(4)] / size;

```

Figure A.5: The quaternion logarithm Matlab code.

```

function [Qt, Qy, Td] = fcn(g, M, ad, psi)
%#eml

a=[ad(1); ad(2); g-ad(3)];
alpha = acos(a(3)/norm(a));
axis = cross(a, [0;0;1]);
size = norm(axis);

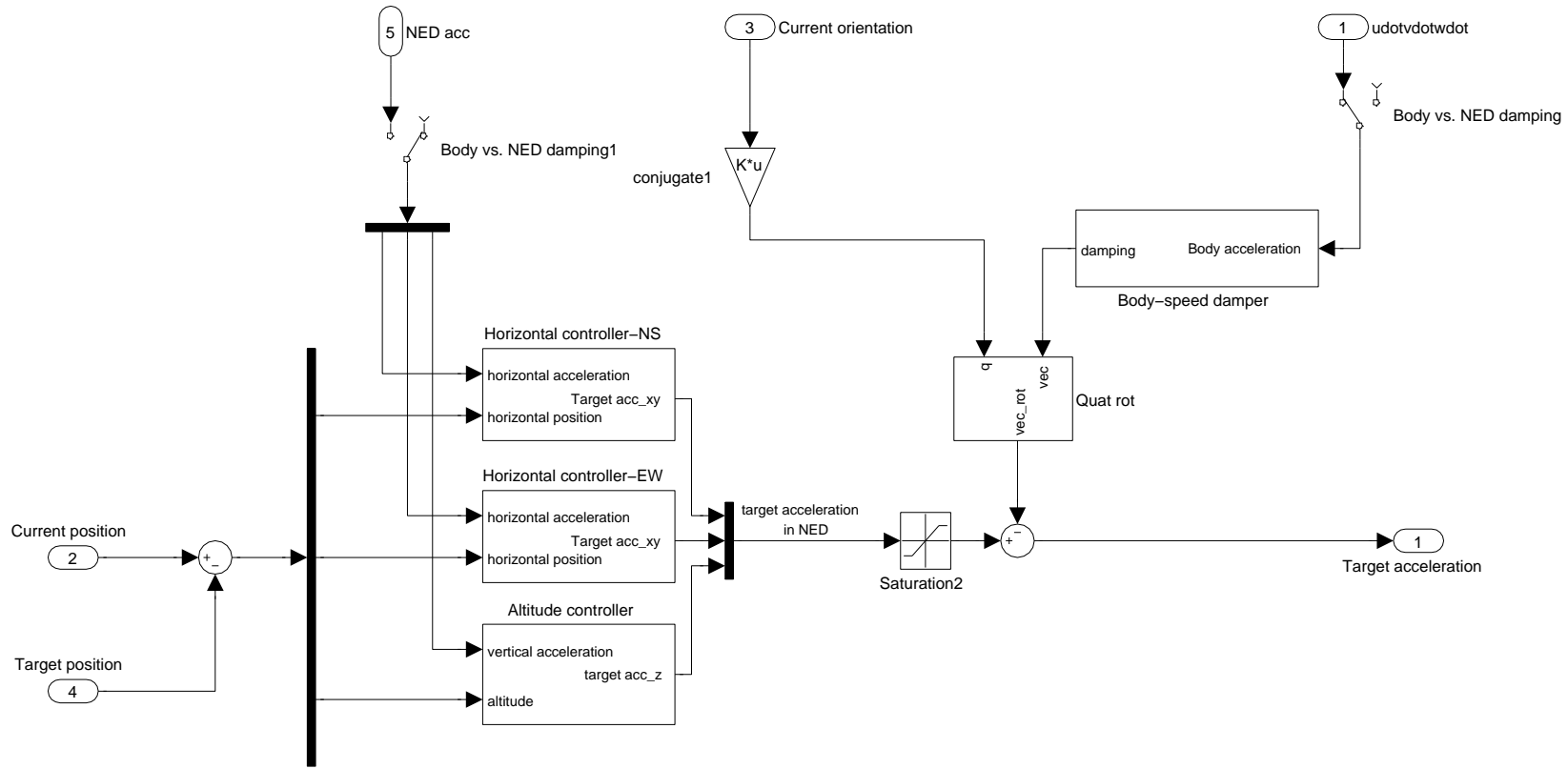
if(size == 0)
    size = 1;
end

Qt = [cos(alpha/2); sin(alpha/2) * axis/size];
Qy = [cos(psi/2); 0; 0; sin(psi/2)];
Td = M * norm(a);

```

Figure A.6: The target orientation and thrust computation Matlab code.

