



**FILTERING AND ADAPTIVE CONTROL  
FOR BALANCING A NANOSATELLITE TESTBED**

**RODRIGO CARDOSO DA SILVA**

**MASTER'S DEGREE THESIS IN THE POSTGRADUATE PROGRAM IN  
ELECTRONIC SYSTEMS AND AUTOMATION  
DEPARTMENT OF ELECTRICAL ENGINEERING**

**FACULTY OF TECHNOLOGY**

**UNIVERSITY OF BRASÍLIA**

**UNIVERSITY OF BRASÍLIA  
FACULTY OF TECHNOLOGY  
DEPARTMENT OF ELECTRICAL ENGINEERING**

**FILTERING AND ADAPTIVE CONTROL  
FOR BALANCING A NANOSATELLITE TESTBED**

**RODRIGO CARDOSO DA SILVA**

**THESIS ADVISOR: ASSISTANT PROFESSOR PH.D. RENATO ALVES BORGES, ENE/UNB**

**THESIS OF MASTER'S DEGREE IN ELECTRICAL ENGINEERING**

**PUBLICATION PPGEA.DM - 706/2018  
BRASÍLIA-DF, 31TH OF JULY OF 2018.**

**UNIVERSITY OF BRASÍLIA  
FACULTY OF TECHNOLOGY  
DEPARTMENT OF ELECTRICAL ENGINEERING**

**FILTERING AND ADAPTIVE CONTROL  
FOR BALANCING A NANOSATELLITE TESTBED**

**RODRIGO CARDOSO DA SILVA**

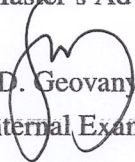
THESIS OF ACADEMIC MASTER'S DEGREE SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING OF THE FACULTY OF TECHNOLOGY OF THE UNIVERSITY OF BRASÍLIA, AS PART OF THE NECESSARY FULLFILMENTS FOR OBTAINING THE DEGREE OF MASTER OF SCIENCE IN ELECTRICAL ENGINEERING.

APPROVED BY:



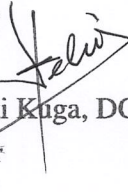
Assistant Professor Ph.D. Renato Alves Borges, ENE/UnB

Master's Advisor



Assistant Professor Ph.D. Geovany Araújo Borges, ENE/UnB

Internal Examiner



Visitant Researcher Ph.D. Hélio Koiti Kuga, DCTA/ITA

External Examiner

**BRASÍLIA, 31TH JULY 2018.**

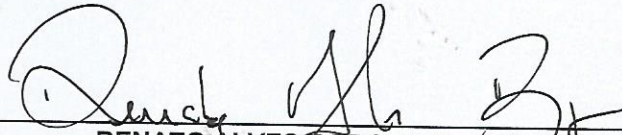
**UNIVERSIDADE DE BRASÍLIA  
FACULDADE DE TECNOLOGIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

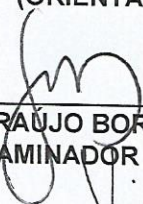
**FILTERING AND ADAPTIVE CONTROL FOR BALANCING A  
NANOSATELLITE TESTBED**

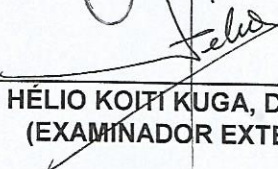
**RODRIGO CARDOSO DA SILVA**

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

  
\_\_\_\_\_  
RENATO ALVES BORGES, Dr., (ENE/UNB  
(ORIENTADOR)

  
\_\_\_\_\_  
GEOVANY ARAUJO BORGES, Dr., ENE/UNB  
(EXAMINADOR INTERNO)

  
\_\_\_\_\_  
HÉLIO KOITI KUGA, Dr., INPE  
(EXAMINADOR EXTERNO)



## **CATALOGRAPHIC CARD**

SILVA, RODRIGO CARDOSO DA

### **FILTERING AND ADAPTIVE CONTROL FOR BALANCING A NANOSATELLITE TESTBED**

**2018, x, 166p., 201 x 297 mm**

(ENE/FT/UnB, Master of Science, ELECTRICAL ENGINEERING, 2018)

Thesis of Master's Degree - University of Brasília

Faculty of Technology - Department of ELECTRICAL ENGINEERING

- |                          |              |
|--------------------------|--------------|
| 1. Kalman                | 2. Lyapunov  |
| 3. air bearing           | 4. balancing |
| 5. nanosatellite testbed | 6. ADCS      |

I. ENE/FT/UnB

## **BIBLIOGRAPHICAL REFERENCE**

RODRIGO CARDOSO DA SILVA (2018) FILTERING AND ADAPTIVE CONTROL FOR BALANCING A NANOSATELLITE TESTBED. Thesis of Master's Degree in ELECTRICAL ENGINEERING, Publication PPGEA.DM 706/2018, Department of ELECTRICAL ENGINEERING, University of Brasília, Brasília, DF, 166p.

## **ASSIGNMENT OF RIGHTS**

AUTHOR: RODRIGO CARDOSO DA SILVA

TITLE: FILTERING AND ADAPTIVE CONTROL FOR BALANCING A NANOSATELLITE TESTBED.

DEGREE: Master of Science YEAR: 2018

It is conceded to the University of Brasília permission to reproduce copies of this thesis of Master's Degree and to lend or sell such copies only for scientific and academic purposes. The author reserves to other rights of publication and no part of this thesis of Master's Degree can be reproduced without the written authorization of the author.

---

RODRIGO CARDOSO DA SILVA

r<sub>c</sub>silva@lara.unb.br.

# Acknowledgments

To my parents, Ari Cardoso da Silva and Maria Rose Meiry Alves da Silva, who dedicated their lives to raising me and my sister giving us all they had not when they were young.

To my sister, Patrícia Cardoso da Silva, who is an example for me of resistance and faith on the pursue of dreams.

To Leandra Lysle Garcia Liguori, for giving me balance in all aspects of my life, for supporting me all the time and for being the person with whom I can share my love eternally.

To all my childhood friends, Fábio, Flávio, Gabriel, Mateus, Pablo e Renato, whose loyalty goes through time without corroding and for bringing “distance” to the length of a mere word.

To all my colleagues in the Laboratory of Aerospace Science and Innovation (LAICA) of the University of Brasília for the partnership, namely Lucas, Igor, João and Fernando.

To professors Ph.D. Renato Alves Borges, Ph.D. Simone Battistini, Ph.D. Chantal Cappelletti, who accompanies me in my journey at the LAICA since the beginning and on who I deposit my respect and admiration. To professor Ph.D Renato Alves Borges, for being my advisor in this work, for believing in this project and for accepting the challenges life give us. To professor Ph.D. Geovany Araújo Borges, for his valuable words of wisdom shared in our always clarifying meetings. To Ph.D. Hélio Koiti Kuga, from the National Institute for Space Research (INPE) of Brazil, who is currently a visitant researcher of the Brazilian Department of Aerospace Science and Technology (DCTA) of the Aeronautics Technological Institute (ITA), for accepting the invitation to evaluate this work. To the University of Brasília (UnB), the Federal District Research Support Foundation (FAPDF), the Coordination for the Improvement of Higher Education Personnel (CAPES) and the National Council for Scientific and Technological Development (CNPq) for supporting this work directly or indirectly.

To the anonymous shadows of destiny, whose combined actions brought me here and to all the people who believe they can impact the society positively.

*“To be great, be whole: do not  
exaggerate or exclude anything from you.*

*Be all in every thing. Put how much you are  
at the very least you do.*

*So in every lake the whole moon  
Shines, because in the high it lives.”*

*Fernando Pessoa  
(personal translation)*

# Abstract

The Laboratory of Application and Innovation in Aerospace Science (LAICA) of the University of Brasília (UnB) is developing a nanosatellite testbed capable of simulating the environment conditions seen in space, specially regarding the Earth magnetic field in orbits, the frictionless rotational movement and the low gravitational torque. This testbed comprises various subsystems, such as an air bearing table, on which nanosatellites are mounted for testing its subsystems; a Helmholtz cage, responsible for simulating the Earth magnetic field present in various kinds of orbit, specially Low Earth Orbits, which is the most common for nanosatellites; actuation systems, such as reaction wheels and magnetorquers, used to study attitude control strategies, and attitude determination systems, such as those based on embedded telemetry or computer vision. The air bearing table is the part responsible for providing the frictionless movement with three rotational degrees of freedom. Also, for providing the low gravitational torque requisite, a method must be developed for balancing the air bearing table. In this work, focus is given for solving this problem. Various methods for balancing the LAICA testbed are presented, specially regarding filtering solutions, such as those using the Kalman Filter and its variations, and adaptive control schemes, aided by the Lyapunov theory. The performance of the proposed balancing methods is evaluated through simulations and experiments.

**Keywords:** Kalman, Lyapunov, air bearing, balancing, nanosatellite testbed, ADCS.



# Resumo

O Laboratório de Aplicação e Inovação em Ciências Aeroespaciais (LAICA) da Universidade de Brasília (UnB) está desenvolvendo uma plataforma de testes de nanossatélites capaz de simular condições ambientais vistas no espaço, especialmente no que diz respeito ao campo magnético da Terra em órbitas, o movimento rotacional livre de atrito e o torque gravitacional baixo. Essa plataforma compreende vários subsistemas, tais como uma mesa com rolamento a ar, na qual nanossatélites são montados para teste de seus subsistemas; uma gaiola de Helmholtz, responsável por simular o campo magnético da Terra presente em vários tipos de órbita, especialmente órbitas de baixa altitude (LOE), que são as mais comuns para nanossatélites; sistemas de atuação, tais como rodas de reação e atuadores magnéticos, usados para estudar estratégias de controle de atitude, e sistemas de determinação de atitude, tais como aqueles baseados em telemetria embarcada ou visão computacional. A mesa com rolamento a ar é a parte responsável por fornecer o movimento livre de atrito com três graus de liberdade rotacionais. Ademais, para fornecer o requisito de torque gravitacional baixo, um método deve ser desenvolvido para balancear a mesa com rolamento a ar. Neste trabalho, foco é dado para a solução desse problema. Vários métodos para balanceamento da plataforma de testes do LAICA são apresentados, especialmente quanto às soluções de filtragem, como aquelas que utilizam o filtro de Kalman e suas variações, e esquemas de controle adaptativo, auxiliados pela teoria de Lyapunov. A performance dos métodos de balanceamento propostos é avaliada por meio de simulações e experimentos.

**Palavras-chave:** Kalman, Lyapunov, rolamento a ar, balanceamento, plataforma de testes de nanossatélites, ADCS.

# SUMMARY

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	KNOWN AIR BEARING PLATFORMS .....	2
1.1.1	INPE - NATIONAL INSTITUTE FOR SPACE RESEARCH .....	2
1.1.2	CALIFORNIA POLYTECHNIC STATE UNIVERSITY (CAL POLY) .....	3
1.2	THE PROBLEM.....	4
1.3	OBJECTIVES .....	5
<b>2</b>	<b>SYSTEM DESIGN .....</b>	<b>7</b>
2.1	SYSTEM OVERVIEW .....	7
2.1.1	THE AIR BEARING TABLE.....	8
2.1.2	THE HELMHOLTZ CAGE .....	10
2.1.3	THE ATTITUDE DETERMINATION WITH COMPUTER VISION SYSTEM (ADCV) .....	10
2.1.4	ACTUATION SYSTEMS.....	11
2.2	THE BALANCING SYSTEM IN DETAIL .....	12
<b>3</b>	<b>THEORETICAL FOUNDATION .....</b>	<b>14</b>
3.1	REFERENCE FRAMES AND KINEMATICS .....	14
3.1.1	ROTATION MATRICES AND SEQUENCE OF ROTATION.....	15
3.1.2	QUATERNIONS.....	19
3.1.3	ATTITUDE DETERMINATION .....	24
3.2	DYNAMIC MODELING .....	30
<b>4</b>	<b>BALANCING TECHNIQUES .....</b>	<b>41</b>
4.1	ESTIMATING THROUGH LEAST SQUARES METHOD .....	41
4.2	ESTIMATING UNBALANCE WITH A KALMAN FILTER .....	48
4.3	NONLINEAR FILTERING APPLIED TO THE BALANCING PROBLEM .....	53
4.3.1	THE EKF APPLIED TO THE BALANCING PROBLEM.....	54
4.3.2	AUGMENTING THE EKF .....	59
4.3.3	THE UKF APPLIED TO THE BALANCING PROBLEM.....	66
4.3.4	DISCUSSIONS ABOUT OBSERVABILITY.....	69
4.3.5	PARAMETER ESTIMATION SUMMARY .....	71
4.4	THE HYBRID ADAPTIVE CONTROL METHOD.....	72

4.4.1	THE TRANSVERSE PLANE COMPENSATION .....	72
4.4.2	THE VERTICAL IMBALANCE COMPENSATION .....	81
<b>5</b>	<b>TESTS AND RESULTS .....</b>	<b>85</b>
5.1	METHODS FOR EVALUATING THE BALANCING PERFORMANCE.....	87
5.2	ANALYSIS .....	88
<b>6</b>	<b>CONCLUSION .....</b>	<b>98</b>
6.1	FUTURE PERSPECTIVES .....	99
	<b>BIBLIOGRAPHICAL REFERENCES.....</b>	<b>100</b>
	<b>APPENDICES .....</b>	<b>105</b>
<b>A</b>	<b>ELECTRONIC BOARD DESIGN AND BLUEPRINT.....</b>	<b>106</b>
<b>B</b>	<b>EULER RATES IN THE ZYX SEQUENCE .....</b>	<b>109</b>
<b>C</b>	<b>THE 6-PARAMETER MAGNETOMETER CALIBRATION.....</b>	<b>111</b>
<b>D</b>	<b>ANGULAR MOMENTUM ABOUT POINT A - SIMPLIFIED EQUATION.....</b>	<b>114</b>
<b>E</b>	<b>QUATERNION ALGEBRA .....</b>	<b>116</b>
<b>F</b>	<b>QUATERNION RATES .....</b>	<b>118</b>
<b>G</b>	<b>THE KALMAN FILTER.....</b>	<b>119</b>
<b>H</b>	<b>THE EKF ALGORITHM .....</b>	<b>121</b>
<b>I</b>	<b>THE COMPLEX STEP DIFFERENTIATION (CSD) .....</b>	<b>123</b>
<b>J</b>	<b>THE UNSCENTED KALMAN FILTER .....</b>	<b>125</b>
<b>K</b>	<b>SOURCE CODES .....</b>	<b>128</b>
K.1	DYNAMIC MODEL COMPARISON .....	128
K.2	KALMAN FILTER.....	131
K.3	EXTENDED KALMAN FILTER.....	138
K.4	UNSCENTED KALMAN FILTER .....	150
K.5	ADAPTIVE CONTROL SIMULATION.....	157

# LIST OF FIGURES

1.1	The INPE Satellite Simulator. From [Gonzales 2009], two rotary tables (left photo, tri-axis at left and single axis at right) and from [Carrara and Milani 2007], right photo (adapted), the triaxial one in detail. ....	3
1.2	The Cal Poly Spacecraft Attitude Dynamics Simulator [Mittelsteadt and Mehiel 2007]. ....	3
2.1	Nanosatellite simulator into the Laboratory of Aerospace Science and Innovation (LAICA/UnB). ....	7
2.2	The air bearing table and its pneumatic system. ....	8
2.3	Bottom view of Air Bearing Table and Z-axis MMU in detail. ....	9
2.4	Upper view of Air Bearing Table (below the aluminum plate). ....	10
2.5	ADCV components. ....	11
2.6	Actuation devices. ....	12
2.7	Electronic board in detail. ....	13
3.1	Inertial and body-fixed frames locations. ....	15
3.2	The ZYX Euler angles sequence. ....	18
3.3	Errors in $\{\phi, \theta, \psi\}$ angles when they are calculated from DCMs. ....	24
3.4	Reference frames in the eCompass algorithm. ....	25
3.5	Yaw distortion when magnetometer is not calibrated. ....	27
3.6	LAICA magnetometer calibration data. ....	29
3.7	Percent error in quaternion determination. ....	30
3.8	An arbitrary rotating rigid body. ....	32
3.9	Error between [Young 1998] and [Chesi et al. 2014] simulated angular velocities. ....	40
4.1	Parameter estimation convergence as function of the batch size. ....	44
4.2	Parameter estimation convergence as function of the batch size. ....	45
4.3	Effect of a synchronization fault. ....	46
4.4	Innovation terms $d^2$ throughout a 100 s simulation of the Kalman Filter. ....	51
4.5	State errors and the $\pm 3\sigma$ intervals. ....	51
4.6	$r_x$ estimation when: (up) normal conditions, (middle) $r$ is 20 times bigger, (bottom) high angular velocities ( $\omega = [1 \ 1 \ 1] \text{ rad/s}^T$ ). ....	52



4.7	$r_x$ estimation when $J_{ij}$ , $i \neq j$ have comparable magnitude with $J_{ii}$ .....	53
4.8	Innovation terms $d^2$ throughout a 100 s simulation of the Extended Kalman Filter. ....	57
4.9	State errors and the $\pm 3\sigma$ intervals. ....	57
4.10	Estimation of $\mathbf{r}$ components in the 6-state EKF. ....	58
4.11	Estimation of $\mathbf{r}$ components in the 6-state EKF when angular velocities are high. ....	58
4.12	Innovation terms $d^2$ throughout a 300 s simulation of the 12-state EKF. ....	63
4.13	$3\sigma$ intervals for the 12-state EKF. ....	64
4.14	Parameter estimation in the 12-state EKF under usual conditions. ....	65
4.15	Parameter estimation in the 12-state EKF (high angular rates). ....	66
4.16	UKF consistency and $3\sigma$ intervals for the estimated states. ....	67
4.17	UKF: Convergence of the unbalance vector components under normal conditions. ....	68
4.18	UKF: Convergence of the unbalance vector components (high angular velocities and unbalance magnitude). ....	68
4.19	Testbed angular velocities during transverse balancing. ....	79
4.20	MMUs positions during transverse balancing. ....	80
4.21	The estimated unbalance vector components ( $\hat{\Theta}$ ). ....	81
4.22	Results of the simulation of the 4-state UKF. ....	83
5.1	Angular velocities and Euler angles of the platform in the unbalanced condition. ....	88
5.2	Unbalance vector components estimated by the KF. ....	89
5.3	Unbalance vector components estimated by the EKF (simplified inertia). ....	90
5.4	Unbalance vector components estimated by the EKF (complete inertia). ....	90
5.5	Unbalance vector components estimated by the UKF. ....	91
5.6	Model validation using the $\mathbf{r}$ vector estimated by the UKF. ....	92
5.7	Spectrum of the roll and pitch signals during the initial condition experiment. .	93
5.8	Energy analysis of the initial condition of the platform. ....	94
5.9	Final MMUs and balancing weight positions for the manual balancing. ....	95
5.10	Performance of the manual balancing. ....	96
5.11	Envelopes of the angular velocity signals during the initial experiment. ....	96
5.12	Exponential curve fittings of the upper bounds of Fig. 5.11. ....	97
A.1	Upper layer of the electronic board (labels only). ....	106
A.2	Bottom layer of the electronic board (copper trails). ....	107
A.3	Electronic board schematics. ....	108

# LIST OF TABLES

2.1	Reaction Wheels specifications. ....	12
2.2	Specifications of the IMU embedded in the LAICA testbed. ....	13
3.1	LSM303DLH chip 6-parameter calibration. ....	28
3.2	Maximum percent error for various simulation conditions. ....	39
4.1	Summary of parameter estimation approaches. ....	71
5.1	Unbalance vector components estimation and corresponding Chi-squared ratings in the initial testbed condition. ....	89
5.2	Curve fitting parameters for $\omega_x$ , $\omega_y$ and $\omega_z$ . ....	95

# LIST OF SOURCE CODES

K.1	Euler equations of motion .....	128
K.2	Young equations of motion.....	129
K.3	Model comparison code for Young and Euler equations of motion .....	130
K.4	Kalman Filter.....	131
K.5	Kalman Filter performance under specific conditions .....	135
K.6	6-state Extended Kalman Filter (analytic jacobians).....	138
K.7	12-state Extended Kalman Filter (jacobians calculated through Complex Step Differentiation) .....	142
K.8	Dynamic model used in the 12-state EKF.....	149
K.9	6-state Unscented Kalman Filter.....	150
K.10	Equations of motion used in the 6-state UKF .....	156
K.11	Adaptive Control scheme.....	157
K.12	Equations of motion used in ChesiBalancing.....	159
K.13	4-state UKF used in the second phase of the hybrid balancing scheme .....	161
K.14	Equations of motion used in the 4-state UKF .....	165

# LIST OF ACRONYMS

ADCS	Attitude Determination and Control Systems
ADCV	Attitude Determination with Computer Vision
BKE	Basic Kinematic Equation
CAD	Computer Aided Design
CSD	Complex Step Differentiation
ITA	Aeronautics Technological Institute
CNPq	National Council for Scientific and Technological Development
CM	Center of Mass
CMG	Control Moment Gyro
CAPES	Coordination for the Improvement of Higher Education Personnel
CR	Center of Rotation
DCM	Direct Cosine Matrix
DCTA	Brazilian Department of Aerospace Science and Technology
DOF	Degree of Freedom
EKF	Extended Kalman Filter
FAPDF	Federal District Research Support Foundation
IMU	Inertial Measurement Unit
INPE	National Institute for Space Research
KF	Kalman Filter
LAICA	Laboratory of Application and Innovation in Aerospace Science
LSM	Least Squares Method
MMU	Movable Mass Unit
NOAA	National Oceanic and Atmospheric Administration
PWM	Pulse Width Modulation
UKF	Unscented Kalman Filter
UnB	University of Brasília
WMM	World Magnetic Model



# LIST OF SYMBOLS

## **Bold symbols**

$T$	Vectors and matrices.
$\phi$	Sampling time
$\theta$	Roll angle
$\psi$	Pitch angle
$\mathbf{J}$	Yaw angle
$\mathbf{R}_{I, \alpha}$	Inertia tensor matrix
$\mathbf{E}$	Rotation of $\alpha$ degrees about I axis.
$\mathbf{q}$	Vector of Euler angles.
$\mathbf{q}$	quaternion
$E$	vector part of quaternion $\mathbf{q}$
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	Energy.
$\times$	Versors of an orthogonal system.
$\cdot$	Cross product.
$[\mathbf{a}\times]$	Dot product.
$\mathbf{L}$ or $\mathbf{H}$ or $\mathbf{h}$	Cross product in matrix form.
$\mathbf{M}$ or $\boldsymbol{\tau}$	Angular momentum.
$\mathbf{v}$	Torque.
$\boldsymbol{\omega}$	Velocity vector.
$\boldsymbol{\rho}$	Angular velocity.
$\mathbf{r}$	Position vector.
$m$	Unbalance vector.
$m_{\text{MMU}}$	mass in general or total mass of the platform.
$\mathbf{J}_i$ or $\mathbf{J}_{ii}$	Moving mass of a MMU.
$\mathbf{J}_{ij}, i \neq j$	Principal moment of inertia around i axis.
Subscript i (1)	Products of inertia between i and j axes.
Subscript i (2)	Quantity related to the inertial frame.
Subscript b	Quantity related to the $i^{\text{th}}$ particle.
Subscript MMU	Quantity related to the body frame.
Subscript O	Quantity related to the Movable Mass Unit.
Subscript G	Quantity related to the center of rotation.
Subscript or superscript $k, k - 1, \dots$	Quantity related to the center of mass.
	Quantity related to the instant $k, k - 1, \dots$

# Chapter 1

## Introduction

*If I have seen further than others, it is by standing upon the shoulders of giants.*

- Isaac Newton

Since the early fifties, scientists all over the world invest effort in space exploration. Motivated by the Cold War, the space race began as a competition between the United States of America and the Soviet Union, trying to outdo each other in aerospace technologies. Spanning for more than twenty years, the space race had seen the man send to space the first living creature - a dog, named Laika -, the first living man to orbit the Earth, Yuri Gagarin, aboard the Vostok I ship, and culminating with the memorable spacewalk at the surface of the Moon, set as goal by the US president Kennedy and accomplished by the NASA astronauts Neil Armstrong and Edwin Aldrin, aboard Apollo XI. Also, it is undeniable that the space race contributed to many of the advances in technology we see today - the GPS is an example.

Although sometimes these accomplishments were made by taking too much risk of mission failures, in the majority of times they were made after an extensive load of tests and simulations. As evolved as the technologies which took the man to space were the technologies that brought space to man. As examples, the vacuum chamber - to simulate vacuum -, thermal chambers - to simulate temperatures in the space environment -, astronaut pools - for training astronauts to move in low gravity environment and others. Nowadays, this philosophy still remains. Regarding the study of satellites motion, one of such testing systems must simulate at least two key conditions: the environmental magnetic field in orbit and the low gravitational torque. For simulating the environmental magnetic field, a well-known approach is to use a combination of Helmholtz coils disposed in a 3D configuration. In this sense, various works were developed at the Laboratory of Aerospace Science and Innovation (LAICA) of the University of Brasília (UnB) resulting in the conception of a Helmholtz cage, capable of simulating the Earth magnetic field in various kinds of orbits [de Loiola et al. 2018]. Further details on this system are given in Sec. 2. On the other hand, to simulate the low gravitational torque the most commonly used systems are gimbals and bearings. Gimbals are pivoted mechanical devices in which the motion can be restricted to a single axis. Combining three gimbals, it is possible to perform complete 3D rotational motion. However, 3D gimbals have specific orientations in which one of the degrees of free-

dom is lost - namely, gimbal locks -, which are difficult to treat and introduces non linear effects in the dynamic modelling. The other testing systems - the bearings - may be mechanically coupled or not. Concerning mechanically coupled bearings, there are the ball bearings that, when combined, may also provide complete three dimensional motion, nevertheless effects such as friction may not be negligible. Finally, uncoupled bearings are those in which a fluid is used to free the rotational joints. This is the most common type of bearing used in satellite simulators and is the one focused in this work, specially the air bearings. This kind of bearing may be planar or rotational depending on the desired application to be simulated. For rendezvous, formation flying and on-orbit construction, planar systems are recommended, since they can provide freedom to the system to rotate and spin. For rotational movement, related to attitude control and pointing maneuvers, spheric systems are used. In this work, a nanosatellite simulator was built on this kind of spheric bearing - namely, the air bearing - in a configuration called tabletop. In this configuration, a table is mounted on a half sphere, giving complete rotational freedom around one axis and approximately  $\pm 45^\circ$  on the other two. Details on the other types of spheric air bearing configurations may be seen in [Schwartz et al. 2003]. Before giving insight in the problem studied in this work, some similar spacecraft simulators will be presented. After that, the main problem concerning the design of this kind of platform is presented, followed by the outline of this dissertation.

## **1.1 Known air bearing platforms**

There are various satellite simulators being developed around the world. As said before, one specific kind of simulator is that which uses pressurized air in order to provide frictionless motion. These platforms are generally based on air bearings. In what follows, some of these platforms are exposed. A historical review of satellite simulators based on air bearings is made in [Schwartz et al. 2003].

### **1.1.1 INPE - National Institute For Space Research**

In Brazil, besides the platform being described in this work, there are also the three platforms present at the National Institute For Space Research (INPE), all of which are dedicated to simulating rotational dynamics. Two simulators are shown in Fig. 1.1 (photo at left) and they divide the task of simulating rotational motion in one or three axes. The mono axial platform is based on a plane bearing which permits rotational motion in only one axis, whereas the three axial platform is based on a spherical bearing and was developed in order to investigate the dynamic behaviour of the nutation damper used in the first brazilian satellite, the SCD-1 [Gonzales 2009].

In [Carrara and Milani 2007] the single axis platform is described (photo at right in Fig. 1.1). This simulator is sufficiently equipped for providing a test environment for attitude

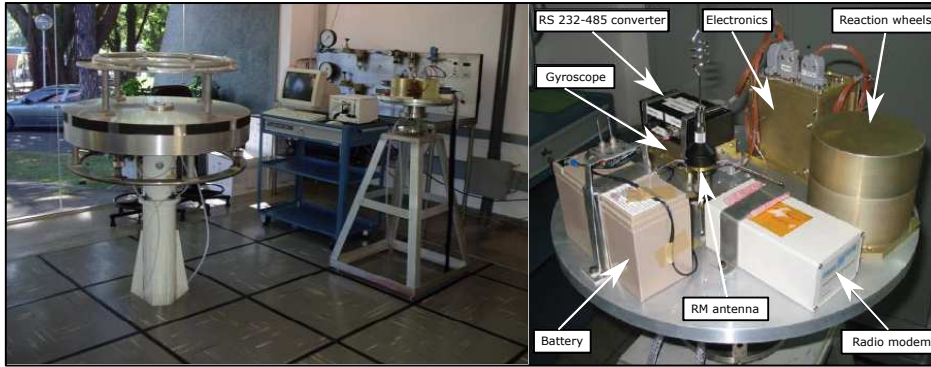


Figure 1.1: The INPE Satellite Simulator. From [Gonzales 2009], two rotary tables (left photo, tri-axis at left and single axis at right) and from [Carrara and Milani 2007], right photo (adapted), the triaxial one in detail.

determination and control algorithms, being equipped with reaction wheel as actuator. The importance of calibrating the mass distribution on the platform is emphasized in this work, which states that the minimum residual torque achieved was of about  $10^{-4} N \cdot m$  while the platform was weighting more than 15 kilograms. This uncalibrated mass distribution is reported as the cause for the platform oscillation when it is not calibrated and it is mentioned that, after balancing the platform, even a counterweight of only 10 grams placed on the platform can make the oscillations rise up again. The two three-axis simulators are described and some tests are reported in [de Oliveira et al. 2013] and [de Oliveira et al. 2015].

### 1.1.2 California Polytechnic State University (Cal Poly)

The Cal Poly developed the Cal Poly Spacecraft Attitude Dynamics Simulator (CP/CADS), which is shown in Fig. 1.2. This simulator rests on a spherical air bearing and provides a simulation environment capable of rotating 360 degrees around the yaw axis and 30 degrees about the roll and pitch axes.

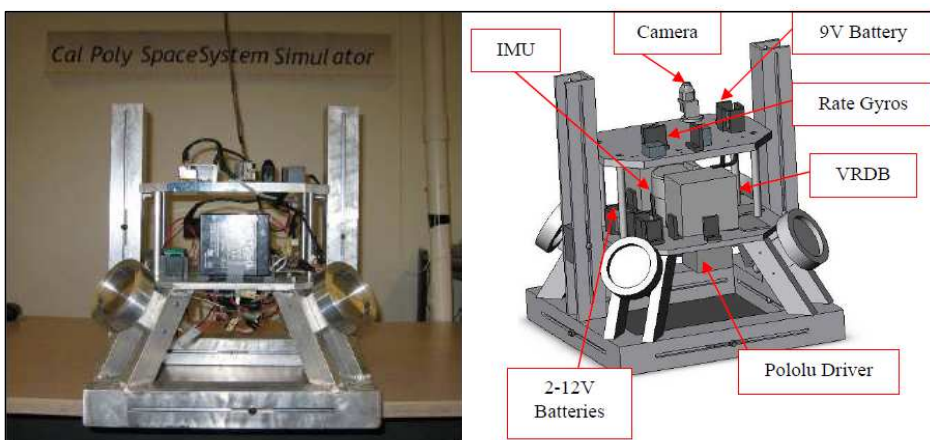


Figure 1.2: The Cal Poly Spacecraft Attitude Dynamics Simulator [Mittelsteadt and Mehiel 2007].

This simulator, differently from other simulators in which three actuators are normally orthogonally disposed, has its actuators in a pyramidal configuration composed of four reaction wheels. It has a simple balancing system made of two rails into which weights are positioned manually, giving evidence of the importance of distributing the masses of the system properly. The project of this simulator foresees space for building an automatic mass balancing system, although it had not been implemented yet at the moment of that publication [Mittelsteadt and Mehiel 2007].

## 1.2 The problem

The main problem concerning the kind of simulation platform being developed is the gravitational torque which arises from the deviation between the center of rotation and the center of mass of the testbed. This torque acts as a disturbance torque in the attitude control system and may limit the actuation devices to function properly. In other words, the actuation devices may saturate before reaching the desired platform orientation. This requisite is specially critic in nanosatellite simulators, since the actuators present in these small-sized satellites can typically provide much less torque than that achieved by other big-sized platforms. From now on, this problem will be attributed to the existence of what is called unbalance vector, imbalance or Center of Mass (CM) offset, *i.e.* the distance between the Center of Rotation (CR) and the CM of the platform, and diminishing this distance to a value as close as zero will be set as goal.

This problem has already been addressed in various works. In [Romano and Agrawal 2003] and [Peck et al. 2003], a minimum gravitational torque of  $0.01 N \cdot m$  was obtained for an air bearing-based platform by performing a manual balancing procedure. However, as reported, this procedure may take hours until a reasonable result is obtained. There are also works in which optimal algorithms were studied in order to obtain the best placement for objects on a spacecraft simulator, such as in [Xu et al. 2016]. In this case, before thinking on the implementation of automatic balancing systems, special care is given to arranging the mass distribution of the platform in such a manner that the CM is already close to the CR. However, the best approach for accomplishing good results with low effort is to design an automatic system capable of moving masses in the system until the CM is placed close to the CR within a previously set tolerance. A cost-effective solution for the balancing problem was implemented using the Least Squares Method (LSM) for obtaining batch estimations of the unbalance vector [Silva et al. 2018]. In [Xu et al. 2015], the balancing was accomplished by implementing a recursive least-squares approach with a tracking differentiator and an Extended Kalman Filter (EKF) and the obtained results were compared. In [de Oliveira et al. 2013], an EKF and the nonlinear least squares method were used for identifying the mass properties of a dumbbell air bearing satellite simulator. The results of this work may be used for implementing an automatic balancing system. In [Kim and Agrawal 2009], an advanced balancing method was developed

based on adaptive control and Lyapunov theories. In this case, Control Moment Gyros (CMGs) are used in order to provide an active three-dimensional torque source which is used in the control scheme. Following the idea of [Kim and Agrawal 2009], but taking into account the unavailability of CMGs for implementing the same procedure, the work developed in [Chesi et al. 2014] used the movable masses itself for providing the necessary control torque of an adapted balancing scheme and compensating the imbalance of the given platform, without neglecting its mass distribution purpose. In this case, since it is possible to generate torques only in a plane orthogonal to the gravitational field, only two of the unbalance vector components are nullified with the adaptive control method, for which reason an Unscented Kalman Filter (UKF) is used for properly identifying the last component and compensating it.

In this work, focus will be given in the filtering solutions for estimating the unbalance vector, specially those based on the Kalman Filter and its variations. Also, the methods based on Lyapunov control theory and adaptive control will be addressed. Specially, the method developed in [Chesi et al. 2014] will be studied, since it presents similar constraints with the testbed being developed at LAICA. The reason for which, at this moment, the method developed in [Kim and Agrawal 2009] is not studied is that, as it requires Control Moment Gyros (CMGs), it is not possible to implement it with the current setup of the LAICA testbed.

### **1.3 Objectives**

After surveying the context of other platforms which holds similarity with the one being developed in this work and after analyzing works in which balancing techniques were developed for this kind of platform, a set of techniques were selected in order to persecute better balancing specifications.

The objective of this work is to implement these advanced balancing techniques for the nanosatellite simulator being developed at the Laboratory of Science and Innovation (LAICA) at the University of Brasília (UnB), Brazil. This will be done in order to provide proper gravitational torque tolerance for future works to be developed in the same project, specially those concerning the development of Attitude Determination and Control (ADC) techniques.

Chapter 2 gives insight in the Hardware and Software design of all the systems in the platform being developed in the LAICA, specially those related to the balancing techniques, including sensor details and electronic schematics. Chapter 3 provides all the mathematical background required for deducing the balancing algorithm strategies used in this work. Details of the design of all the filters developed in this work are given, namely the Kalman Filter, the Extended Kalman Filter and the Unscented Kalman Filter. Also, details of a balancing algorithm based on Lyapunov theory and adaptive control theory is presented. Further

demonstrations are presented in the course of this chapter or left as references or appendices. Chapter 5 shows the results obtained with the new implemented algorithms, compares these results to those obtained in previous works and presents an analysis of the achieved performance with judicious experiments. Finally, some concluding remarks and future works perspectives are left to Chap. 6.

# Chapter 2

## System Design

In this chapter, the nanosatellite simulator project being developed in the LAICA is presented. First an overview of the various subsystems of this project is made. Then, details are given concerning the components related to the balancing system, which is the main system used in this work.

### 2.1 System Overview

Fig. 2.1 depicts the portion of the LAICA in which the nanosatellite simulator was constructed. The overall simulator may be divided in four main subsystems, as follows.



Figure 2.1: Nanosatellite simulator into the Laboratory of Aerospace Science and Innovation (LAICA/UnB).



## 2.1.1 The Air Bearing Table

The air bearing table is the system in which the nanosatellite motion is tested. After mounting the nanosatellite on it, it becomes possible to test attitude control algorithms, given it provides adequately low gravitational torques and frictionless movement. Considering the air bearing table was developed in the tabletop configuration, the roll and pitch angles are limited to the  $\pm 45^\circ$  range, whereas the yaw angle is unlimited. Also, two important characteristics of the air bearing table are its total mass and its inertia. Its total mass was obtained by separately measuring each component of the table and was obtained as  $13.901 \text{ kg}$  (without any nanosatellite mockup mounted on it), whereas its inertia is given by the following matrix

$$\mathbf{J} = \begin{bmatrix} 0.265 & -0.014 & -0.035 \\ -0.014 & 0.246 & -0.018 \\ -0.035 & -0.018 & 0.427 \end{bmatrix} [\text{kg} \cdot \text{m}^2], \quad (2.1)$$

which was obtained via CAD software. The meaning of the matrix  $\mathbf{J}$  will become apparent in Sec. 3.

Fig. 2.2 shows the air bearing table placed on the air bearing base (left) and the pneumatic system of the air bearing (right). In order to provide the frictionless contact between the two parts of the air bearing - the air bearing sphere (see Fig. 2.3) and its base -, a pneumatic system composed by an air compressor and air filters is connected to the air bearing base, creating an air cushion between the parts. The air compressor is capable of providing  $10 \text{ bar}$  or approximately  $145 \text{ psi}$ , whereas the air bearing base nominal pressure is  $80 \text{ psi}$ . The pressure sent to the air bearing base is controlled in the air filters, which are capable of filtering particles as small as  $0.01 \mu\text{m}$ . It must be emphasized the importance of the air filters: although the surrounding air appear to be sufficiently pure, in the operating pressure of the air bearing the impurities of the air become too concentrated. These impurities may become stuck in the air bearing, lowering its performance.

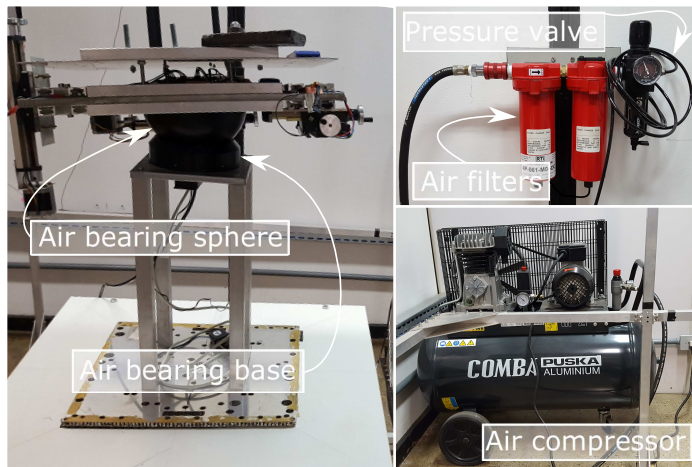


Figure 2.2: The air bearing table and its pneumatic system.

Fig. 2.3 shows components related to the balancing system, namely the Movable Mass

Units (MMU) and the balancing weight. These elements are used to change the mass distribution of the table, either manually, as in the case of the balancing weight, either by controlling the stepper motors of the MMUs electronically. As one may realize from Fig. 2.3, the balancing weight is needed for counterbalancing the weight of the Y-axis MMU, whereas both the Z-axis and X-axis MMUs counterbalance each other. Looking to the Z-axis in detail (right portion of Fig. 2.3), it is possible to see that there are two movable axes in a MMU, both accessible via cranks: a main one which is controlled by a stepper motor and an auxiliary one, which is used only manually for fine adjustments of the mass distribution of the table. Two of the MMUs properties are important, concerning the balancing system: the total excursion of a MMU in its main axis is about  $134\text{ mm}$  and the mass displaced by an MMU, comprising its movable part (the mass of a stepper motor plus the mass of the other moving parts), is about  $0.78\text{ kg}$ . Also, each turn of the crank of a MMU displaces this  $0.78\text{ kg}$  for exactly  $1\text{ mm}$ .

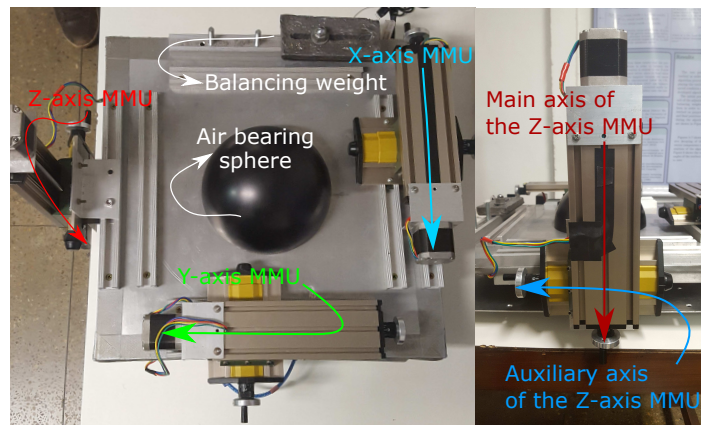


Figure 2.3: Bottom view of Air Bearing Table and Z-axis MMU in detail.

Fig. 2.4 shows the air bearing table when the nanosatellite mounting plate is removed. In this figure, it is possible to see four main components:

1. Batteries: two batteries are used, one as the stepper motors supply ( $12\text{ V}$ ) and another for the electronic board ( $5\text{ V}$ ).
2. Movable Mass Units (MMUs): the three MMUs, as explained.
3. Electronic Board: an electronic board, responsible for controlling the MMUs, collecting telemetry data and interfacing with an external computer wirelessly.

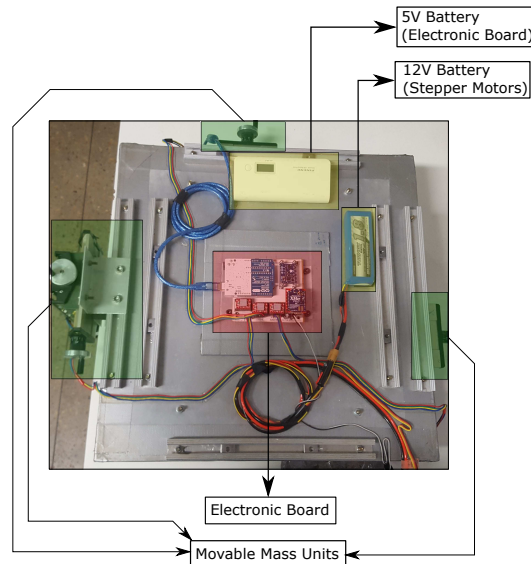


Figure 2.4: Upper view of Air Bearing Table (below the aluminum plate).