Figure A.7: The translation controller diagram.



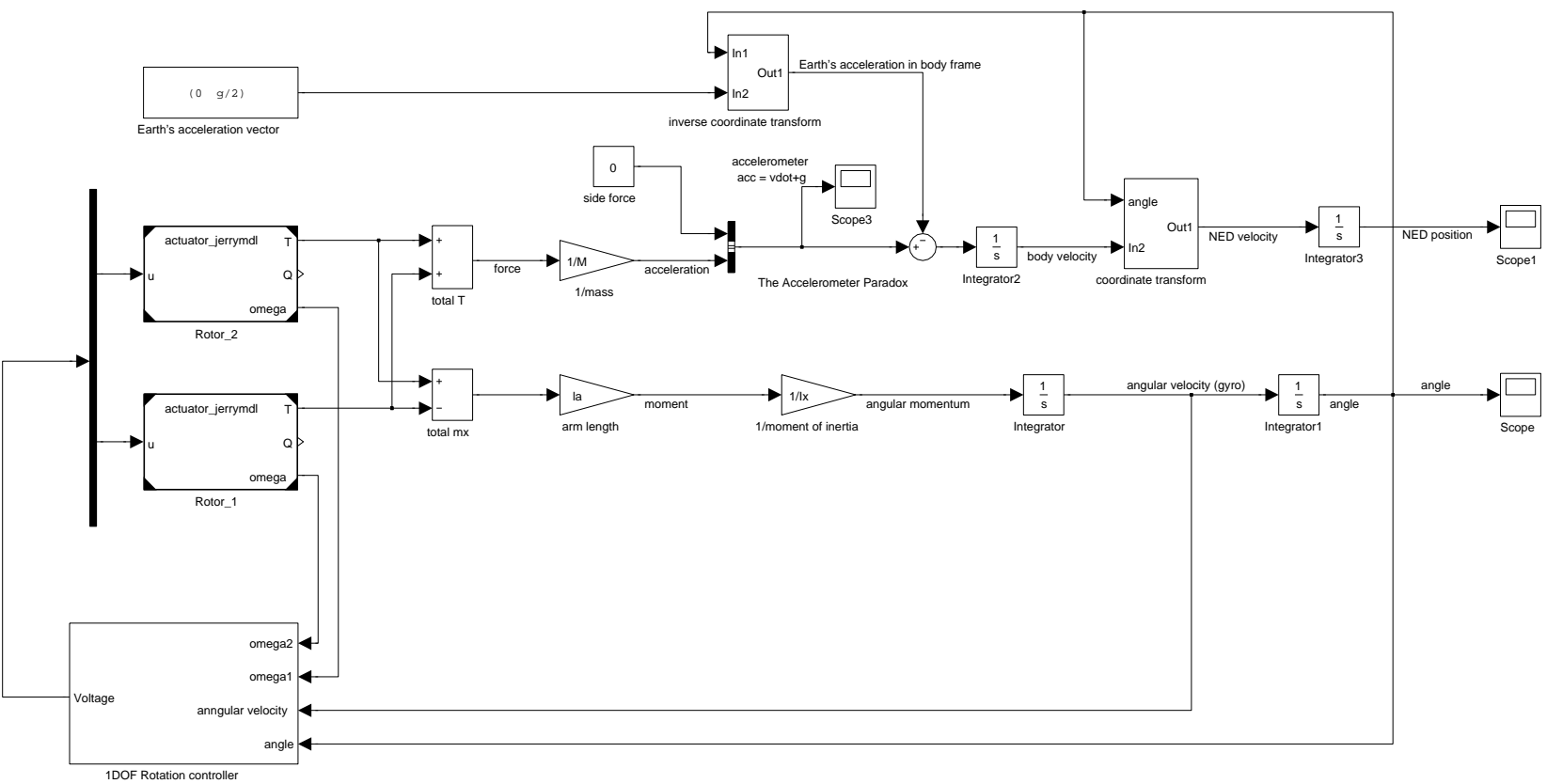


Figure A.8: The simplified 2D-quadrotor model with rotation controller diagram.