## 2.1.2 The Helmholtz Cage

The Helmholtz cage is a huge structure that involves the air bearing table. This structure, which can be seen in Fig. 2.1, is composed by three orthogonally disposed pairs of 2.5 m-sided square coils, meaning that it can generate magnetic field in its interior pointing in any direction. The current in each of the cage axes is controlled via DC sources connected to an external computer (Fig. 2.5 shows the three DC sources below one of the ADCV monitors).

By controlling the magnetic field inside the cage, it is possible to replicate the magnetic field seen in various orbits around Earth. Having the possibility of controlling this field is specially important in attitude control algorithms which require the magnetic interaction between the magnetic actuators of the nanosatellite, such as those which will be described in Sec. 2.1.4, and the environmental magnetic field.

Further details on the design of the Helmholtz cage may be seen in [van der Ploeg 2017], whereas other studies may be found in [de Loiola et al. 2018], such as a study of the magnetic field capacity of the cage and the homogeneity of the field in its interior.

## 2.1.3 The Attitude Determination with Computer Vision System (ADCV)

Fig. 2.5 shows the components of the Attitude Determination with Computer Vision (ADCV) system developed in the LAICA. This system, which is composed by an USB camera connected to an external computer (the same responsible for controlling the Helmholtz cage), is responsible for providing a ground truth reference frame for the air bearing table, providing the orientation of the table in real-time. Its functioning is based on the proper placement of square shaped patterns (the ArUco patterns) in strate-



Figure 2.5: ADCV components.

gic points of the air bearing table. These patterns are detected by the computer vision software, which determines the Euler angles of the testbed. The reader may refer to the works described in [Garrido-Jurado et al. 2016],[Garrido-Jurado et al. 2014] and [Romero-Ramirez et al. 2018] for further details on the ArUco library, its theoretical basis and current development. Regarding embedded attitude determination, some works were already developed in the LAICA, such as [Guimarães et al. 2017], in which an EKF is used for determining the Euler angles of the testbed using the data collected by the IMU embedded in the table.

#### 2.1.4 Actuation Systems

To run attitude control algorithms, it is necessary to embed actuation devices capable of providing torques on the air bearing table. Fig. 2.6 shows the actuation devices available in the LAICA. These torques may be internal or external. In the case of reaction wheels, this torque is internal and the testbed moves by following the conservation of angular momentum principle. Masses are mounted on brushless DC motors which are commanded by a microcontroller embedded in the testbed. The properties of this actuator are shown in Table 2.1. In [de Loiola et al. 2017], the reaction wheels of Fig. 2.6 were used for controlling the orientation of the air bearing table accordingly to the orientation described by a nanosatellite in a previously set orbit.

In the case of the magnetorquers, the controlling torque is external and is provided by the interaction between the magnetic field in the laboratory - controlled by the Helmholtz cage - and the magnetic field generated by the magnetorquers. The magnetorquers consist in sets

Table 2.1: Reaction Wheels specifications.

Maximum torque	$3.7 \cdot 10^{-3} \text{ N} \cdot \text{m}$
Maximum speed	$7000 \text{ rpm}$
Wheels inertia	$32 \cdot 10^{-6} \text{ kg} \cdot \text{m}^2$
Momentum	$23.4 \cdot 10^{-3} \text{ N} \cdot \text{m} \cdot \text{s}$

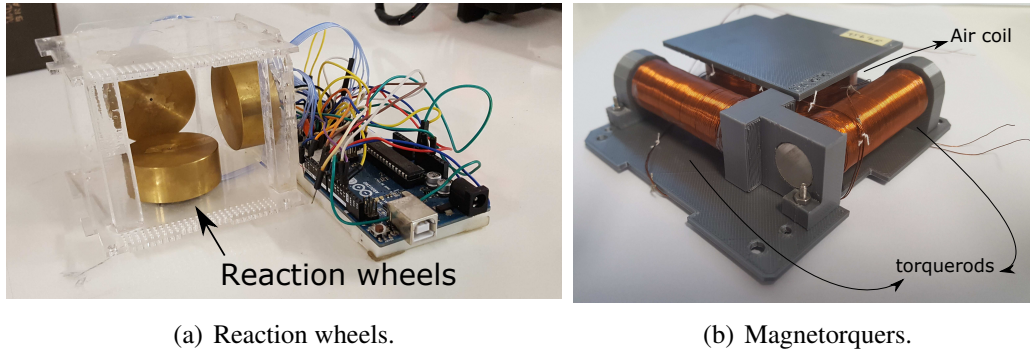


Figure 2.6: Actuation devices.

of coils in which an electronic controlled current passes through. Fig. 2.6 shows two kinds of magnetorquers: torquerods, in which the coils are placed around a metal rod made of a high magnetic permeability material; and air coils, in which the core is mostly air and the supporting structure made of plastic. The design of the magnetorquers developed in LAICA and shown in Fig. 2.6 was presented in [Ishioka et al. 2017].

## 2.2 The Balancing System In Detail

The main part of the balancing system of the air bearing table is the electronic board shown in Fig. 2.7. This board, developed during this work as an upgrade of the electronics presented in [Da Silva and Rodrigues 2015], contains the following components:

- An Arduino: the Arduino board (Arduino Uno model), which contains an ATMEGA8 microcontroller, is responsible for collecting the telemetry data from the Inertial Measurement Unit (IMU) and for sending it to an external computer, on which the balancing algorithms are tested. Also, it is responsible for controlling the stepper motors of the MMUs.
- Drivers: three drivers are placed in this board for controlling each of the three motors of the MMUs placed on the air bearing table.
- Inertial Measurement Unit (IMU): an IMU (9DOF IMU, manufactured by Adafruit) containing three sensors - namely, a 3-axis magnetometer, a 3-axis gyroscope and a 3-axis accelerometer - is responsible for the telemetry data.



Table 2.2: Specifications of the IMU embedded in the LAICA testbed.

	L3GD20 chip	LSM303DLHC	
	Gyroscope	Accelerometer	Magnetometer
Full scale	250/500/2000 <i>dps</i>	$\pm 2/ \pm 4/ \pm 8/ \pm 16 \text{ g}$	$\pm 1.3 \text{ to } \pm 8.1 \text{ Gauss}$
Resolution	16 bits in two's complement.		

- And a communication module: a XBee module (Pro Series S2 model) is used for providing wireless communication between the board and the external computer.

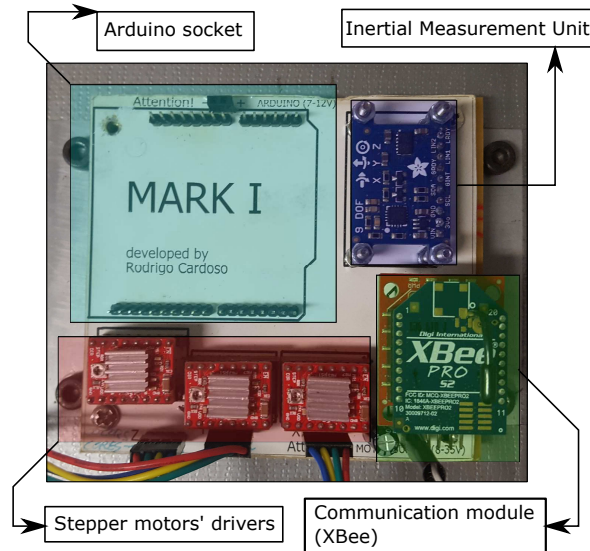


Figure 2.7: Electronic board in detail.

Concerning the stepper motor drivers, they are based on the A4988 chip and are responsible for sending the Pulse Width Modulation (PWM) signals which controls the stepper motors. The drivers are configured in a 200-step mode, which means that each turn of a stepper motor around its corresponding axis is accomplished by sending 200 PWM pulses to the stepper driver. Also, since each turn of a stepper motor moves a MMU for exactly 1 *mm*, the minimum distance that may be moved by an MMU is two hundredths of a millimeter ( $5 \mu m$ ), *i.e.* the distance obtained by sending a single PWM pulse to the driver. Regarding the IMU, its sensors are contained in 2 chips: the L3GD20 chip holds the gyroscopes, whereas the LSM303DLHC chip holds both the magnetometer and the accelerometer. The specifications of each of these sensor are given in Table 2.2.

The reader may find the design schematics of the electronic board of Fig. 2.7, as well as the blueprints of this board, in the Appendix A.

# Chapter 3

## Theoretical Foundation

*The enchanting charms of this sublime science reveal themselves in all their beauty only to those who have the courage to go deeply into it.*

- Johann Carl Friedrich Gauss

### 3.1 Reference frames and kinematics

Before going further into the dynamic model description and equations, it is necessary to establish a standardization to the problem. Such standardization provides means for the understanding of different equations related to the same physical problem. In other words, different equations relating the same problem may be both correct.

One of these standards are the reference frames, which are sets of orthogonal axes whose origins may be positioned in different places of the system space. Its positioning in the system is important for the understanding of the different vector representations that may appear. For example, assuming the gravity vector may be represented in an inertial frame in which the xy plane is parallel to the ground, its z-axis component may have positive or negative magnitude depending on whether this axis is pointing to Earth or in the reverse direction.

In this work, two reference frames are defined, an inertial and a body-fixed one. Inertial frames are assumed to have constant orientation and position in the problem space, while the body-fixed frame is, as its name indicates, a set of orthogonal axes that are fixed to the moving body. In the case of the body-fixed frame, its orientation and position is constant only with relation to the moving body. The inertial frame is represented by the  $(x_i, y_i, z_i)$  set of axes, while the  $(x_b, y_b, z_b)$  set is used for the body-fixed frame.

Since the Air bearing platform has, as a consequence of the bearing design, a well-known center of rotation (CR), this point is taken as the origin for both reference frames. The  $x_i$  and  $y_i$  axes form a plane parallel to the ground of the laboratory, while the  $z_i$  axis points outwards the Earth (or, in the perspective of the laboratory, it points to the roof). As a consequence,

the gravity vector representation is  $\mathbf{g} = [0 \ 0 \ -g]^T$ ,  $g \in \mathbb{R}$ ,  $g > 0$ . The set  $(x_b, y_b, z_b)$  of orthogonal axes are placed fixed to the platform in such a manner that the movement of each of the Movable Mass Units (MMUs) is parallel to one of these axis. Besides this, the  $z_b$  axis is normal to the mounting plate of the platform. Figure 3.1 shows how the body axes - represented by  $X_\psi, Y_\psi, X_\theta, Z_\theta, Y_\phi$  and  $Z_\phi$ , depending on the rotation axis - moves when the body performs rotations of  $\phi, \theta, \psi$  about its  $X_b, Y_b$  and  $Z_b$  axes, respectively. The

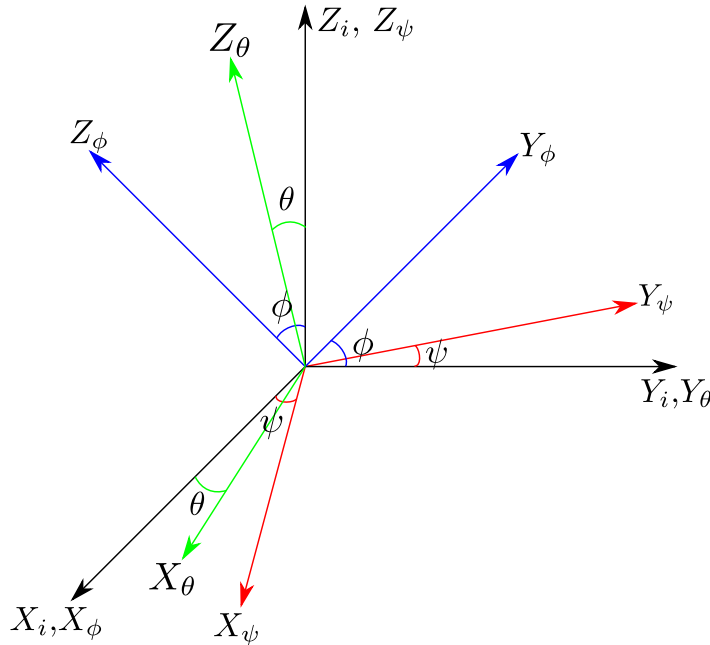


Figure 3.1: Inertial and body-fixed frames locations.

chosen directions for the x and y axes will become clear in Sec. 3.1.3, where the attitude determination algorithm is explained.

After establishing the reference frames used in this work, there must be a mathematical form to describe the kinematics of the system. In this work, two forms are explained: the **Rotation Matrices** with Euler angles and the “four Euler parameters”, also known as **Quaternions**. The rotation matrices will be introduced in order to understand the convention for the rotation sequence adopted. This convention will be later used in the demonstration of some quaternion equations.

### 3.1.1 Rotation matrices and sequence of rotation

In the discussion that follows, vectors are going to be represented as boldfaced lowercase letters. The same applies to the unit vectors used for describing an orthogonal set of axis, like  $\mathbf{i}$ ,  $\mathbf{j}$  and  $\mathbf{k}$ .

In  $\mathbb{R}^3$  a rotation may be seen as the transformation

$$\mathbf{v}_a = \mathbf{R}_{3 \times 3} \mathbf{v}_b, \quad (3.1)$$



in which  $\mathbf{R}$  is a  $3 \times 3$  transformation matrix  $\mathbf{R} : \mathbf{v}_b \mapsto \mathbf{v}_a$ . This rotation matrix obeys some properties, such as

1. Its determinant is equal to  $\pm 1$ . If the coordinate frames and its rotation operations obey the right-hand rule, the determinant equals 1.
2.  $\mathbf{R}^{-1} = \mathbf{R}^T$ , *i.e.* its inverse equals its transpose.
3.  $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}$ , *i.e.* rotation matrices are orthogonal.

The first aspect to be exalted is that a rotation may be interpreted in many ways, *e.g.* a rotation of a vector in a fixed coordinate frame, a rotation of the coordinate frame while the vectors remains fixed or simply the orientation of a transformed frame with respect to a fixed frame [Spong et al. 2006, p. 40]. The interpretation to be considered must be made clear by the context of the analysis.

In [Kuipers et al. 1999], it is emphasized that rotation operators may be viewed as rotations of the coordinate frame or the vectors while the other remains fixed. Also, it is shown that one is the inverse of the other or, considering the rotation operator properties, one is the transpose of the other. It means that a rotation of  $\theta$  degrees of the coordinate frame while the vectors remains fixed may be viewed as a rotation of  $-\theta$  degrees of the vectors while the coordinate frame remains fixed, *i.e.*  $\mathbf{R}_{frame}(\theta) = \mathbf{R}_{vectors}(-\theta)$ .

Considering that, in this work, rotations are performed only in order to reference vectors in different frames - the inertial and the body frames - while the vectors remains fixed, the standard X, Y and Z axes rotations are given by

$$\begin{aligned}
 \mathbf{R}_{X,\phi} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \\
 \mathbf{R}_{Y,\theta} &= \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \\
 \mathbf{R}_{Z,\psi} &= \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} .
 \end{aligned} \tag{3.2}$$

This is called the *usage* of the rotation operator and, in the case of this work, the operator is used as *passive*, *i.e.* the rotations are performed *world-to-body* (PWTB). Some fructiferous discussions on the different rotation operators usage may be seen in [Sommer et al. 2018, Appendix C] and [Shuster 1993, p. 494]. In fact, books commonly define the rotation operator without explicitly showing these two usual conventions, which may lead to confusion when somebody is used to adopt and know only one of them. Taking as examples the books referenced in this work, [Kuipers et al. 1999] adopts the *passive* usage, whereas

[Spong et al. 2006] adopts the *active* usage. Also, two of the most classical books in attitude determination and control utilizes the passive usage, *Spacecraft Attitude Dynamics* [Hughes 2012] and *Spacecraft Attitude Determination and Control* [Wertz 2012]. The duality of passive and active conventions is also known as *alias* and *alibi*, respectively, and is well explained in [Shuster 1993], pointing the convention adopted by some classical works in attitude determination and control literature. The relation between these two conventions may be briefly described by Eq. 3.3,

$$\mathbf{R}_{active} = \mathbf{R}_{passive}^T, \quad (3.3)$$

which means that if one needs to change between *world-to-body* (PWTB or passive) and *body-to-world* (PBTW or active) perspectives, the only needed operation is a matrix transposition.

The matrices in Eq. 3.2 may be combined in various sequences of rotations to relate a frame with another. To obtain the coordinates of a vector in a rotated frame, the rotation matrices are pre-multiplied in the order of the desired sequence, *i.e.* supposing a frame is obtained from the inertial frame after performing a rotation of  $\alpha$  degrees about the Z inertial axis and a rotation of  $\beta$  degrees about the new x axis, then the coordinates of a vector  $\mathbf{v}_i$  in the resulting frame are given by

$$\begin{aligned} \mathbf{v}_{b'} &= \mathbf{R}_{Z,\alpha} \mathbf{v}_i \\ \mathbf{v}_b &= \mathbf{R}_{X,\beta} \mathbf{v}_{b'} \\ \mathbf{v}_b &= \mathbf{R}_{X,\beta} \mathbf{v}_{b'} = (\mathbf{R}_{X,\beta} \mathbf{R}_{Z,\alpha}) \mathbf{v}_i \end{aligned} \quad (3.4)$$

in which the subscript  $b'$  denotes the intermediate frame between the initial and the final frames  $i$  and  $b$ , respectively. Again, caution must be taken to notice that the composition of rotations rule adopted in this work follows the *passive* usage of the rotation operator.

Leonhard Euler (1707-1783), who made great contributions on the field of dynamics and mechanics, is known for being the first person to use sequences of rotations to determine orbit relationships in orbital mechanics. Specially, he proved that

**Theorem 3.1.1 Theorem of Euler angles.** *Two coordinate frames may be related by a sequence of three rotations in which two consecutive rotations are not about the same axis.*

This statement results in a total of twelve combinations of the  $x, y, z$  set of axes:  $x-y-z$ ,  $x-y-x$ ,  $x-z-y$ ,  $x-z-x$ ,  $y-x-z$ ,  $y-x-y$ ,  $y-z-x$ ,  $y-z-y$ ,  $z-x-y$ ,  $z-x-z$ ,  $z-y-x$  and  $z-y-z$ . Moreover, the sequence  $\mathbf{z} - \mathbf{y} - \mathbf{x}$  is known for being used in aerospace applications. In this sequence, a rotation about the inertial Z-axis is performed, followed by rotations about the new y-axis (body axis) and the newest x-axis (body axis), in this order. Following the rotation flowchart convention adopted in [Kuipers et al. 1999], this sequence is represented in Fig.3.2.

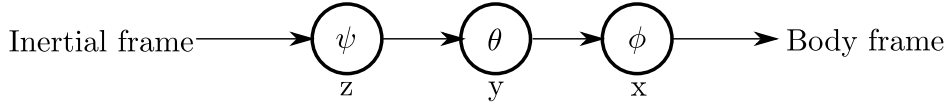


Figure 3.2: The ZYX Euler angles sequence.

From this sequence, the relation between the coordinates of a vector in the body and inertial frames is given by  $\mathbf{v}_b = (\mathbf{R}_{X,\phi}\mathbf{R}_{Y,\theta}\mathbf{R}_{Z,\psi})\mathbf{v}_i$ . The resulting rotation matrix which corresponds to this sequence may be obtained by

$$\begin{aligned}
\mathbf{R} &= \mathbf{R}_i^b = \mathbf{R}_{X,\phi}\mathbf{R}_{Y,\theta}\mathbf{R}_{Z,\psi} = \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \cdot \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \cdot \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \\
&= \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\theta s_\phi \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\theta c_\phi \end{bmatrix}, \tag{3.5}
\end{aligned}$$

in which  $\mathbf{R}_i^b$  is read as the rotation matrix that transforms a vector from the inertial to the body frames.

One of the advantages of this Euler angles sequence, as will be shown in 4.1, is that the gravity vector may be described in the body frame as a function of only two angles, the roll ( $\phi$ ) and pitch ( $\theta$ ) angles.

Another important equation concerns the Euler rates of the adopted sequence, *i.e.* the derivatives of the Euler angles. As one may notice, the propagation in time of the Euler angles is not accomplished by applying directly the angular velocities of the body, but rather by applying the Euler rates  $\dot{\mathbf{E}}$  as in Eq. 3.6

$$\mathbf{E}_{k+1} = \mathbf{E}_k + \dot{\mathbf{E}} \cdot T \Rightarrow \begin{bmatrix} \phi_{k+1} \\ \theta_{k+1} \\ \psi_{k+1} \end{bmatrix} = \begin{bmatrix} \phi_k \\ \theta_k \\ \psi_k \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \cdot T \tag{3.6}$$

in which  $T$  is the sampling time and the discretization is obtained by using the forward Euler method as an example.

The time propagation of the Euler angles is required when simulating the rigid body dynamics. For the known ZYX aerospace sequence, the Euler rates are given by Eq. 3.7 (See Appendix B)

$$\dot{\mathbf{E}} = [\dot{e}_{ij}]_{3 \times 3} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan \theta \sin \phi & \tan \theta \cos \phi \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sec \theta \sin \phi & \sec \theta \cos \phi \end{bmatrix}. \tag{3.7}$$

Eq. 3.7 shows that there exists a set of angles in which the Euler rates are not defined. Basically, due to the presence of the  $\tan$  and  $\sec = \cos^{-1}$  functions in  $\{\dot{e}_{12}, \dot{e}_{13}\}$  and  $\{\dot{e}_{32}, \dot{e}_{33}\}$ , respectively, the Euler rates are not defined when  $\theta = \pm \frac{\pi}{2} \text{ rad}$ . This singularity in the Euler rates equation is inherent to all sequences and is one of the main disadvantages of using the ZYX Euler angles to define attitude, since filters must include algorithms for avoiding the error introduced when the singular attitude is reached.

### 3.1.2 Quaternions

It was on October 16, 1843, when, while walking with his wife along the Royal Canal, Sir William Rowan Hamilton discovered a way to “multiply triples”, after struggling for more than 15 years in the search for a way of treating vectors with complex algebra. At that moment, the enthusiasm about his discovery made him carve on a stone of Brougham Bridge the symbols that were going to rule quaternion algebra [Piaggio 1943].

Quaternions are hyper-complex numbers of rank 4, which means that it has four components. Although the order of its components have variations in literature, as will be explained posteriorly, it can be defined as having a scalar in the first component and other three different imaginary parts, also called the vector part.

**Definition 3.1.2** *A quaternion is a hyper-complex number of the form*

$$q = q_1 + q_2\mathbf{i} + q_3\mathbf{j} + q_4\mathbf{k}, \quad q_i \in \mathbb{R} \forall i, \quad (3.8)$$

in which  $\mathbf{i} = (1, 0, 0)$ ,  $\mathbf{j} = (0, 1, 0)$  and  $\mathbf{k} = (0, 0, 1)$  form the standard orthonormal basis in  $\mathbb{R}^3$ .

Following this definition, comes the celebrated rule developed by William Rowan Hamilton when discovering the quaternions, as shown in Eq. 3.9.

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \quad (3.9)$$

Some additional quaternion properties may be found on Appendix E. Here, special attention is given to the multiplication formula. Being  $p$  a quaternion, the following notation will be used

$$p = p_0 + \mathbf{p}, \quad p_0 \in \mathbb{R}, \quad \mathbf{p} \in \mathbb{R}^3, \quad (3.10)$$

where  $\mathbf{p}$  is the vector part of the quaternion and may be written as  $\mathbf{p} = (p_1, p_2, p_3)$  or  $\mathbf{p} = p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$ . Considering the quaternion rule in Eq. (3.9), the product between two

quaternions may be calculated as

$$\begin{aligned}
p \odot q &= p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3) + \\
&+ p_0(\mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3) + q_0(\mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3) + \\
&\mathbf{i}(p_2q_3 - p_3q_2) + \mathbf{j}(p_3q_1 - p_1q_3) + \mathbf{k}(p_1q_2 - p_2q_1)
\end{aligned} \tag{3.11}$$

in which the  $\odot$  symbol is introduced as notation for the Hamiltonian quaternion product. Using the standard dot and cross products of vectors in  $\mathbb{R}^3$ , this operation may be written as follows.

**Definition 3.1.3 Product.** *Being  $p$  and  $q$  two different quaternions and  $p_i$  and  $q_i$ ,  $i = 1, \dots, 4$  the respective components, its product is defined as*

$$p \odot q = p_0q_0 - \mathbf{p} \cdot \mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q} \tag{3.12}$$

This product may also be written in matrix form as

$$p \odot q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & -p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.13}$$

The importance of quaternions in this work relies on the utility of one of its subgroups: the unit quaternions, also known as *Euler-Rodrigues symmetric parameters*, *Euler symmetric parameters* or *quaternions of rotation*. Unit quaternions may be used for performing rotations in 3D space in a similar manner to that accomplished by using rotation matrices.

**Definition 3.1.4 The rotation quaternion [Kuipers et al. 1999].** *Being  $q = q_0 + \mathbf{q}$  an unit quaternion,  $\|q\| = 1$ , it is called a “rotation quaternion” if it can be written as*

$$q = \cos\left(\frac{\theta}{2}\right) + \mathbf{u} \sin\left(\frac{\theta}{2}\right), \quad \mathbf{u} \in \mathbb{R}^3, \quad -\pi < \theta < \pi \tag{3.14}$$

in which  $\mathbf{u}$  is the rotation axis for a given rotation angle  $\theta$ .

As one may notice, the argument of the rotation quaternion is half the desired rotation angle. Indeed, only unit quaternions may be used to perform rotations, which is why it is rigorously wrong to refer to quaternions *lato sensu* as rotation operators. In other words, every  $4 \times 1$  vector of Euler-Rodrigues symmetric parameters is a quaternion, but the inverse is not true [Shuster 1993]. Following Def. 3.1.4, given a vector  $\mathbf{v} \in \mathbb{R}^3$ , its rotation by an angle  $\theta$  yields a vector  $\mathbf{w} \in \mathbb{R}^3$  and may be defined as

$$\mathbf{w} = \bar{q}\mathbf{v}q \tag{3.15}$$

where  $q = \cos\left(\frac{\theta}{2}\right) + \mathbf{u} \sin\left(\frac{\theta}{2}\right)$  is the associated rotation quaternion and  $\bar{q}$  is the quaternion conjugate of  $q$ . Whenever a vector, such as  $\mathbf{w}$  and  $\mathbf{v}$  in Eq. (3.15), is introduced in a formula containing quaternions, it is treated as a purely imaginary quaternion, overloading the quaternion product for pairs in  $\mathbb{R}^3 \times \mathbb{R}^4$  and  $\mathbb{R}^4 \times \mathbb{R}^3$ . It must be emphasized that the rotation defined by Eq. (3.15) is performed *world-to-body* (PWTB, a frame rotation), following the *passive* convention. In case the active convention is needed, *i.e.* in a context where vectors are rotated and the frame is hold still, the desired rotation may be obtained by swapping the conjugated quaternion as  $\mathbf{w} = q\mathbf{v}\bar{q}$ .

The *world-to-body* rotation performed with unit quaternions may also be described matrixially as

$$\mathbf{w} = \bar{q}\mathbf{v}q = \mathbf{Q}_i^b \mathbf{v}, \quad (3.16)$$

in which the matrix  $[\mathbf{Q}_i^b]_{3 \times 3}$  is given by

$$\mathbf{Q}_i^b = \begin{bmatrix} 2q_0^2 - 1 & 2q_1q_2 & 2q_1q_3 \\ +2q_1^2 & +2q_0q_3 & -2q_0q_2 \\ 2q_1q_2 & 2q_0^2 - 1 & 2q_2q_3 \\ -2q_0q_3 & +2q_2^2 & +2q_0q_1 \\ 2q_1q_3 & 2q_2q_3 & 2q_0^2 - 1 \\ +2q_0q_2 & -2q_0q_1 & +2q_3^2 \end{bmatrix}. \quad (3.17)$$

At this point, it is prudent to mention the differences of notation seen in literature. In [Markley and Crassidis 2014], it is mentioned that there are two usual notations for representing a quaternion. These notations differ basically by the order in which the vector part of the quaternion is disposed. In the work originally developed by Sir William Rowan Hamilton (1805-1865), the idea of having an hypercomplex number with three imaginary parts was adopted, resulting in a notation in which the vector part comes after the scalar part, *i.e.* the Hamiltonian quaternion is defined as

$$q_{\mathbb{H}} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} = q_0 + \mathbf{q}_{1:3} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \quad (3.18)$$

On the other hand, another convention was created and vastly adopted in works related to aerospace engineering, which defines the quaternion as a four-component vector in which the vector part comes first. This convention, assigned to Malcon D. Shuster in this work, is defined as

$$q_{\mathbb{S}} = \mathbf{q}_{1:3} + q_4 = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}. \quad (3.19)$$

Historically, these differences come from mathematical preferences or even as a consequence of the chosen programming language used to implement quaternion equations at the epoch [Shuster 1993, p. 467].

Along with this difference between quaternion definitions, different formulae for quaternion multiplications were also developed. In [Shuster 1993], the use of the  $\otimes$  quaternion product was advocated. The definition of the  $\otimes$  quaternion product is obtained from Eq. (3.12) by swapping the signal of the cross product as

$$p \otimes q = p_0 q_0 - \mathbf{p} \cdot \mathbf{q} + p_0 \mathbf{q} + q_0 \mathbf{p} - \mathbf{p} \times \mathbf{q}, \quad (3.20)$$

*i.e.*  $\mathbf{p} \otimes \mathbf{q} = \mathbf{q} \odot \mathbf{p}$ . The advantage of the Shuster quaternion product becomes apparent for compositions of rotations. Considering the set of unit length quaternions,  $\mathbb{U}$ , the Special Orthogonal Group  $SO(3)$  and the mapping  $\mathbb{C} : \mathbb{U} \mapsto SO(3)$ , which assigns direct cosine matrices to the rotation quaternions, it can be seen that

$$\forall p, q \in \mathbb{U} : \mathbb{C}(p \odot q) = \mathbb{C}(q) \cdot \mathbb{C}(p), \text{ while, for the } \otimes \text{ product,}$$

$$p \otimes q := q \odot p \Rightarrow \mathbb{C}(p \otimes q) = \mathbb{C}(p) \cdot \mathbb{C}(q).$$

In other words, using the Shuster product rule changes the composition of rotations mapping from an antihomomorphism to an homomorphism. Recent discussions about these conventions were made recently, in an attempt to discontinuing the usage of the  $\otimes$  product [Sommer et al. 2018].

From this point on, whenever one of the cited works differ from the notations adopted in [Kuipers et al. 1999], the formulas will be shown in the notation of the corresponding author and how it should be in the notation of this work. Additionally, whenever the operator  $\odot$  or  $\otimes$  is omitted, the product of quaternions is made using the classical Hamiltonian product.

Rotation quaternions provide some strong advantages with relation to rotation matrices for attitude representation, such as requiring 4 instead of 9 elements for its complete representation, fewer constraints - one, about the unit norm of the rotation quaternion, instead of the six constraints of a rotation matrix which imply that all of its rows and columns are unit vectors - and less math operations in its composition rule (16 instead of 27) [Shuster 1993]. Another great advantage of the quaternion approach for describing attitude is that its kinematic equation, the quaternion rates equation, does not present any singularity, in opposition to the ZYX Euler rates equation shown in Eq. 3.7. The quaternion rates equation is shown

in Eq. 3.21,

$$\begin{aligned} \frac{dq}{dt} &= \frac{1}{2} \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \end{aligned} \quad (3.21)$$

in which it can be seen that the quaternion rates are linear within its parameters. Its demonstration is shown in Appendix F, page 118, with little adaption from [Kuipers et al. 1999, p. 263].

However, a rotation quaternion does not have an intuitive meaning as that of Euler angles. In this sense, in order to provide means of comparing the attitude obtained with quaternion algorithms with that obtained using the ZYX Euler angles sequence, the attitude quaternion will be translated to the corresponding roll, pitch and yaw angles. This may be done by comparing Eqs. 3.5 and 3.17, the composite ZYX rotation matrix and the matrix version of the rotation operation with quaternions, respectively,

$$\begin{aligned} \mathbf{R}_i^b &= \mathbf{Q}_i^b, \\ [r_{ij}]_{3 \times 3} &= [q_{ij}]_{3 \times 3}, \end{aligned} \quad (3.22)$$

$$\begin{bmatrix} \boxed{c_\psi c_\theta} & \boxed{s_\psi c_\theta} & \boxed{-s_\theta} \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & \boxed{c_\theta s_\phi} \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & \boxed{c_\theta c_\phi} \end{bmatrix} = \begin{bmatrix} \boxed{2q_0^2 - 1} & \boxed{2q_1 q_2} & \boxed{2q_1 q_3} \\ \boxed{+2q_1^2} & \boxed{+2q_0 q_3} & \boxed{-2q_0 q_2} \\ 2q_1 q_2 & 2q_0^2 - 1 & 2q_2 q_3 \\ -2q_0 q_3 & +2q_2^2 & +2q_0 q_1 \\ 2q_1 q_3 & 2q_2 q_3 & 2q_0^2 - 1 \\ +2q_0 q_2 & -2q_0 q_1 & +2q_3^2 \end{bmatrix}.$$

It may be seen that, considering the cosine of the pitch angle ( $\theta$ ) is not null, the roll and yaw angles may be calculated, using the blue and red portions of Eq. (3.22), respectively, by

$$\phi = \tan_2^{-1} \left( \frac{r_{23}}{r_{33}} \right) = \tan_2^{-1} \left( \frac{q_{23}}{q_{33}} \right) = \tan_2^{-1} \left( \frac{2(q_2 q_3 + q_0 q_1)}{2(q_0^2 + q_3^2) - 1} \right), \quad (3.23)$$

$$\psi = \tan_2^{-1} \left( \frac{r_{12}}{r_{11}} \right) = \tan_2^{-1} \left( \frac{q_{12}}{q_{11}} \right) = \tan_2^{-1} \left( \frac{2(q_1 q_2 + q_0 q_3)}{2(q_0^2 + q_1^2) - 1} \right), \quad (3.24)$$

in which  $\tan_2^{-1}$  is the two argument arc tangent function, whereas the pitch angle may be



calculated directly from the green portion of Eq. (3.22) by

$$\theta = \sin^{-1}(-r_{13}) = \sin^{-1}(2q_0q_2 - 2q_1q_3) . \quad (3.25)$$

Fig. 3.3 shows the percent errors in the roll, pitch and yaw angles when they are calculated directly from time propagation of euler angles and when they are calculated from Eqs. 3.23, 3.25 and 3.24, during a 100 s simulation (0.1 s sampling time). This graph shows that both representations, Euler angles and quaternions, indicates the same attitude, considering that the maximum error is 0.18% and is mainly introduced by the usage of the  $\tan^{-1}$  and  $\tan_2^{-1}$  functions, since these functions introduce discontinuities in the attitude graphs.

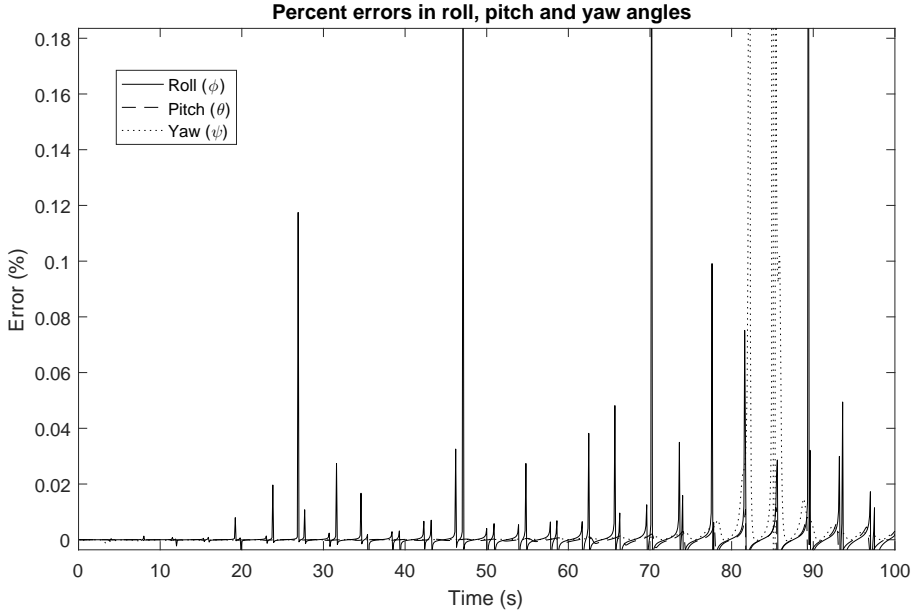


Figure 3.3: Errors in  $\{\phi, \theta, \psi\}$  angles when they are calculated from DCMs.

### 3.1.3 Attitude determination

In this work the eCompass system described in [Semiconductors 2018] is used for determining the attitude of the platform based on the measurements of the three-axis accelerometer and the three-axis magnetometer, both embedded in the LAICA testbed. This system follows the basic idea of the TRIAD algorithm, which is obtaining the attitude with relation to two inertial vectors (in this case, the local magnetic field and the local gravity). Further details on the TRIAD algorithm may be found in [Bar-Itzhack and Harman 1997].

The eCompass algorithm starts by considering an initial position of the body frame in which the magnetometer points to the direction of the planar component of the local magnetic field and the accelerometer is positioned flat with relation to the surface of the Earth, considering the axes of both sensors are aligned with the axes of the body frame (see Fig. 3.4). In this configuration, the corresponding measurements,  $\mathbf{g}_{\text{accel}}$  and  $\mathbf{B}_{\text{mag}}$  for the accelerometer and

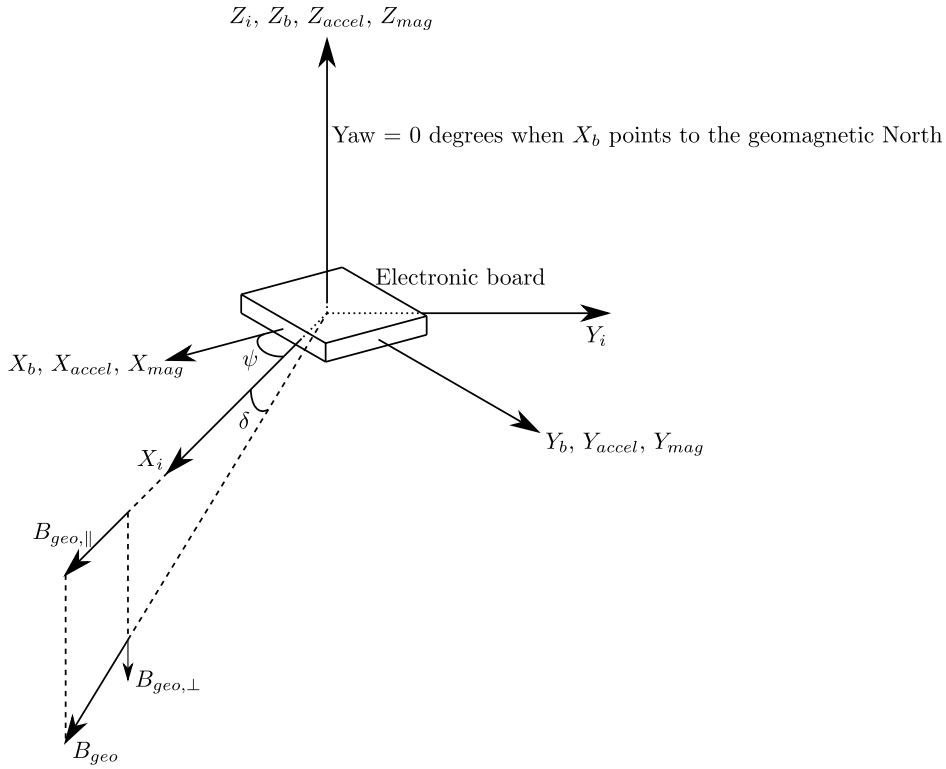


Figure 3.4: Reference frames in the eCompass algorithm.

the magnetometer, respectively, may be given as

$$\mathbf{B}_{\text{mag}} = B \begin{bmatrix} c_{\delta} \\ 0 \\ -s_{\delta} \end{bmatrix}, \quad (3.26)$$

$$\mathbf{g}_{\text{accel}} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}. \quad (3.27)$$

The local magnetic field intensity  $B$  and the declination angle  $\delta$  of the local magnetic field vector  $\mathbf{B}$  may be obtained by using the local GPS coordinates in one of the documented geomagnetic field maps available, such as the WMM (World Magnetic Model) from the National Oceanic and Atmospheric Administration (NOAA) [Oceanic and Administration 2018]. Although these information is useful for calibrating the magnetometer and will be determined afterwards, it does not interfere with the eCompass algorithm, since both values are canceled in the calculations.

Following the adopted Euler angle convention, the ZYX sequence, these measurement

vectors may be given, after a general rotation of the electronic board, as

$$\begin{aligned}\mathbf{B}_{\text{rot}} &= \begin{bmatrix} B_{rx} \\ B_{ry} \\ B_{rz} \end{bmatrix} = \mathbf{R}_i^b \mathbf{B}_{\text{mag}} = \mathbf{R}_{x,\phi} \mathbf{R}_{y,\theta} \mathbf{R}_{z,\psi} \mathbf{B}_{\text{mag}} , \\ \mathbf{g}_{\text{rot}} &= \begin{bmatrix} g_{rx} \\ g_{ry} \\ g_{rz} \end{bmatrix} = \mathbf{R}_i^b \mathbf{g}_{\text{accel}} = \mathbf{R}_{x,\phi} \mathbf{R}_{y,\theta} \mathbf{R}_{z,\psi} \mathbf{g}_{\text{accel}} .\end{aligned}\quad (3.28)$$

in which the subscript  $ri, i \in \{x, y, z\}$  is used for the components of the rotated  $\mathbf{B}_{\text{rot}}$  and  $\mathbf{g}_{\text{rot}}$  vectors. Pre-multiplying Eq. 3.28 by  $\mathbf{R}_{x,-\phi}$  first and then by  $\mathbf{R}_{y,-\theta}$  to remove the roll and pitch rotations gives

$$\begin{aligned}\mathbf{R}_{y,-\theta} \mathbf{R}_{x,-\phi} \mathbf{g}_{\text{rot}} &= \mathbf{R}_{y,-\theta} \mathbf{R}_{x,-\phi} \mathbf{R}_{x,\phi} \mathbf{R}_{y,\theta} \mathbf{R}_{z,\psi} \mathbf{g}_{\text{accel}} = \mathbf{R}_{z,\psi} \mathbf{g}_{\text{accel}} \\ &\quad \mathbf{R}_{y,\theta}^T \mathbf{R}_{x,\phi}^T \mathbf{g}_{\text{rot}} = \mathbf{R}_{z,\psi} \mathbf{g}_{\text{accel}} \\ \begin{bmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \begin{bmatrix} g_{rx} \\ g_{ry} \\ g_{rz} \end{bmatrix} &= \mathbf{R}_{z,\psi} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} ,\end{aligned}\quad (3.29)$$

in which the property  $\mathbf{R}_{-\alpha} = \mathbf{R}_\alpha^{-1} = \mathbf{R}_\alpha^T$  is used. The trick is to notice that any rotation of the gravity vector about the z-axis when it is pointing in this axis does not change the vector components. From Eq. 3.29, it is possible to extract the following formulas from the second and first rows, necessarily in this order,

$$\begin{aligned}g_{ry} c_\phi - g_{rz} s_\phi &= 0 \Rightarrow \tan \phi = \frac{g_{ry}}{g_{rz}} , \\ g_{rx} c_\theta + g_{ry} s_\phi s_\theta + g_{rz} s_\theta c_\phi &= 0 \Rightarrow \tan \theta = \frac{-g_{rx}}{g_{ry} s_\phi + g_{rz} c_\phi} ,\end{aligned}\quad (3.30)$$

which, after applying the adequate inverse trigonometric function, gives the roll and pitch angles of the electronic board. With the roll and pitch angles computed, the magnetic field vector may be rotated back to its original position

$$\begin{aligned}\mathbf{R}_{y,-\theta} \mathbf{R}_{x,-\phi} \mathbf{B}_{\text{rot}} &= \mathbf{R}_{y,-\theta} \mathbf{R}_{x,-\phi} \mathbf{R}_{x,\phi} \mathbf{R}_{y,\theta} \mathbf{R}_{z,\psi} \mathbf{B}_{\text{mag}} = \mathbf{R}_{z,\psi} \mathbf{B}_{\text{mag}} \\ &\quad \mathbf{R}_{y,\theta}^T \mathbf{R}_{x,\phi}^T \mathbf{B}_{\text{rot}} = \mathbf{R}_{z,\psi} \mathbf{B}_{\text{mag}} \\ \begin{bmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \begin{bmatrix} B_{rx} \\ B_{ry} \\ B_{rz} \end{bmatrix} &= \mathbf{R}_{z,\psi} \cdot \begin{bmatrix} B c_\delta \\ 0 \\ -B s_\delta \end{bmatrix} = \begin{bmatrix} c_\psi B c_\delta \\ -s_\psi B c_\delta \\ -B s_\delta \end{bmatrix} \\ \begin{bmatrix} B_{rx} c_\theta + B_{ry} s_\phi s_\theta + B_{rz} s_\theta c_\phi \\ B_{ry} c_\phi - B_{rz} s_\phi \\ -B_{rx} s_\theta + B_{ry} c_\theta s_\phi + B_{rz} c_\theta c_\phi \end{bmatrix} &= \begin{bmatrix} c_\psi B c_\delta \\ -s_\psi B c_\delta \\ -B s_\delta \end{bmatrix} ,\end{aligned}\quad (3.31)$$

and the yaw angle can then be computed, using the first and second rows of Eq. 3.31, as

$$\begin{cases} c_\psi B C_\delta = (B_{rx} c_\theta + B_{ry} s_\theta s_\phi + B_{rz} s_\theta c_\phi) \\ s_\psi B C_\delta = -(B_{ry} c_\phi - B_{rz} s_\phi) \end{cases} \Rightarrow \tan \psi = \frac{B_{rz} s_\phi - B_{ry} c_\phi}{B_{rx} c_\theta + B_{ry} s_\theta s_\phi + B_{rz} s_\theta c_\phi}. \quad (3.32)$$

Since there is more than one solution to the set of angles  $(\phi, \theta, \psi)$ , which is a consequence of the 360-degree periodicity of the trigonometric functions, a convention is adopted: roll and yaw angles are computed with  $\tan_2^{-1}$  - the two-argument arc tangent function - and may vary from  $-180^\circ$  to  $180^\circ$ , whereas the pitch angle is computed with the  $\tan^{-1}$  function and may vary from  $-90^\circ$  to  $90^\circ$ . In Eq. 3.32 it is assumed that the  $B_{ri}$ ,  $i = x, y, z$  components are calibrated for the hard-iron effect in the magnetometer. If it is not true, then the hard-iron offsets  $V_i$ ,  $i = x, y, z$  must be determined with a proper calibration procedure and the yaw angle is determined from raw data as

$$\tan \psi = \frac{(B_{rz} - V_z) s_\phi - (B_{ry} - V_y) c_\phi}{(B_{rx} - V_x) c_\theta + (B_{ry} - V_y) s_\theta s_\phi + (B_{rz} - V_z) s_\theta c_\phi}. \quad (3.33)$$

To illustrate the effect of poor calibration in the magnetometer, a simulation was run with the testbed dynamic model described in this work. In this simulation, an initial angular velocity is applied to the body z-axis of the testbed only, which means that the yaw angle is expected to increase over time infinitely (no friction is considered). Raw data for accelerometer and magnetometer is simulated, considering noise for all measurements and  $V$  offsets for the magnetometer measurements, and the attitude of the testbed is determined. Fig. 3.5 depicts the simulated attitude and the simulated measurements obtained with the eCompass algorithm, showing that, when the magnetometer readings are not calibrated for the  $V$  offsets, the yaw angle appear not to be linear (see expected yaw angle in dashed line).

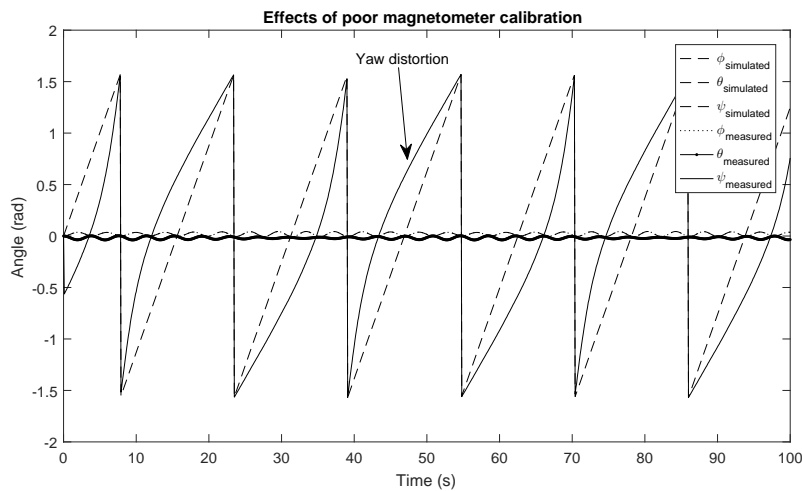


Figure 3.5: Yaw distortion when magnetometer is not calibrated.

Since the balancing algorithms described in this work depend on good attitude estimation, a calibration procedure must be used before running any experiment. The magnetometer

Table 3.1: LSM303DLH chip 6-parameter calibration.

Axis	x	y	z
Offset $V$ ( $\mu T$ )	3.4770	-11.8123	4.9395
Scaling Factor $S$ ( $\mu T$ )	29.9352	29.3514	27.4197

readings are typically submitted to five different sources of errors, namely scaling factors, null shift errors, misalignment errors and hard and soft iron effects. The null shift errors - or offset errors - and the scaling factors are inherent to the sensor manufacturing and are not the same for each of the axes of a three-axial sensor. Also, the misalignment errors are errors introduced by the sensor manufacturing and are perceptible when the sensors in each axis are not perfectly orthogonal within each other. Hard and soft iron effects come as side effects of magnetic field interaction with permanent and induced magnetic fields, respectively, and the first, as a consequence of being constant in the body frame, is usually mathematically indistinguishable from null shift errors [Foster and Elkaim 2008]. Referring to [STMicroelectronics 2018], it is possible to determine 6 calibration parameters for the 3-axis magnetometer: 3 offsets ( $V_{\{x,y,z\}}$ ) and 3 scaling factors ( $S_{\{x,y,z\}}$ ), all of them estimated by fitting an ellipsoid to a set of calibration data through Least Squares Method. One may refer to Appendix C for further details on the adopted magnetometer calibration method and [Foster and Elkaim 2008] for deeper contextualization on the matter. Fig. 3.6 shows the data obtained by rotating the LAICA testbed covering a  $4\pi$  steradian surface and used in the 6-parameter calibration of the 3-axis magnetometer. Table 3.1 shows the obtained calibration parameters used for calibrating the IMU magnetometer. Applying these values to the magnetometer raw data, the magnetic field vector may be normalized to an unit vector which may be then scaled with the local magnetic field intensity (obtained with magnetic maps database such as NOAA WMM or with another standard calibrated magnetometer).

The eCompass algorithm may result in erratic behaviour of the attitude determination because it is known that Eqs. 3.30 and 3.32 do not work well in some singular orientations. However, it is assumed that these orientations will not be achieved during the movement of the platform, due to the restriction imposed by the air bearing configuration as tabletop.

In the following developments, when the attitude must be determined in terms of a quaternion, this quaternion will be determined by transforming the ZYX Euler angles obtained with the method explained in this section to the quaternion representation. This could be done by using the relationships between  $\{\phi, \theta, \psi\}$  and  $\{q_0, q_1, q_2, q_3\}$  in Eq. 3.22. Considering, however, that the direct cosine matrix built by inserting the roll, pitch and yaw angles in Eq. (3.5) may not be precise, the optimal method described in [Bar-Itzhack 2000] will be used to obtain the “best fit” quaternion. In [Bar-Itzhack 2000], it is proposed three versions of the *DCM-to-quaternion* method, two of them for obtaining the quaternion from orthogonal DCMs and one for nonorthogonal DCMs. The third version, which serves the desired purpose, is given by the following steps

1. Given a nonorthogonal  $3 \times 3$  direct cosine matrix corresponding to the testbed attitude,

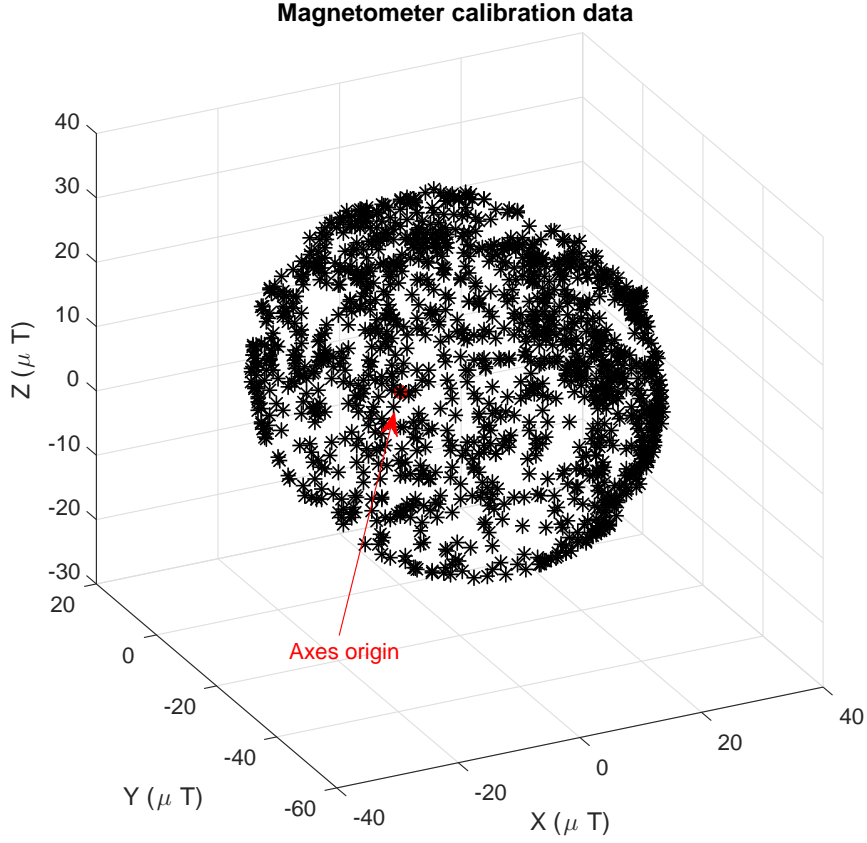


Figure 3.6: LAICA magnetometer calibration data.

such as that in Eq. (3.5), the  $\mathbf{K}_3$  matrix is formed as in (3.34)

$$\mathbf{K}_3 = \frac{1}{3} \begin{bmatrix} d_{11} - d_{22} - d_{33} & d_{21} + d_{12} & d_{31} + d_{13} & d_{23} - d_{32} \\ d_{21} + d_{12} & d_{22} - d_{11} - d_{33} & d_{32} + d_{23} & d_{31} - d_{13} \\ d_{31} + d_{13} & d_{32} + d_{23} & d_{33} - d_{11} - d_{22} & d_{12} - d_{21} \\ d_{23} - d_{32} & d_{31} - d_{13} & d_{12} - d_{21} & d_{11} + d_{22} + d_{33} \end{bmatrix}, \quad (3.34)$$

considering that the attitude DCM of the testbed is given by

$$\mathbf{D} = [d_{ij}]_{3 \times 3} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}. \quad (3.35)$$

2. The  $\lambda_i$  eigenvalues of  $\mathbf{K}_3$  are obtained.
3. The  $\lambda_{max}$  eigenvalue is selected as  $\max_i(\lambda_i)$ .
4. The desired attitude quaternion is calculated as being the eigenvector  $\mathbf{q}_\lambda = [q_{1\lambda} \ q_{2\lambda} \ q_{3\lambda} \ q_{4\lambda}]$  associated with the  $\lambda_{max}$  eigenvalue, in which  $q_{4\lambda}$  is the scalar part of the quaternion.

An important observation is that, as in [Bar-Itzhack 2000] it is considered that the quaternion is given in the  $q_{1:3} + q_4$  order, the obtained quaternion must be rearranged as

$$\mathbf{q}_{attitude} = [q_{4\lambda} \ q_{1\lambda} \ q_{2\lambda} \ q_{3\lambda}]^T \quad (3.36)$$

to follow the convention adopted in this work. Fig. 3.7 shows the percent error of the quaternion components between when they are calculated from time propagation and when they are calculated from [Bar-Itzhack 2000] methods.

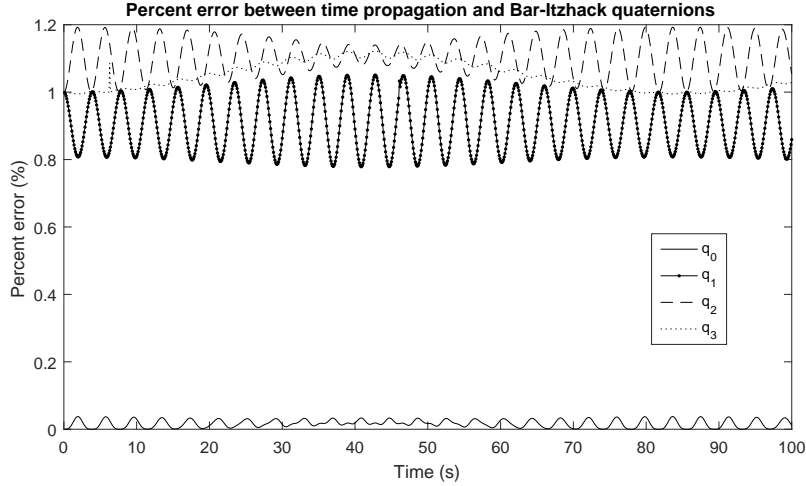


Figure 3.7: Percent error in quaternion determination.

## 3.2 Dynamic Modeling

In order to implement filters and control laws in the testbed, it is necessary to determine a dynamic model for it. The dynamic model rules the relationship between the torques and forces applied to the testbed and translates it into positions, velocities and accelerations. In the present case, the testbed is not capable of translating, which means that only rotational dynamics is developed.

In this work, three models are studied: two of them - a complete and a simplified ones - were studied by [Young 1998], whereas the third one is used recurrently in most works, including [Chesi et al. 2014], and is referred to as the Euler Equations of Motion.

From now on, whenever convenient, the notation used for the cross product between two vectors will be  $[\mathbf{a} \times]$ , which means that this operation is taken in its matrix form, as shown in Eq. 3.37

$$[\mathbf{a} \times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \quad (3.37)$$

in which  $[\mathbf{a} \times]$  is a skew-symmetric matrix.

Also, when time differentiating a vector in a fixed reference frame - *e.g.* the inertial frame - using information about this vector in a rotating reference frame - *e.g.* the body frame, the following definition is used [Hibbeler 2010], which is referred to as the Transport Theorem or the Basic Kinematic Equation (BKE).

**Theorem 3.2.1 Basic Kinematic Equation (BKE).** *Given two reference frames  $\mathbf{i}$  and  $\mathbf{b}$ , being  $\mathbf{b}$  a frame which rotates about the fixed frame  $\mathbf{i}$  with angular velocity  $\omega_i^b$ , the time derivative of a vector  $\mathbf{a}$  expressed in terms of the inertial frame may be obtained as*

$$\left. \frac{d\mathbf{a}}{dt} \right|_i = \left. \frac{d\mathbf{a}}{dt} \right|_b + \omega_i^b \times \mathbf{a} \Big|_i , \quad (3.38)$$

*in which  $\omega_i^b$  is the angular velocity of the rotating frame  $\mathbf{b}$  in relation to the frame  $\mathbf{i}$ , measured in the frame  $\mathbf{i}$ . The subscripts  $\mathbf{b}$  and  $\mathbf{i}$  after the bar are used to define the frame in which the quantity is taken.*

The development of all the dynamic models in this work start with the law of moment of momentum [Wittenburg 2013]. This law, which is a parallel to the Newton's second axiom on translational motion, states that the absolute time derivative, *i.e.* the time derivative taken in a fixed reference frame, of the absolute angular momentum with respect to a reference point O is given by the resultant torque applied to this same point, as shown in Eq. 3.39

$$\frac{d\mathbf{L}_O}{dt} = \mathbf{M}_O . \quad (3.39)$$

Following the procedure adopted by [Young 1998], the angular momentum about the Center of Rotation (CR) of the testbed must be determined as function of the unbalance vector and the angular momentum about the Center of Mass (CM) of the testbed.

Before developing Eq. 3.39, the expressions for the angular momentum about some strategic points must be determined. First, consider an arbitrary rigid body in a inertial reference frame rotating with angular velocity  $\omega$  and consider it is composed of infinite particles of mass  $m_i$ ,  $i = 1, \dots \infty$ . Fig. 3.8 shows this rigid body and the vector quantities to be considered [Hibbeler 2010].

The angular momentum of an arbitrary particle of this body  $\mathbf{H}_{A,i}$  about the arbitrary point A is given by Eq. 3.40,

$$(\mathbf{H}_{A,i}) \Big|_i = \boldsymbol{\rho}_{A,i} \Big|_i \times m_i \mathbf{v}_i \Big|_i , \quad (3.40)$$

in which the right bar indicates the reference system to which each quantity is related,  $\mathbf{v}_i$  is the velocity of the  $i$ -th particle and  $\boldsymbol{\rho}_{A,i}$  is the position vector of the  $i$ -th particle with relation to the point A. This bar is considered implicit from now on. The velocity of this particle contains two portions: one given by translation and another given by the rotation of the body. These quantities are addressed in Eq. 3.41,

$$\mathbf{v}_i = \mathbf{v}_{\text{trans}} + \mathbf{v}_{\text{rot}} = \mathbf{v}_A + \boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i} , \quad (3.41)$$



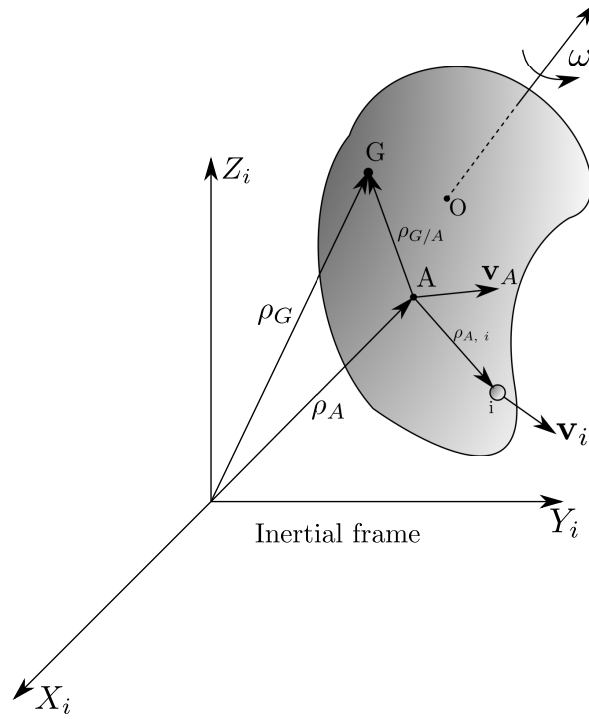


Figure 3.8: An arbitrary rotating rigid body.

in which  $\mathbf{v}_A$  is the translation velocity of the rigid body with relation to the point A. Upon substitution of Eq. 3.41 in Eq. 3.40, Eq. 3.42 is obtained,

$$(\mathbf{H}_{A,i}) = \boldsymbol{\rho}_{A,i} \times m_i \mathbf{v}_A + \boldsymbol{\rho}_{A,i} \times m_i (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}). \quad (3.42)$$

Eq. 3.42 may be integrated for the entire set of particles of the rigid body, considering infinitesimal particles of mass  $dm$ , to obtain the angular momentum of the body about the point A,  $\mathbf{H}_A$ , in Eq. 3.43,

$$\mathbf{H}_{A,i} = \lim_{m_i \rightarrow dm} \sum_i (\mathbf{H}_{A,i}) = \left( \int_m \boldsymbol{\rho}_{A,i} dm \right) \times \mathbf{v}_A + \int_m \boldsymbol{\rho}_{A,i} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) dm. \quad (3.43)$$

Important expressions arise when the point A is strategically selected as the Center of Mass (point G) or the Center of Rotation (point O) of the rotating rigid body. Eqs. 3.44 and 3.45,

$$\mathbf{H}_O = \underbrace{\left( \int_m \boldsymbol{\rho}_O dm \right)}_{\mathbf{0}} \times \underbrace{\mathbf{v}_O}_{\mathbf{0}} + \int_m \boldsymbol{\rho}_O \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_O) dm = \int_m \boldsymbol{\rho}_O \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_O) dm \quad (3.44)$$

and

$$\mathbf{H}_G = \underbrace{\left( \int_m \boldsymbol{\rho}_G dm \right)}_0 \times \mathbf{v}_G + \int_m \boldsymbol{\rho}_O \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_G) dm = \int_m \boldsymbol{\rho}_G \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_G) dm , \quad (3.45)$$

shows that the angular momentum assumes the general form  $\mathbf{H} = \int_m \boldsymbol{\rho} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}) dm$  in these cases. It must be noticed that, from the definition of the Center of Mass, the mass distribution gives  $\int_m \boldsymbol{\rho}_G dm = 0$ , whereas the velocity of the Center of Rotation is given by  $\mathbf{v}_O = 0$ , following from the fact that the CR is a fixed point.

Proceeding from Eq. 3.43 and given that the vector  $\boldsymbol{\rho}_{A,i}$  may be obtained from the vector  $\boldsymbol{\rho}_G$  using the relative position vector  $\boldsymbol{\rho}_{G/A,i}$ , Eq. 3.46 states an useful expression for the angular momentum about an arbitrary point A considering knowledge of the angular momentum  $\mathbf{H}_G$  (see Appendix D),

$$\mathbf{H}_A = \boldsymbol{\rho}_{G/A} \times m\mathbf{v}_G + \mathbf{H}_G , \quad (3.46)$$

Eq. 3.46 will be useful when deriving the dynamic model used by [Young 1998] and to analyze the difference between the models to be considered. Before that, the inertia tensor is defined and its relationship with the angular momentum equation is given.

**Definition 3.2.2** *The Inertia Tensor of a three-dimensional rigid body is defined as the  $3 \times 3$  symmetric matrix  $\mathbf{J}$ , given by*

$$J = \begin{bmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{yx} & J_{yy} & -J_{yz} \\ -J_{zx} & -J_{zy} & J_{zz} \end{bmatrix} . \quad (3.47)$$

*in which the principal moments of inertia are given by*

$$\begin{aligned} J_{xx} &= \int_m (y^2 + z^2) dm , \\ J_{yy} &= \int_m (x^2 + z^2) dm , \\ J_{zz} &= \int_m (x^2 + y^2) dm \end{aligned} \quad (3.48)$$

and the products of inertia are given by

$$\begin{aligned}
 J_{xy} &= J_{yx} = \int_m xy \, dm , \\
 J_{xz} &= J_{zx} = \int_m xz \, dm , \\
 J_{yz} &= J_{zy} = \int_m yz \, dm .
 \end{aligned}
 \tag{3.49}$$

It may be noticed from Eqs. 3.44 and 3.45 that the angular momentum about these points are of the form  $\mathbf{H} = \int_m \boldsymbol{\rho} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}) \, dm$ . Following the procedure described in [Greenwood 1965], this integral is developed as follows. From Eq. 3.50,

$$\mathbf{H} = \int_m \boldsymbol{\rho} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}) \, dm ,
 \tag{3.50}$$

upon substitution of Eqs. 3.51 and 3.52,

$$\boldsymbol{\rho} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}
 \tag{3.51}$$

and

$$\boldsymbol{\omega} = \omega_x\mathbf{i} + \omega_y\mathbf{j} + \omega_z\mathbf{k} ,
 \tag{3.52}$$

the cross products may be developed as shown in Eqs. 3.53 and 3.54,

$$\boldsymbol{\omega} \times \boldsymbol{\rho} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \omega_x & \omega_y & \omega_z \\ x & y & z \end{vmatrix} = (z\omega_y - y\omega_z)\mathbf{i} + (x\omega_z - z\omega_x)\mathbf{j} + (y\omega_x - x\omega_y)\mathbf{k}
 \tag{3.53}$$

and

$$\begin{aligned}
 \boldsymbol{\rho} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}) &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x & y & z \\ (z\omega_y - y\omega_z) & (x\omega_z - z\omega_x) & (y\omega_x - x\omega_y) \end{vmatrix} \\
 &= [(y^2 + z^2)\omega_x - xy\omega_y - xz\omega_z]\mathbf{i} \\
 &\quad + [-yx\omega_x + (x^2 + z^2)\omega_y - yz\omega_z]\mathbf{j} \\
 &\quad + [-zx\omega_x - zy\omega_y + (x^2 + y^2)\omega_z]\mathbf{k} .
 \end{aligned}
 \tag{3.54}$$

Using these expressions for the cross products, the angular momentum equation may be obtained, as in Eq. 3.55, in which the definitions of the principal moments of inertia and the

products of inertia becomes present.

$$\begin{aligned}
\int_m \boldsymbol{\rho} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}) dm &= \left[ \int_m (y^2 + z^2) \omega_x dm + \int_m -xy \omega_y dm + \int_m -xz \omega_z dm \right] \mathbf{i} \\
&+ \left[ \int_m -yx \omega_x dm + \int_m (x^2 + z^2) \omega_y dm + \int_m -yz \omega_z dm \right] \mathbf{j} \\
&+ \left[ \int_m -zx \omega_x dm + \int_m -zy \omega_y dm + \int_m (x^2 + y^2) \omega_z dm \right] \mathbf{k}.
\end{aligned} \tag{3.55}$$

Using the definition of the inertia tensor, the angular momentum may be finally obtained as shown in Eq. 3.56,

$$\begin{aligned}
\mathbf{H} &= (I_{xx} \cdot \omega_x + I_{xy} \cdot \omega_y + I_{xz} \cdot \omega_z) \mathbf{i} \\
&+ (I_{yx} \cdot \omega_x + I_{yy} \cdot \omega_y + I_{yz} \cdot \omega_z) \mathbf{j} \\
&+ (I_{zx} \cdot \omega_x + I_{zy} \cdot \omega_y + I_{zz} \cdot \omega_z) \mathbf{k} \\
&= \mathbf{J} \cdot \boldsymbol{\omega}
\end{aligned} \tag{3.56}$$

In the present case, the system is modeled as a rigid body rotating around a fixed CR and having its CM displaced by an offset vector, the unbalance vector. To develop the Equations Of Motion (EOM) of this system, the angular momentum must be differentiated, as shown in Eq. 3.39. From Eq. 3.46, selecting the arbitrary point A as being the CR of the testbed (point O), gives

$$\mathbf{H}_O = \boldsymbol{\rho}_{G/O} \times m \mathbf{v}_G + \mathbf{H}_G. \tag{3.57}$$

Eq. 3.57 shows that the angular momentum about the CR of the testbed may be obtained as function of the angular momentum about the CM -  $\mathbf{H}_G$  - and the momentum of the linear momentum  $m \mathbf{v}_G$ . Differentiating this equation and applying the BKE, gives

$$\begin{aligned}
\dot{\mathbf{H}}_O &= \frac{d}{dt} \mathbf{H}_O = \frac{d}{dt} (\mathbf{r} \times m \mathbf{v}_G) + \frac{d}{dt} (\mathbf{H}_G) \\
&= \frac{d}{dt} (\mathbf{r} \times m \mathbf{v}_G) \Big|_b + \omega_i^b \times (\mathbf{r} \times m \mathbf{v}_G) + \frac{d}{dt} (\mathbf{H}_G) \Big|_b + \omega_i^b \times (\mathbf{H}_G)
\end{aligned}$$

in which the applicability of the BKE becomes apparent from the fact that the components of  $\mathbf{H}_G$  are taken in the body frame, since the Inertia Tensor is constant in the body frame, whereas is a function of time when taken in the inertial frame. Since  $\frac{d}{dt} \mathbf{r} = \mathbf{v}_G$  by definition,

$$\frac{d}{dt} (\mathbf{r} \times m \mathbf{v}_G) \Big|_b = \underbrace{\frac{d}{dt} \mathbf{r} \times m \mathbf{v}_G + \mathbf{r} \times m \frac{d}{dt} \mathbf{v}_G}_{\mathbf{v}_G \times \mathbf{v}_G = \mathbf{0}} = \mathbf{r} \times m \mathbf{a}_G \tag{3.58}$$

and considering

$$\left. \frac{d}{dt} (\mathbf{H}_G) \right|_b = \left. \frac{d}{dt} (\mathbf{J}\boldsymbol{\omega}_i^b) \right|_b = \underbrace{\frac{d}{dt} \mathbf{J}}_0 \boldsymbol{\omega}_i^b + \mathbf{J} \frac{d}{dt} \boldsymbol{\omega}_i^b = \mathbf{J}\dot{\boldsymbol{\omega}}_i^b \quad (3.59)$$

then, the model developed by [Young 1998] is achieved:

**Definition 3.2.3 Equation of Motion [Young 1998].** The motion of the spacecraft simulator is given by

$$\begin{aligned} \frac{d\mathbf{H}_o}{dt} &= \mathbf{M}_O, \\ \dot{\mathbf{H}}_o &= \mathbf{r} \times m\dot{\mathbf{r}} + \boldsymbol{\omega} \times (\mathbf{r} \times m\dot{\mathbf{r}}) + \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) \end{aligned}$$

in which  $\mathbf{M}_o$  is the external torque applied on the testbed,  $\mathbf{r} = \boldsymbol{\rho}_{G/O}$  is the unbalance vector and  $\boldsymbol{\omega} = \boldsymbol{\omega}_i^b$  is the angular velocity of the body with respect to the inertial frame.

To simulate this model, its equations may be further developed. First, considering Eq. (3.56), the  $\mathbf{J}\boldsymbol{\omega}$  factor may be written as

$$\mathbf{J}\boldsymbol{\omega} = \begin{bmatrix} J_{xx}\omega_x + J_{xy}\omega_y + J_{xz}\omega_z \\ J_{yx}\omega_x + J_{yy}\omega_y + J_{yz}\omega_z \\ J_{zx}\omega_x + J_{zy}\omega_y + J_{zz}\omega_z \end{bmatrix}, \quad (3.60)$$

in which the inertia tensor  $\mathbf{J}$  is the inertia tensor around the center of mass of the testbed. Also, the external moments applied to the testbed may be written as

$$\mathbf{M}_O = \boldsymbol{\tau}_{dev} + \boldsymbol{\tau}_{aero} + \boldsymbol{\tau}_G \quad (3.61)$$

$$\boldsymbol{\tau}_{dev} = \begin{bmatrix} \tau_{dev_x} & \tau_{dev_y} & \tau_{dev_z} \end{bmatrix}^T, \quad (3.62)$$

$$\boldsymbol{\tau}_{aero} = \begin{bmatrix} -B_X\omega_x^2 & -B_Y\omega_y^2 & -B_Z\omega_z^2 \end{bmatrix}, \quad (3.63)$$

$$\boldsymbol{\tau}_G = \mathbf{r} \times (m\mathbf{g}_b) = m g \begin{bmatrix} -r_y c_\phi c_\theta + r_z s_\phi c_\theta \\ r_x c_\phi c_\theta + r_z s_\theta \\ -r_x s_\phi c_\theta - r_y s_\theta \end{bmatrix}, \quad (3.64)$$

in which  $\mathbf{g}_b$  is the gravity vector represented in the body frame and  $\boldsymbol{\tau}_G$  is the gravitational torque. For simulation purposes, aerodynamic drag torque,  $\boldsymbol{\tau}_{aero}$ , characterized by the  $B_X$ ,  $B_Y$  and  $B_Z$  coefficients, and actuator devices torques,  $\boldsymbol{\tau}_{dev}$ , may be added.

Making the proper substitutions, the model equation may be written as

$$\mathbf{A}\dot{\boldsymbol{\omega}} + \mathbf{B} = \mathbf{M}, \quad (3.65)$$

in which the  $\mathbf{A}$ ,  $\mathbf{B}$  and  $\mathbf{M}$  matrices are given by Eqs. (3.66),

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} mr_y^2 + mr_z^2 + J_{xx} & -mr_x r_y + J_{xy} & -mr_x r_z + J_{xz} \\ -mr_x r_y + J_{xy} & mr_x^2 + mr_z^2 + J_{yy} & -mr_y r_z + J_{yz} \\ -mr_x r_z + J_{xz} & -mr_y r_z + J_{yz} & mr_x^2 + mr_y^2 + J_{zz} \end{bmatrix}_{3 \times 3}, \\
\mathbf{B} &= \begin{bmatrix} [(-2mr_y r_z + J_{zy})\omega_y^2 + (2mr_y r_z - J_{yz})\omega_z^2 + \\ + (-mr_x r_z + J_{xz})\omega_x \omega_y + (mr_x r_y - J_{xy})\omega_x \omega_z + \\ + (mr_y^2 - mr_z^2 - J_{yy} + J_{zz})\omega_y \omega_z] \\ [ (2mr_x r_z - J_{zx})\omega_x^2 + (-2mr_x r_z + J_{xz})\omega_z^2 + \\ + (mr_y r_z - J_{zy})\omega_x \omega_y + (-mr_x r_y + J_{xy})\omega_y \omega_z + \\ + (-mr_x^2 - mr_z^2 + J_{xx} - J_{zz})\omega_x \omega_z] \\ [(-2mr_x r_y + J_{xy})\omega_x^2 + (2mr_x r_y - J_{xy})\omega_y^2 + \\ + (-mr_y r_z + J_{yz})\omega_x \omega_z + (-mr_x r_z + J_{xz})\omega_y \omega_z + \\ + (mr_x^2 - mr_y^2 - J_{xx} + J_{yy})\omega_x \omega_y] \end{bmatrix}_{3 \times 1}, \\
\mathbf{M} &= \begin{bmatrix} \tau_{dev_x} - B_x \omega_x^2 - mgr_y c_\phi c_\theta + mgr_z s_\phi c_\theta \\ \tau_{dev_y} - B_y \omega_y^2 + mgr_x c_\phi c_\theta + mgr_z s_\theta \\ \tau_{dev_z} - B_z \omega_z^2 - mgr_x s_\phi c_\theta - mgr_y s_\theta \end{bmatrix}_{3 \times 1},
\end{aligned} \tag{3.66}$$

which can be isolated for the angular accelerations as

$$\dot{\boldsymbol{\omega}} = \mathbf{A}^{-1}(\mathbf{M} - \mathbf{B}). \tag{3.67}$$

This model can be simulated along with the Euler rates equation to obtain the motion of the testbed. The code developed by [Young 1998] for simulating this model was adapted and is present in Appendix K.

Def. 3.2.3 can be simplified considering some assumptions, such as considering  $\mathbf{r}$ ,  $\boldsymbol{\omega}$  and the products of inertia  $J_{ij}$ ,  $i \neq j$  small with relation to the other terms, as well as considering that the only external torque present in the system is the gravitational torque. This gives the second model described in this work:

**Definition 3.2.4 Simplified Equation of Motion [Young 1998].** *The simplified motion of the spacecraft simulator is given by*

$$\dot{\boldsymbol{\omega}} = \begin{bmatrix} \frac{mg}{J_x} (-r_y c_\phi c_\theta + r_z s_\phi c_\theta) \\ \frac{mg}{J_y} (r_x c_\phi c_\theta + r_z s_\theta) \\ \frac{mg}{J_z} (-r_x s_\phi c_\theta - r_y s_\theta) \end{bmatrix} \tag{3.68}$$

*in which  $c$ . and  $s$ . operators are used to denote the sine and cosine of the roll and pitch angles,  $\phi$  and  $\theta$ .*

Finally, the third model to be considered is that used by [Chesi et al. 2014] and [Kim and Agrawal 2009]. This model considers the dynamics of the simulator with respect to the center of rotation as being given by the known

Euler Equations of Motion. In this case, the angular momentum  $\mathbf{H}_O$  is used directly. Besides this, the only external torque to be considered in this model is the gravitational torque. This model is defined as follows.

**Definition 3.2.5 Euler Equations of Motion [Chesi et al. 2014].** *The motion of the spacecraft simulator is given by the Euler EOM with respect to the CR,*

$$\begin{aligned}\frac{d\mathbf{H}_O}{dt} &= \mathbf{M}_O, \\ \dot{\mathbf{H}}_O &= \dot{\mathbf{H}}_O|_b + \boldsymbol{\omega} \times \mathbf{H}_O|_b = \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \\ \mathbf{M}_O &= \mathbf{r} \times \mathbf{F}_G = \mathbf{r} \times m\mathbf{g}_b,\end{aligned}$$

*in which  $\mathbf{g}_b$  is the gravity vector with respect to the body frame.*

One of the main differences in this model is that the Inertia Tensor  $\mathbf{J}$  is not the same as that used in the model developed by [Young 1998]. In this case, the Inertia Tensor is developed with relation to the CR of the system, whereas the inertia tensor used in the Young's model is developed with relation to the CM of the system. Still, both tensors are constant with relation to the body frame and can be used without changing the model complexity.

To understand the difference between these tensors, one must record the parallel axis theorem applied to 3D motion, which states that

**Theorem 3.2.6 Parallel Axis Theorem** *Considering the principal moments of inertia,  $J_{xx}$ ,  $J_{yy}$  and  $J_{zz}$ , and the products of inertia  $J_{xy}$ ,  $J_{xz}$  and  $J_{yz}$ , of a rigid body composed the inertia tensor with respect to the CM,  $\mathbf{J}_G$ , then the parallel axis theorem states that the inertia tensor about any arbitrary point may be obtained as*

$$\mathbf{J}_A = \begin{bmatrix} J'_{xx} & -J'_{xy} & -J'_{xz} \\ -J'_{yx} & J'_{yy} & -J'_{yz} \\ -J'_{zx} & -J'_{zy} & J'_{zz} \end{bmatrix}. \quad (3.69)$$

*in which the principal moments of inertia are given by*

$$\begin{aligned}J'_{xx} &= J_{xx} + m(y_G^2 + z_G^2), \\ J'_{yy} &= J_{yy} + m(x_G^2 + z_G^2), \\ J'_{zz} &= J_{zz} + m(x_G^2 + y_G^2),\end{aligned} \quad (3.70)$$

Table 3.2: Maximum percent error for various simulation conditions.

Simulation conditions			Maximum error in $\omega$ (%)
<b>J</b>	$\omega_{t=0}^a$	<b>r</b>	0.0049346%
<b>10J</b>	$\omega_{t=0}$	<b>r</b>	$2.1227 \cdot 10^{-5}\%$
<b>10<sup>2</sup>J</b>	$\omega_{t=0}$	<b>r</b>	$4.5847 \cdot 10^{-8}\%$
<b>10<sup>3</sup>J</b>	$\omega_{t=0}$	<b>r</b>	$1.9938 \cdot 10^{-10}\%$
<b>J</b>	$10\omega_{t=0}$	<b>r</b>	0.021518%
<b>J</b>	$10^2\omega_{t=0}$	<b>r</b>	0.32%
<b>J</b>	$10^3\omega_{t=0}$	<b>r</b>	190.50%
<b>J</b>	$\omega_{t=0}$	<b>10r</b>	1.96%
<b>J</b>	$\omega_{t=0}$	<b>10<sup>2</sup>r</b>	6.27%
<b>J</b>	$\omega_{t=0}$	<b>10<sup>3</sup>r</b>	0.078532%

$$^a: \omega_{t=0} = [0.01 \ 0.01 \ 0.01]^T \text{ rad/s}$$

*the products of inertia are given by*

$$\begin{aligned} J'_{xy} &= J_{xy} + mx_G y_G, \\ J'_{xz} &= J_{xz} + mx_G z_G, \\ J'_{yz} &= J_{yz} + my_G z_G \end{aligned} \quad (3.71)$$

*and the components  $x_G$ ,  $y_G$  and  $z_G$  define the position of the point with respect to the CM.*

In other words, the relationship between the inertia tensor considered about the center of rotation and considered about the center of mass is given by

$$\mathbf{J}_O = \mathbf{J}_G + \mathbf{J}_+, \quad (3.72)$$

in which  $\mathbf{J}_+$  is given by

$$\mathbf{J}_+ = \begin{bmatrix} m(y_G^2 + z_G^2) & -mx_G y_G & -mx_G z_G \\ -mx_G y_G & m(x_G^2 + z_G^2) & -mz_G y_G \\ -mx_G z_G & -mz_G y_G & m(x_G^2 + y_G^2) \end{bmatrix}. \quad (3.73)$$

Fig. 3.9 shows the difference between the angular velocities obtained by simulating the models by [Chesi et al. 2014] and [Young 1998] using the same model conditions (same inertia tensor and unbalance vector, angular velocity initially null and attitude angles initially null) for a 100 s simulation. Also, Table 3.2 shows how the magnitude of the error increases when the values of angular velocity, inertia and unbalance distances increases.

Analyzing the data of these tests, it could be seen that the error caused by **r** is the most critical and has the trend to increase with time, while others tend to vary cyclically.