Appendix B

Simulation plots

A steps were applied to the target position and heading commands, namely from zero to $\hat{X}_x = 10\text{m}$ at $t = 1\text{s}$, $\hat{X}_y = 2\text{m}$ at $t = 3\text{s}$, $\hat{\psi} = \pi/2\text{rad}$ at $t = 5\text{s}$, $\hat{X}_z = 4\text{m}$ at $t = 7\text{s}$ while the key closed-loop system variables were observed and are plotted below.

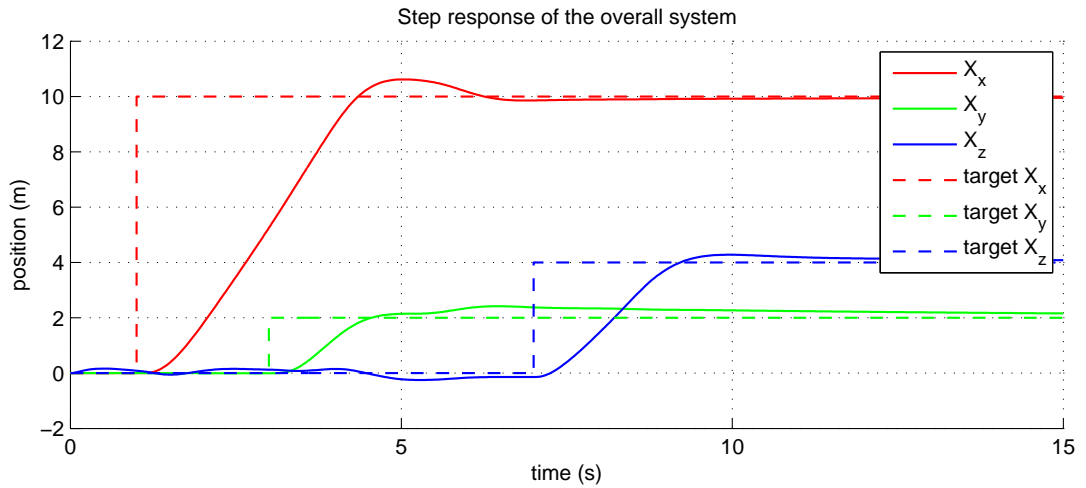


Figure B.1: Position history.

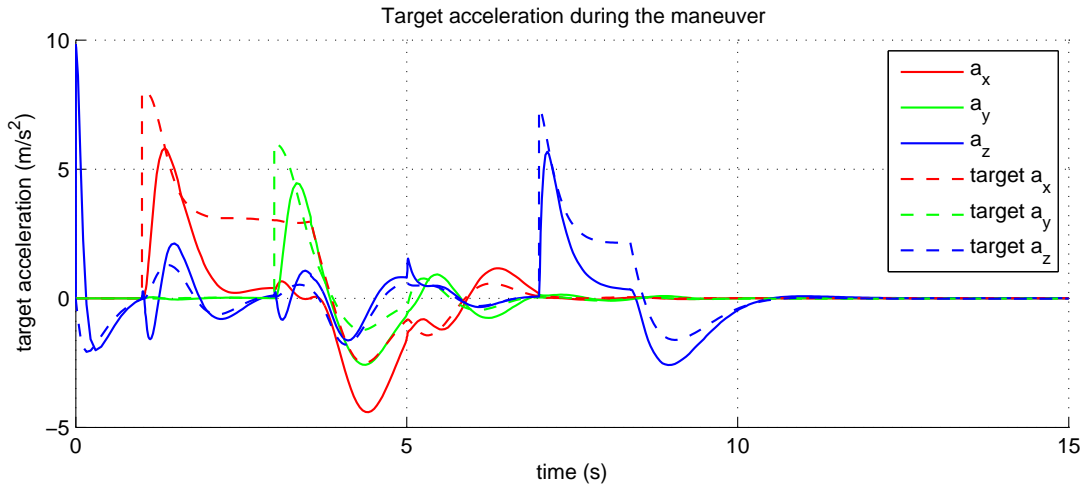


Figure B.2: Target acceleration history.