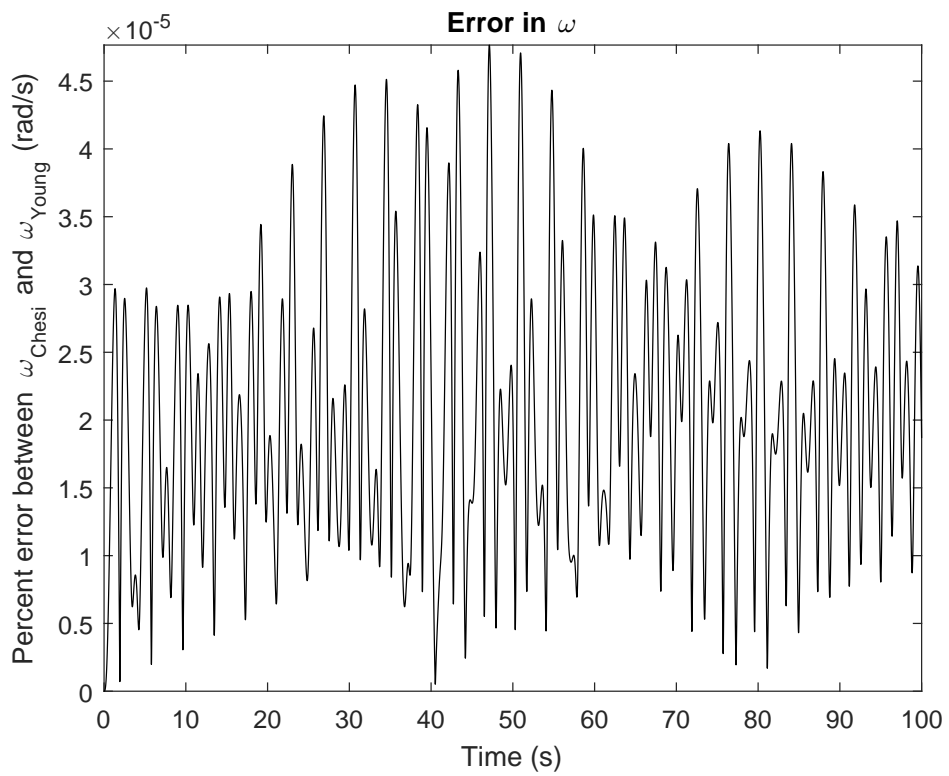


Figure 3.9: Error between [Young 1998] and [Chesi et al. 2014] simulated angular velocities.

# Chapter 4

## Balancing Techniques

*Control, control, you must learn control!*  
- Yoda

After having described the kinematic and dynamic models to be considered in this work, the balancing techniques can now be proposed. In this chapter the following techniques will be explained:

- Batch and recursive Least Squares Methods (LSM), page 41,
- Linear Kalman filtering, page 48,
- Nonlinear Kalman filtering (EKF and UKF), page 53 and
- Hybrid adaptive control method, page 72.

### 4.1 Estimating through Least Squares Method

The Least Squares Method was extensively studied in [Young 1998], [Da Silva and Rodrigues 2015], [Da Silva et al. 2016] and [Silva et al. 2018], for which reason this procedure is simply summarized in this section. Rewriting Eq. (3.68),

$$\dot{\boldsymbol{\omega}} = \begin{bmatrix} \frac{mg}{J_x} (-r_y c_\phi c_\theta + r_z s_\phi c_\theta) \\ \frac{mg}{J_y} (r_x c_\phi c_\theta + r_z s_\theta) \\ \frac{mg}{J_z} (-r_x s_\phi c_\theta - r_y s_\theta) \end{bmatrix}, \quad (4.1)$$

it must be noticed that  $m, g, J_{xx}, J_{yy}, J_{zz}$  and  $\mathbf{r} = [r_x \ r_y \ r_z]^T$  are all constants and the  $\mathbf{r}$  vector is the parameters vector to be estimated. The measurements in this model are the angular acceleration components, which will be estimated, through discretization, with the angular velocity components  $\omega_x, \omega_y$  and  $\omega_z$ , and the orientation given in the ZYX Euler angles, roll ( $\phi$ ), pitch ( $\theta$ ) and yaw ( $\psi$ ).

The angular acceleration may be discretized as

$$\dot{\boldsymbol{\omega}} \approx \frac{\Delta\boldsymbol{\omega}^k}{T} = \frac{1}{T} \begin{bmatrix} \omega_x^{k+1} - \omega_x^k \\ \omega_y^{k+1} - \omega_y^k \\ \omega_z^{k+1} - \omega_z^k \end{bmatrix}, \quad (4.2)$$

in which  $T$  is the sampling time. Using the trapezoidal rule, the equations of the simplified model may be written as

$$\begin{aligned} \Delta\omega_x^k &= \omega_x^{k+1} - \omega_x^k = \frac{-mgT}{2J_{xx}} \{ [(c_\phi c_\theta)^{k+1} + (c_\phi c_\theta)^k] r_y - [(s_\phi c_\theta)^{k+1} + (s_\phi c_\theta)^k] r_z \}, \\ \Delta\omega_y^k &= \omega_y^{k+1} - \omega_y^k = \frac{mgT}{2J_{yy}} \{ [(c_\phi c_\theta)^{k+1} + (c_\phi c_\theta)^k] r_x + [(s_\theta)^{k+1} + (s_\theta)^k] r_z \} \\ \Delta\omega_z^k &= \omega_z^{k+1} - \omega_z^k = \frac{-mgT}{2J_{zz}} \{ [(s_\phi c_\theta)^{k+1} + (s_\phi c_\theta)^k] r_x + [(s_\theta)^{k+1} + (s_\theta)^k] r_y \}. \end{aligned} \quad (4.3)$$

Eq. 4.3 can be written in a linear form as

$$\begin{aligned} \Delta\boldsymbol{\Omega} &= \boldsymbol{\phi} \mathbf{r} \\ \underbrace{\begin{bmatrix} \Delta\omega_x^k \\ \Delta\omega_y^k \\ \Delta\omega_z^k \end{bmatrix}}_{\Delta\boldsymbol{\Omega}} &= \underbrace{\begin{bmatrix} 0 & \phi_{12} & \phi_{13} \\ \phi_{21} & 0 & \phi_{23} \\ \phi_{31} & \phi_{32} & 0 \end{bmatrix}}_{\boldsymbol{\phi}} \underbrace{\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}}_{\mathbf{r}}, \end{aligned} \quad (4.4)$$

where the  $\phi_{ij}^k$  terms are given by

$$\left[ \begin{array}{c|c} \phi_{12}^k = -\frac{mgT}{2J_{xx}} ((c_\phi c_\theta)^{k+1} + (c_\phi c_\theta)^k) & \phi_{13}^k = \frac{mgT}{2J_{xx}} ((s_\phi c_\theta)^{k+1} + (s_\phi c_\theta)^k) \\ \phi_{21}^k = \frac{mgT}{2J_{yy}} ((c_\phi c_\theta)^{k+1} + (c_\phi c_\theta)^k) & \phi_{23}^k = \frac{mgT}{2J_{yy}} ((s_\theta)^{k+1} + (s_\theta)^k) \\ \phi_{31}^k = -\frac{mgT}{2J_{zz}} ((s_\phi c_\theta)^{k+1} + (s_\phi c_\theta)^k) & \phi_{32}^k = -\frac{mgT}{2J_{zz}} ((s_\theta)^{k+1} + (s_\theta)^k) \end{array} \right]. \quad (4.5)$$

The Least Squares Method applied to this problem consists in augmenting the matrix equation in Eq. 4.4 for obtaining a more accurate estimative of the unbalance vector, since the noise inherent to the angular velocities  $\boldsymbol{\omega}$  and the attitude angles  $\{\phi, \theta, \psi\}$  prevents the possibility of obtaining a good estimative through simple matrix inversion. Then, augment-

ing Eq. 4.4 leads to

$$\Delta\Omega_{aug} = \phi_{aug}\mathbf{r} \quad (4.6)$$

$$\underbrace{\begin{bmatrix} \Delta\omega_x^0 \\ \Delta\omega_y^0 \\ \Delta\omega_z^0 \\ \hline \Delta\omega_x^1 \\ \Delta\omega_y^1 \\ \Delta\omega_z^1 \\ \hline \vdots \end{bmatrix}}_{\Delta\Omega_{aug}} \quad 3n \times 1 = \underbrace{\begin{bmatrix} 0 & \phi_{12}^0 & \phi_{13}^0 \\ \phi_{21}^0 & 0 & \phi_{23}^0 \\ \phi_{31}^0 & \phi_{32}^0 & 0 \\ \hline 0 & \phi_{12}^1 & \phi_{13}^1 \\ \phi_{21}^1 & 0 & \phi_{23}^1 \\ \phi_{31}^1 & \phi_{32}^1 & 0 \\ \hline \vdots & \vdots & \vdots \end{bmatrix}}_{\phi_{aug}} \quad 3n \times 3 \quad \underbrace{\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}}_{\mathbf{r}} \quad 3 \times 1, \quad (4.7)$$

in which  $n$  is the selected number of batch estimates of Eq. (4.4). The unbalance vector may then be determined, using the pseudoinverse of  $\phi_{aug}$ , as

$$\mathbf{r} = [\phi_{aug}^T \phi_{aug}]^{-1} \phi_{aug}^T \Delta\Omega_{aug}. \quad (4.8)$$

The adjustment of the CM position may be then accomplished by moving the movable masses orthogonally disposed onto the testbed. The calculation may be simplified by considering that the testbed without the movable masses is already balanced and the unbalance is due solely by the addition of the movable mass on the table. Considering this, the unbalance vector may be calculated as

$$\begin{aligned} \mathbf{r} &= \frac{1}{m} \sum_i m_i \mathbf{r}_i, \\ &= m_0 \mathbf{0} + m_x r_{mx} \mathbf{i} + m_y r_{my} \mathbf{j} + m_z r_{mz} \mathbf{k}, \\ &= \begin{bmatrix} \frac{m_x}{m} r_{mx} \\ \frac{m_y}{m} r_{my} \\ \frac{m_z}{m} r_{mz} \end{bmatrix}, \end{aligned} \quad (4.9)$$

$$m = m_0 + m_x + m_y + m_z,$$

in which  $m_0$  is the mass of the testbed without movable masses,  $m_{\{x,y,z\}}$  are the masses of each movable mass and  $r_{\{mx,my,mz\}}$  are the positions of the masses along its corresponding axes. Considering that the interest relies on the variation  $\delta_m$  applied to the mass positions to compensate the unbalancing, one may write

$$\mathbf{r}_{new} = \mathbf{0} = \begin{bmatrix} \frac{m_x}{m} (r_{mx} + \delta_{mx}) \\ \frac{m_y}{m} (r_{my} + \delta_{my}) \\ \frac{m_z}{m} (r_{mz} + \delta_{mz}) \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{m_x}{m} r_{mx} \\ \frac{m_y}{m} r_{my} \\ \frac{m_z}{m} r_{mz} \end{bmatrix}}_{=\mathbf{r}, \text{ Eq. 4.9}} + \begin{bmatrix} \frac{m_x}{m} \delta_{mx} \\ \frac{m_y}{m} \delta_{my} \\ \frac{m_z}{m} \delta_{mz} \end{bmatrix} \quad (4.10)$$

Thus, isolating the  $\delta_m = [\delta_{mx} \delta_{my} \delta_{mz}]^T$  vector from Eq. (4.10), the position variations of the movable masses are determined as

$$\delta_m = -m \begin{bmatrix} \frac{r_x}{m_x} \\ \frac{r_y}{m_y} \\ \frac{r_z}{m_z} \end{bmatrix} \quad (4.11)$$

which, considering all the movable masses displace equal amounts of mass, *i.e.*  $k = m_x = m_y = m_z$ , may be simplified as

$$\delta_m = -\frac{m}{k} \mathbf{r} . \quad (4.12)$$

This was the balancing procedure adopted in the work developed in [Da Silva and Rodrigues 2015]. It is most unlikely that the testbed balancing will reach reasonable results in only one batch estimation. For this reason, multiple balancing iterations (“estimation-plus-actuation”) must be performed. To optimize this process, the batch size was studied in order to establish the minimum usable batch size. Fig. 4.1 shows how the unbalance vector estimate converges as function of the batch size, evidencing that after about 50 *s* of measurements acquired at 10 *Hz* rate (a total of 500 measurements) the final estimate and its standard deviation has acceptable variation.

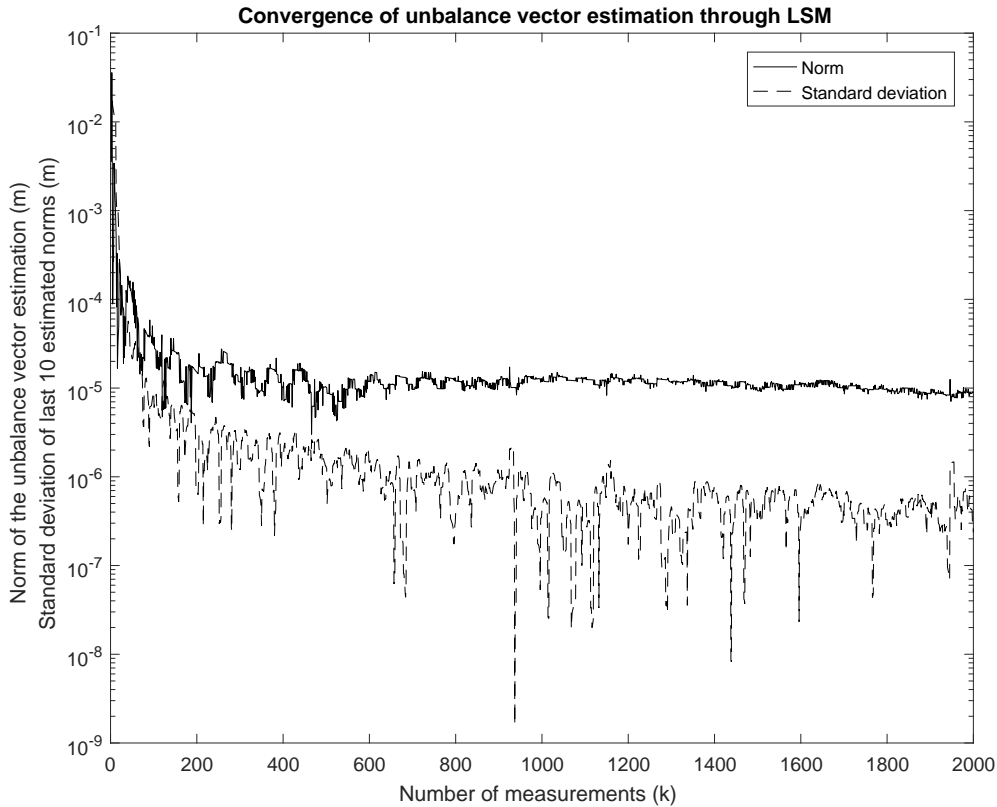


Figure 4.1: Parameter estimation convergence as function of the batch size.

Also, considering the practical design of the balancing system, sometimes the measure-

ments may be acquired at a variable sampling rate or even present huge time gaps between measurements when the electronic system fails (*e.g.* when the battery level goes low or the wireless communication fails). To study the effect of this problem in the estimation process, a batch of measurements was obtained through simulation and portions of the data were eliminated, *i.e.* if the set of measurements is given by  $\{\mathbf{x}_0, \dots, \mathbf{x}_k\}$ , then the modified set is given by  $\{\mathbf{x}_0, \dots, \mathbf{x}_n, \mathbf{x}_{n+l}, \dots, \mathbf{x}_k\}$ , in which the set  $\{\mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+l-1}\}$  was removed. Fig. 4.2 shows the estimate convergence when two sets of measurements are eliminated,  $\{\mathbf{x}_{500}, \dots, \mathbf{x}_{700}\}$  and  $\{\mathbf{x}_{1500}, \dots, \mathbf{x}_{1700}\}$ .

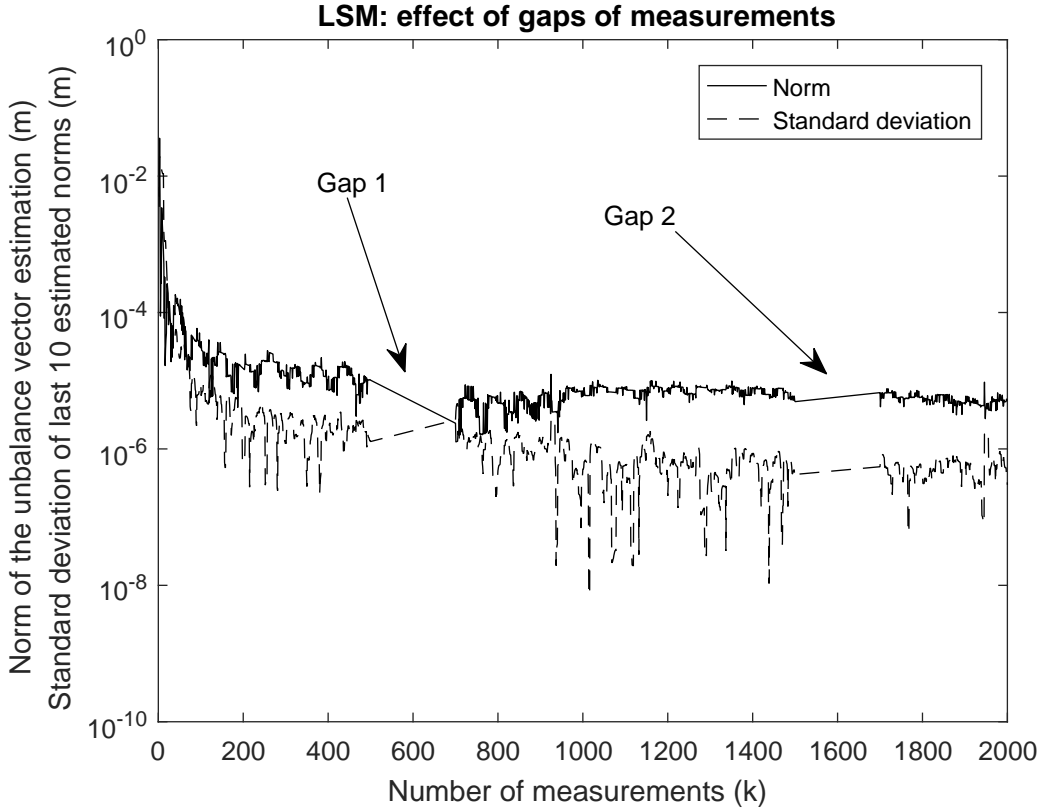


Figure 4.2: Parameter estimation convergence as function of the batch size.

As can be seen in Fig. 4.2, there is little effect of these measurement gaps in the parameter convergence. However, attention must be given for ensuring that the measured attitude angles are synchronized with the measured angular velocities. Fig. 4.3 depicts the effect of the synchronization fault of the roll angle in a extreme example in which the roll angle becomes 50 samples advanced in time with relation to the other measurements. This figure shows that the unbalance vector estimation starts to oscillate right after the data becomes unsynchronized.

There are some improvements that may be made in the balancing method presented in this section. One of these improvements is described in [Xu et al. 2015], [Sharifi et al. 2017], [Krishnanunni et al. 2018] and, primarily, in [Kim and Agrawal 2009] and refers to the possibility of estimating not only the unbalance vector, but also the inertia tensor components. Referring to the model in 3.2.5, a term for

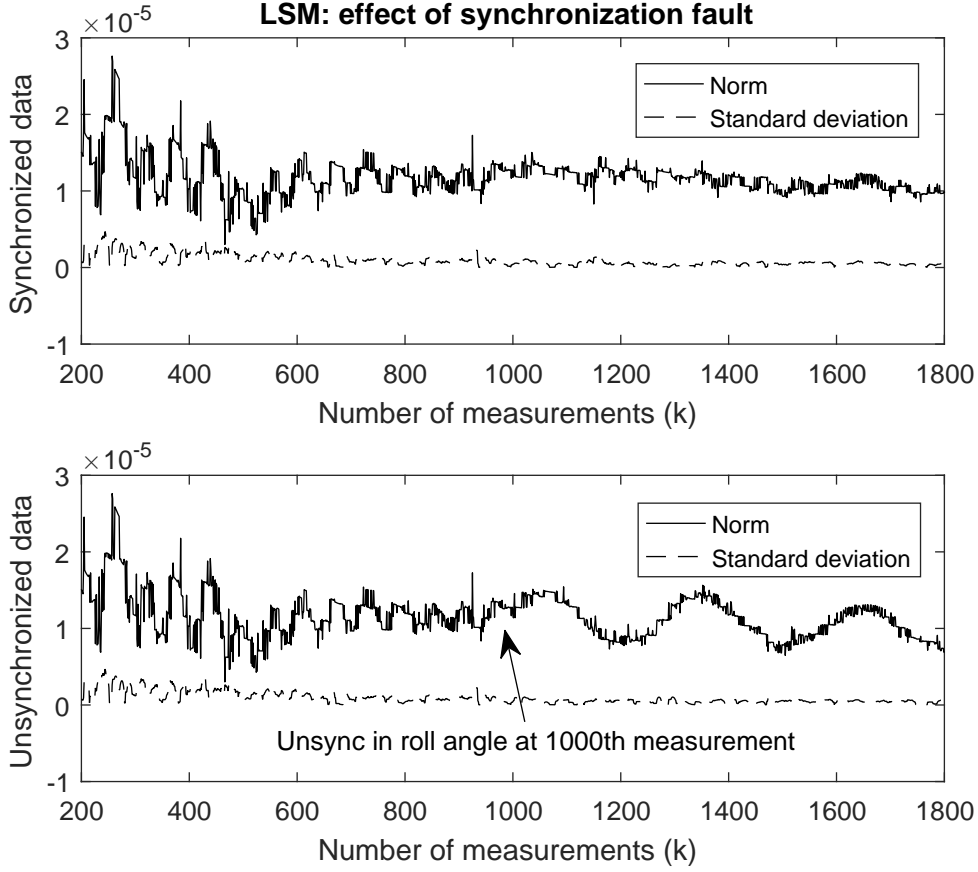


Figure 4.3: Effect of a synchronization fault.

representing the torque generated by momentum exchanging devices is added, which means that

$$\frac{d\mathbf{H}_O}{dt} = \frac{d\mathbf{J}\boldsymbol{\omega}}{dt} + \frac{d\mathbf{h}}{dt} = \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \dot{\mathbf{h}} + \boldsymbol{\omega} \times \mathbf{h}, \quad (4.13)$$

in which the BKE is applied and the superscripts and subscripts referring to coordinate frames are suppressed. Considering the derivative of the angular momentum equals the applied external torque, considering only the gravitational torque is present and rearranging terms, Eq. 4.13 becomes

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = -\dot{\mathbf{h}} - \boldsymbol{\omega} \times \mathbf{h} - [\mathbf{g}\times]m\mathbf{r}, \quad (4.14)$$

in which the matrix form of the cross product and the identity  $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$  are used. Considering the parameters vector to be estimated is  $\mathbf{x} = [J_x \ J_y \ J_z \ J_{xy} \ J_{xz} \ J_{yz} \ mr_x \ mr_y \ mr_z]^T$ , Eq. 4.13 may be rearranged in the linear form as

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{\omega} & [\mathbf{g}\times] \end{bmatrix} \underbrace{\begin{bmatrix} \tilde{\mathbf{J}} \\ m\mathbf{r} \end{bmatrix}}_{\mathbf{x}} = -\dot{\mathbf{h}} - \boldsymbol{\omega} \times \mathbf{h} \quad (4.15)$$

in which  $\Omega$ ,  $\tilde{\mathbf{J}}$  and  $[\mathbf{g}\times]$  are given by

$$\Omega = \begin{bmatrix} \omega_1 & 0 & 0 & \omega_2 & \omega_3 & 0 \\ 0 & \omega_2 & 0 & \omega_1 & 0 & \omega_3 \\ 0 & 0 & \omega_3 & 0 & \omega_1 & \omega_2 \end{bmatrix} \quad (4.16)$$

$$\tilde{\mathbf{J}} = \begin{bmatrix} J_x & J_y & J_z & J_{xy} & J_{xz} & J_{yz} \end{bmatrix}^T \quad (4.17)$$

$$[\mathbf{g}\times] = [\mathbf{g}_b\times] = g \begin{bmatrix} 0 & -c_\phi c_\theta & s_\phi c_\theta \\ c_\phi c_\theta & 0 & s_\theta \\ -s_\phi c_\theta & -s_\theta & 0 \end{bmatrix}. \quad (4.18)$$

Before applying the Least Squares Method (LSM), it must be noticed that Eq. (4.15) utilizes the derivative of the angular rates, which is not desirable, since it can amplify the noises in these signals. To relieve some of this effect, Eq. (4.15) is integrated over time

$$\underbrace{\begin{bmatrix} \Omega + \int_{t_0}^t \boldsymbol{\omega} \times \Omega dt & \int_{t_0}^t [\mathbf{g}\times] dt \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} \tilde{\mathbf{J}} \\ m\mathbf{r} \end{bmatrix}}_{\mathbf{x}} = \underbrace{-\mathbf{h} - \int_{t_0}^t \boldsymbol{\omega} \times \mathbf{h} dt}_{\mathbf{H}} \quad (4.19)$$

and the LSM for  $n$  samples may be applied as

$$\mathbf{x} = (\Phi_{aug}^T \Phi_{aug})^{-1} \Phi_{aug}^T \mathbf{H}_{aug}, \quad (4.20)$$

in which  $\Phi_{aug}$  and  $\mathbf{H}_{aug}$  are given by

$$\Phi_{aug} = \begin{bmatrix} \Phi_0 \\ \Phi_1 \\ \vdots \\ \Phi_k \end{bmatrix}_{3n \times 9}, \quad (4.21)$$

$$\Phi_k = \begin{bmatrix} \Omega + \int_{t_0}^{t_k} \boldsymbol{\omega} \times \Omega dt & \int_{t_0}^{t_k} [\mathbf{g}\times] dt \end{bmatrix}_{3 \times 9}, \quad (4.22)$$

$$\mathbf{H}_{aug} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_k \end{bmatrix}_{3n \times 1}, \quad (4.23)$$

$$\mathbf{H}_k = \begin{bmatrix} -\mathbf{h} - \int_{t_0}^{t_k} \boldsymbol{\omega} \times \mathbf{h} dt \end{bmatrix}_{3 \times 1}. \quad (4.24)$$

Another improvement would be implementing this parameter estimation recursively. In this way, there is no need of acquiring a set of measurements to proceed with a batch estimation of the desired parameters. This has special utility in embedded systems, in which memory and computational effort requisites are more restrict. Starting from the modified



loss function in Eq. (4.25),

$$V(t) = \sum_{n=1}^k \lambda^{k-n} (\mathbf{H}_n - \Phi_n^T x), \quad (4.25)$$

the work described in [Sharifi et al. 2017] designs the Recursive Least Squares (RLS) approach for Eq. (4.19), following the procedure described in [Haykin 1986], as

$$\begin{aligned} \mathbf{K}_k &= \frac{\lambda^{-1} \mathbf{P}_{k-1} \Phi_k}{1 + \lambda^{-1} \Phi_k^T \mathbf{P}_{k-1} \Phi_k}, \\ \alpha_k &= \mathbf{H}_k - \Phi_k^T \mathbf{x}_{k-1}, \\ \mathbf{x}_k &= \mathbf{x}_{k-1} + \mathbf{K}_k \alpha_k, \\ \mathbf{P}_k &= \lambda^{-1} \mathbf{P}_{k-1} - \lambda^{-1} \mathbf{K}_k \Phi_k^T \mathbf{P}_{k-1}, \end{aligned}$$

where  $\alpha_k$  is a residual estimation error at time  $k$  and  $\lambda$  is the forgetting factor, a scalar which defines how the past observations are considered in the estimation process. The  $\lambda$  parameter ranges in the  $(0; 1]$  interval and values near 1 indicates that only recent observations are taken into account. Another advantage of this recursive estimation is that it does not require any matrix inversion. Another technique, based on the Classical Levenberg-Marquadt (CLM) algorithm, is explained in [Sharifi et al. 2017] and compared with the results obtained by RLS estimation. For further details, one must refer to this work. Recursive estimation may be also achieved using the well-known Kalman filter, as will be shown in the next section.

## 4.2 Estimating unbalance with a Kalman Filter

Section 4.1 showed how to estimate the unbalance vector recursively. Another recursive approach may be given to this problem. As it can be seen in Def. 3.2.4, the simplified dynamic model is linear with respect to the unbalance vector parameters. To explicit this relation, Eq. 4.4 may be rewritten as

$$\begin{bmatrix} \omega_x^{k+1} \\ \omega_y^{k+1} \\ \omega_z^{k+1} \end{bmatrix} = \begin{bmatrix} \omega_x^k \\ \omega_y^k \\ \omega_z^k \end{bmatrix} + \begin{bmatrix} 0 & \phi_{12}^k & \phi_{13}^k \\ \phi_{21}^k & 0 & \phi_{23}^k \\ \phi_{31}^k & \phi_{32}^k & 0 \end{bmatrix} \begin{bmatrix} r_x^k \\ r_y^k \\ r_z^k \end{bmatrix} \quad (4.26)$$

and, choosing the augmented state vector  $\mathbf{x}^k = [\omega_x^k \ \omega_y^k \ \omega_z^k \ r_x^k \ r_y^k \ r_z^k]^T$ , in which the unbalance vector is added as a static parameter (*i.e.*  $\dot{\mathbf{r}} = 0$ ), the discretized dynamic equation becomes

$$\mathbf{x}_{k+1} = \begin{bmatrix} \omega_x^{k+1} \\ \omega_y^{k+1} \\ \omega_z^{k+1} \\ \gamma_x^{k+1} \\ \gamma_y^{k+1} \\ \gamma_z^{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \phi_{12} & \phi_{13} \\ 0 & 1 & 0 & \phi_{21} & 0 & \phi_{23} \\ 0 & 0 & 1 & \phi_{31} & \phi_{32} & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}_k} \begin{bmatrix} \omega_x^k \\ \omega_y^k \\ \omega_z^k \\ \gamma_x^k \\ \gamma_y^k \\ \gamma_z^k \end{bmatrix} + \mathbf{w}_{6 \times 1}, \quad (4.27)$$

$$= \mathbf{F}_k \mathbf{x}_k + \mathbf{w}_k \quad (4.28)$$

in which  $\mathbf{w}_{6 \times 1}$  represents the process noise. The angular velocities may be obtained from the sensor, whereas the unbalance vector components may be only estimated. In other words, the output equation is given by

$$\mathbf{y}_k = \mathbf{H} \cdot \mathbf{x}_k + \mathbf{v}_k \quad (4.29)$$

in which  $\mathbf{H} = [\mathbf{I}_{3 \times 3} \mathbf{0}_{3 \times 3}]$  and  $\mathbf{v}_k$  is the  $3 \times 1$  vector of measurement noises. The observability of the state vector may be analyzed by checking the rank of the observability matrix

$$\mathcal{O} = \begin{bmatrix} \mathbf{H}_{3 \times 6} \\ (\mathbf{H}\mathbf{F})_{3 \times 6} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{I}_{3 \times 3} & \Phi_{3 \times 3} \end{bmatrix}. \quad (4.30)$$

The determinant of this block matrix equals the determinant of the  $\Phi_{3 \times 3}$  matrix, which depends on the values of the  $\phi_{ij} = f(m, g, T, I_{xx}, I_{yy}, I_{zz}, \phi, \theta, \psi)$  terms. By analyzing the expression for the  $\phi_{ij}$  terms in Eq. 4.5, it can be concluded that the determinant of the  $\phi$  matrix will become zero only when the roll and pitch angles of the system are  $0^\circ$ ,  $\pm 90^\circ$  or  $\pm 180^\circ$  for at least two consecutive instants. This observability issue may be avoided by simply guaranteeing that the system stays oscillating around these angles.

Following the format of the Kalman Filter shown in G, the KF is implemented as

1. Time update:

$$[\mathbf{P}_k^-]_{6 \times 6} = [\mathbf{F}_{k-1}]_{6 \times 6} [\mathbf{P}_{k-1}^+]_{6 \times 6} [\mathbf{F}_{k-1}^T]_{6 \times 6} + [\mathbf{Q}_{k-1}]_{6 \times 6} \quad (4.31)$$

$$\hat{\mathbf{x}}_k = \mathbf{F}_{k-1} \hat{\mathbf{x}}_{k-1} \quad (4.32)$$

2. Measurement update:

$$[\mathbf{K}_k]_{6 \times 3} = [\mathbf{P}_k^-]_{6 \times 6} [\mathbf{H}_k^T]_{6 \times 3} ([\mathbf{H}_k]_{3 \times 6} [\mathbf{P}_k^-]_{6 \times 6} [\mathbf{H}_k^T]_{6 \times 3} + [\mathbf{R}_k]_{3 \times 3})^{-1} \quad (4.33)$$

$$[\hat{\mathbf{x}}_k^+]_{6 \times 1} = [\hat{\mathbf{x}}_k^-]_{6 \times 1} + [\mathbf{K}_k]_{6 \times 3} ([\mathbf{y}_k]_{3 \times 1} - [\mathbf{H}_k]_{3 \times 6} [\hat{\mathbf{x}}_k^-]_{6 \times 1}) \quad (4.34)$$

$$[\mathbf{P}_k^+]_{6 \times 6} = ([\mathbf{I}]_{6 \times 6} - [\mathbf{K}_k]_{6 \times 3} [\mathbf{H}_k]_{3 \times 6}) [\mathbf{P}_k^-]_{6 \times 6} \quad (4.35)$$

with  $\mathbf{P}_0$ , the initial state covariance, set as  $\mathbf{P}_0 = \mathbf{0}_{6 \times 6}$  and the process noise and measurement

noise matrices,  $\mathbf{Q}_k$  and  $\mathbf{R}_k$ , respectively, given by

$$\mathbf{Q}_k = E(\mathbf{w}\mathbf{w}^T) = \text{diag} \left( \begin{bmatrix} 5 \cdot 10^{-4} \text{ rad}^2/\text{s}^2 \\ 5 \cdot 10^{-4} \text{ rad}^2/\text{s}^2 \\ 5 \cdot 10^{-4} \text{ rad}^2/\text{s}^2 \\ 1 \cdot 10^{-8} \text{ m}^2 \\ 1 \cdot 10^{-8} \text{ m}^2 \\ 25 \cdot 10^{-8} \text{ m}^2 \end{bmatrix} \right) \text{ and} \quad (4.36)$$

$$\mathbf{R}_k = E(\mathbf{v}\mathbf{v}^T) = \text{diag} \left( \begin{bmatrix} 0.05^2 \text{ rad}^2/\text{s}^2 \\ 0.05^2 \text{ rad}^2/\text{s}^2 \\ 0.05^2 \text{ rad}^2/\text{s}^2 \end{bmatrix} \right), \quad (4.37)$$

in which it is observed that both matrices are diagonal. This fact is coherent with the assumption that all the axes of the gyroscope are uncorrelated (*e.g.* there is no misalignment in the gyroscope axes). Besides considering the hardware setup, the approach adopted in order to select these noise matrices was to analyze the filter consistency. The Normalized Innovation Squared (NIS) test states that, under the hypothesis that a filter is consistent, its NIS will present a chi-square ( $\chi^2$ ) distribution with  $n$  degrees of freedom, in which  $n$  is the number of measurements. Following the definition shown in [Bar-Shalom et al. 2004, p. 236], the *a priori* distribution of the squared innovation terms  $d_k^2$  is given by

$$d_k^2 = (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-), \quad (4.38)$$

which is the called NIS. From the number of degrees of freedom  $n$  and by setting the confidence level - which, in this work, is always set as 95% - the value  $X$  of the cumulative distribution of the  $\chi^2(n)$  may be obtained, *e.g.* from a predetermined table. The test consists in verifying if the value of the  $d_k^2$  terms stay below this  $X$  value, but close to it.

Also, the error signals given by  $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$ , which must rely on the  $\pm 3\sigma_i$  intervals, were analyzed. The  $\sigma_i$ 's are calculated in each time step  $k$  as  $\sigma_i = \sqrt{P_{ii}}$ , in which  $P_{ii}$  are the diagonal terms of the  $\mathbf{P}_k^+$  matrix.

Fig. 4.4 shows the innovation terms obtained. It is known that these innovation terms have a Chi-square distribution. To perform this analysis, the cumulative value of the Chi-square distribution with the 95% confidence interval and 3 degrees of freedom (number of measurements) is determined and used as the limit, obtained as 7.815 in this case. It is expected that, at maximum, only 5% of the samples appear above this limit.

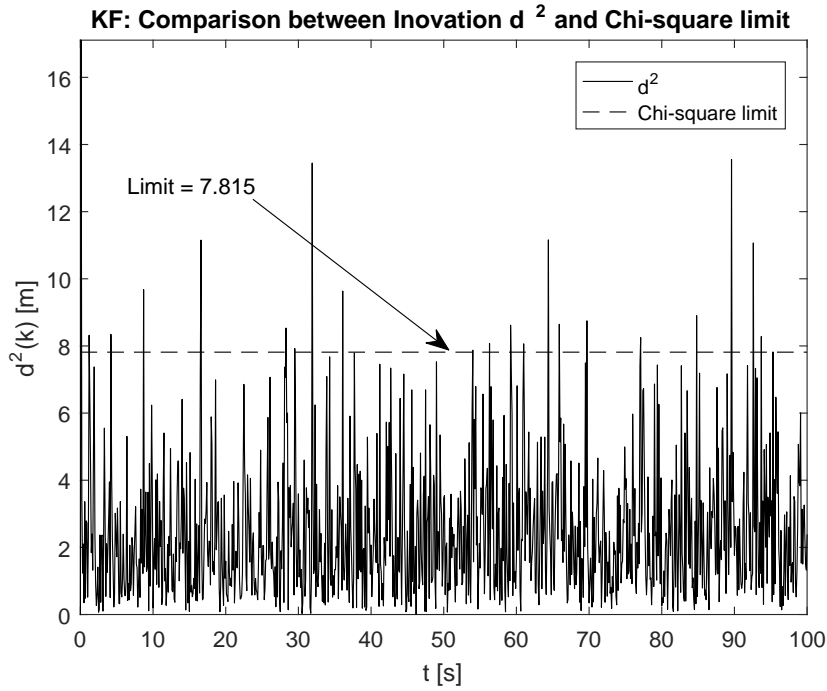


Figure 4.4: Innovation terms  $d^2$  throughout a 100 s simulation of the Kalman Filter.

Simulating the Kalman Filter with the characteristics described in this section resulted that 96.90% of the samples were below this limit. Fig. 4.5 shows that all the state errors remained in its corresponding  $\pm 3\sigma$  intervals, which is the 99.7% confidence interval.

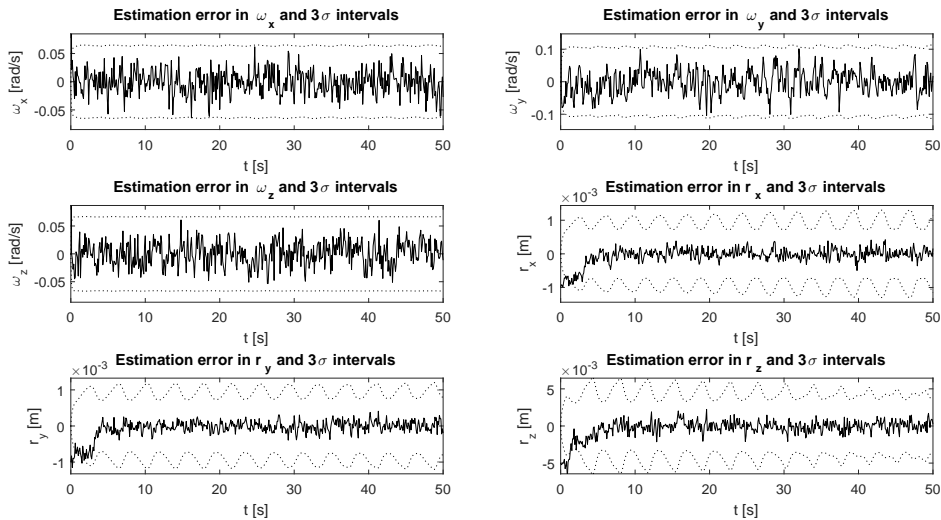


Figure 4.5: State errors and the  $\pm 3\sigma$  intervals.

The estimated unbalance vector components were estimated as  $-1.0349541074 \text{ mm}$ ,  $-1.0008567124 \text{ mm}$  and  $-4.9812694038 \text{ mm}$  when the simulated unbalance vector was  $\mathbf{r} = [-1 \ -1 \ -5]^T \text{ mm}$ . This filter, however, does not work well when the conditions assumed in the model simplification do not hold anymore. Fig. 4.6 shows, as an example, the estimation of the  $r_x$  component of the unbalance vector in three different cases: under normal

conditions; when the  $r$  components are 20 times bigger; and when the angular velocities are much higher than normal operation ( $\omega = [1 \ 1 \ 1] \text{ rad/s}^T$ ).

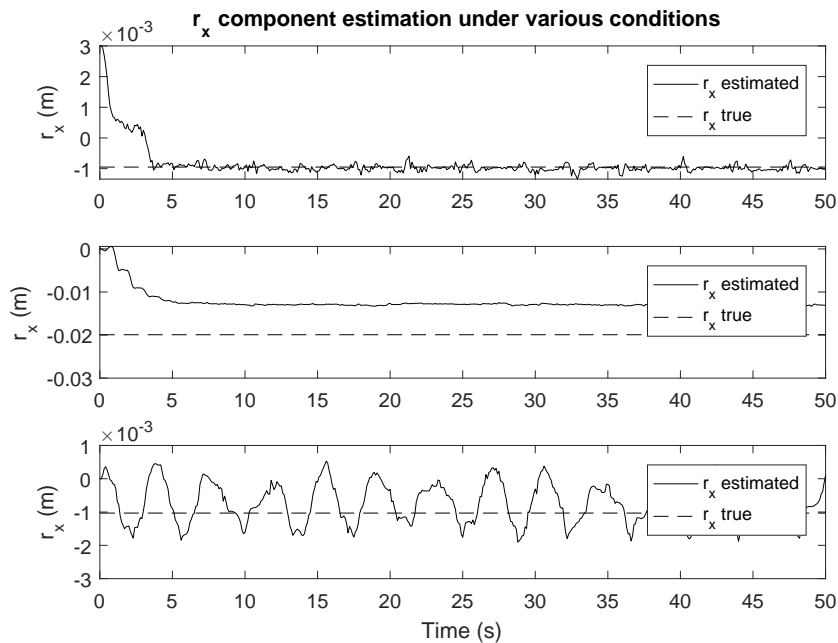


Figure 4.6:  $r_x$  estimation when: (up) normal conditions, (middle)  $r$  is 20 times bigger, (bottom) high angular velocities ( $\omega = [1 \ 1 \ 1] \text{ rad/s}^T$ ).

As can be seen in Fig. 4.6, when the  $r$  components are big enough compared to the other terms in the complete model, Eq. (3.2.3), the filter converges to a value within an offset of the true value. From simulation, it was concluded that this offset grows with the  $r$  components. Also, from this figure it can be seen that, when the angular velocities are high enough, the estimation oscillates around the true value, with amplitude getting as big as bigger become the angular velocities of the system. Additionally, Fig. 4.7 depicts the estimation of the  $r_x$  component when the inertia products have considerable value when compared with the principal moments of inertia, showing that the estimation starts to present an unexpected behaviour, never converging to the true value of  $r_x$ .

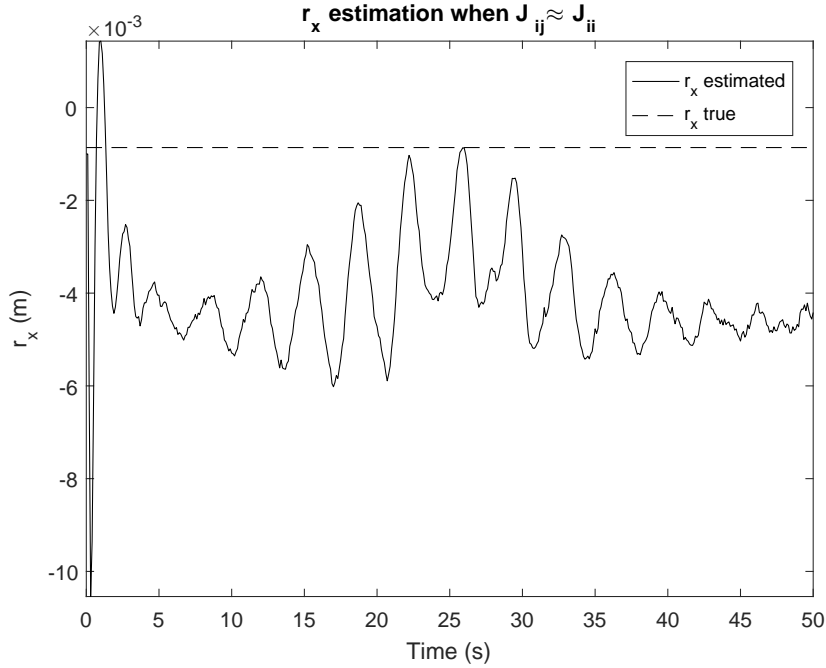


Figure 4.7:  $r_x$  estimation when  $J_{ij}$ ,  $i \neq j$  have comparable magnitude with  $J_{ii}$ .

In all these conditions, the KF converged to erratic values. To improve the filter robustness in these conditions, the complete nonlinear model for the testbed dynamics must be considered, which implies the use of the nonlinear Kalman filters. The following section explains how the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) may be used for this purpose.

### 4.3 Nonlinear filtering applied to the balancing problem

In Sec. 3.2, the testbed was modeled as a rigid body rotating without translation about a center of rotation (CR). As shown in the previous sections, this model may be simplified and the unbalance vector may be estimated using the Least Squares Method or the classical linear Kalman Filter. However, if the experiment does not satisfy one of the listed assumptions, the linear model will not hold anymore and the effect of the nonlinearities cannot be neglected.

Adopting the complete dynamic model and considering the angular velocities as part of the state vector of the filter, it can be seen that the relationship between the state vector and the dynamic model will not be linear. In this section, to overcome the limitations imposed by the usage of the simplified dynamic model, two nonlinear filters will be used: the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF).

First, it will be explained how the EKF may be used to estimate the unbalance vector only. Then, the procedure for augmenting the state vector and estimating the inertia tensor components will be explained. This procedure was presented in the *4th IAA Conference on University Satellite Missions and CubeSat Workshop* [Silva et al. 2017] and is based on

the method described in [Xu et al. 2015]. Finally, the UKF approach will be presented. Observability analyses will be made in the course of the section, as well as simulations to validate the proposed methods.

### 4.3.1 The EKF applied to the balancing problem

The Extended Kalman Filter was developed as an approach for applying the Kalman Filter in nonlinear problems. To accomplish this goal, the main idea of this filter is that a simple Kalman Filter may be used if the model is linearized around the current state. Following the procedure for the discrete EKF shown in [Simon 2006] and summarized in Appendix H, the model must be written in the state-space form as

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}_{k-1}(\boldsymbol{\omega}_{k-1}, \mathbf{r}_{k-1}, \mathbf{w}_{k-1}) \\ \mathbf{y}_k &= \mathbf{h}_k(\boldsymbol{\omega}_k, \mathbf{r}_k, \mathbf{v}_k)\end{aligned}\quad (4.39)$$

in which the state vector is defined as  $\mathbf{x} = [\boldsymbol{\omega} \ \mathbf{r}]^T = [\omega_x \ \omega_y \ \omega_z \ r_x \ r_y \ r_z]^T$  and  $\mathbf{f}_{k-1}$  is obtained by discretizing the Def. 3.2.5 and considering  $\mathbf{r}_k = \mathbf{r}_{k-1}$ . Also, considering that the angular velocities are directly obtained from the sensors, the same output equation used in the Kalman Filter approach may be used (Eq. (4.29)).

The EKF requires the calculation of four jacobians. Considering that the process noise  $\mathbf{w}_k$  and the measurements noise  $\mathbf{v}_k$  are additive gaussian noises, the  $\mathbf{L}_{k-1}$  and  $\mathbf{M}_k$  jacobians are  $3 \times 3$  identity matrices. Also, considering that the output equation is linear, the  $\mathbf{H}_k$  jacobian is given by the  $\mathbf{H}$  matrix itself, which is a  $3 \times 6$  matrix  $\mathbf{H} = [\mathbf{I}_{3 \times 3} \ \mathbf{0}_{3 \times 3}]$ .

The remaining task is to determine the  $\mathbf{F}_{k-1}$  jacobian, which is given by

$$\mathbf{F}_{k-1} = \begin{bmatrix} \frac{\partial f_1^{k-1}}{\partial x_1^{k-1}} & \cdots & \frac{\partial f_1^{k-1}}{\partial x_6^{k-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6^{k-1}}{\partial x_1^{k-1}} & \cdots & \frac{\partial f_6^{k-1}}{\partial x_6^{k-1}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{f}_{k-1}^{1:3}}{\partial \mathbf{x}_{k-1}} \\ \frac{\partial \mathbf{f}_{k-1}^{4:6}}{\partial \mathbf{x}_{k-1}} \end{bmatrix}, \quad (4.40)$$

implying that the  $f_i, i \in \{1, \dots, 6\}$  functions must be determined. Dividing  $\dot{\boldsymbol{\omega}}$  given by Def. 3.2.5 in two different portions, it follows that

$$\dot{\boldsymbol{\omega}} = \underbrace{\mathbf{J}^{-1}[\mathbf{J}\boldsymbol{\omega} \times] \boldsymbol{\omega}}_{\text{Term 1}} + \underbrace{\mathbf{J}^{-1}[-m\mathbf{g}_b \times] \mathbf{r}}_{\text{Term 2}}, \quad (4.41)$$

in which the cross products are taken in matrix form. Assuming that the inertia products are negligible when comparing with the main inertia terms - which is a reasonable consideration, given the real inertia tensor of the testbed -, the  $\mathbf{J}$  is diagonal, which simplifies the development of Eq. (4.41). The term 1 in Eq. (4.41), aided by the developments in [Shen et al. 2004],

is given by

$$\mathbf{J}^{-1}[\mathbf{J}\boldsymbol{\omega} \times] \boldsymbol{\omega} = \begin{bmatrix} J_x^{-1} & 0 & 0 \\ 0 & J_y^{-1} & 0 \\ 0 & 0 & J_z^{-1} \end{bmatrix} \begin{bmatrix} 0 & -J_z\omega_z & J_y\omega_y \\ J_z\omega_z & 0 & -J_x\omega_x \\ -J_y\omega_y & J_x\omega_x & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (4.42)$$

$$= \begin{bmatrix} \omega_z\omega_y \left( \frac{J_y - J_z}{J_x} \right) \\ \omega_x\omega_z \left( \frac{J_z - J_x}{J_y} \right) \\ \omega_x\omega_y \left( \frac{J_x - J_y}{J_z} \right) \end{bmatrix}, \quad (4.43)$$

whereas the term 2 is given by

$$\mathbf{J}^{-1}[-m\mathbf{g}_b \times] \mathbf{r} = \begin{bmatrix} J_x^{-1} & 0 & 0 \\ 0 & J_y^{-1} & 0 \\ 0 & 0 & J_z^{-1} \end{bmatrix} \begin{bmatrix} 0 & mg_{bz} & -mg_{by} \\ -mg_{bz} & 0 & mg_{bx} \\ mg_{by} & -mg_{bx} & 0 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \quad (4.44)$$

$$= \begin{bmatrix} \frac{1}{J_x} (mg_{bz}r_y - mg_{by}r_z) \\ \frac{1}{J_y} (-mg_{bz}r_x + mg_{bx}r_z) \\ \frac{1}{J_z} (mg_{by}r_x - mg_{bx}r_y) \end{bmatrix}, \quad (4.45)$$

following that the  $f_i$  terms are given by

$$\begin{cases} f_1^{k-1} = \omega_{x,k} = \omega_x + \dot{\omega}_x T = \omega_x + \left[ \omega_z\omega_y \left( \frac{J_y - J_z}{J_x} \right) + \frac{1}{J_x} (mg_{bz}r_y - mg_{by}r_z) \right] T \\ f_2^{k-1} = \omega_{y,k} = \omega_y + \dot{\omega}_y T = \omega_y + \left[ \omega_x\omega_z \left( \frac{J_z - J_x}{J_y} \right) + \frac{1}{J_y} (-mg_{bz}r_x + mg_{bx}r_z) \right] T \\ f_3^{k-1} = \omega_{z,k} = \omega_z + \dot{\omega}_z T = \omega_z + \left[ \omega_x\omega_y \left( \frac{J_x - J_y}{J_z} \right) + \frac{1}{J_z} (mg_{by}r_x - mg_{bx}r_y) \right] T \end{cases}, \quad (4.46)$$

in which the  $\boldsymbol{\omega}$ ,  $\mathbf{g}_b$  and  $\mathbf{r}$  components at the right-hand side are taken at the instant  $k - 1$ .

Then, the partial derivatives in  $\frac{\partial \mathbf{f}_{k-1}^{1:3}}{\partial \mathbf{x}_{k-1}}$  are obtained as

$\frac{\partial f_1}{\partial \omega_{x,k-1}} = 1$	$\frac{\partial f_2}{\partial \omega_{x,k-1}} = \omega_{z,k-1} \left( \frac{J_z - J_x}{J_y} \right) T$	$\frac{\partial f_3}{\partial \omega_{x,k-1}} = \omega_{y,k-1} \left( \frac{J_x - J_y}{J_z} \right) T$
$\frac{\partial f_1}{\partial \omega_{y,k-1}} = \omega_{z,k-1} \left( \frac{J_y - J_z}{J_x} \right) T$	$\frac{\partial f_2}{\partial \omega_{y,k-1}} = 1$	$\frac{\partial f_3}{\partial \omega_{y,k-1}} = \omega_{x,k-1} \left( \frac{J_x - J_y}{J_z} \right) T$
$\frac{\partial f_1}{\partial \omega_{z,k-1}} = \omega_{y,k-1} \left( \frac{J_y - J_z}{J_x} \right) T$	$\frac{\partial f_2}{\partial \omega_{z,k-1}} = \omega_{x,k-1} \left( \frac{J_z - J_x}{J_y} \right) T$	$\frac{\partial f_3}{\partial \omega_{z,k-1}} = 1$
$\frac{\partial f_1}{\partial r_{x,k-1}} = 0$	$\frac{\partial f_2}{\partial r_{x,k-1}} = -\frac{1}{J_y} mg_{bz} T$	$\frac{\partial f_3}{\partial r_{x,k-1}} = \frac{1}{J_z} mg_{by} T$
$\frac{\partial f_1}{\partial r_{y,k-1}} = \frac{1}{J_x} mg_{bz} T$	$\frac{\partial f_2}{\partial r_{y,k-1}} = 0$	$\frac{\partial f_3}{\partial r_{y,k-1}} = -\frac{1}{J_z} mg_{bx} T$
$\frac{\partial f_1}{\partial r_{z,k-1}} = -\frac{1}{J_x} mg_{by} T$	$\frac{\partial f_2}{\partial r_{z,k-1}} = \frac{1}{J_y} mg_{bx} T$	$\frac{\partial f_3}{\partial r_{z,k-1}} = 0$

(4.47)

For the partial derivatives in  $\frac{\partial \mathbf{f}_{k-1}^{4:6}}{\partial \mathbf{x}_{k-1}}$ , one must first notice that the  $f_{k-1}^{4:6}$  functions are given



by

$$f_4^{k-1} = r_x^k = r_x^{k-1} \quad (4.48)$$

$$f_5^{k-1} = r_y^k = r_y^{k-1} \quad (4.49)$$

$$f_6^{k-1} = r_z^k = r_z^{k-1}, \quad (4.50)$$

which implies that  $\frac{\partial \mathbf{f}_{k-1}^{4:6}}{\partial \boldsymbol{\omega}_{k-1}} = \mathbf{0}$  and  $\frac{\partial \mathbf{f}_{k-1}^{4:6}}{\partial \mathbf{r}_{k-1}} = \mathbf{I}_{3 \times 3}$ . In other words, the  $\mathbf{F}_{k-1}$  jacobian is given by

$$\mathbf{F}_{k-1} = \begin{bmatrix} \left[ \begin{array}{c} \frac{\partial f_{1:3}^{k-1}}{\partial \mathbf{x}_{k-1}} \end{array} \right]_{3 \times 6} \\ \mathbf{0}_{3 \times 3} \quad \mathbf{I}_{3 \times 3} \end{bmatrix}. \quad (4.51)$$

Having described the filter design, the EKF equations may be used as follows, in which the matrix dimensions are made explicit to verify compatibility,

### 1. Time update:

$$[\mathbf{P}_k^-]_{6 \times 6} = [\mathbf{F}_{k-1}]_{6 \times 6} [\mathbf{P}_{k-1}^+]_{6 \times 6} [\mathbf{F}_{k-1}^T]_{6 \times 6} + [\mathbf{L}_{k-1}]_{6 \times 6} [\mathbf{Q}_{k-1}]_{6 \times 6} [\mathbf{L}_{k-1}^T]_{6 \times 6} \quad (4.52)$$

$$= [\mathbf{F}_{k-1}]_{6 \times 6} [\mathbf{P}_{k-1}^+]_{6 \times 6} [\mathbf{F}_{k-1}^T]_{6 \times 6} + [\mathbf{Q}_{k-1}]_{6 \times 6} \quad (4.53)$$

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\omega}_x^k \\ \hat{\omega}_y^k \\ \hat{\omega}_z^k \\ \hat{r}_x^k \\ \hat{r}_y^k \\ \hat{r}_z^k \end{bmatrix} = \begin{bmatrix} \hat{\omega}_x^{k-1} + \left[ \hat{\omega}_z^{k-1} \hat{\omega}_y^{k-1} \left( \frac{J_y - J_z}{J_x} \right) + \frac{1}{J_x} (mg_{bz} \hat{r}_y^{k-1} - mg_{by} \hat{r}_z^{k-1}) \right] T \\ \hat{\omega}_y^{k-1} + \left[ \hat{\omega}_x^{k-1} \hat{\omega}_z^{k-1} \left( \frac{J_z - J_x}{J_y} \right) + \frac{1}{J_y} (mg_{bz} \hat{r}_x^{k-1} - mg_{bx} \hat{r}_z^{k-1}) \right] T \\ \hat{\omega}_z^{k-1} + \left[ \hat{\omega}_x^{k-1} \hat{\omega}_y^{k-1} \left( \frac{J_x - J_y}{J_z} \right) + \frac{1}{J_z} (mg_{by} \hat{r}_x^{k-1} - mg_{bx} \hat{r}_y^{k-1}) \right] T \\ \hat{r}_x^{k-1} \\ \hat{r}_y^{k-1} \\ \hat{r}_z^{k-1} \end{bmatrix} \quad (4.54)$$

### 2. Measurement update:

$$[\mathbf{K}_k]_{6 \times 3} = [\mathbf{P}_k^-]_{6 \times 6} [\mathbf{H}_k^T]_{6 \times 3} ([\mathbf{H}_k]_{3 \times 6} [\mathbf{P}_k]_{6 \times 6} [\mathbf{H}_k^T]_{6 \times 3} + \underbrace{[\mathbf{M}_k]_{3 \times 3} [\mathbf{R}_k]_{3 \times 3} [\mathbf{M}_k^T]_{3 \times 3}}_{=\mathbf{I}_{3 \times 3}})^{-1} \quad (4.55)$$

$$= [\mathbf{P}_k^-]_{6 \times 6} [\mathbf{H}_k^T]_{6 \times 3} ([\mathbf{H}_k]_{3 \times 6} [\mathbf{P}_k]_{6 \times 6} [\mathbf{H}_k^T]_{6 \times 3} + [\mathbf{R}_k]_{3 \times 3})^{-1} \quad (4.56)$$

$$[\hat{\mathbf{x}}_k^+]_{6 \times 1} = [\hat{\mathbf{x}}_k^-]_{6 \times 1} + [\mathbf{K}_k]_{6 \times 3} ([\mathbf{y}_k]_{3 \times 1} - [\mathbf{H}_k]_{3 \times 6} [\hat{\mathbf{x}}_k^-]_{6 \times 1}) \quad (4.57)$$

$$[\mathbf{P}_k^+]_{6 \times 6} = ([\mathbf{I}]_{6 \times 6} - [\mathbf{K}_k]_{6 \times 3} [\mathbf{H}_k]_{3 \times 6}) [\mathbf{P}_k^-]_{6 \times 6} \quad (4.58)$$

A 50 s simulation was performed to test the proposed filter. Figs. 4.8 and 4.9 shows the Chi-squared test and the  $3\sigma$  intervals, whereas Fig. 4.10 shows the estimation of the  $\mathbf{r}$  components throughout the simulation (the simulated unbalance vector is  $\mathbf{r} = [-1 \ -1 \ -5]^T 10^{-3} \text{ m}$ ). Fig. 4.11 shows the estimation when the angular velocities magnitude is  $\boldsymbol{\omega} = [10 \ 10 \ 10]^T \text{ rad/s}$ .

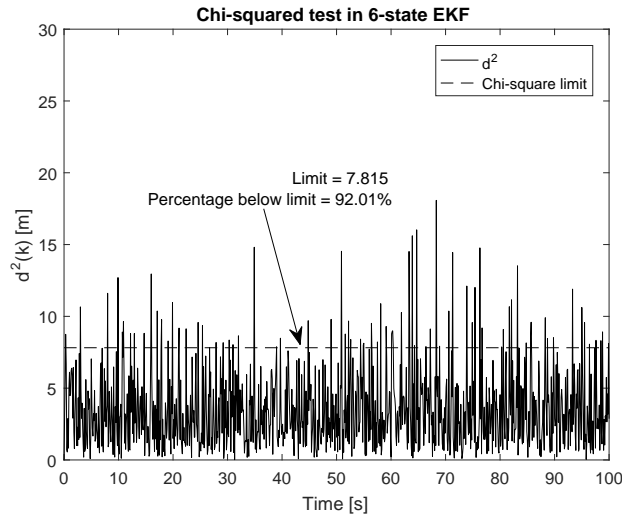


Figure 4.8: Innovation terms  $d^2$  throughout a 100 s simulation of the Extended Kalman Filter.

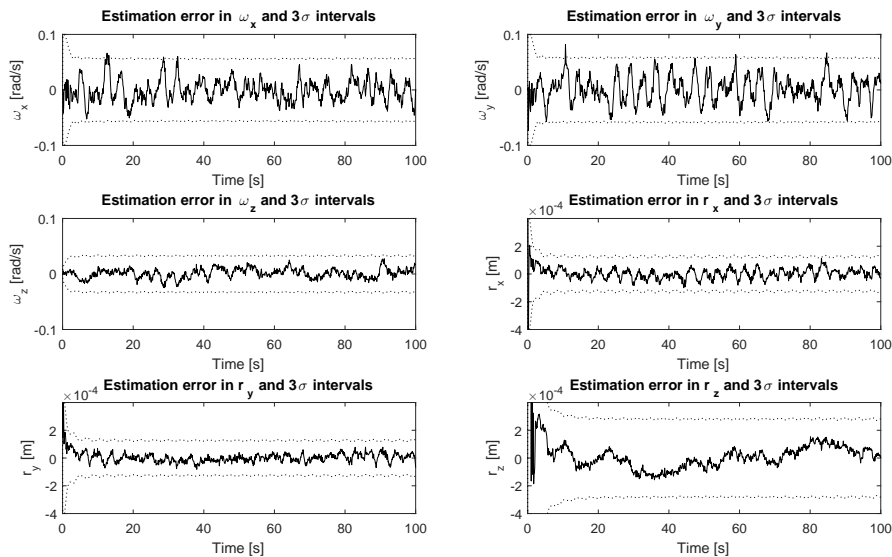


Figure 4.9: State errors and the  $\pm 3\sigma$  intervals.

As can be seen in Fig. 4.11, the Extended Kalman Filter does not perform well when high angular velocities are considered. It is a consequence of unmodelled dynamics, since the simulated model considers the complete inertia tensor matrix, augmented with the terms of the parallel axis theorem, whereas the filter considers only the diagonal terms. To enhance the filter performance for this situation, one must consider the entire inertia tensor matrix as will be explained in the next section.

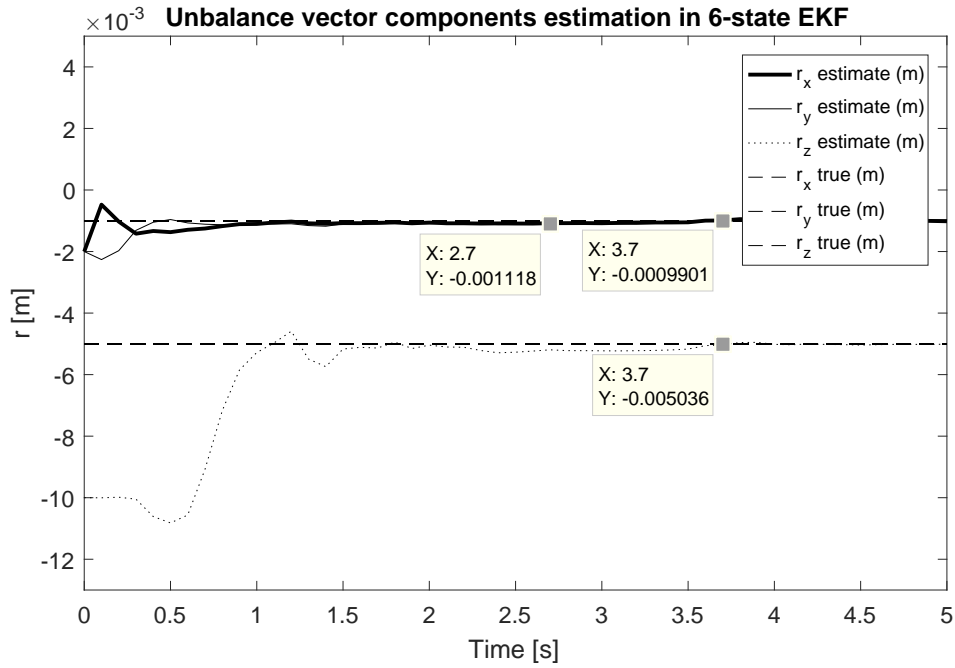


Figure 4.10: Estimation of  $r$  components in the 6-state EKF.

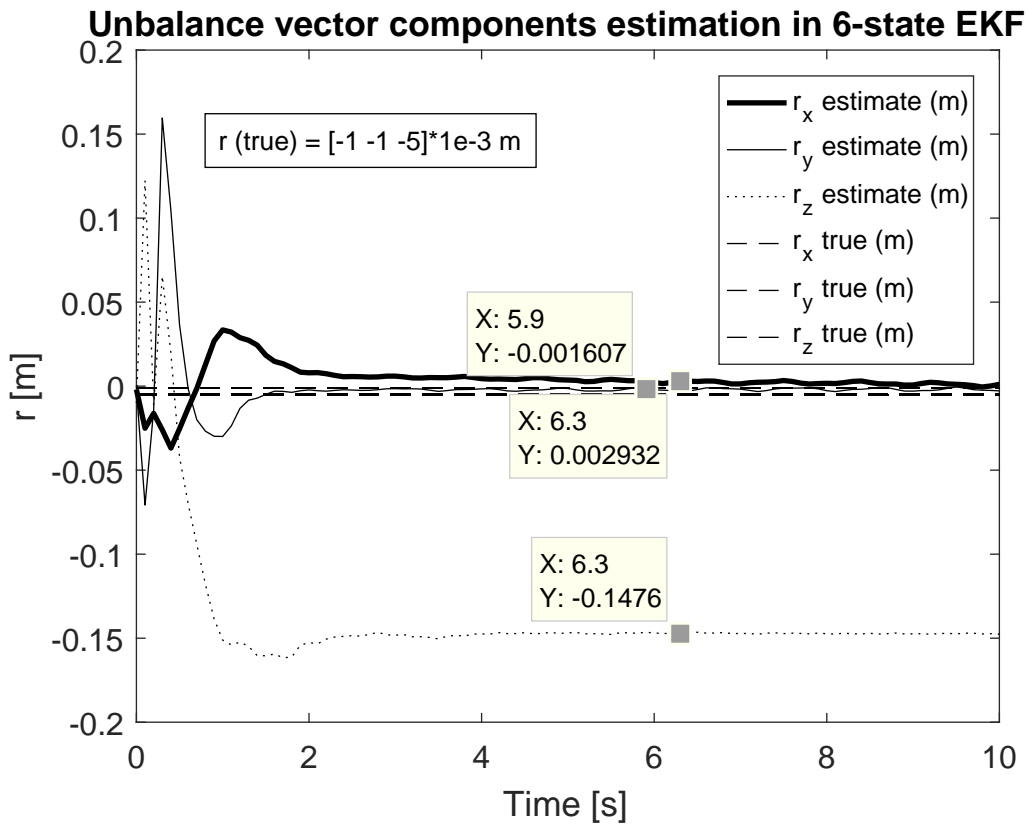


Figure 4.11: Estimation of  $r$  components in the 6-state EKF when angular velocities are high.

### 4.3.2 Augmenting the EKF

One of the possible ways of controlling the platform or estimating its parameters in order to compensate the unbalance vector is by using momentum exchange devices, such as reaction wheels, to excitate the platform along with the movable masses. This is done, for example, in the works developed by [Xu et al. 2015], which uses reaction wheels, and [Kim and Agrawal 2009], which uses Control Moment Gyros (CMGs). In this section the method developed by [Xu et al. 2015] is focused.

The adaptation required in this case is simply adding an angular momentum term  $\mathbf{h}$  to the total angular momentum of the platform  $\mathbf{H}_o$ . It must be noticed that the additional term  $\mathbf{h}$  is not added as external torque, which remains the same. From the Euler equation of motion defined in Def. 3.2.5,

$$\begin{aligned} \frac{d}{dt} \mathbf{h} \Big|_i &= \dot{\mathbf{h}} \Big|_b + \boldsymbol{\omega} \times \mathbf{h} \\ \frac{d\mathbf{H}_o}{dt} &= \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \frac{d}{dt} \mathbf{h} \Big|_i = \\ &= \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + (\dot{\mathbf{h}} + \boldsymbol{\omega} \times \mathbf{h}) = \boldsymbol{\rho}_{CM} \times m\mathbf{g}_b + \mathbf{T}, \end{aligned} \quad (4.59)$$

where  $\mathbf{h}$  is the added momentum generated by the reaction wheels,  $\mathbf{T}$  is the total external torque disturbances and the BKE is applied to the reaction wheels angular momentum, since its components are measured in the body frame.

Similarly to the first EKF approach, the angular velocities of the platform composes the state vector  $\mathbf{x}_1 = [\omega_x \ \omega_y \ \omega_z]^T$ . This state vector may be augmented with the physical parameters of the system, based on the same procedure described by [Simon 2006]. In this case, the vector with the augmented parameters comprises the components of the inertia tensor and the unbalance vector components, as shown in Eq. 4.60

$$\mathbf{x}_2 = \begin{bmatrix} J_x & J_y & J_z & J_{xy} & J_{xz} & J_{yz} & m\mathbf{r}_x & m\mathbf{r}_y & m\mathbf{r}_z \end{bmatrix}_{9 \times 1}^T, \quad (4.60)$$

in which the mass  $m$  of the system is grouped with the unbalance vector and estimated together. The system dynamics in state-space form can then be represented by Eq.4.61, in which the complete state vector is represented by  $\mathbf{x} = [\mathbf{x}_1 \ \mathbf{x}_2]^T$ ,

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = F(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}) = \begin{bmatrix} f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}) \\ 0 \end{bmatrix}, \quad (4.61)$$

where  $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}) = \dot{\boldsymbol{\omega}} = J^{-1}(J\boldsymbol{\omega} \times \boldsymbol{\omega} + \mathbf{r} \times m\mathbf{g}_b - (\dot{\mathbf{h}} + \boldsymbol{\omega} \times \mathbf{h}))$  and, in discretized form,

as

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}_{k-1}(\boldsymbol{\omega}_{k-1}, \mathbf{r}_{k-1}, \mathbf{w}_{k-1}) \\ \mathbf{y}_k &= \mathbf{h}_k(\boldsymbol{\omega}_k, \mathbf{r}_k, \mathbf{v}_k) .\end{aligned}\quad (4.62)$$

As in the previous cases, the output equation is linear, since the measurements are taken directly from the sensor. Besides it, the state vector contains 12 elements and only the first three are measured, which are the components of the angular velocity vector  $\boldsymbol{\omega}$ . Having it said, the output equation is given by

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k = \mathbf{H} \mathbf{x}_k + \mathbf{v}_k , \quad (4.63)$$

in which the  $\mathbf{H}$  matrix is constant and given by  $\mathbf{H} = [\mathbf{I}_{3 \times 3} \ \mathbf{0}_{3 \times 9}]$ . Also, the same considerations of the last section are used with relation to the process and measurement noises, *i.e.* both are additive noises, which have, as consequence, that the  $\mathbf{L}_{k-1}$  and  $\mathbf{M}_k$  matrices are given by

$$\mathbf{L}_{k-1} = \mathbf{I}_{12 \times 12} \quad (4.64)$$

$$\mathbf{M}_k = \mathbf{I}_{3 \times 3} , \quad (4.65)$$

in which attention must be given to the adapted size of the matrices, that are compatible with the new state vector size. Again, the main challenge concerning the usage of this algorithm is determining the  $\mathbf{F}_{k-1}$  jacobian. By discretizing Eq. 4.61, the dynamic discrete equations are obtained as

$$\omega_x^k = f_1^{k-1}(\mathbf{x}) = \omega_x^{k-1} + \dot{\omega}_x^{k-1} T + w_1^{k-1} \quad (4.66)$$

$$\omega_y^k = f_2^{k-1}(\mathbf{x}) = \omega_y^{k-1} + \dot{\omega}_y^{k-1} T + w_2^{k-1} \quad (4.67)$$

$$\omega_z^k = f_3^{k-1}(\mathbf{x}) = \omega_z^{k-1} + \dot{\omega}_z^{k-1} T + w_3^{k-1} \quad (4.68)$$

$$J_x^k = f_4^{k-1}(\mathbf{x}) = J_x^{k-1} + w_4^{k-1} \quad (4.69)$$

$$J_y^k = f_5^{k-1}(\mathbf{x}) = J_y^{k-1} + w_5^{k-1} \quad (4.70)$$

$$J_z^k = f_6^{k-1}(\mathbf{x}) = J_z^{k-1} + w_6^{k-1} \quad (4.71)$$

$$J_{xy}^k = f_7^{k-1}(\mathbf{x}) = J_{xy}^{k-1} + w_7^{k-1} \quad (4.72)$$

$$J_{xz}^k = f_8^{k-1}(\mathbf{x}) = J_{xz}^{k-1} + w_8^{k-1} \quad (4.73)$$

$$J_{yz}^k = f_9^{k-1}(\mathbf{x}) = J_{yz}^{k-1} + w_9^{k-1} \quad (4.74)$$

$$(mr_x)^k = f_{10}^{k-1}(\mathbf{x}) = (mr_x)^{k-1} + w_{10}^{k-1} \quad (4.75)$$

$$(mr_y)^k = f_{11}^{k-1}(\mathbf{x}) = (mr_y)^{k-1} + w_{11}^{k-1} \quad (4.76)$$

$$(mr_z)^k = f_{12}^{k-1}(\mathbf{x}) = (mr_z)^{k-1} + w_{12}^{k-1} \quad (4.77)$$

The set of functions  $f_{4:12}^{k-1}$  depend only on one of the 12 states, which means that, con-

sidering the noises  $w_{4:12}^{k-1}$  are independent with relation to the state vector, the  $\mathbf{F}_{k-1}$  jacobian may be simplified as

$$\mathbf{F}_{k-1} = \begin{bmatrix} \frac{\partial f_1^{k-1}(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1^{k-1}(\mathbf{x})}{\partial x_3} & \frac{\partial f_1^{k-1}(\mathbf{x})}{\partial x_4} & \dots & \frac{\partial f_1^{k-1}(\mathbf{x})}{\partial x_9} & \frac{\partial f_1^{k-1}(\mathbf{x})}{\partial x_{10}} & \dots & \frac{\partial f_1^{k-1}(\mathbf{x})}{\partial x_{12}} \\ \frac{\partial f_2^{k-1}(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_2^{k-1}(\mathbf{x})}{\partial x_3} & \frac{\partial f_2^{k-1}(\mathbf{x})}{\partial x_4} & \dots & \frac{\partial f_2^{k-1}(\mathbf{x})}{\partial x_9} & \frac{\partial f_2^{k-1}(\mathbf{x})}{\partial x_{10}} & \dots & \frac{\partial f_2^{k-1}(\mathbf{x})}{\partial x_{12}} \\ \frac{\partial f_3^{k-1}(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_3^{k-1}(\mathbf{x})}{\partial x_3} & \frac{\partial f_3^{k-1}(\mathbf{x})}{\partial x_4} & \dots & \frac{\partial f_3^{k-1}(\mathbf{x})}{\partial x_9} & \frac{\partial f_3^{k-1}(\mathbf{x})}{\partial x_{10}} & \dots & \frac{\partial f_3^{k-1}(\mathbf{x})}{\partial x_{12}} \\ \hline & & \mathbf{0}_{9 \times 3} & & & \mathbf{I}_{9 \times 9} & & & \end{bmatrix} \quad (4.78)$$

$$= \begin{bmatrix} \left[ \frac{\partial f_{1:3}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}} \right]_{3 \times 3} & \left[ \frac{\partial f_{1:3}^{k-1}}{\partial \mathbf{J}_{k-1}} \right]_{3 \times 6} & \left[ \frac{\partial f_{1:3}^{k-1}}{\partial m\mathbf{x}_{k-1}} \right]_{3 \times 6} \\ \hline \mathbf{0}_{9 \times 3} & & \mathbf{I}_{9 \times 9} \end{bmatrix} \quad (4.79)$$

As the set of functions in  $f_{1:3}^{k-1}$  are nonlinear with relation to the state vector, determining the  $\left[ \frac{\partial f_{1:3}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}} \right]_{3 \times 3}$ ,  $\left[ \frac{\partial f_{1:3}^{k-1}}{\partial \mathbf{J}_{k-1}} \right]_{3 \times 6}$  and  $\left[ \frac{\partial f_{1:3}^{k-1}}{\partial m\mathbf{x}_{k-1}} \right]_{3 \times 6}$  jacobians turn out to be a complex task, since the analytic solution may be cumbersome to determine or result in enormous equations. In this work, to avoid this obstacle, these jacobians were determined numerically using the Complex Step Differentiation (CSD) method. The CSD implementation by Yi Cao was used and is available in [Cao 2018]. The basic idea of this differentiation method is given in Appendix I, whereas the reader may refer to [Al-Mohy and Higham 2010, Martins et al. 2001, Martins et al. 2003] for further details. To exemplify how it is done, the  $\left[ \frac{\partial f_{1:3}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}} \right]_{3 \times 3}$  jacobian is calculated by first determining the  $\frac{\partial \dot{\boldsymbol{\omega}}^{k-1 T}}{\partial \boldsymbol{\omega}_{k-1}} = T \frac{\partial \dot{\boldsymbol{\omega}}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}}$  term through CSD and adding the identity matrix, resulting

$$\left[ \frac{\partial f_{1:3}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}} \right]_{3 \times 3} = \mathbf{I}_{3 \times 3} + T \left[ \frac{\partial \dot{\boldsymbol{\omega}}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}} \right]_{CSD} \quad (4.80)$$

The identity matrix is added as a result of the derivative of the  $\omega_{x,y,z}^{k-1}$  terms. To calculate the  $\left[ \frac{\partial \dot{\boldsymbol{\omega}}^{k-1}}{\partial \boldsymbol{\omega}_{k-1}} \right]_{CSD}$  term through CSD each column is determined with the following steps

1. First, the point  $\mathbf{x}_{k-1}^-$  is taken as the reference point.
2. The point is incremented in its  $k^{th}$  independent variable by a complex infinitesimal value  $h \cdot i$ , in which  $i$  is the complex variable. The  $h$  value is chosen near to the minimum scalar that can be represented in the digital system. For example, the minimum value in the double floating point arithmetic used in MATLAB is  $2.2204 \cdot 10^{-16}$ .
3. The function is evaluated at this point, the imaginary part of the result is taken and divided by  $h$ . The result is the  $k^{th}$  column of the CSD jacobian. The process is repeated

for all the variables in  $\mathbf{x}_{k-1}^-$  and is given, in the prescribed example, by

$$\left[ \frac{\partial \dot{\boldsymbol{\omega}}^{k-1}}{\partial \boldsymbol{\omega}^{k-1}} \right]_{CSD} = \left[ \begin{array}{c} \text{Im} \left[ \dot{\boldsymbol{\omega}}^{k-1} \begin{pmatrix} \omega_x^{k-1} + h \cdot i \\ \omega_y^{k-1} \\ \omega_z^{k-1} \end{pmatrix} \right] \\ \hline h \quad \dots \end{array} \right] \quad (4.81)$$

Repeating the methodology used in the 6-state EKF shown in the last section, the time update and measurement update phases are shown to facilitate the verification of matrix compatibility and to summarize the algorithm,

### 1. Time update:

$$\begin{aligned} [\mathbf{P}_k^-]_{12 \times 12} &= [\mathbf{F}_{k-1}]_{12 \times 12} [\mathbf{P}_{k-1}^+]_{12 \times 12} [\mathbf{F}_{k-1}^T]_{12 \times 12} + [\mathbf{L}_{k-1}]_{12 \times 12} [\mathbf{Q}_{k-1}]_{12 \times 12} [\mathbf{L}_{k-1}^T]_{12 \times 12} \\ &= [\mathbf{F}_{k-1}]_{12 \times 12} [\mathbf{P}_{k-1}^+]_{12 \times 12} [\mathbf{F}_{k-1}^T]_{12 \times 12} + [\mathbf{Q}_{k-1}]_{12 \times 12} \\ \hat{\mathbf{x}}_k &= \begin{bmatrix} \hat{\omega}_x^k \\ \hat{\omega}_y^k \\ \hat{\omega}_z^k \\ \hat{r}_x^k \\ \hat{r}_y^k \\ \hat{r}_z^k \end{bmatrix} = \begin{bmatrix} \hat{\omega}_x^{k-1} + \hat{\omega}_x^{k-1} T \\ \hat{\omega}_y^{k-1} + \hat{\omega}_y^{k-1} T \\ \hat{\omega}_z^{k-1} + \hat{\omega}_z^{k-1} T \\ j_x^{k-1} \\ j_y^{k-1} \\ j_z^{k-1} \\ j_{xy}^{k-1} \\ j_{xz}^{k-1} \\ j_{yz}^{k-1} \\ \hat{m}r_x^{k-1} \\ \hat{m}r_y^{k-1} \\ \hat{m}r_z^{k-1} \end{bmatrix} \end{aligned}$$

### 2. Measurement update:

$$\begin{aligned} [\mathbf{K}_k]_{12 \times 3} &= [\mathbf{P}_k^-]_{12 \times 12} [\mathbf{H}_k^T]_{12 \times 3} \left( [\mathbf{H}_k]_{3 \times 12} [\mathbf{P}_k]_{12 \times 12} [\mathbf{H}_k^T]_{12 \times 3} + \underbrace{[\mathbf{M}_k]_{3 \times 3} [\mathbf{R}_k]_{3 \times 3} [\mathbf{M}_k^T]_{3 \times 3}^{-1}}_{=\mathbf{I}_{3 \times 3}} \right)^{-1} \\ &= [\mathbf{P}_k^-]_{12 \times 12} [\mathbf{H}_k^T]_{12 \times 3} \left( [\mathbf{H}_k]_{3 \times 12} [\mathbf{P}_k]_{12 \times 12} [\mathbf{H}_k^T]_{12 \times 3} + [\mathbf{R}_k]_{3 \times 3} \right)^{-1} \\ [\hat{\mathbf{x}}_k^+]_{12 \times 1} &= [\hat{\mathbf{x}}_k^-]_{12 \times 1} + [\mathbf{K}_k]_{12 \times 3} \left( [\mathbf{y}_k]_{3 \times 1} - [\mathbf{H}_k]_{3 \times 12} [\hat{\mathbf{x}}_k^+]_{12 \times 1} \right) \\ [\mathbf{P}_k^+]_{12 \times 12} &= ([\mathbf{I}]_{12 \times 12} - [\mathbf{K}_k]_{12 \times 3} [\mathbf{H}_k]_{3 \times 12}) [\mathbf{P}_k^-]_{12 \times 12} \end{aligned} \quad (4.82)$$

A 100 s experiment was run with the following initial conditions and matrices setup:

### 1. Process noise:

$$\mathbf{Q} = \text{diag} \left( \left[ (5 \cdot 10^{-6})_{1 \times 3} (10^{-6})_{1 \times 3} (10^{-7})_{1 \times 3} (2 \cdot 10^{-8})_{1 \times 3} \right]^T \right) \quad (4.83)$$

$$\left[ (\text{rad}^2/\text{s}^2)_{1 \times 3} (\text{kg}^2\text{m}^4)_{1 \times 6} (\text{kg}^2\text{m}^2)_{1 \times 3} \right]$$

### 2. Measurement noise:

$$\mathbf{R} = \text{diag} \left( \left[ 0.005^2 \ 0.005^2 \ 0.005^2 \right]^T \text{rad}^2/\text{s}^2 \right) \quad (4.84)$$

The  $\mathbf{R}$  matrix is coherent with the noise level in the IMU embedded in the LAICA testbed.