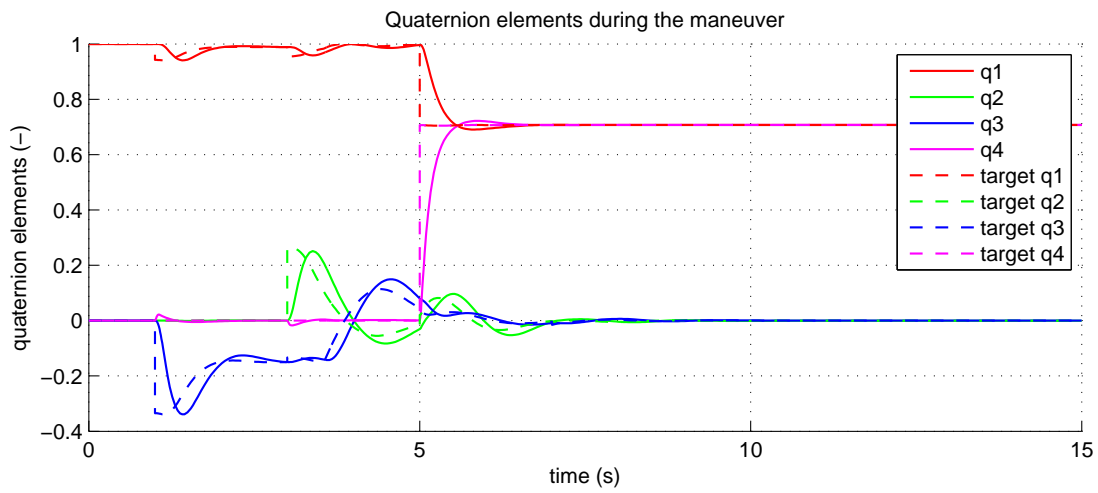


Figure B.3: Quaternion elements history.

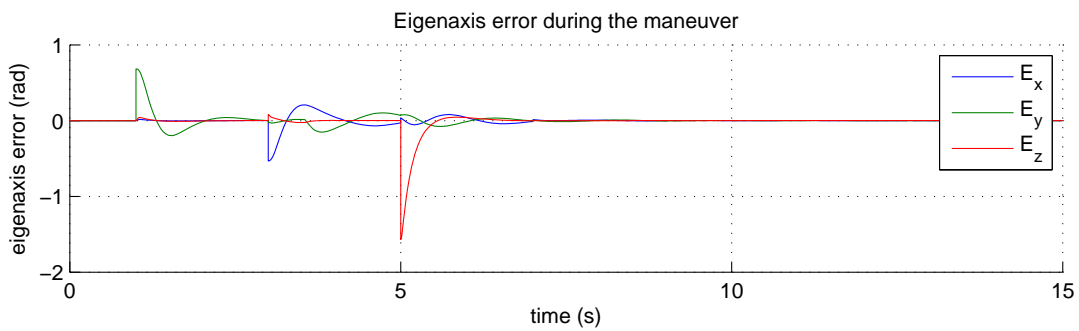


Figure B.4: Eigenaxis error history.