### 3. Initial conditions:

$$\mathbf{P}_0 = \text{diag}(\mathbf{0}_{3 \times 1}[\text{rad}^2/\text{s}^2] \quad \mathbf{0}_{3 \times 1}[\text{m}^2]) \quad (4.85)$$

$$\boldsymbol{\omega} = [0 \ 0 \ 0]^T \text{ rad/s} \quad (4.86)$$

$$\phi = 0 \text{ rad (roll angle)} \quad (4.87)$$

$$\theta = 0 \text{ rad (pitch angle)} \quad (4.88)$$

$$\psi = 0 \text{ rad (yaw angle)} \quad (4.89)$$

$$\dot{\mathbf{h}} = [0.05 \ 0.05 \ 0.05]^T \text{ N} \cdot \text{m (RW torques)} \quad (4.90)$$

### 4. Testbed and general characteristics:

$$\mathbf{J} = \begin{bmatrix} 0.265 & -0.014 & -0.035 \\ -0.014 & 0.246 & -0.018 \\ -0.035 & -0.018 & 0.427 \end{bmatrix} \text{ kgm}^2 \quad (4.91)$$

$$\mathbf{r} = [-1 \ -1 \ -5]^T \cdot 10^{-3} \text{ m} \quad (4.92)$$

$$m = 14.307 \text{ kg} \quad (4.93)$$

$$g = 9.78 \text{ m/s}^2 \text{ (local gravity acceleration)} \quad (4.94)$$

The  $\mathbf{R}$  matrix is set up considering the characteristics of the testbed IMU, whereas the  $\mathbf{Q}$  is adjusted in accordance with the  $3\sigma$ -interval graphs and the chi-squared test, which are shown in Figs. 4.13 and 4.12. The chi-squared test resulted in 82.44 % samples inside the 95 % confidence interval and it is possible to notice that the majority of the outgoing samples are concentrated in the first 50 s of the simulation. Simulating this experiment with longer periods, e.g 500 s, result in even higher percentages of innovation samples inside the proper confidence level. In Fig. 4.13, in the  $3\sigma$  intervals graph, it may be noticed that each parameter may delay until approximately 50 s of the simulation to converge to the confidence interval. However, after this period, all parameters stay in its proper interval, as could be seen in longer simulations.

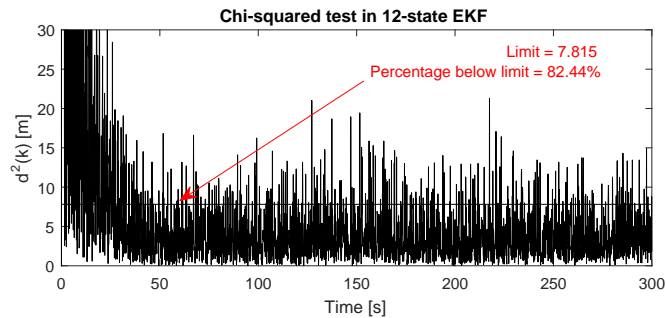


Figure 4.12: Innovation terms  $d^2$  throughout a 300 s simulation of the 12-state EKF.



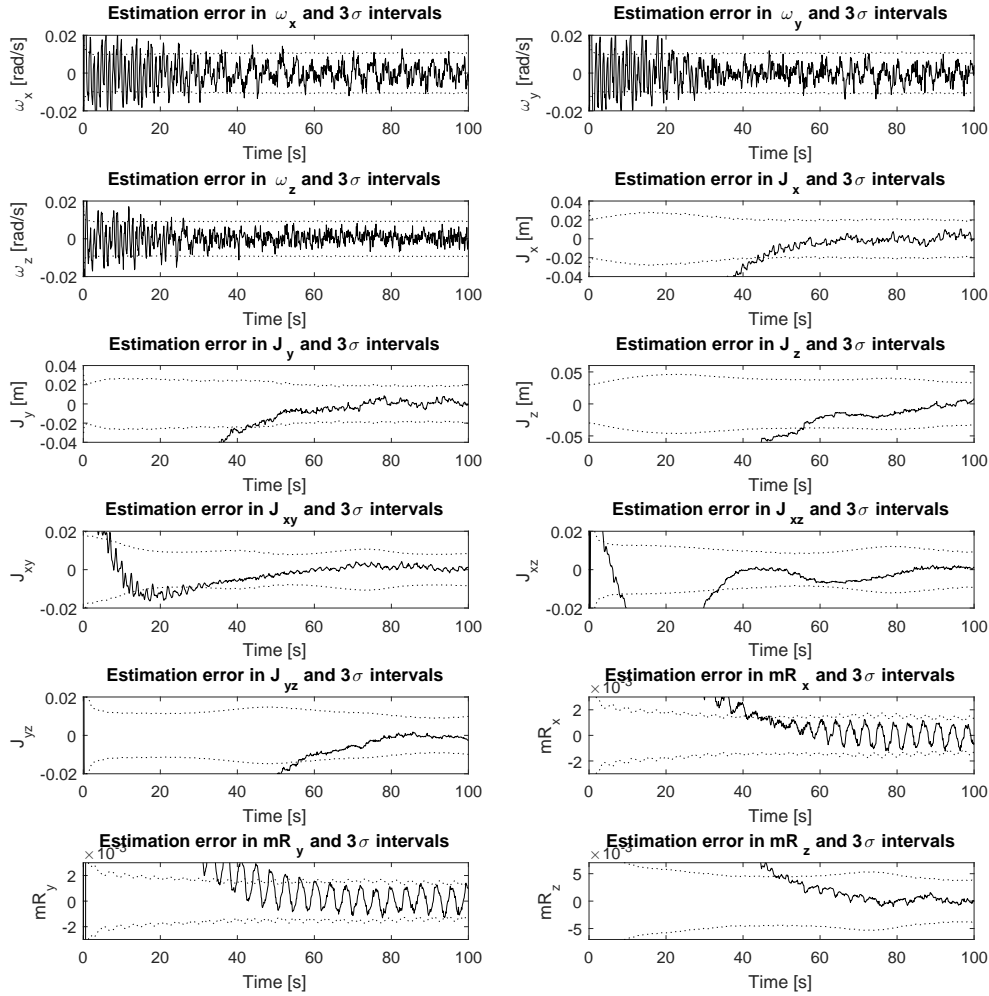


Figure 4.13:  $3\sigma$  intervals for the 12-state EKF.

Fig. 4.14 shows the convergence of the 9 parameters (three for the unbalance vector and six for the inertia tensor). This graph shows that the convergence of the parameters is slower than that obtained in the 6-state EKF, but it tends to the correct parameters values properly. Also, a simulation was performed considering high angular rates, similarly to that tested in the 6-state EKF (in this case,  $\omega = [10 \ 10 \ 10]^T \text{ rad/s}$ ) and is shown in Fig. 4.15.

As can be seen in Fig. 4.15, the estimation of the  $r$  components still does not perform well, although the complete dynamic model is considered in the filter design, and the  $J$  components estimation also present erratic behaviour. It turned out that tuning the  $\mathbf{R}$  and  $\mathbf{Q}$  matrices of the filter become a difficult task. The same noise levels of the low angular rates experiment cannot be used, otherwise the filter would be using almost noiseless signals, since the magnitude of the angular rates are much higher than the previous noise level. Besides this, the sampling time also cannot be the same. As the angular rates become higher, the  $0.1 \text{ s}$  sampling time turned out to be inadequate, since too much information is lost from

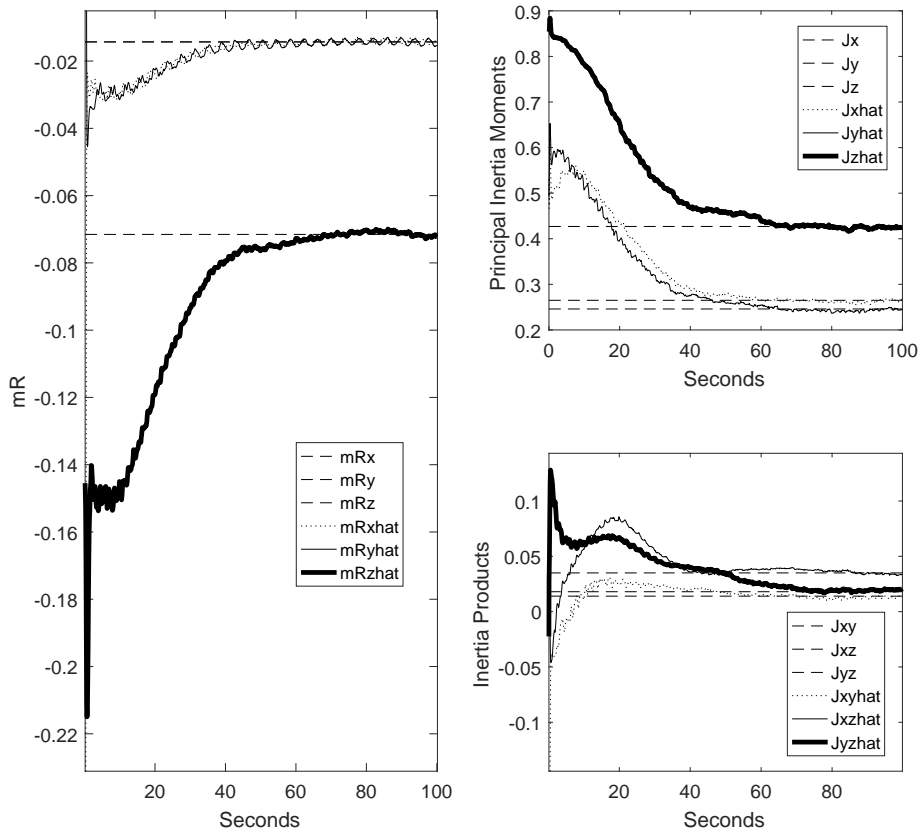


Figure 4.14: Parameter estimation in the 12-state EKF under usual conditions.

the signals. The high omega experiment, when running with lower sampling times, such as  $0.01\text{ s}$ , still does not provide good estimation results, presenting responses similar to those shown in Fig. 4.11, for the 6-state EKF.

It must be emphasized that testing the filters with high angular rates is considered just for study purposes, since the testbed will not reach this velocities in a normal experiment. In the next section, the design of an Unscented Kalman Filter (UKF) is shown, as well as results showing that this high velocity limitation may be overcome.

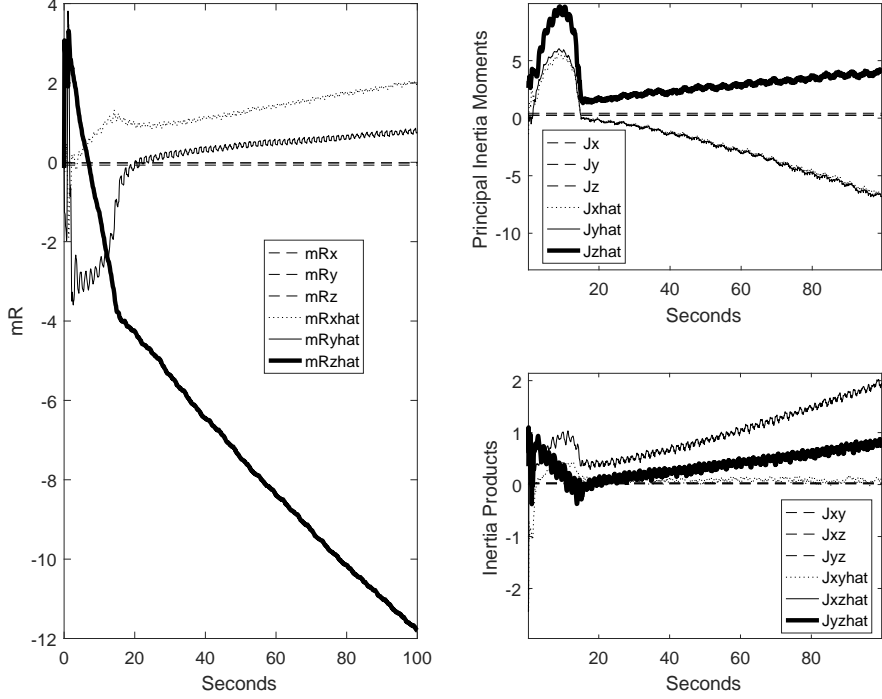


Figure 4.15: Parameter estimation in the 12-state EKF (high angular rates).

### 4.3.3 The UKF applied to the balancing problem

The Unscented Kalman Filter is another method for adapting the Kalman Filter to nonlinear problems. In this case, the approach for approximating the system dynamics is accomplished by determining an optimal set of samples of the system - the sigma points - which can represent the same probability distribution of the nonlinear problem [Simon 2006]. One of the main advantages of using the UKF is that there is no need of determining the jacobians of the model, which may represent great reduction in computational efforts - if they are determined numerically - or no need of determining them analytically - which may be difficult, as in the case of the model in Eq. (4.59).

To apply the UKF, the same state-space form presented in Eq.4.39 may be used, *i.e.* the inertia terms will not be grouped in the state vector  $\mathbf{x}$  as in the augmented EKF of Sec. 4.3.2. The only difference is that, since the complete dynamic model is used, whenever a time update of the state or the sigma points is needed, the following discretization rule is applied:

$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_{k-1} + \dot{\boldsymbol{\omega}}T, \quad (4.95)$$

in which the  $\dot{\boldsymbol{\omega}}$  term is that from Def. 3.2.5 and  $T$  is the sampling time. Following the procedure for implementing the discrete UKF described in [Simon 2006] and presented in Appendix J, in the time update phase the *a priori* estimated state  $\hat{\mathbf{x}}_k^-$  and the *a priori* state covariance matrix  $\mathbf{P}_k^-$  are determined by using the Unscented Transformation (UT). In the measurement update phase, the current output estimate  $\hat{\mathbf{y}}_k$ , the output covariance matrix  $\mathbf{P}_y$

and the cross-covariance matrix  $\mathbf{P}_{xy}$  are determined with through UT and the corrected state  $\hat{\mathbf{x}}_k^-$ , the updated Kalman gain  $\mathbf{K}_k$  and the corrected state covariance matrix  $\mathbf{P}_k^+$  are obtained by using the following equations

$$\begin{aligned}\hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) = \text{a posteriori state estimate} \\ \mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \\ &= [(\mathbf{P}_k^-)^{-1} + \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k]^{-1} \\ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-, \end{aligned}$$

in which the usual KF equations are used for the measurement update phase, since the measurement equation is linear (the angular velocities, present in the state vector, are measured directly by the IMU).

A 50 s simulation is performed using the same conditions listed from Eq. 4.83 to Eq. 4.94. Fig. 4.16 shows the Chi-squared test and the  $3\sigma$  intervals used for tuning the filter, whereas Fig. 4.17 shows the estimation of the unbalance vector components throughout the simulation. In this simulation, the chi-squared test with 95% confidence gave a 90.22% rating, but this rate naturally becomes better when longer simulations are made (e.g. 100 s or 200 s), achieving a  $> 95\%$  rate.

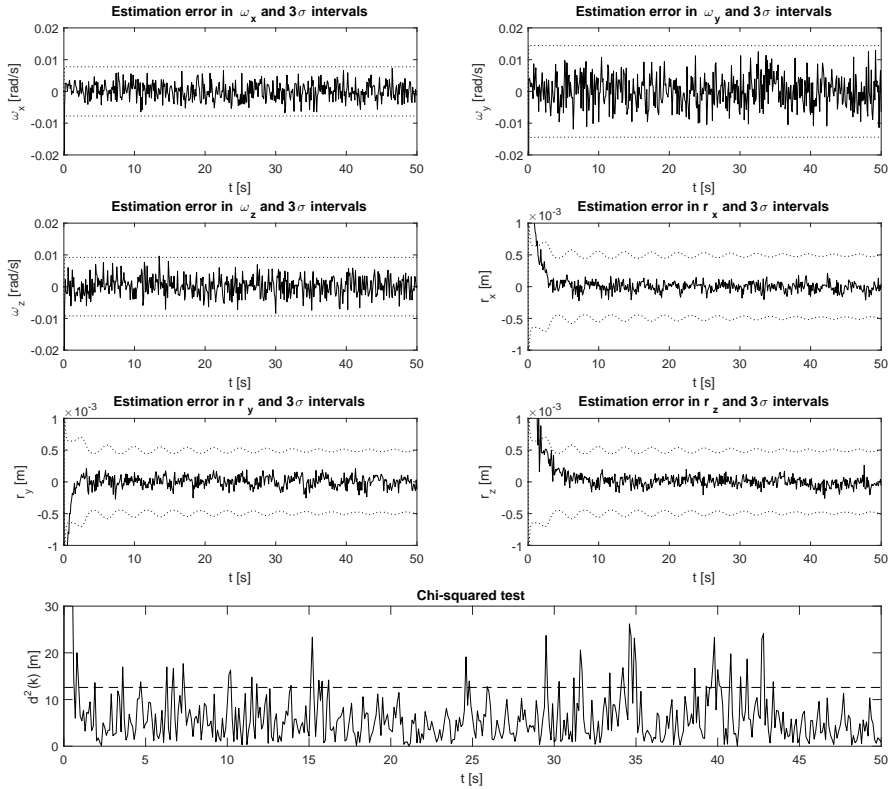


Figure 4.16: UKF consistency and  $3\sigma$  intervals for the estimated states.

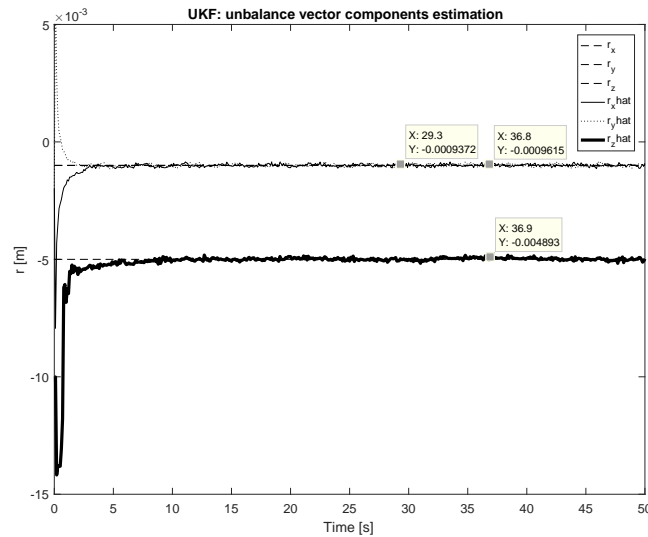


Figure 4.17: UKF: Convergence of the unbalance vector components under normal conditions.

An improvement obtained by using the UKF approach is that the unbalance vector components may be estimated under extreme conditions, specially high angular velocities and higher unbalance magnitude. In a first moment, by simply setting up the initial velocity of the experiment shown in Fig. 4.17 to  $\omega = [10 \ 10 \ 10]^T \text{ rad/s}$ , the parameters estimation diverge as in Fig. 4.15. Also, when the unbalance vector is set up as 20 times bigger (*i.e.*  $r = [20 \ 60 \ 100]^T \text{ mm}$ ), the parameters estimation diverge. However, by simply reducing the sampling time from 0.1 s to 0.01 s, the estimation converges properly with any of these two conditions, alone or together. The result of applying both conditions is shown in Fig. 4.18 and corresponds to a 89.16% rating in the Chi-squared test. As one may notice, the depen-

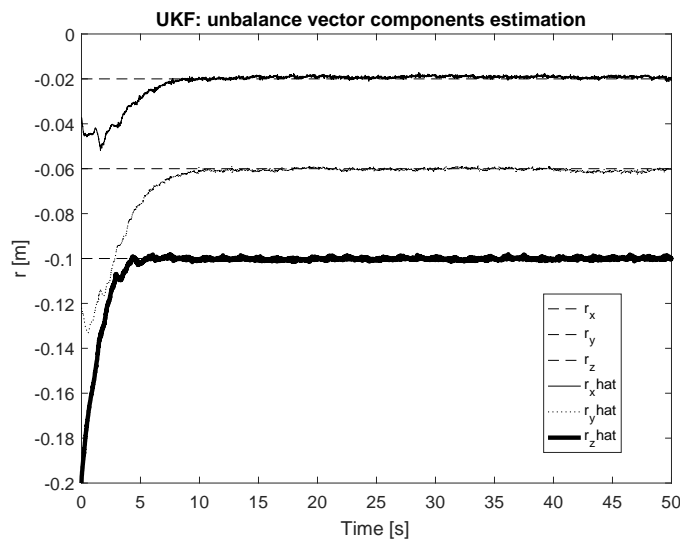


Figure 4.18: UKF: Convergence of the unbalance vector components (high angular velocities and unbalance magnitude).

dependency of the filter on the sampling time is a consequence of the information loss that occurs

when the platform achieves high angular velocities and this dependency is present even when only the unbalance magnitude is raised because a natural consequence of a bigger unbalance vector is that the platform will achieve higher angular velocities (given the initial attitude is the same as in the previous tests). Also, the erroneous tuning of the filter sampling time may lead to the occurrence of aliasing, lowering the filter performance.

The UKF approach turns out to be better than the EKF approaches, since it gives reasonable parameter estimates in any condition and with lower convergence time. A possible reason for this performance may be the strong nonlinearities in the dynamic model of the platform, which could not be properly treated by the local linearization performed by the EKF. Also, this approach may be augmented with the inertia terms as made in Sec. 4.3.2, but it will not be studied further because, as will be explained in Sec. 5, the momentum exchanging devices available until the moment that this work is being written are not capable of providing enough excitation to the platform.

#### 4.3.4 Discussions about observability

The filter in order to develop well has to guarantee some assumptions, such as the observability of the state equation. In this section the observability of the model used for the augmented EKF given by Eq. 4.62 is analyzed, following the same procedure adopted in [Xu et al. 2015]. However, in the case of nonlinear models the observability of the system is not determined as directly as in the linear case, for which reason the following concepts, from [Kang and Barbot 2007], are used:

**Definition 4.3.1 Observability.** *The function  $z = z(t, \epsilon(t), u(t))$  is said to be observable in  $U$  if for any two trajectories  $(t, \epsilon^i(t), u^i(t))$ ,  $i = 1, 2$  in  $U$  defined on a same interval  $[t_0, t_1]$ , the equality*

$$h(\epsilon^1(t), u^1(t)) = h(\epsilon^2(t), u^2(t)), \text{ almost everywhere in } [t_0, t_1] \quad (4.96)$$

*implies*

$$z(t, \epsilon^1(t), u^1(t)) = z(t, \epsilon^2(t), u^2(t)) \quad (4.97)$$

*almost everywhere in  $[t_0, t_1]$ . Suppose for any trajectory  $(t, \epsilon(t), u(t))$  in  $U$  there always exists an open set  $U_1 \in U$  so that  $(t, \epsilon(t), u(t))$  is contained in  $U_1$  and  $z(t, \epsilon, u)$  is observable in  $U_1$ . Then,  $z = z(t, \epsilon, u)$  is said to be locally observable in  $U$ .*

**Definition 4.3.2 Observability (lemma).** *Consider a system without control*

$$\dot{\epsilon} = f(t, \epsilon), \quad \epsilon \in \mathbb{R}^n \quad Y = h(t, \epsilon) \quad (4.98)$$

Let  $U \in \mathbb{R} \times \mathbb{R}^n$  be an open set. Consider

$$V = (Y^T, DY^T, \dots, \dots, D^{l-1}Y^T)^T \quad (4.99)$$

for some  $l > 0$ , where  $D$  is the differentiation operator. If

$$\text{rank} \left( \frac{\partial V}{\partial \epsilon} \right) = n \quad (4.100)$$

for  $(t, \epsilon) \in U$ , then  $z = z(t, \epsilon)$  is locally observable in  $U$ .

In the case of the augmented EKF, the vector  $\mathbf{Y}$  is given by  $\mathbf{Y} = \boldsymbol{\omega} = [\omega_x \ \omega_y \ \omega_z]^T$  and  $D\mathbf{Y}$  is given by  $D\mathbf{Y} = \dot{\boldsymbol{\omega}}$ , in which  $\dot{\boldsymbol{\omega}}$  is given by  $f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u})$  of Eq. 4.61. Then, it follows that the value of  $n$  in Def. 4.3.2 is 6, *i.e.* for the system to be observable the jacobian  $\frac{\partial V}{\partial \epsilon}$  must have rank equal to 6. Considering the definitions of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  previously given in Sec. 4.3.2 and considering  $\mathbf{u}$  is given by

$$\mathbf{u} = [\dot{h}_x \ \dot{h}_y \ \dot{h}_z]^T, \quad (4.101)$$

in which the  $h_i = J_{rw}\omega_{rw,i}$ ,  $i \in \{x, y, z\}$  terms are the angular momenta of the reaction wheels, used as input torque, then the jacobian  $\frac{\partial V}{\partial \epsilon}$  is given by

$$\frac{\partial V}{\partial [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{u}]} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 9} & \mathbf{0}_{3 \times 3} \\ \left[ \frac{\partial D\mathbf{Y}}{\partial \mathbf{x}_1} \right]_{3 \times 3} & \left[ \frac{\partial D\mathbf{Y}}{\partial \mathbf{x}_2} \right]_{3 \times 9} & \left[ \frac{\partial D\mathbf{Y}}{\partial \mathbf{u}} \right]_{3 \times 3} \end{bmatrix}_{6 \times 15}, \quad (4.102)$$

where  $D\mathbf{Y} = \dot{\mathbf{x}}_1 = \mathbf{J}^{-1}(\mathbf{J}\mathbf{x}_1 \times \mathbf{x}_1 + \mathbf{r} \times m\mathbf{g}_b - (\dot{\mathbf{h}} + \mathbf{x}_1 \times \mathbf{h}))$ .

The  $\frac{\partial D\mathbf{Y}}{\partial \mathbf{u}}$  jacobian may be calculated as

$$\frac{\partial D\mathbf{Y}}{\partial \mathbf{u}} = \frac{\partial D\mathbf{Y}}{\partial \dot{\mathbf{h}}} = \frac{\partial}{\partial \dot{\mathbf{h}}} \dot{\mathbf{x}}_1 = \frac{\partial}{\partial \dot{\mathbf{h}}} \mathbf{J}^{-1}(\mathbf{J}\mathbf{x}_1 \times \mathbf{x}_1 + \mathbf{r} \times m\mathbf{g}_b - (\dot{\mathbf{h}} + \mathbf{x}_1 \times \mathbf{h})) \quad (4.103)$$

$$= \frac{\partial}{\partial \dot{\mathbf{h}}} \mathbf{J}^{-1}(-(\dot{\mathbf{h}} + \mathbf{x}_1 \times \mathbf{h})) \quad (4.104)$$

$$= \frac{\partial}{\partial \dot{\mathbf{h}}} \mathbf{J}^{-1}(-(\dot{\mathbf{h}} + \mathbf{x}_1 \times \int \dot{\mathbf{h}} dt)) \quad (4.105)$$

$$= \frac{\partial}{\partial \dot{\mathbf{h}}} \mathbf{J}^{-1}(-(\dot{\mathbf{h}} + \mathbf{x}_1 \times \dot{\mathbf{h}}t)) \quad (4.106)$$

$$= \frac{\partial}{\partial \dot{\mathbf{h}}} \mathbf{J}^{-1}(-(\mathbf{I} + \mathbf{x}_1 \times \mathbf{I}t)), \quad (4.107)$$

in which it was considered that all the reaction wheels provide square waves of output torques, *i.e.* the  $\dot{h}_i$  terms are constant most of the time and independent of the time  $t$ . Thus, from Eq. 4.107 it follows that the rank of the  $\frac{\partial \mathbf{Y}}{\partial \mathbf{u}}$  is 3. Without continuing with the calculus of the jacobians  $\frac{\partial D\mathbf{Y}}{\partial \mathbf{x}_1}$  and  $\frac{\partial D\mathbf{Y}}{\partial \mathbf{x}_2}$ , by simply analyzing the rank of the jacobian matrix in Eq. 4.102, it follows that

$$6 \geq \text{rank} \left( \frac{\partial V}{\partial [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{u}]} \right) = \text{rank}(\mathbf{I}_{3 \times 3}) + \text{rank} \left( \begin{bmatrix} \frac{\partial V}{\partial \mathbf{x}_2} & \frac{\partial V}{\partial \mathbf{u}} \end{bmatrix} \right) \geq \text{rank}(\mathbf{I}_{3 \times 3}) + \text{rank} \left( \frac{\partial V}{\partial \mathbf{u}} \right) = 6, \quad (4.108)$$

thus the rank of  $\frac{\partial V}{\partial [x_1 \ x_2 \ u]}$  must be equal to 6 and the system is observable. In other words, as long as the torques provided by the momentum exchange devices remain different of zero, the observability of the augmented 12-state EKF shown in Sec. 4.3.2 is guaranteed.

### 4.3.5 Parameter estimation summary

After analyzing the various approaches for estimating the unbalance vector of the platform, it is possible to summarize the main results obtained in Secs. 4.1 through 4.3.3 as shown in Table 4.1.

Table 4.1: Summary of parameter estimation approaches.

Approach	Description
LSM	As an advantage, it can be cited that the LSM method presents reasonable fast convergence. However, as disadvantages, it can be cited that its computation effort ( <i>e.g.</i> matrix inversion) and memory requisites increases with the size of the batch of data. Also, it is susceptible to mismodeling, <i>i.e.</i> linear models do not provide the best accuracy.
KF	The Kalman Filter presents an advantage when compared with the LSM method: it is recursive, meaning that it requires less memory. However, it is susceptible to unmodeled dynamics errors as well.
EKF	Compared to the linear KF, the EKF is better since it provides means for considering the complete non-linear model of the platform, which may increase the parameter estimation accuracy. However, it has the disadvantage of requiring the determination of complex jacobians through numerical or analytical methods. Also, it still presents erratic behaviour when extreme conditions are applied to the simulation ( <i>e.g.</i> high angular velocities).
UKF	The UKF, when compared with the EKF, has the advantage of not requiring the calculation of any jacobian. It relies on the idea that it is easier to approximate a probability distribution than an arbitrary nonlinear function. Also, the UKF presented the most robust behaviour of all previously presented methods, giving reasonable estimations under any condition.



## 4.4 The hybrid adaptive control method

In this work, because of the actuation capabilities inherent to the platform, there is no manner to reproduce a similar work to that presented in [Kim and Agrawal 2009], since the only torque exchanging devices in the testbed are the MMUs, which provide only torques orthogonal to gravity. In the other hand, this limitation was already encountered by Chesi et al. and a solution was proposed [Chesi et al. 2014].

The proposed solution, following the procedure developed in [Chesi et al. 2014], can be divided in two major steps.

1. Transverse plane compensation
2. Vertical unbalance compensation

In the first step, the transverse plane compensation, an adaptive control technique is used in order to compensate the unbalance vector for its transverse components. This, as will become clear in the next sections, happens as a consequence of the MMUs being capable of generating only torques that are perpendicular to the gravitational field.

In the second step, assuming that the transverse components (x and y components in the body frame) of the unbalance vector were already nullified, an Unscented Kalman Filter (UKF) is applied in order to estimate the remaining vertical component. The UKF step is similar to that described in Sec. 4.3 with the difference that, instead of estimating the entire unbalance vector, the model assumption  $r_x = r_y = 0$  is used to simplify the filter.

After these two steps, the platform is assumed to be balanced and results may be collected. As follows, these steps are explained in detail.

### 4.4.1 The transverse plane compensation

As stated by Eq. (3.2.5), the dynamic model of the whole platform, considering it as a rigid body performing only rotational movement about a center of rotation (CR), is given by

$$\frac{d\mathbf{H}_o}{dt} = \dot{\mathbf{H}}_o + \boldsymbol{\omega} \times \mathbf{H}_o = \mathbf{M}_o = \mathbf{r} \times M\mathbf{g}_b \quad (4.109)$$

where the BKE is applied to  $\mathbf{H}_o$ , the only external torque being considered is the gravitational torque and all quantities are related to the body frame. Also, the angular momentum, as shown in Eq. (3.56), is given by

$$\mathbf{H}_o = \mathbf{J}\boldsymbol{\omega} . \quad (4.110)$$

During the transverse plane compensation, the MMUs will displace masses accordingly to the adaptive law to be designed. As a consequence, the inertia tensor  $\mathbf{J}$ , which is used in the model and assumed to be known, will be updated as shown in Eq. (4.111)

$$\mathbf{J} = \mathbf{J}_o - m_{\text{MMU}} \sum_i [\mathbf{r}_i \times][\mathbf{r}_i \times], \quad (4.111)$$

where  $m_{\text{MMU}}$  is the amount of mass displaced by a single MMU, which is considered to be equal to the three MMUs placed embedded in the system,  $\mathbf{J}_o$  is the inertia tensor of the system when the MMUs are in their initial positions and  $\mathbf{r}_i$  are the positions of the MMUs in the body frame.

From Eq. (4.109), using Eq. (4.110), the system model may be written as

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \mathbf{r} \times M\mathbf{g}_b, \quad (4.112)$$

in which the gravity vector referred to the body frame  $\mathbf{g}_b$  is obtained by

$$\mathbf{g}_b = \mathbf{Q}_i^b \mathbf{g}_i, \quad \mathbf{g}_i = [0 \ 0 \ -g]^T. \quad (4.113)$$

The assumptions made for the terms in Eq. (4.112) may be summarized in the following list:

- The initial inertia tensor  $\mathbf{J}_o$  is diagonal, *i.e.* the body coordinate system is parallel to the principal inertia axes of the simulator.
- The MMUs are perfectly aligned with the body axes, meaning that each of the  $\mathbf{r}_i$  in Eq. (4.111) has only one variable component.
- The velocities of the MMUs displacements are slow enough to neglect the dynamics of the inertia tensor, *i.e.*  $\dot{\mathbf{J}} = 0$ .

In [Chesi et al. 2014] and [Chesi 2015], the  $\mathbf{Q}_i^b$  direct cosine matrix is described in the “vector-scalar” order of the quaternions, *i.e.* the quaternions are given by  $\mathbf{q} = [q_1 \ q_2 \ q_3 \ q_4]^T$ . To adapt the notation used in these works to the notation adopted here, one may simply replace the  $q_4$  terms with  $q_0$  terms and make algebraic manipulations with the diagonal terms considering  $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ . However, even after making this change, the obtained matrix is not the same as in Eq. (3.17). It was noticed that the matrix adopted by [Chesi et al. 2014] should be transposed. Also, the formula in the  $a_{32}$  term should be  $2(q_2q_3 + q_0q_1)$  (see red portion). These changes are left as an *addendum* to the original works.

$$\mathbf{Q}_i^b = \begin{bmatrix} 1 - 2q_2^2 & 2q_1q_2 & 2q_1q_3 \\ -2q_3^2 & -2q_4q_3 & +2q_4q_2 \\ 2q_1q_2 & 1 - 2q_1^2 & 2q_2q_3 \\ +2q_4q_3 & -2q_3^2 & -2q_4q_1 \\ 2q_1q_3 & 2q_1q_3 & 1 - 2q_1^2 \\ -2q_4q_2 & +2q_4q_1 & -2q_2^2 \end{bmatrix} \quad \text{Chesi version.} \quad (4.114)$$

$$\mathbf{Q}_i^b = \begin{bmatrix} 2q_0^2 - 1 & 2q_1q_2 & 2q_1q_3 \\ +2q_1^2 & -2q_0q_3 & +2q_0q_2 \\ 2q_1q_2 & 2q_0^2 - 1 & 2q_2q_3 \\ +2q_0q_3 & +2q_2^2 & -2q_0q_1 \\ 2q_1q_3 & 2q_1q_3 & 2q_0^2 - 1 \\ -2q_0q_2 & +2q_0q_1 & +2q_3^2 \end{bmatrix} \quad \text{After } q_4 \rightarrow q_0 \text{ and manipulating diagonal.} \quad (4.115)$$

$$\mathbf{Q}_i^b = \begin{bmatrix} 2q_0^2 - 1 & 2q_1q_2 & 2q_1q_3 \\ +2q_1^2 & +2q_0q_3 & -2q_0q_2 \\ 2q_1q_2 & 2q_0^2 - 1 & 2q_2q_3 \\ -2q_0q_3 & +2q_2^2 & +2q_0q_1 \\ 2q_1q_3 & 2q_2q_3 & 2q_0^2 - 1 \\ +2q_0q_2 & -2q_0q_1 & +2q_3^2 \end{bmatrix} \quad \text{Notation of this work..} \quad (4.116)$$

Still regarding the quaternion notation adopted in [Chesi et al. 2014], the quaternion derivatives are also given with the “vector-scalar” order of the quaternions, as shown in Eq. 4.117,

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}. \quad (4.117)$$

It is possible to verify that, to adapt Eq. (4.117) to the notation used in this work, one must simply replace  $q_4$  by  $q_0$  and rearrange the order of the lines in the matrices, leading to Eq. 3.21.

**The design of the adaptive control law** The main concept used to design the adaptive control law is the conservation of the angular momentum. If the external torques applied to the system are null, then the angular momentum is conserved, whereas when they are not - if the gravitational torque is present, for example - the angular momentum is not conserved. Then, the goal is to achieve, with the design of the control torque, a condition in which the derivative of the angular momentum becomes null.

First, considering the design of the system, the control torque is provided by the displace-

ments performed by the MMUs. This torque can be written as

$$\tau_r = m_{\text{MMU}} \sum_i \mathbf{r}_{\text{MMU},i} \times \mathbf{g}^b, \text{ for } i = x, y, z \quad (4.118)$$

where  $\mathbf{r}_{\text{MMU},i}$  is the displacement of the  $i$ th MMU and  $m_p$  is the mass moved by a single MMU.

Also, by adding to Eq. 4.112 the control torque, which is external, since it is a consequence of the gravitational field, the system model is given by

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \mathbf{r} \times M\mathbf{g}_b + \tau_r. \quad (4.119)$$

From the successful design of the control torque  $\tau_r$ , the MMU displacements  $\mathbf{r}_{\text{MMU}}$  may be obtained, as explained posteriorly. First, the  $\Phi$  is defined as  $\Phi = \mathbf{r}$ , where  $\mathbf{r}$  is the initial unbalance vector parameter, a constant parameter of the system. Then, the gravitational torque resulting from the entire system may be written as

$$\Theta \times m\mathbf{g}_b = \Phi\Theta, \quad (4.120)$$

$$\Phi(\mathbf{q}) = -m\mathbf{g}^b \times, \quad (4.121)$$

in which  $\Phi(\mathbf{q})$  is a function of the attitude quaternion  $\mathbf{q}$ , since it depends on the gravity vector referred to the body frame. Considering the  $\Theta$  parameter is an unknown, its estimate is given by  $\hat{\Theta}$ , whereas the error in estimation is given by

$$\tilde{\Theta} = \Theta - \hat{\Theta}(t). \quad (4.122)$$

Also, the following quantities are defined:

- $\mathbf{X} = [\mathbf{q} \ \boldsymbol{\omega} \ \tilde{\Phi}]^T$  - the state vector
- $\boldsymbol{\omega}_g$  - vertical component of the system angular velocity  $\boldsymbol{\omega}$ , parallel to  $\mathbf{g}_b$ .
- $\boldsymbol{\omega}_p$  - transverse component of the system angular velocity  $\boldsymbol{\omega}$ , orthogonal to  $\mathbf{g}_b$ . This component may be calculated as

$$\boldsymbol{\omega}_p = \mathbf{P}_p(\mathbf{q})\boldsymbol{\omega}, \quad (4.123)$$

in which the  $\mathbf{P}_p(\mathbf{q})$  operator is a function of the attitude quaternion  $\mathbf{q}$ , since it depends on the gravity field referred to the body frame, and is given by

$$\mathbf{P}_p(\mathbf{q}) = \left[ \mathbf{I} - \frac{\mathbf{g}^b(\mathbf{g}^b)^T}{\|\mathbf{g}^b\|^2} \right]. \quad (4.124)$$

- The following rules apply for the  $\omega_g$  and  $\omega_p$  vectors

$$\omega = \omega_g + \omega_p, \quad (4.125)$$

$$\omega_g^T \omega_p = 0, \quad (4.126)$$

$$(g^b)^T \omega_p = 0. \quad (4.127)$$

The state feedback control law and the adaptive law are designed by considering the following Lyapunov function,

$$V(\mathbf{X}) = V(\mathbf{q}, \omega, \tilde{\Phi}) = \frac{1}{2} \omega^T \mathbf{J} \omega + \frac{1}{2} \tilde{\Phi}^T \tilde{\Phi} + \frac{1}{2} \mathbf{q}^T \mathbf{q}, \quad (4.128)$$

which was designed in [Chesi 2015]. Further details on the Lyapunov theory as well as common practices when designing Lyapunov functions may be found in [Haddad and Chellaboina 2011].

The derivative of this Lyapunov function is given by

$$\dot{V}(t) = \omega^T (\mathbf{J} \dot{\omega}) + \tilde{\Theta}^T \dot{\tilde{\Theta}} + \underbrace{\mathbf{q}^T \dot{\mathbf{q}}}_{\substack{0 \\ \text{Eq. (4.117), } \dot{\mathbf{q}}}} \quad (4.129)$$

$$= \omega^T \underbrace{(-\omega \times \mathbf{J} \omega + \Phi \Theta + \tau_r)}_{\substack{\mathbf{J} \dot{\omega} \text{ from Eq. (4.119)}}} + \tilde{\Theta}^T \dot{\tilde{\Theta}} \quad (4.130)$$

$$= \omega^T \Phi \Theta + \omega^T \tau_r + \tilde{\Theta}^T \dot{\tilde{\Theta}} \quad (4.131)$$

Replacing  $\Theta = \hat{\Theta} + \tilde{\Theta}$ ,

$$\dot{V} = \omega^T \Phi \hat{\Theta} + \tilde{\Theta}^T (\Phi^T \omega + \dot{\tilde{\Theta}}) + \omega^T \tau_r \quad (4.132)$$

Then, choosing the adaptive law  $\dot{\hat{\Theta}}$  as

$$\dot{\hat{\Theta}} = \Phi^T \omega \quad (4.133)$$

and the control torque to be

$$\tau_r = -\Phi \hat{\Theta} - k_p \omega_p, \quad (4.134)$$

the Lyapunov derivative may be further developed as

$$\dot{V} = \omega^T \Phi \hat{\Theta} - \omega^T \Phi \hat{\Theta} - k_p \omega^T \omega_p \quad (4.135)$$

$$= -k_p \omega^T \omega_p \quad (4.136)$$

$$= -k_p (\omega_g + \omega_p)^T \omega_p \quad (4.137)$$

Finally, considering  $\omega_g^T \omega_p = 0$ , the derivative of the chosen Lyapunov function  $V$  results

in

$$\dot{V} = -k_p \|\boldsymbol{\omega}_p\|^2, \quad (4.138)$$

which is negative semidefinite. Thus, the closed loop system is stable in the Lyapunov sense. Additionally, as shown in [Chesi 2015], following as a consequence of the LaSalle Invariance Principle, the system will converge to the largest invariant set  $\Omega$  contained in

$$\{\dot{V}(t) \equiv 0\} = \{\mathbf{X} : \dot{V}(\mathbf{q}, \boldsymbol{\omega}, \tilde{\boldsymbol{\Phi}}) \equiv 0\} = \{\boldsymbol{\omega}_p(t) \equiv 0\} \quad (4.139)$$

$$\text{i.e., } \lim_{t \rightarrow \infty} \boldsymbol{\omega}_p = 0 \quad (4.140)$$

**Determining the control torque parameters** Given that the designed control torque  $\boldsymbol{\tau}_r$  accomplishes the desired goal, the next task is mapping the obtained control torque to MMU displacements. This may be obtained as follows. First, to verify that the designed control torque can be generated by simply displacing the movable masses, one may pre-multiply Eq. (4.134) by  $(\mathbf{g}_b)^T$ , use the rules in Eqs. (4.125) to (4.127) and the definition of  $\boldsymbol{\Phi}$  from Eq. (4.121), giving

$$(\mathbf{g}_b)^T \boldsymbol{\tau}_r = -(\mathbf{g}_b)^T \boldsymbol{\Phi} \hat{\boldsymbol{\Theta}} - k_p (\mathbf{g}_b)^T \boldsymbol{\omega}_p = 0, \quad (4.141)$$

*i.e.* since the scalar product between  $(\mathbf{g}_b)$  and  $\boldsymbol{\tau}_r$  equals zero, these vectors are mutually orthogonal. Transcribing Eq. (4.118) after writing the movable masses positions in a single vector  $\mathbf{r}_{\text{MMU}}$ , it follows that

$$\boldsymbol{\tau}_r = m_{\text{MMU}} (-\mathbf{g}_b \times \mathbf{r}_{\text{MMU}}), \quad (4.142)$$

which gives a solution for  $\mathbf{r}_{\text{MMU}}$ . As proved in [Chesi 2015], a solution to Eq. (4.142) is

$$\mathbf{r}_{\text{MMU}} = \frac{\mathbf{g}_b \times \boldsymbol{\tau}_r}{\|\mathbf{g}_b\|^2 m_{\text{MMU}}}, \quad (4.143)$$

which can be verified by substituting  $\mathbf{r}$  given by Eq. (4.143) into Eq. (4.142), as follows

$$-m_{\text{MMU}} \left( \frac{\mathbf{g}_b \times \mathbf{g}_b \times \boldsymbol{\tau}_r}{\|\mathbf{g}_b\|^2 m_{\text{MMU}}} \right) = -m_{\text{MMU}} \left( \frac{(\mathbf{g}_b \cdot \boldsymbol{\tau}_r) \mathbf{g}_b - (\mathbf{g}_b \cdot \mathbf{g}_b) \boldsymbol{\tau}_r}{\|\mathbf{g}_b\|^2 m_{\text{MMU}}} \right) = \boldsymbol{\tau}_r. \quad (4.144)$$

In summary, to properly control the movable masses, one must determine the control signal  $\boldsymbol{\tau}_r$  from Eq. (4.134) and then substitute its value in Eq. (4.143) to obtain  $\mathbf{r}_{\text{MMU}}$ .

**About the convergence of the estimated unbalance vector  $\hat{\boldsymbol{\Theta}}$**  The convergence of the estimated unbalance vector  $\hat{\boldsymbol{\Theta}}$  was already studied in [Chesi et al. 2014]. However, for convenience purposes, it is replicated in what follows. Considering, from application of the LaSalle Invariance Principle to the problem, that  $\boldsymbol{\omega}_p \rightarrow 0$ , then the adaptation law gives that

$$\dot{\hat{\boldsymbol{\Theta}}} = \boldsymbol{\Phi}^T \boldsymbol{\omega} = \boldsymbol{\Phi}^T \boldsymbol{\omega}_p + \boldsymbol{\Phi}^T \boldsymbol{\omega}_g = \boldsymbol{\Phi}^T \boldsymbol{\omega}_g = \mathbf{0}, \quad (4.145)$$

considering  $\Phi^T \omega_g$  represents the cross-product between two vectors point in the same direction, the gravity field  $\mathbf{g}_b$  and  $\omega_g$ , which is the component of  $\omega$  pointing in the same direction of  $\mathbf{g}_b$ . To study the effect of the proposed control law,  $\dot{\omega}$  is isolated from Eq. (4.119) after substitution of the control law, giving

$$\dot{\omega} = \mathbf{J}^{-1}(-\omega \times \mathbf{J}\omega + \Phi \tilde{\Theta} - k_p \omega_p). \quad (4.146)$$

The projection factor shown in Eq. (4.124) may be developed, in this case, as

$$\mathbf{P}_p(\mathbf{q}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} (g_{b,x})^2 & g_{b,x}g_{b,y} & g_{b,x}g_{b,z} \\ g_{b,y}g_{b,x} & (g_{b,y})^2 & g_{b,y}g_{b,z} \\ g_{b,z}g_{b,x} & g_{b,z}g_{b,y} & (g_{b,z})^2 \end{bmatrix} \frac{1}{\|\mathbf{g}_b\|^2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (4.147)$$

which follows from the fact that  $g_{b,x} = g_{b,y} = 0$ , since when  $\omega_p = 0$  the platform rotates around the gravity field, which is along the Z-axis of the body frame, and from the fact that when  $g_{b,x} = g_{b,y} = 0$ , then  $\mathbf{g}_b = g_{b,z} \mathbf{e}_z$ , following that  $\|\mathbf{g}_b\| = g_{b,z}$ . The square exponent in red is an addendum to the analysis shown in [Chesi et al. 2014] and [Chesi 2015]. Applying the projection operator to both sides of Eq. (4.146), it follows, considering the projection factor is constant when  $\omega_p = 0$ , that

$$\mathbf{P}_p \dot{\omega} = \frac{d(\mathbf{P}_p \omega)}{dt} = \dot{\omega}_p = \mathbf{P}_p \mathbf{J}^{-1}(-\omega \times \mathbf{J}\omega + \Phi \tilde{\Theta} - K_p \omega_p). \quad (4.148)$$

Knowing that

$$\Phi = M \begin{bmatrix} 0 & g_{b,z} & 0 \\ -g_{b,z} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (4.149)$$

$$\omega = \omega_p + \omega_g \quad (4.150)$$

$$\mathbf{0} = \omega_g \times \mathbf{J}\omega_g \quad (4.151)$$

then Eq. (4.148) may be further simplified as

$$\mathbf{0} = \mathbf{P}_p \mathbf{J}^{-1}(-\omega_g \times \mathbf{J}\omega_g + \Phi \tilde{\Theta}) \quad (4.152)$$

$$\mathbf{0} = \mathbf{P}_p \mathbf{J}^{-1}(\Phi \tilde{\Theta}) \quad (4.153)$$

$$\mathbf{0} = M \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} J_x^{-1} & 0 & 0 \\ 0 & J_y^{-1} & 0 \\ 0 & 0 & J_z^{-1} \end{bmatrix} \begin{bmatrix} 0 & g_{b,z} & 0 \\ -g_{b,z} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{\Theta}_x \\ \tilde{\Theta}_y \\ \tilde{\Theta}_z \end{bmatrix}, \quad (4.154)$$

which gives

$$J_x^{-1} g_{b,z} \tilde{\Theta}_y = 0 \quad (4.155)$$

$$-J_y^{-1} g_{b,z} \tilde{\Theta}_x = 0 \quad (4.156)$$

$$0\tilde{\Theta}_z = 0 \quad (4.157)$$

In conclusion, since  $J_x^{-1}$ ,  $J_y^{-1}$  and  $g_{b,z}$  are all terms different from zero, then both  $\tilde{\Theta}_x$  and  $\tilde{\Theta}_y$  are null, as desired. Also, nothing can be said about  $\tilde{\Theta}_z$ , which is the reason why  $\tilde{\Theta}_z$  will be estimated by using an UKF, as explained in the next section.

**Simulations** In order to test the transverse unbalance compensation method described in this section, simulations were performed. Fig. 4.19 shows the angular velocities of the platform, whereas Fig. 4.20 shows the balance masses positions throughout the balancing procedure and Fig. 4.21 shows the estimated unbalance vector components obtained from the  $\hat{\Theta}$  vector. In this simulation, the testbed parameters and the control gain  $k_p$  were set as

$$\mathbf{J} = \begin{bmatrix} 0.265 & -0.014 & -0.035 \\ -0.014 & 0.246 & -0.018 \\ -0.035 & -0.018 & 0.427 \end{bmatrix} kg \cdot m^2, \quad (4.158)$$

$$\mathbf{r}^{off} = [-1 \ -2 \ -5] \cdot 10^{-3} m, \quad (4.159)$$

$$M = 14.307 kg, \quad (4.160)$$

$$m_{MMU} = 0.7 kg, \quad (4.161)$$

$$k_p = 0.5, \quad (4.162)$$

in which the control gain  $k_p$  was empirically set, since the purpose of this simulation is to test the control scheme convergence only, without setting any control performance specifications.

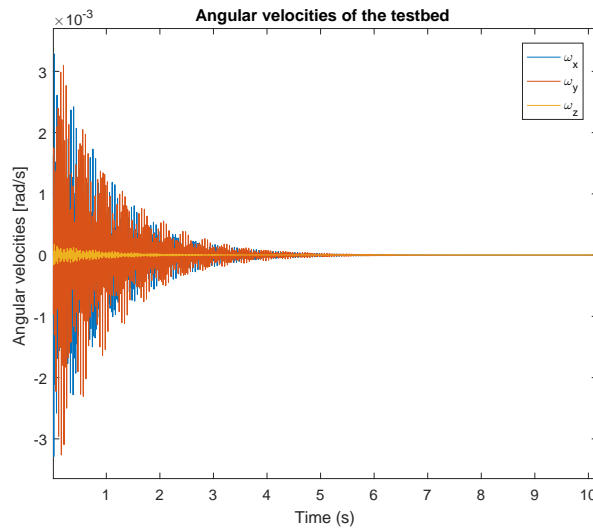


Figure 4.19: Testbed angular velocities during transverse balancing.



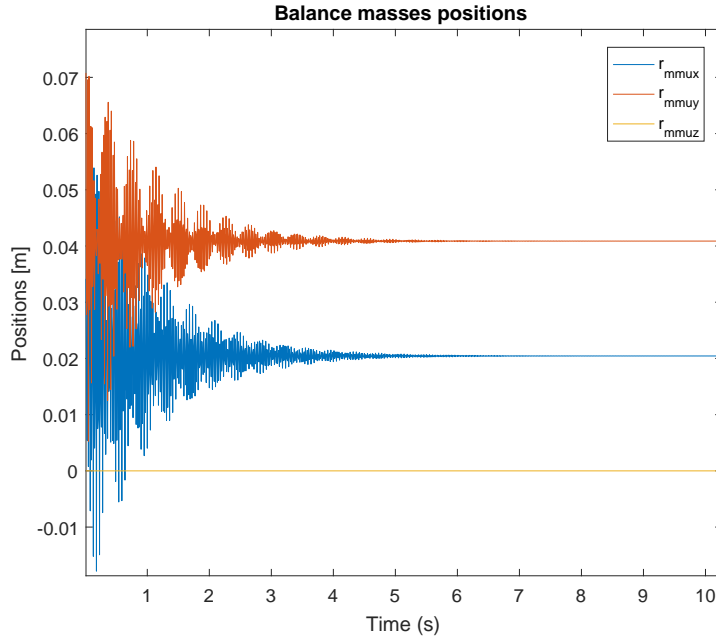


Figure 4.20: MMUs positions during transverse balancing.

As one may notice, the graph in Fig. 4.20 shows fast variations in the MMUs positions, since the dynamics of the MMUs were not considered, *i.e.* the MMUs change its positions instantly during the simulation. In fact, to implement this scheme physically, the dynamics must be considered, which means that the convergence of the terms in all graphs will delay more than what is shown. From all these graphs, it also can be seen that the terms related to the z-axis does not vary. It is inherent to the limitation of the MMUs to generate torque in the transverse plane only, which means that the z-axis related terms will stay in the initial condition set up in the simulation. Also, the little variation in the  $\omega_z$  signal in Fig. 4.19 is a consequence of the presence of the products of inertia in the simulated body model. Although the balancing scheme developed by Chesi considers a diagonal matrix of inertia  $\mathbf{J}$ , the simulated dynamics considered an inertia tensor will all elements different from zero, as shown in Eq. 4.158. As a final reminder for the transverse balancing phase, additional care must be taken when selecting the gain  $k_p$ , since the magnitude of the signals in the graph of Fig. 4.20 cannot be bigger than the excursion of the MMUs.

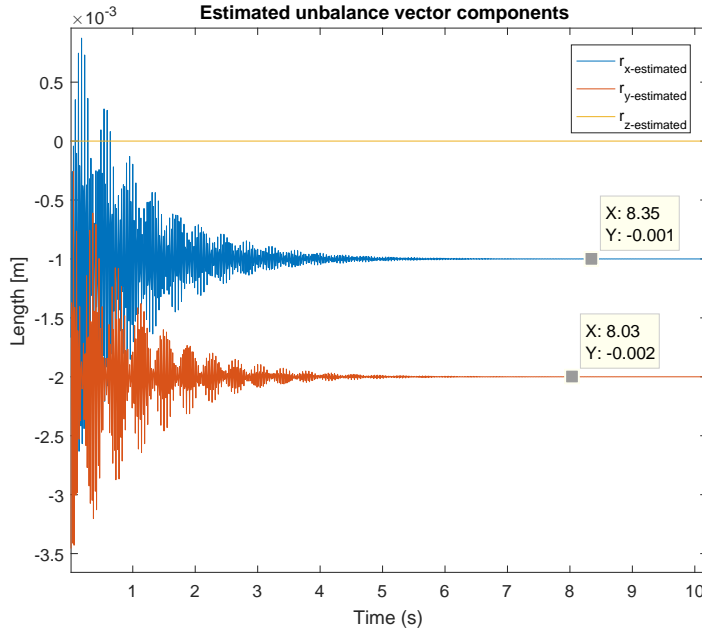


Figure 4.21: The estimated unbalance vector components ( $\hat{\Theta}$ ).

#### 4.4.2 The vertical imbalance compensation

As explained previously, the adaptive control can only compensate the transverse components of the unbalance vector, since the torque provided by the actuator is always perpendicular to the gravity vector. After performing the transverse plane compensation, a filter is used to estimate the remaining unbalance.

Following the procedure developed by [Chesi 2015], an Unscented Kalman Filter is used. The state vector of the proposed filter contains the angular velocities of the system, as well as the desired parameter to be estimated, as shown in Eq. 4.163,

$$\mathbf{x} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ r_z \end{bmatrix}. \quad (4.163)$$

This procedure is similar to that used in the EKF imbalance compensation shown in Sec. 4.3.1, in which the state vector containing the dynamics of the system is augmented to estimate some of the parameters of the system and may also be viewed as a particular case of the UKF developed in Sec. 4.3.3. However, some differences are perceptible when comparing to the models of the EKF or UKF methods shown previously, such as considering  $\mathbf{J}$  a diagonal matrix, which is a reasonable consideration given the real testbed inertia and simplifies the foregoing analyses (*e.g* the observability analysis). In this case, since the vertical imbalance is considered to be constant throughout the estimation, the corresponding dynamics is given by  $\dot{r}_z = 0$ . The dynamics of the simulator remains the same used in

the transverse plane compensation, except for the gravitational torque, which is given by Eq. 4.164,

$$\mathbf{r} \times m\mathbf{g}^b = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & r_z \\ mg_x^b & mg_y^b & mg_z^b \end{vmatrix} = \begin{bmatrix} -mg_y^b r_z \\ mg_x^b r_z \\ 0 \end{bmatrix}, \quad (4.164)$$

since it is assumed that the  $r_x$  and  $r_y$  components were nullified in the transverse plane compensation. Using this assumptions, the dynamics of the system in state-space form may be given, in continuous time, by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{w}(\mathbf{t}), \quad (4.165)$$

in which  $\mathbf{f}$  is given by

$$\mathbf{f} = \begin{bmatrix} \mathbf{J}^{-1} \left( -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \begin{bmatrix} -mg_y^b r_z \\ mg_x^b r_z \\ 0 \end{bmatrix} \right) \\ 0 \end{bmatrix}_{4 \times 1} \quad (4.166)$$

and  $\mathbf{w}(\mathbf{t})$  is the process noise. The angular velocity measurements may be directly obtained from a gyroscope, which means that the output equation is linear on the state vector. The discretized output equation is given by Eq. 4.167

$$[\mathbf{y}_k]_{3 \times 1} = \mathbf{H}_k(\mathbf{x}_k) + \mathbf{v}_k = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 1} \end{bmatrix}_{3 \times 4} [\mathbf{x}_k]_{4 \times 1} + \mathbf{v}_k \quad (4.167)$$

in which  $\mathbf{I}$  is the identity matrix and  $\mathbf{v}_k$  represents the measurement noise, *i.e.* the noise from the gyroscope. To run the UKF, the sigma points  $\hat{\mathbf{x}}_k^{(i)}$  in the time update phase are calculated as

$$\hat{\mathbf{x}}_k^{(i)} = \hat{\mathbf{x}}_{k-1}^{(i)} + \mathbf{f} \cdot dt, \quad (4.168)$$

in which  $dt$  is the sampling time, and the filter equations are given by

$$\begin{aligned} [\mathbf{K}_k]_{4 \times 3} &= [\mathbf{P}_{xy}]_{4 \times 3} [\mathbf{P}_y^{-1}]_{3 \times 3} \\ [\hat{\mathbf{x}}_k^+]_{4 \times 1} &= [\hat{\mathbf{x}}_k^-]_{4 \times 1} + [\mathbf{K}_k]_{4 \times 3} ([\mathbf{y}_k]_{3 \times 1} - [\hat{\mathbf{y}}_k]_{3 \times 1}) \\ [\mathbf{P}_k^+]_{4 \times 4} &= [\mathbf{P}_k^-]_{4 \times 4} - [\mathbf{K}_k]_{4 \times 3} [\mathbf{P}_y]_{3 \times 3} [\mathbf{K}_k^T]_{3 \times 4}, \end{aligned}$$

Fig. 4.22 shows the estimation of the  $r_z$  component when the simulation settings are

given by

$$\mathbf{R} = \text{diag}([0.005^2 \ 0.005^2 \ 0.005^2]) \text{ rad}^2/\text{s}^2 \quad (4.169)$$

$$\mathbf{Q} = \text{diag}([0.00005 \ 0.00005 \ 0.00005 \ (5 \cdot 10^{-4})^2]) [\text{rad}^2/\text{s}^2 \ \text{m}^2] \quad (4.170)$$

$$dt = 0.1 \text{ s} \quad (4.171)$$

$$r_z = 5 \cdot 10^{-3} \text{ m} \quad (4.172)$$

$$\hat{r}_{z,0} = 10 \cdot 10^{-3} \text{ m} \quad (4.173)$$

$$J_x = 0.265 \text{ kg} \cdot \text{m}^2 \quad (4.174)$$

$$J_y = 0.246 \text{ kg} \cdot \text{m}^2 \quad (4.175)$$

$$J_z = 0.427 \text{ kg} \cdot \text{m}^2, \quad (4.176)$$

as well as the  $3\sigma$  intervals and the chi-squared test with 95% confidence interval, which, in this case, gave a 97.1% rating.

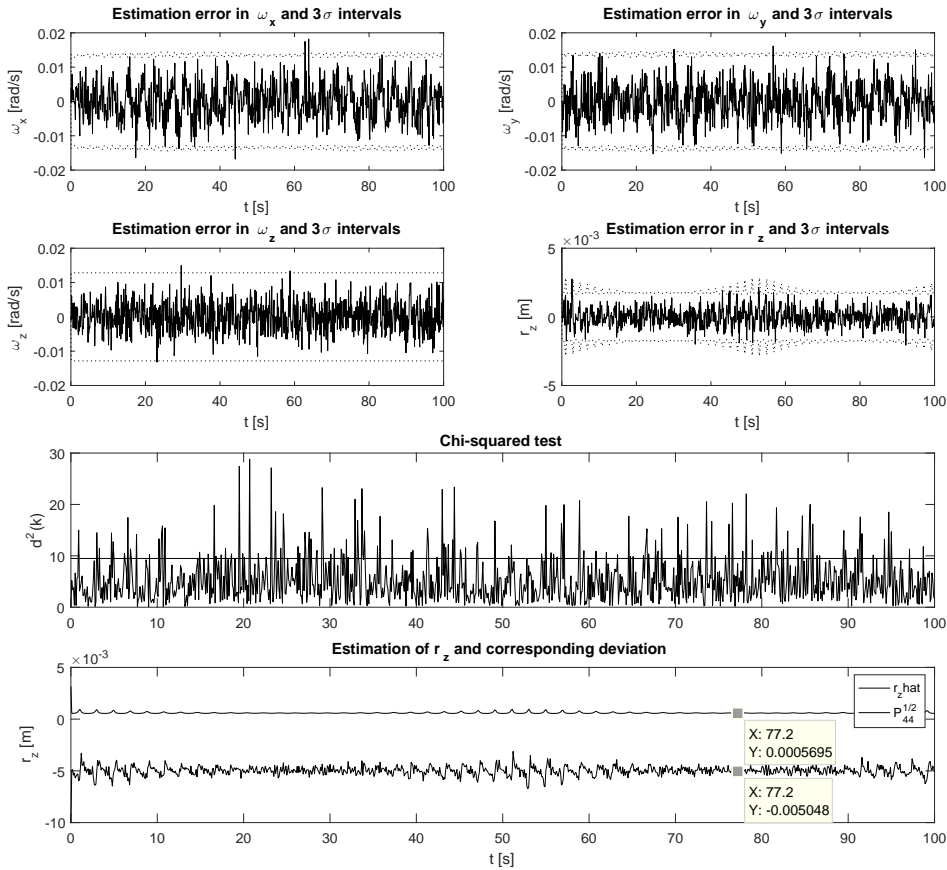


Figure 4.22: Results of the simulation of the 4-state UKF.

Following the same procedure adopted when analyzing the observability of the EKF balancing procedure, Eq. 4.100 may be applied along with the previously given observability definition. From Eq. 4.166 and considering that the measurements are given by  $h(t) =$

$[\omega_x \ \omega_y \ \omega_z]$ , the variable to be estimated is  $z(t, \epsilon) = r_z$ . Being  $D$  the differentiation operator, the function  $\mathbf{V}$  described in Def. 4.3.2 is developed by considering  $\mathbf{Y}$  and  $D\mathbf{Y}$  given by

$$\mathbf{Y} = [\boldsymbol{\omega}] \text{ and by}$$

$$D\mathbf{Y} = [\dot{\boldsymbol{\omega}}] = \begin{bmatrix} \mathbf{J}^{-1} \left( -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \begin{bmatrix} -mg_y^b r_z \\ mg_x^b r_z \\ 0 \end{bmatrix} \right) \\ 0 \end{bmatrix}.$$

As follows from Def. 4.3.2, the interest relies on discovering on whether the following matrix  $\mathbf{V}$

$$\mathbf{V} = \frac{\partial}{\partial \mathbf{X}} \begin{bmatrix} \mathbf{Y} \\ D\mathbf{Y} \end{bmatrix} = \begin{bmatrix} \left[ \frac{\partial \mathbf{Y}}{\partial \mathbf{X}} \right]_{4 \times 3} \\ \left[ \frac{\partial D\mathbf{Y}}{\partial \mathbf{X}} \right]_{4 \times 3} \end{bmatrix} \quad (4.177)$$

has full column rank or not, *i.e.* if its rank equals 4 or not (4 is the length of the state vector). As  $\boldsymbol{\omega}$  is in the state vector itself,  $\frac{\partial \mathbf{Y}}{\partial \boldsymbol{\omega}} = \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\omega}} = \mathbf{I}_{3 \times 3}$ . Also, as  $r_z$  is not contained in  $\mathbf{Y}$ ,  $\frac{\partial \mathbf{Y}}{\partial r_z} = \frac{\partial \boldsymbol{\omega}}{\partial r_z} = \mathbf{0}_{3 \times 1}$  and the  $\mathbf{V}$  is simplified as

$$[\mathbf{V}]_{6 \times 4} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{V}_{21} & \mathbf{V}_{22} \end{bmatrix}, \quad (4.178)$$

in which  $[\mathbf{V}_{21}]_{3 \times 3}$ , differentiating the result in Eq. (4.43) with respect to  $\boldsymbol{\omega}$ , and  $[\mathbf{V}_{22}]_{3 \times 1}$ , differentiating Eq. (4.166) with respect to  $r_z$ , are given by

$$\mathbf{V}_{21} = \begin{bmatrix} 0 & \omega_z \left( \frac{J_y - J_z}{J_x} \right) & \omega_y \left( \frac{J_y - J_z}{J_x} \right) \\ \omega_z \left( \frac{J_z - J_x}{J_y} \right) & 0 & \omega_x \left( \frac{J_z - J_x}{J_y} \right) \\ \omega_y \left( \frac{J_x - J_y}{J_z} \right) & \omega_x \left( \frac{J_x - J_y}{J_z} \right) & 0 \end{bmatrix} \quad (4.179)$$

$$\mathbf{V}_{22} = \frac{\partial \dot{\boldsymbol{\omega}}}{\partial r_z} = \begin{bmatrix} -\frac{mg_y^b}{J_{xx}} \\ -\frac{mg_x^b}{J_{yy}} \\ 0 \end{bmatrix}. \quad (4.180)$$

In other words, to provide the required observability of the state vector it suffices to guarantee that  $g_x^b$  or  $g_y^b$  in  $\mathbf{V}_{22}$  do not equal zero. It can be done by tumbling the simulator in order to prevent the roll and pitch angles of reaching 0 *rad*.

# Chapter 5

## Tests and Results

In this chapter the balancing methods shown in Sec. 4 are tested in the physical platform. The analysis will comprise the following techniques:

- The Kalman Filter;
- The 6-state Extended Kalman Filter (simplified inertia);
- The 6-state Extended Kalman Filter (complete inertia); and
- The 6-state Unscented Kalman Filter.

As one may notice, further experiments concerning the Least Squares Method are not performed, since it was already studied in [Silva et al. 2018], being its appearance in this section summarized by the results obtained in that work. Also, neither the Extended Kalman Filter, nor the Unscented Kalman Filter are tested in its 12-state forms, which also estimates the inertia tensor components. The main reason for which these methods were not tested is that these methods require an input torque and the only available momentum exchange devices - the reaction wheels and the magnetorquers - cannot provide the required torque magnitude. Magnetorquers are used in slower dynamics experiments when compared with reaction wheels, since they provide torques with low magnitude. On the other hand, the reaction wheels, which would be the best actuator candidate, also cannot provide an adequate torque level, taking into account the resolution of the gyroscopes in the IMU, which is  $0.01 \text{ rad/s}$ , and the signal-noise ratio in this sensor. To illustrate it, it is possible to estimate the reachable angular velocity of the testbed when the reaction wheels saturate at maximum velocity by applying the angular momentum conservation principle, which in this case states that

$$\Delta \mathbf{H} = \mathbf{0} \Rightarrow \mathbf{J}\boldsymbol{\omega} + \mathbf{J}_{\text{RW}}\boldsymbol{\omega}_{\text{RW}} = \mathbf{0} , \quad (5.1)$$

in which  $\mathbf{J}_{\text{RW}}$  and  $\boldsymbol{\omega}_{\text{RW}}$  are the inertia tensor of the reaction wheels, whose characteristics are available in Sec. 2.1.4, and the angular velocity of the reaction wheels, respectively.

Simplifying the analysis for one of the axis, the maximum angular velocity reachable by the testbed in this axis is obtained as

$$\omega_z = -\frac{J_{RW,z}}{J_z}\omega_{RW} = -\frac{0.000032}{0.427}7000 \text{ rpm} \approx -0.5246 \text{ rpm} \approx -0.0549 \text{ rad/s} \quad (5.2)$$

in which the analysis was made in the  $Z_b$  axis. Comparing this angular velocity with the resolution of the IMU, the resolution represents approximately 20% of the signal magnitude, which indicates an ill-conditioned signal and worsen when the noise in this signal or lower angular velocities are also considered. Moreover, this analysis was made in the  $Z_b$  axis since the gravitational torque, which must also be considered, is more critical in the  $X_b$  and  $Y_b$  axes, *i.e.* the  $Z_b$  axis represents the best case scenario. The maximum torque of the reaction wheels also cannot help the testbed to reach higher angular velocities, since the highest attainable angular acceleration is given by (in the  $Z_b$  axis)

$$\tau = J_z\dot{\omega}_z \Rightarrow \dot{\omega}_z = \frac{0.0037}{0.427} = 0.0087 \text{ rad/s}^2, \quad (5.3)$$

meaning that the testbed would need 6 s to reach the 0.05 rad/s velocity, which is sufficient time for the reaction wheels to saturate at its maximum velocity. In conclusion, the available actuators are not capable of guaranteeing a good filter performance or even the proper observability of the state vector, as analyzed in Sec. 4.3.4. This problem will be avoided with the acquisition of new actuators, capable of providing enough torque, for which reason the validation of the 12-state filters are left for future works.

Another balancing technique that was not tested is the two-step balancing method developed in [Chesi et al. 2014]. In this case, an obstacle is introduced by the adopted hardware architecture, which uses the embedded electronic board for actuation and telemetry purposes only, whereas an external computer is responsible for running the algorithms and heavy calculations. This architecture is not adequate in this case since the delays introduced by the wireless communication between the external computer and the electronic board highly affects the stability of the controller, as concluded from simulations. As an example, to determine the current control torque magnitude a request would be sent from the external computer to the electronic board, which would respond to this command with the actuator telemetry. However, the received signal would correspond to a past instant, considering the time for the message to be sent and to be received, about 0.1 s at best. Another problem concerns the actuation: when the external computer sends an actuation command, the microcontroller in the electronic board becomes dedicated to moving the stepper motors and does not respond to other requests - such as telemetry from the testbed or the actuators - until the actuator reaches the position indicated by the command. These obstacles indicate that the proper hardware setup for implementing this balancing technique requires a real-time embedded system, similarly to what is done in [Chesi et al. 2014], and it is left for future works.

## 5.1 Methods for evaluating the balancing performance

In order to evaluate the performance of the balancing techniques, there are some experiments that may be made. The first and simpler one, also used in [Da Silva et al. 2016], is to consider the movement of the platform as analogous to a 3D physical pendulum. In this case, the mass of the platform is considered to be concentrated in the Center of Mass position, the pivot is the Center of Rotation and the length of the rod is given by the unbalance vector magnitude. In this scenario, the experiment consists in observing the oscillation period of the pendulum, which, considering the oscillation period of the 2D pendulum, is given by

$$T = 2\pi \sqrt{\frac{J_i}{mg\|\mathbf{r}\|}}, \quad (5.4)$$

in which  $J_i$  is the inertia moment of the selected axis  $i$  of the testbed. Eq. (5.4) indicates that the oscillation period of the platform is inversely proportional to the unbalance vector magnitude, *i.e.* as higher is the oscillation period, better is the balancing. This selection is made arbitrarily, but the experiment must be performed accordingly to the selected axis, *i.e.* when one of the axes is analyzed, then effort is made to make oscillations in this axis only. For example, if the  $X_b$  axis is selected, then  $J_x$  is used in Eq. 5.4 and the testbed must oscillate only around this axis. This is made by giving an initial impulse in the testbed manually. The reason for it is that, as the testbed has freedom to rotate around its three rotational axes, energy may flow from one axis to another, making it difficult to observe the oscillation period graphically and introducing errors when using Eq. 5.4. As can be seen in [Da Silva and Rodrigues 2015], evaluating the oscillation period with this experiment incurs in high order percent errors between the measured period and that estimated from Eq. 5.4. In this work, considering this difficulty, the oscillations periods are estimated by obtaining the frequency spectrum of the attitude signals and extracting the corresponding dominant frequencies.

The second method, shown in [Chesi et al. 2014], consists in evaluating the potential energy of the testbed in time. In this method, an initial impulse is given to the testbed, which introduces mechanical energy to the system. This mechanical energy, neglecting eventual energy losses due to aerodynamic drag and residual friction, is composed by kinetic ( $E_k$ ) and potential ( $E_u$ ) energies. For a rigid body, these energies may be calculated as

$$E_{mech} = E_k + E_u = \frac{1}{2}\boldsymbol{\omega}^T \mathbf{J} \boldsymbol{\omega} + mgh, \quad (5.5)$$

in which  $h$  is the height of the CM while it moves, with relation to an inertial reference system. When the testbed is unbalanced, this height oscillates, as well as the potential energy, consequently. In this way, the mechanical energy flows between its kinetic and potential portions. This can be seen graphically as oscillations in the kinetic energy. If the testbed is balanced, the height  $h$  in Eq. 5.5 becomes almost constant, as well as the potential energy,



as consequence. In other words, one may evaluate the balancing performance by analyzing the reduction in the kinetic energy oscillations: if there is no potential energy oscillating in the system, then the kinetic energy is constant, considering the total mechanical energy has to be constant (neglecting friction and other sources of energy loss).

## 5.2 Analysis

In what follows, the performance of the proposed balancing methods is analyzed. To allow a fair comparison between the different techniques, an effort was made in order to guarantee that all experiments start with the same CM position. However, since the CM position may not be measured, the adopted way was to certify that all balancing masses start in the middle of the corresponding excursions. Since all MMUs move accordingly to the signals sent by the electronic board, it is possible to track the distance moved by each MMU, which allows to send the MMUs to its initial positions with fair precision.

After setting up the testbed in this initial condition, an initial impulse was given to the testbed and a set of measurements were acquired. Fig. 5.1 shows the angular velocities and attitude angles of the testbed while it was swinging.

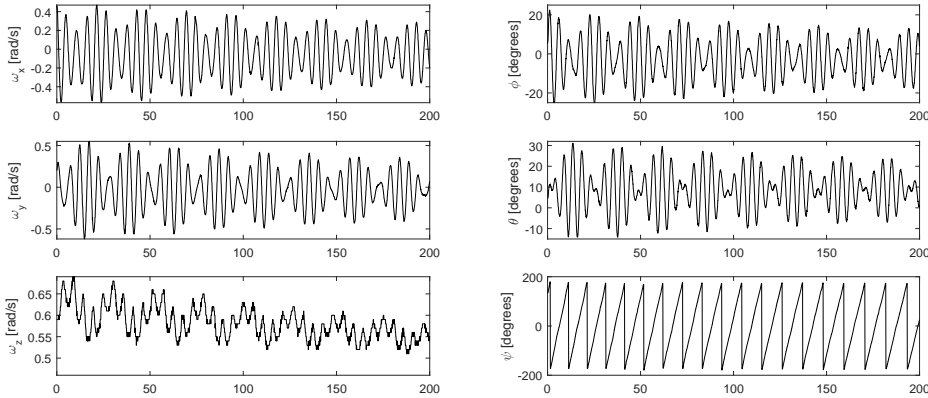


Figure 5.1: Angular velocities and Euler angles of the platform in the unbalanced condition.

From this initial condition and considering this initial data set, all the filters were run and the corresponding unbalance vector estimations may be seen in Figs. 5.2, 5.3, 5.4 and 5.5 for the Kalman Filter, the Extended Kalman Filter (simplified inertia), the Extended Kalman Filter (complete inertia) and the Unscented Kalman Filter, respectively. In these filters, both the  $\mathbf{R}$  and the  $\mathbf{Q}$  matrices were tuned to the same values, which are

$$\mathbf{R} = \text{diag}([0.01^2 \ 0.01^2 \ 0.01^2]) \text{ rad}^2/\text{s}^2 \text{ and} \quad (5.6)$$

$$\mathbf{Q} = \text{diag}([0.00005 \ 0.00005 \ 0.00005 \ 10^{-8} \ 10^{-8} \ 10^{-8}]) [\text{rad}^2/\text{s}^2 \ \text{m}^2], \quad (5.7)$$

whereas the initial value for the state covariance matrix  $\mathbf{P}$  was set as  $\mathbf{P}_0 = \mathbf{0}_{6 \times 1}$ .

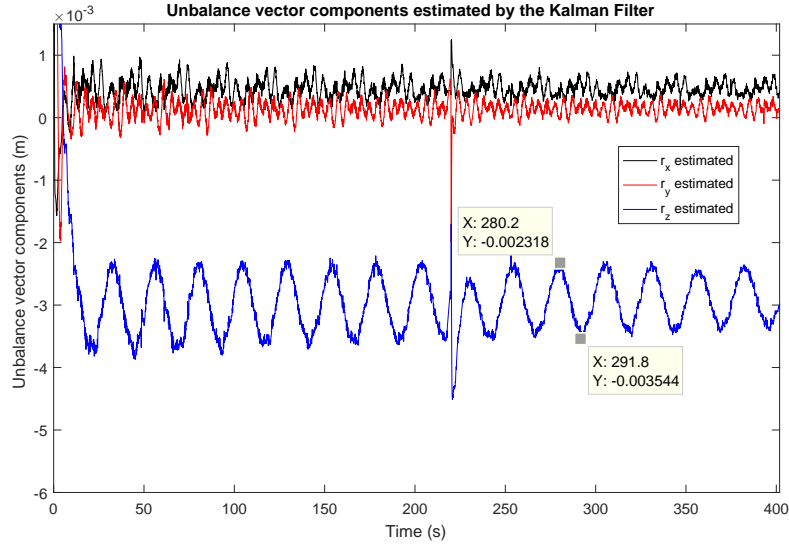


Figure 5.2: Unbalance vector components estimated by the KF.

Table 5.1: Unbalance vector components estimation and corresponding Chi-squared ratings in the initial testbed condition.

Filter	Unbalance components			Chi-squared rate
	$r_x$ (mm)	$r_y$ (mm)	$r_z$ (mm)	
KF	0.4647498458	0.1486030602	-2.9139714047	99.5%
EKF1 <sup>a</sup>	0.4507034361	0.1551217675	-2.6196104581	93.75%
EKF2 <sup>b</sup>	0.3722184772	0.1201874945	-2.5159876292	90.78%
UKF	0.3725541379	0.1202941854	-2.5180617614	99.02%

<sup>a</sup>: Simplified inertia.

<sup>b</sup>: Complete inertia.

The final estimated unbalance vector components are shown in Table 5.1 with the corresponding Chi-squared test ratings in the 95% confidence interval. Since the estimated values may oscillate more or less depending on the selected filter, the strategy of taking the mean value of the final third portion of the graphs was adopted. It is specially relevant when considering, for example, the  $r_z$  component estimation shown in Fig. 5.2 for the Kalman Filter, which oscillates from approximately 2.3 mm to 3.5 mm, a wide range considering the application.

Fig. 5.2 shows remarkable oscillations in the estimation of the  $Z_b$  axis component of the unbalance vector. Referring to Sec. 4.1, one may notice that these oscillations remember the same effect caused by various conditions, such as synchronization errors in the telemetry, high angular velocity conditions or erratic inertia modeling. Considering the testbed

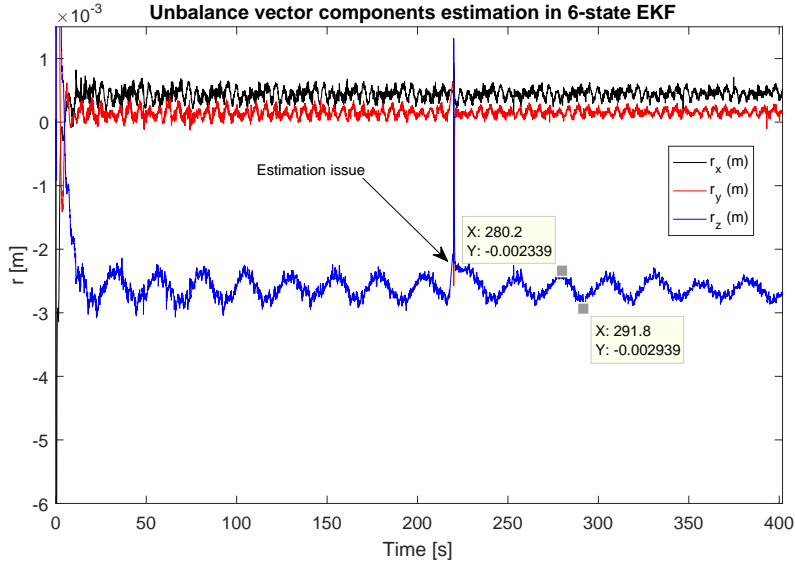


Figure 5.3: Unbalance vector components estimated by the EKF (simplified inertia).

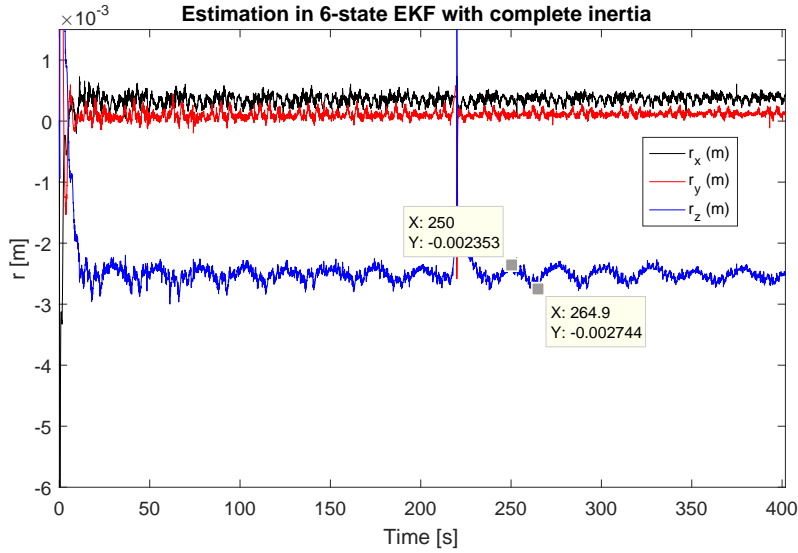


Figure 5.4: Unbalance vector components estimated by the EKF (complete inertia).

parameters were set as (updated before running the experiments)

$$\mathbf{J} = \begin{bmatrix} 0.265 & -0.014 & -0.035 \\ -0.014 & 0.246 & -0.018 \\ -0.035 & -0.018 & 0.427 \end{bmatrix} \text{ kg m}^2 \quad (5.8)$$

$$m = 13.901 \text{ kg} , \quad (5.9)$$

in which the  $\mathbf{J}$  matrix is that obtained from the CAD model of the platform, the most reasonable argument is that these oscillations are caused by erratic inertia modeling, since the KF assumes that the inertia tensor  $\mathbf{J}$  is a diagonal matrix, when it is not. In fact, estimating the inertia tensor matrix through CAD modeling may incur in considerable errors, as it depends on the skills of the CAD modeler for reproducing the platform model accurately. These os-

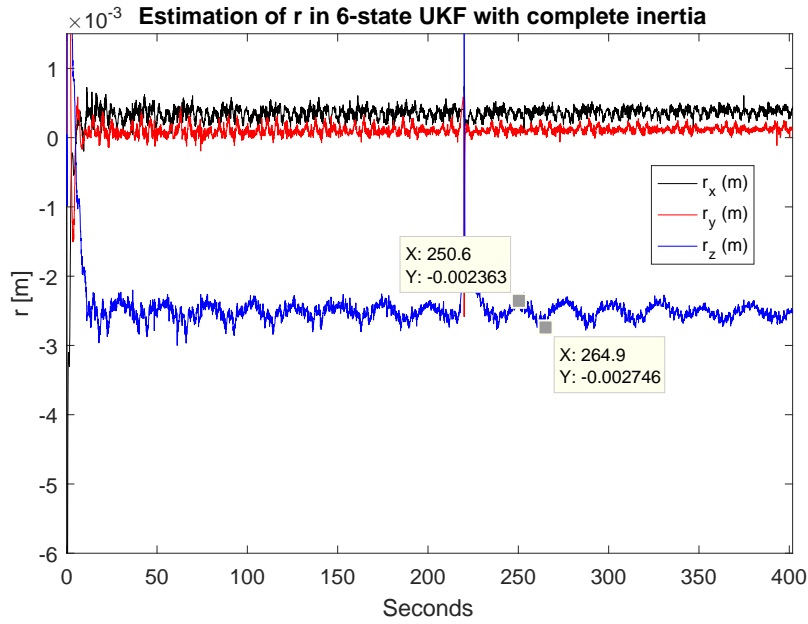


Figure 5.5: Unbalance vector components estimated by the UKF.

cillations in the parameter estimation may also be seen in the graphs corresponding to the EKF and the UKF. However, it may be observed that these oscillations occur in a narrower range for the EKF of Fig. 5.3, even considering this filter does not model the products of inertia  $J_{ij}$ ,  $i \neq j$ , as well. The narrowest oscillation range is obtained from the EKF (complete inertia) and the UKF, shown in Figs. 5.4 and 5.5. It is coherent with the fact that both these two filters model the inertia tensor with all its 9 elements.