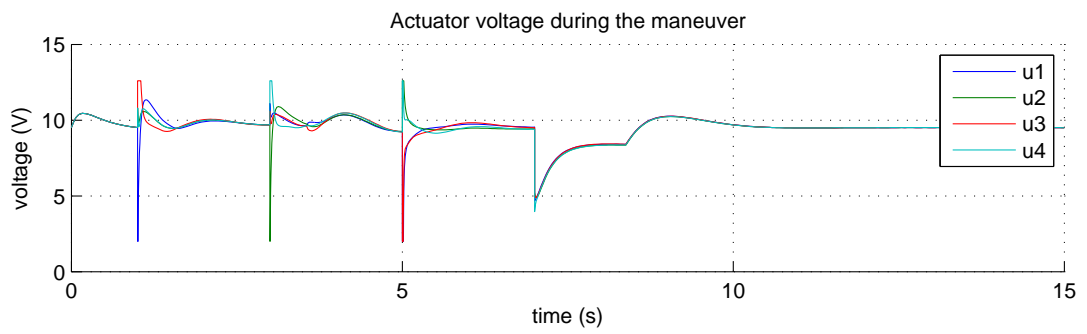


Figure B.5: Actuator voltage history.

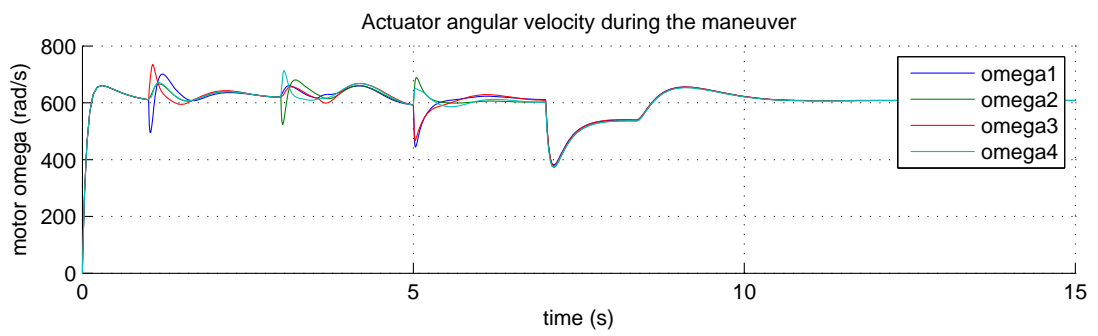


Figure B.6: Actuator angular velocity history.

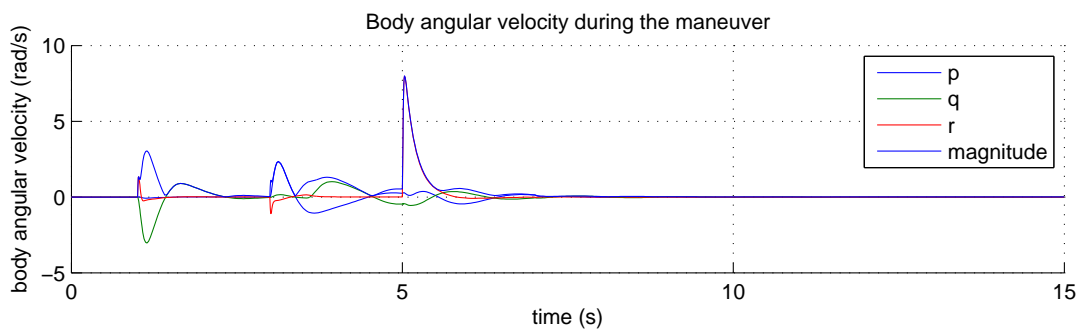


Figure B.7: Body angular velocity history.

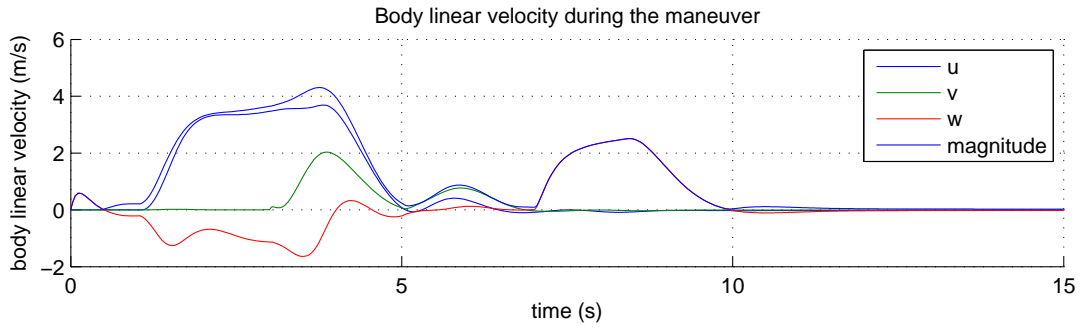


Figure B.8: Body linear velocity history.

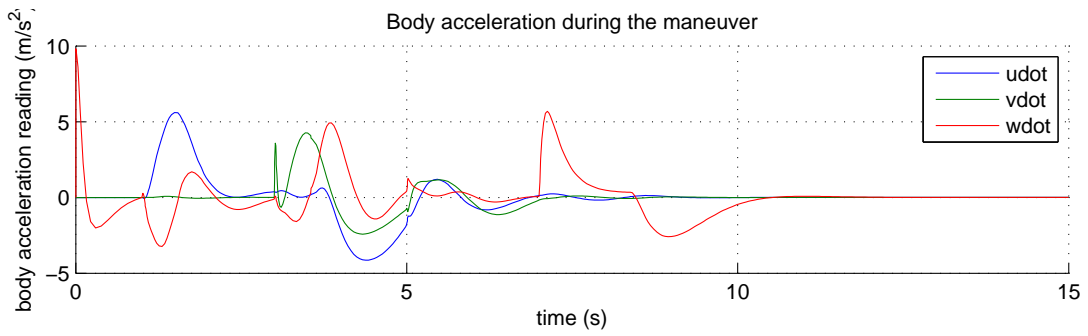


Figure B.9: Body acceleration history.

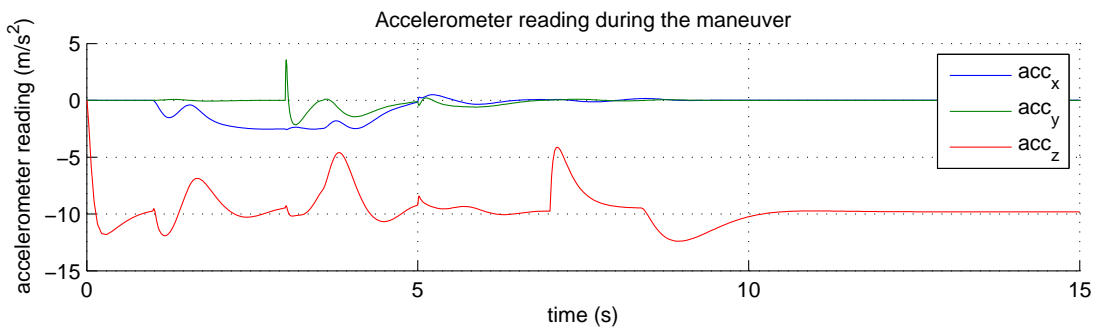


Figure B.10: Accelerometer reading history.

Appendix C

Appended CD

A CD is appended to this work, with the following contents:

- Model: Complete simulink model with all necessary files. To run the simulations, please load the constants first from *quadrotor_par.m* script file and then synthesize the rotation controller by running *ctrlsynth.m*. The complete model is saved in *quadrotor.mdl* and the additional, simplified 2D-model in *quadrotor2D.mdl*.
- Experiments: Two real flights were made and the key system variables were recorded at native sampling frequency $f_s = 100Hz$. In the first experiment, the translation damper was disengaged (acro-mode) whereas in the second, the altitude hold-mode was selected using the altitude reference from onboard rangefinder with the translation damper engaged. The data format is self-explanatory. Each experiment was also recorded using a camcorder.
- Gallery: Few photos and videos from the project development history.