Another interesting remark concerns the estimation issue pointed in Fig. 5.3 and present in Figs. 5.2 to 5.5. This issue was artificially made by interrupting the wireless communication during the estimation process and shows that the filters have high sensibility regarding this error source. It indicates that unsynchronization in the telemetry may not be the cause of the oscillations seen in these graphs, since it would represent gross estimation errors instead of the observed oscillations.

After this initial estimation, the values obtained for the unbalance vector were used for simulating the testbed model, for model validation purposes. Since it is difficult to determine which of the unbalance vector estimations shown in Table 5.1 is the closest to the real unbalance vector, the values obtained for the UKF in Table 5.1 are used. Fig. 5.6 shows the simulated and the measured angular velocities of the testbed.

From Fig. 5.6, it can be seen that the adopted model provides a reasonable fit to the measured data, neglecting the phase difference between the real and measured signals and considering the similarity between the shapes of the graphs. Also, the amplitudes of the graphs are similar. However, as one may notice, in all the three axes the oscillations appear to have different periods when comparing the simulated data with the measured data. It indicates that the unbalance vector estimated in the filters presents a difference with relation to the real value, as the oscillation period is closely related to the magnitude of the unbalance

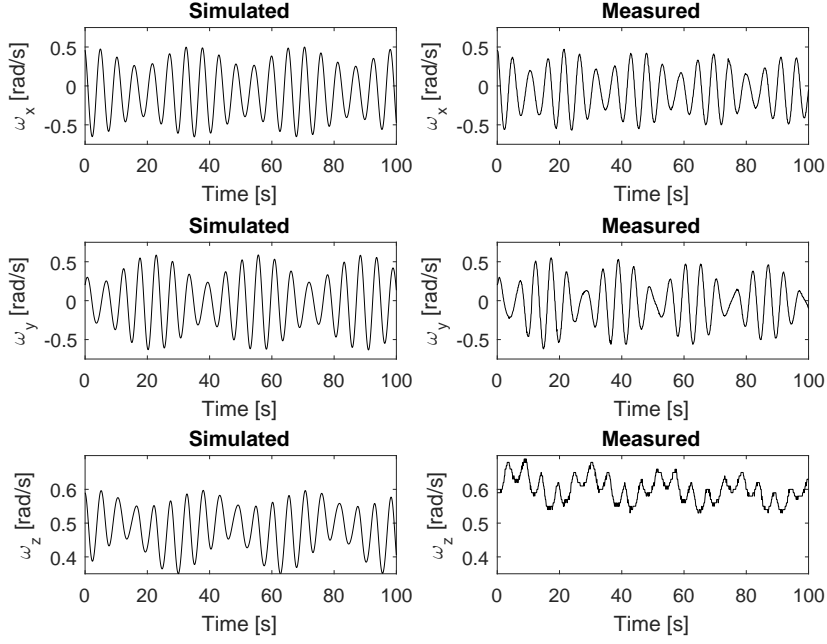


Figure 5.6: Model validation using the  $r$  vector estimated by the UKF.

vector (as discussed previously, in the 3D pendulum analogy). Also, it may be noticed a difference in the magnitude of the simulated and measured angular velocities in the  $Z_b$  axis. It indicates that the inertia terms related to the  $Z_b$  axis - the  $J_z$ ,  $J_{xz}$  and  $J_{yz}$  terms - may present some difference with relation to the real ones. In other words, the current analysis indicates that the proper functioning of the described filters is highly dependent on a good estimation of the inertia tensor.

An additional analysis concerns the frequency spectrum of the roll and pitch signals obtained from the measured data and the corresponding simulated model. These spectra are shown in Fig. 5.7. As can be seen in Fig. 5.7, the dominant frequencies in both the simulated and measured graphs are almost the same, which indicates that, despite the differences previously pointed, the adopted model provides reasonably similar results. Moreover, the measured angular velocities of the platform during the initial position experiment may be used to calculate the kinetic energy of the system  $E_K$  accordingly to Eq. (5.5). The upper portion of Fig. 5.8 shows the kinetic energy signal, whereas the middle portion shows only the oscillating part of the kinetic energy signal. The lower portion of Fig. 5.8 shows the amplitude of the oscillating part and is used as a unbalance performance index: the unbalance performance becomes as better as smaller is the overall amplitude. In this case, the maximum amplitude is around  $0.09 J$ . Unfortunately, this performance index must be analyzed carefully, since this amplitude does not depend only on the unbalance vector magnitude, but also on the initial attitude angles and initial angular velocities imposed on the platform.

Before testing the estimations shown in Table 5.1, a manual balance is performed in order to acquire a notion of the real unbalance vector. Starting from the same initial con-

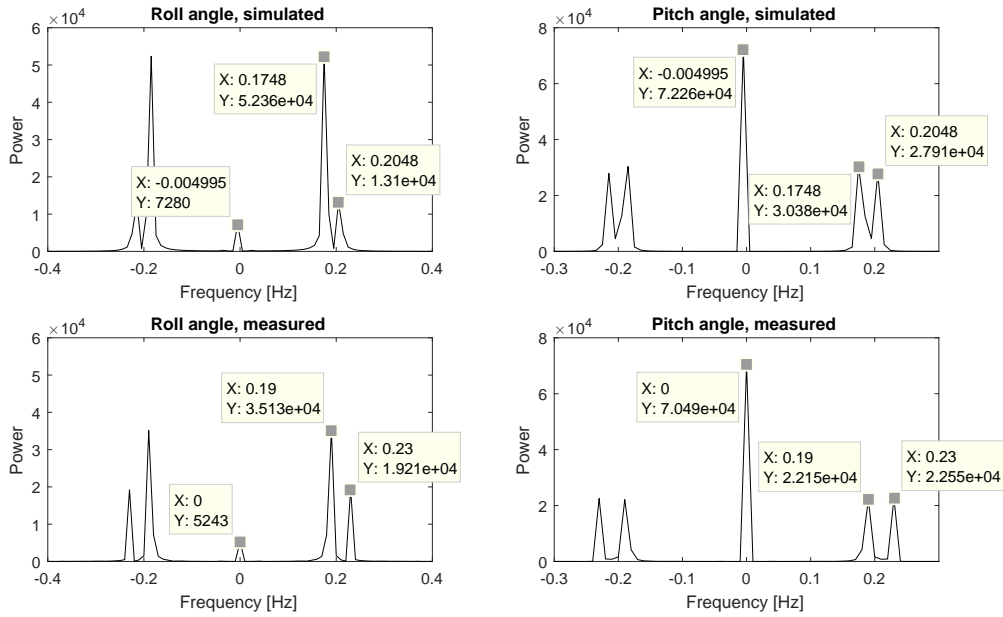


Figure 5.7: Spectrum of the roll and pitch signals during the initial condition experiment.

dition of the previous experiments, the procedure consists in applying reverse engineering on Eq. (4.12): by moving the MMUs until the platform reaches a reasonable balancing performance and taking note of the MMUs position, the unbalance vector may be estimated through Eq. (4.12), given both the platform mass  $m$  and the MMUs masses  $m_{\text{MMU}}$  are known. After this procedure, which took about an hour, the variations in the positions of the MMUs were obtained as  $\delta_m \approx [7 \ 4 \ 51]^T \text{ mm}$ , which, after multiplying by the ratio between  $m_{\text{MMU}}$  (0.78 kg) and  $m$  (13.901 kg), gives  $\mathbf{r} \approx [-0.39 \ -0.22 \ -2.86]^T \text{ mm}$ . Fig. 5.9 shows the measuring scales used for estimating the MMUs positions, as well as the position of the balancing weight.

The frequency spectrum of the roll angle and the energy graphs for this manual balancing are shown in Fig. 5.10 and indicates an oscillation period of about 30 s and energy amplitude of about 0.003 J at maximum.

After the analysis made so far, the unbalance vector estimations shown in Table 5.1 were used to move the MMUs in order to nullify the unbalance vector. The adopted methodology was to use Eq. (4.12) to determine how much each MMU has to move, followed by the corresponding actuation. Then, a data set is collected in order to evaluate the balancing performance and the MMUs are placed in the initial condition again. Using the collected data set, the frequency spectrum and the amplitude of the energy oscillation are determined, similarly to the experiments previously made. Since both the EKF2 and UKF filters in Table 5.1 show similar estimations of the unbalance vector, they are analyzed first. After performing the required MMUs movements for nullifying the unbalance vector, a data set was collected and a new oscillation period of about 10 s was obtained. Also, a maximum energy oscillation of 0.2 J was obtained - a reduction of almost 80% compared to the initial condition in Fig. 5.8. When using this data set in the EKF2 or UKF for estimating the unbalance vector again, the

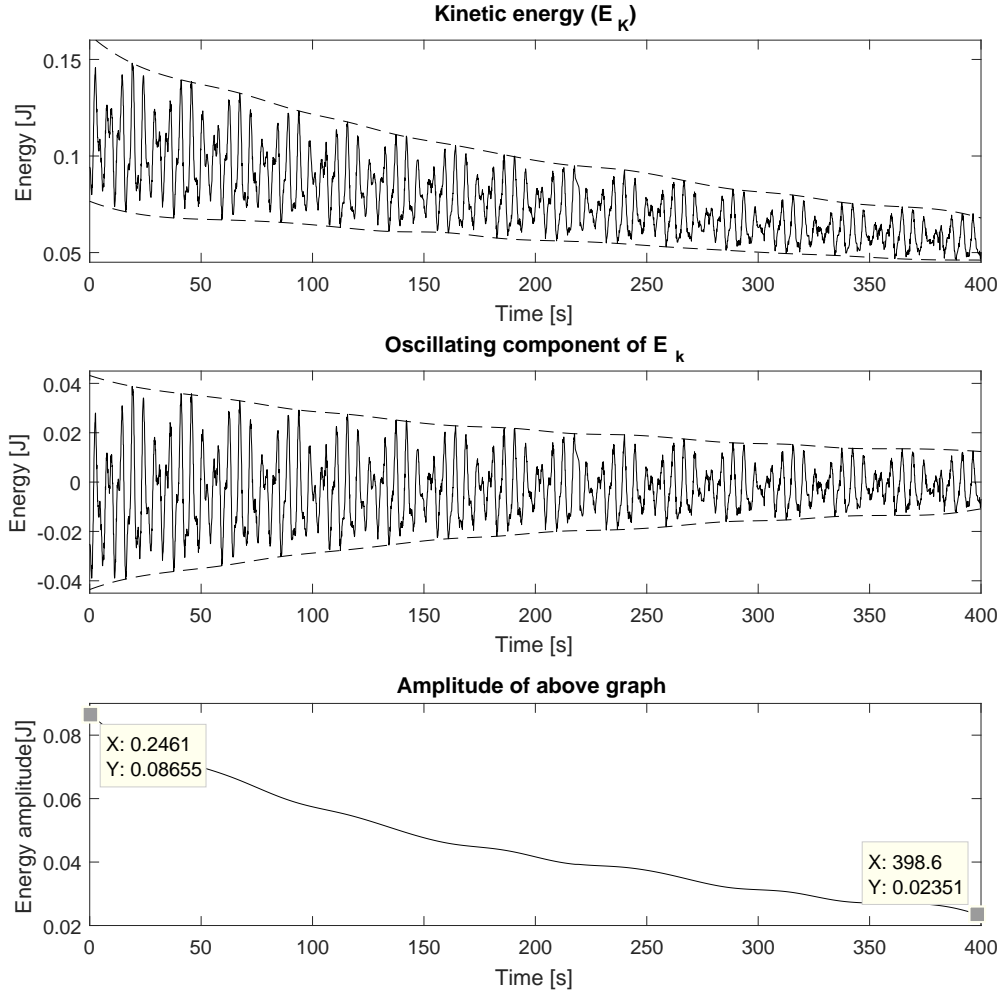


Figure 5.8: Energy analysis of the initial condition of the platform.

new obtained unbalance vector was  $\mathbf{r} \approx [-0.074 \ 0.082 \ -0.332]^T \text{ mm}$ , distant from the null value expected. The EKF2 presented a slightly better performance, giving 12.5 s for the oscillation period. Moreover, the performance of the balancing using the unbalance vector estimated by the KF could not be evaluated, since the testbed reached an inverted pendulum configuration after moving the MMUs. These results indicates that the filters present limited performance which may be explained by low accuracy on the assumed model parameters, such as the inertia tensor  $\mathbf{J}$  and the platform mass  $m$ .

A final analysis can be made concerning the energy losses in the system. As can be seen in Fig. 5.1, the angular velocities decay in time. This effect, which is a consequence of the low - but existent - friction in the air bearing, was not modeled in the filters. By analyzing Figs. 5.2, 5.3, 5.3 and 5.5, it appears that this effect does not interfere with the estimation of the unbalance vectors components, neither creating a biased estimation nor introducing undesired oscillations. In either case, this effect is estimated as follows. Fig. 5.11 shows the envelopes of the angular velocity  $\omega$  components, whereas the upper bounds are used in

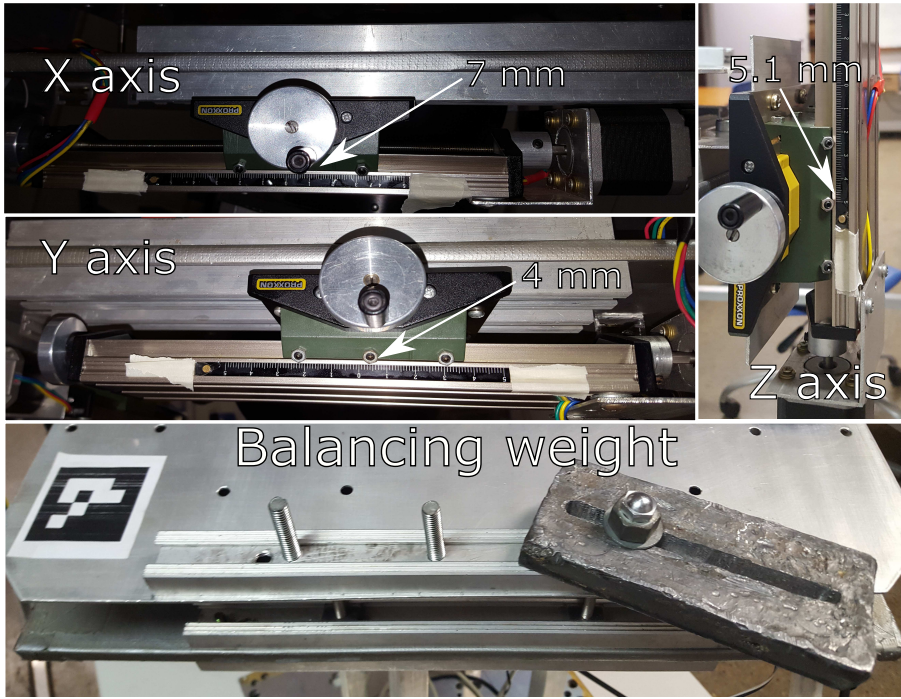


Figure 5.9: Final MMUs and balancing weight positions for the manual balancing.

Table 5.2: Curve fitting parameters for  $\omega_x$ ,  $\omega_y$  and  $\omega_z$ .

Curve	$a$ [rad/s] <sup>a</sup>	$\tau$ [s] <sup>a</sup>
$\omega_x$	0.4856 (0.485, 0.4862)	362.3188 (361.4022, 363.1082)
$\omega_y$	0.5638 (0.5631, 0.5646)	407.8303 (406.6694, 408.8307)
$\omega_z$	0.6879 (0.6875, 0.6883)	1316.1 (1312.2, 1320.0)

<sup>a</sup>: The values of  $a$  and  $\tau$  are accompanied by the corresponding 95% confidence upper and lower bounds in parenthesis.

Fig. 5.12 for the curve fitting of an exponential model of the form

$$f(t) = a \cdot e^{-\frac{t}{\tau}}, \quad (5.10)$$

in which  $a$  represents the initial value of the exponential and  $\tau$  its time constant or mean lifetime. The curve fitting parameters obtained from these curves are shown in Table 5.2.

As can be seen in Table 5.2, the obtained values for  $\tau$  indicates that it delays at least approximately 350 s for any angular velocity to decay about 63% of its initial value. Considering all the tested filters - KF, EKF or UKF - have a convergence for the unbalance vector components much lower than this time - usually less than 50 s -, it can be concluded that, unless the system energy is fully dissipated, the friction effect does not interfere with the unbalance vector estimation at all.



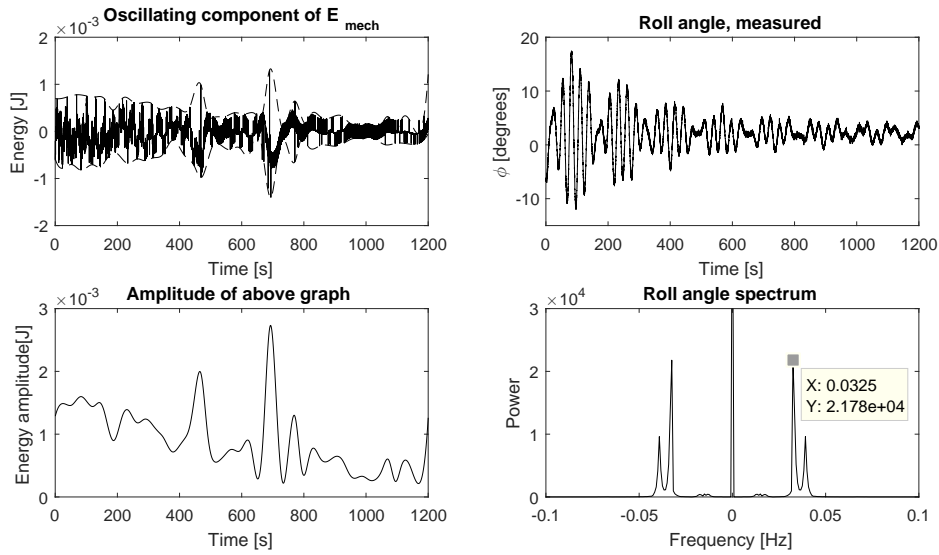


Figure 5.10: Performance of the manual balancing.

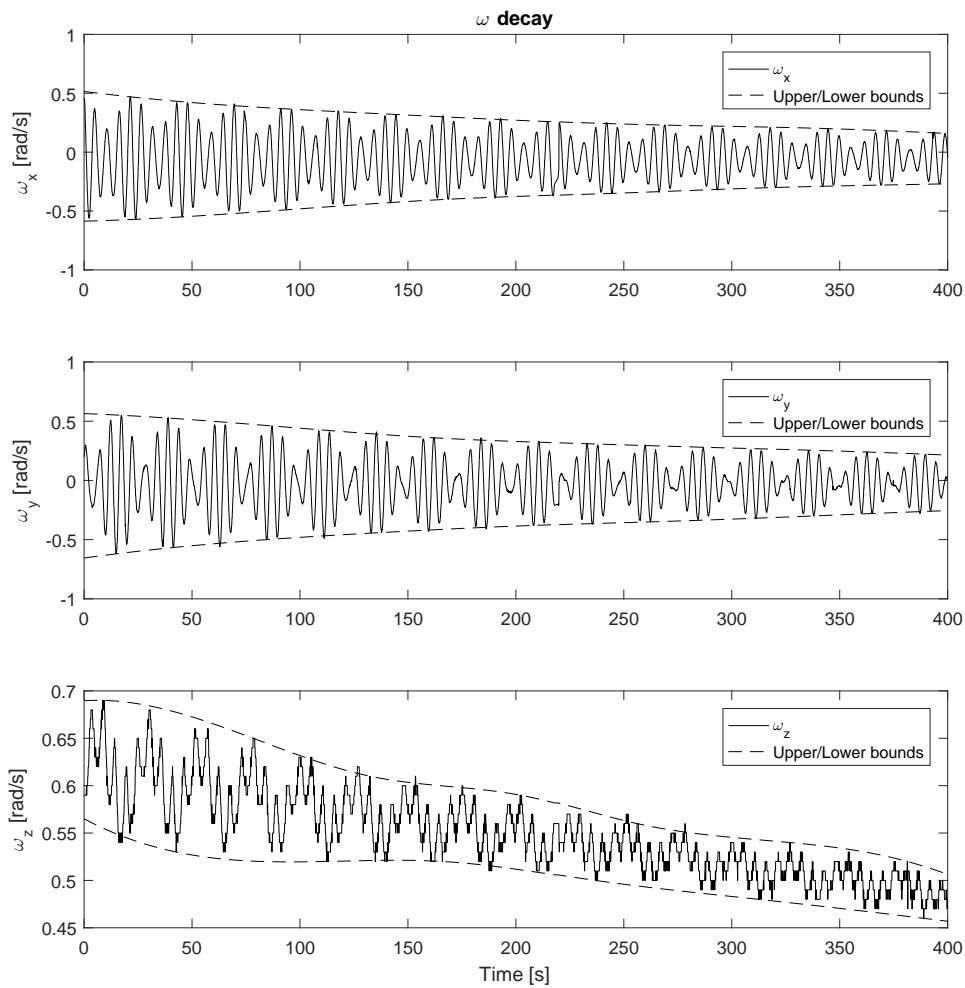


Figure 5.11: Envelopes of the angular velocity signals during the initial experiment.

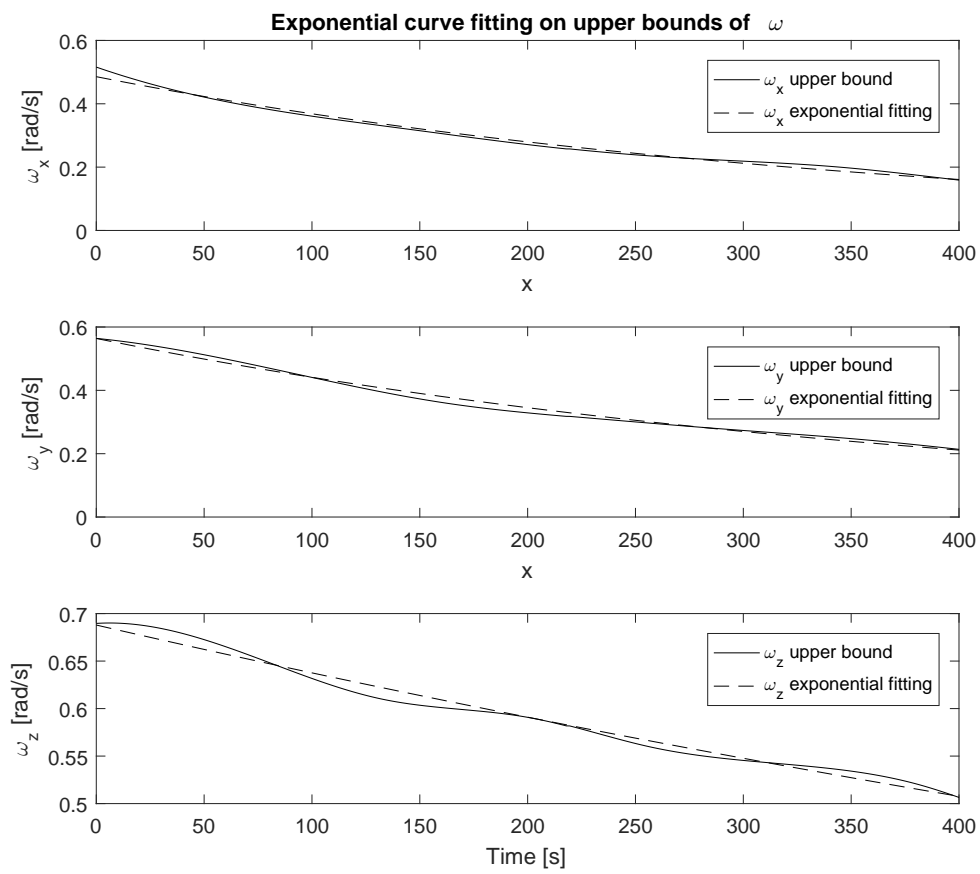


Figure 5.12: Exponential curve fittings of the upper bounds of Fig. 5.11.

# Chapter 6

## Conclusion

This work showed various methods for solving a remarkable problem in attitude simulators for spacecrafts: the offset between the CM and the CR. As explained, this problem is responsible for limiting the capacity of this kind of simulator of performing attitude control experiments, specially when the goal is to control the roll and pitch angles of the platform. The reduction of this offset - named throughout the text as unbalance vector - may allow even low torque actuators to perform attitude control properly. In others words, even the low torque actuators present in nanosatellites such as CubeSats may be tested in this kind of simulator if the adopted balancing method has adequate performance.

In this work, the Kalman filter and its most popular nonlinear versions - the EKF and the UKF - were tested with various filter designs and simulation conditions. It was shown that these filters may provide reasonably good estimations of the unbalance vector. Also, the method developed in [Chesi et al. 2014] was presented along with simulation results. In all these cases the filters consistency was guaranteed and the observability of the states was analyzed.

On the other hand, when experimentally testing the proposed filters some obstacles arise. The current hardware setup available at the LAICA is not capable of running the filters which entirely estimates the inertia tensor, for which reason all the tested filters consider the inertia tensor as a previously known parameter. In fact, this parameter may be obtained by CAD models of the platform. However, this approach turned out to be imprecise, since the unbalance vector estimations presented undesirable oscillations and were biased from the real value. Possibly, this obstacle may be avoided by developing a detailed and accurate CAD model of the platform.

Better results may be obtained by implementing the balancing method developed in [Chesi et al. 2014], but special attention must be given to the second phase of this balancing method, since it also assumes the knowledge of the inertia tensor. A more efficient approach may be upgrading this method for using a 12-state UKF instead of the 4-state UKF originally adopted in this work, since estimation errors concerning erratic inertia tensor com-

ponents would not be introduced. Moreover, after upgrading the actuators available in the LAICA, it will be possible to test the method developed by [Kim and Agrawal 2009], which, differently from the Hybrid Adaptive Control method, is a one-step method, since it may generate torques in all three directions.

## 6.1 Future perspectives

The project being developed at LAICA involves a wide variety of research topics. As could be seen throughout this work, one of these possibilities is related with attitude determination. The precision of the attitude determination method interferes in all supervenient aspects of the project, such as attitude controlling and is specially critical when implementing advanced balancing methods.

Considering attitude determination, there are many improvements that may be made, such as improving the calibration, the precision and even the hardware of the attitude determination methods based on computer vision, as well as implementing various attitude determination filters in the embedded systems to be tested on the table, including the model of the sensors in these filters. Also, the calibration procedures adopted may be further improved, specially when considering the accelerometers and gyroscopes, that were not calibrated except for the static bias offset, which is known to change dynamically.

Concerning the balancing methods, possible improvements include developing procedures for determining the testbed characteristics - such as its inertia tensor - accurately, *e.g.* developing a minimallistic CAD model for the testbed. The augmented versions of the filters may also be implemented for estimating the inertia tensor components. Moreover, the balancing techniques described in [Chesi et al. 2014] and in [Kim and Agrawal 2009] may be implemented, involving hardware advances such as developing a real-time embedded system for the platform and actuators with higher torque capacity.

Minor improvements include adding endstop sensors for preventing eventual damage in the stepper motors of the MMUs when they reach the end of its excursion. Also, another model of MMU may be developed, decreasing the mass displaced by the MMU and allowing even more fine adjustment of the CM position.

The development of the platform will also provide the necessary experimental apparatus for designing and implementing algorithms for attitude control.

# BIBLIOGRAPHICAL REFERENCES

- [Al-Mohy and Higham 2010] Al-Mohy, A. H. and Higham, N. J. (2010). The complex step approximation to the fréchet derivative of a matrix function. *Numerical Algorithms*, 53(1):133.
- [Bar-Itzhack 2000] Bar-Itzhack, I. Y. (2000). New method for extracting the quaternion from a rotation matrix. *Journal of guidance, control, and dynamics*, 23(6):1085–1087.
- [Bar-Itzhack and Harman 1997] Bar-Itzhack, I. Y. and Harman, R. R. (1997). Optimized triad algorithm for attitude determination. *Journal of guidance, control, and dynamics*, 20(1):208–211.
- [Bar-Shalom et al. 2004] Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2004). *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons.
- [Cao 2018] Cao, Y. (2018). Mathworks. file exchange. complex step jacobian. <https://www.mathworks.com/matlabcentral/fileexchange/18176-complex-step-jacobian>. Accessed: 2018-06-23.
- [Carrara and Milani 2007] Carrara, V. and Milani, P. G. (2007). Controle de uma mesa de mancal a ar de um eixo equipada com giroscópio e roda de reação. *V SBEIN-Simpósio Brasileiro de Engenharia Inercial. Rio de Janeiro*, pages 26–29.
- [Chesi 2015] Chesi, S. (2015). *Attitude control of nanosatellites using shifting masses*. University of California, Santa Cruz.
- [Chesi et al. 2014] Chesi, S., Gong, Q., Pellegrini, V., Cristi, R., and Romano, M. (2014). Automatic mass balancing of a spacecraft three-axis simulator: Analysis and experimentation. *Journal of Guidance, Control, and Dynamics*.
- [Da Silva and Rodrigues 2015] Da Silva, R. and Rodrigues, U. (2015). Simulador de sistema de determinação e controle de atitude de pequenos satélites. Technical report, University of Brasília.
- [Da Silva et al. 2016] Da Silva, R., Rodrigues, U., Borges, R., Sampaio, M., Beghelli, P., da Costa, S., Popov, B., Battistini, S., and Cappelletti, C. (2016). A testbed for attitude

- and determination control of spacecrafts. In *II Latin American IAA CubeSat workshop, Florianopolis, Brazil*, volume 18.
- [de Loiola et al. 2017] de Loiola, J. V. L., da Silva, L. M. B., Battistini, S., Cappelletti, C., and Borges, R. A. (2017). Development of a hardware-in-the-loop test platform for nanosatellites adcs integrated with an ukf. In *4th IAA Conference on University Satellite Missions and CubeSat Workshop*.
- [de Loiola et al. 2018] de Loiola, J. V. L., van der Ploeg, L. C., da Silva, R. C., aes, F. C. G., Borges, R. A., Borges, G. A., Battistini, S., and Cappelletti, C. (2018). 3 axis simulator of the earth magnetic field. In *2018 IEEE Aerospace Conference*, pages 1–8.
- [de Oliveira et al. 2013] de Oliveira, A. M., Kuga, H. K., and Carrara, V. (2013). Estimating the mass characteristics of a dumbbell air bearing satellite simulator. In *22nd International Congress of Mechanical Engineering, Ribeirao Preto, Brazil*, pages 3–7.
- [de Oliveira et al. 2015] de Oliveira, A. M., Kuga, H. K., and Carrara, V. (2015). Air bearing platforms for simulation of spacecraft attitude control systems. In *International Symposium on Dynamic Problems of Mechanics*.
- [Foster and Elkaim 2008] Foster, C. C. and Elkaim, G. H. (2008). Extension of a two-step calibration methodology to include nonorthogonal sensor axes. *IEEE Transactions on Aerospace and Electronic Systems*, 44(3).
- [Garrido-Jurado et al. 2014] Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F., and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292.
- [Garrido-Jurado et al. 2016] Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F., and Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481 – 491.
- [Gonzales 2009] Gonzales, R. G. (2009). *Utilização dos métodos SDRE e Filtro de Kalman para o controle de atitude de simuladores de satélites*. PhD thesis, Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espacial, DEM/INPE, São Jose dos Campos, SP.
- [Greenwood 1965] Greenwood, D. T. (1965). *Principles of dynamics*. Prentice Hall.
- [Guimarães et al. 2017] Guimarães, F. C., da Silva, R. C., ao Victor Lopes de Loiola, J., Borges, G. A., Borges, R. A., Battistini, S., and Cappelletti, C. (2017). Aplicação do filtro de kalman para a determinação de atitude de plataforma de testes de pequenos satélites. In *XIII Simpósio Brasileiro de Automação Inteligente*, pages 1784–1789.
- [Haddad and Chellaboina 2011] Haddad, W. M. and Chellaboina, V. (2011). *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton University Press.

- [Haykin 1986] Haykin, S. (1986). *Adaptive filter theory*. Prentice-Hall, Inc.
- [Hibbeler 2010] Hibbeler, R. C. (2010). *Engineering mechanics Dynamics*. Pearson education.
- [Hughes 2012] Hughes, P. C. (2012). *Spacecraft attitude dynamics*. Courier Corporation.
- [Ishioka et al. 2017] Ishioka, I. S. K., Battistini, S., Cappelletti, C., and Borges, R. A. (2017). Design and development of an active magnetic actuator for attitude control system of nanosatellites. In *4th IAA Conference on University Satellite Missions and CubeSat Workshop*.
- [Julier and Uhlmann 2004] Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.
- [Kang and Barbot 2007] Kang, W. and Barbot, J.-P. (2007). Discussions on observability and invertibility. In *NOLCOS*.
- [Kim and Agrawal 2009] Kim, J. J. and Agrawal, B. N. (2009). Automatic mass balancing of air-bearing-based three-axis rotational spacecraft simulator. *Journal of Guidance, Control, and Dynamics*, 32(3):1005–1017.
- [Krishnanunni et al. 2018] Krishnanunni, A. R., Jayadevan, S., Mony, A., and Sharma, G. (2018). Inertia and center of mass estimation of a 3 dof air bearing platform. In *Proceedings of the Third IFAC International Conference on Advances in Control and Optimization of Dynamical Systems (ACODS 2018)*.
- [Kuipers et al. 1999] Kuipers, J. B. et al. (1999). *Quaternions and rotation sequences*, volume 66. Princeton university press Princeton.
- [Markley and Crassidis 2014] Markley, F. L. and Crassidis, J. L. (2014). *Fundamentals of spacecraft attitude determination and control*, volume 33. Springer.
- [Martins et al. 2001] Martins, J., Sturdza, P., and Alonso, J. J. (2001). The connection between the complex-step derivative approximation and algorithmic differentiation. *AIAA paper*, 921:2001.
- [Martins et al. 2003] Martins, J. R., Sturdza, P., and Alonso, J. J. (2003). The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262.
- [Mittelsteadt and Mehiel 2007] Mittelsteadt, C. and Mehiel, E. (2007). Cal poly spacecraft attitude dynamics simulator. *AIAA Paper*, 6443:20–23.
- [Oceanic and Administration 2018] Oceanic, N. and Administration, A. (2018). NOAA magnetic field calculators. <https://www.ngdc.noaa.gov/geomag-web/#igrfwmm>. Accessed: 2018-05-03.

- [Peck et al. 2003] Peck, M. A., Miller, L., Cavender, A. R., Gonzalez, M., and Hintz, T. (2003). An airbearing-based testbed for momentum control systems and spacecraft line of sight. *Advances in the Astronautical Sciences*, 114:427–446.
- [Piaggio 1943] Piaggio, H. (1943). The significance and development of hamilton’s quaternions. *Nature*, 152(3863):553.
- [Romano and Agrawal 2003] Romano, M. and Agrawal, B. N. (2003). Acquisition, tracking and pointing control of the bifocal relay mirror spacecraft. *Acta Astronautica*, 53(4):509–519.
- [Romero-Ramirez et al. 2018] Romero-Ramirez, F. J., Muñoz-Salinas, R., and Medina-Carnicer, R. (2018). Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38 – 47.
- [Schwartz et al. 2003] Schwartz, J. L., Peck, M. A., and Hall, C. D. (2003). Historical review of air-bearing spacecraft simulators. *Journal of Guidance, Control, and Dynamics*.
- [Semiconductors 2018] Semiconductors, N. (2018). NXP implementing a tilt-compensated ecompass using accelerometer and magnetometer sensors. [https://cache.freescale.com/files/sensors/doc/app\\_note/AN4248.pdf](https://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf). Accessed: 2018-05-03.
- [Sharifi et al. 2017] Sharifi, G., Mirshams, M., and Shahmohamadi Ousaloo, H. (2017). Mass properties identification and automatic mass balancing system for satellite attitude dynamics simulator. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, page 0954410017742932.
- [Shen et al. 2004] Shen, J., Sanyal, A. K., Chaturvedi, N. A., Bernstein, D., and McClamroch, H. (2004). Dynamics and control of a 3d pendulum. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 323–328. IEEE.
- [Shuster 1993] Shuster, M. D. (1993). A survey of attitude representations. *Navigation*, 8(9):439–517.
- [Silva et al. 2017] Silva, R., Battistini, S., Borges, R., and Cappelletti, C. (2017). Center of mass compensation of a nanosatellite testbed based on the extended kalman filter. In *4th IAA Conference on University Satellite Missions and CubeSat Workshop*, pages 595–606.
- [Silva et al. 2018] Silva, R. C., Guimarães, F. C., Loiola, J. V. L., Borges, R. A., Battistini, S., and Cappelletti, C. (2018). Tabletop testbed for attitude determination and control of nanosatellites. *Journal of Aerospace Engineering*. Accepted for publication.
- [Simon 2006] Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.



- [Sommer et al. 2018] Sommer, H., Gilitschenski, I., Bloesch, M., Weiss, S. M., Siegwart, R., and Nieto, J. (2018). Why and how to avoid the flipped quaternion multiplication. *arXiv preprint arXiv:1801.07478*.
- [Spong et al. 2006] Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot modeling and control*, volume 3. Wiley New York.
- [STMicroelectronics 2018] STMicroelectronics (2018). Using lsm303dlh for a tilt compensated electronic compass. [www.st.com/resource/en/application\\_note/cd00269797.pdf](http://www.st.com/resource/en/application_note/cd00269797.pdf). Accessed: 2018-06-21.
- [van der Ploeg 2017] van der Ploeg, L. C. (2017). Desenvolvimento de sistema para simulação do campo magnético terrestre em órbitas baixas. Technical report, University of Brasília.
- [Wertz 2012] Wertz, J. R. (2012). *Spacecraft attitude determination and control*, volume 73. Springer Science & Business Media.
- [Wie 1998] Wie, B. (1998). *Space vehicle dynamics and control*. Aiaa.
- [Wittenburg 2013] Wittenburg, J. (2013). *Dynamics of systems of rigid bodies*, volume 33. Springer-Verlag.
- [Xu et al. 2016] Xu, Z., Chen, Y., Qi, N., Sun, Q., Fan, Y., and Wang, C. (2016). Inertia parameters optimisation method for three-axis spacecraft simulator. *Electronics Letters*, 52(20):1675–1677.
- [Xu et al. 2015] Xu, Z., Qi, N., and Chen, Y. (2015). Parameter estimation of a three-axis spacecraft simulator using recursive least-squares approach with tracking differentiator and extended kalman filter. *Acta Astronautica*, 117:254–262.
- [Young 1998] Young, J. S. (1998). *Development of an automatic balancing system for a small satellite attitude control simulator*. PhD thesis, Utah State University. Department of Mechanical and Aerospace Engineering.

# **Appendices**

# Appendix A

## Electronic board design and blueprint

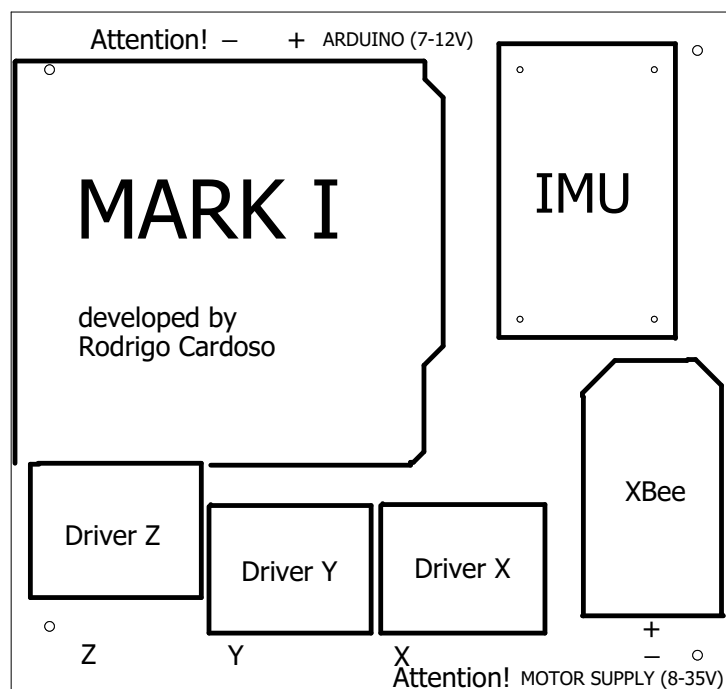


Figure A.1: Upper layer of the electronic board (labels only).

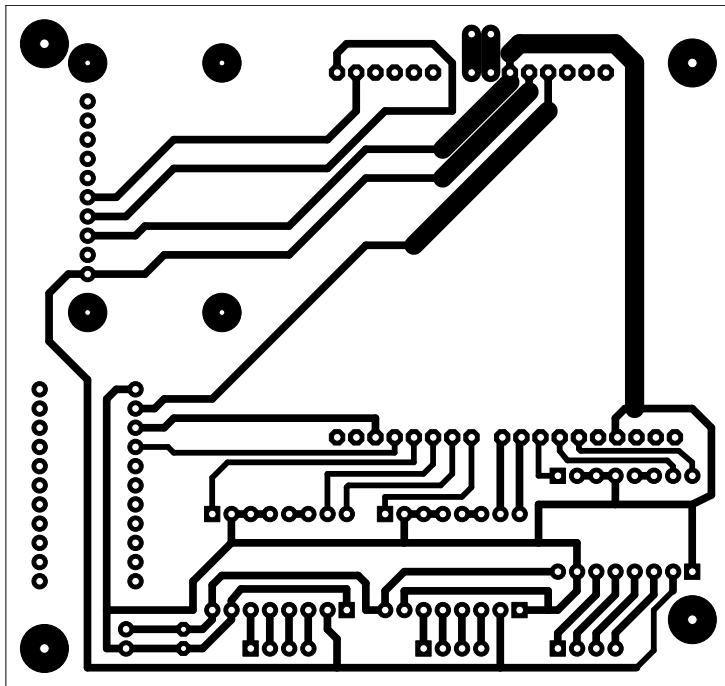


Figure A.2: Bottom layer of the electronic board (copper trails).

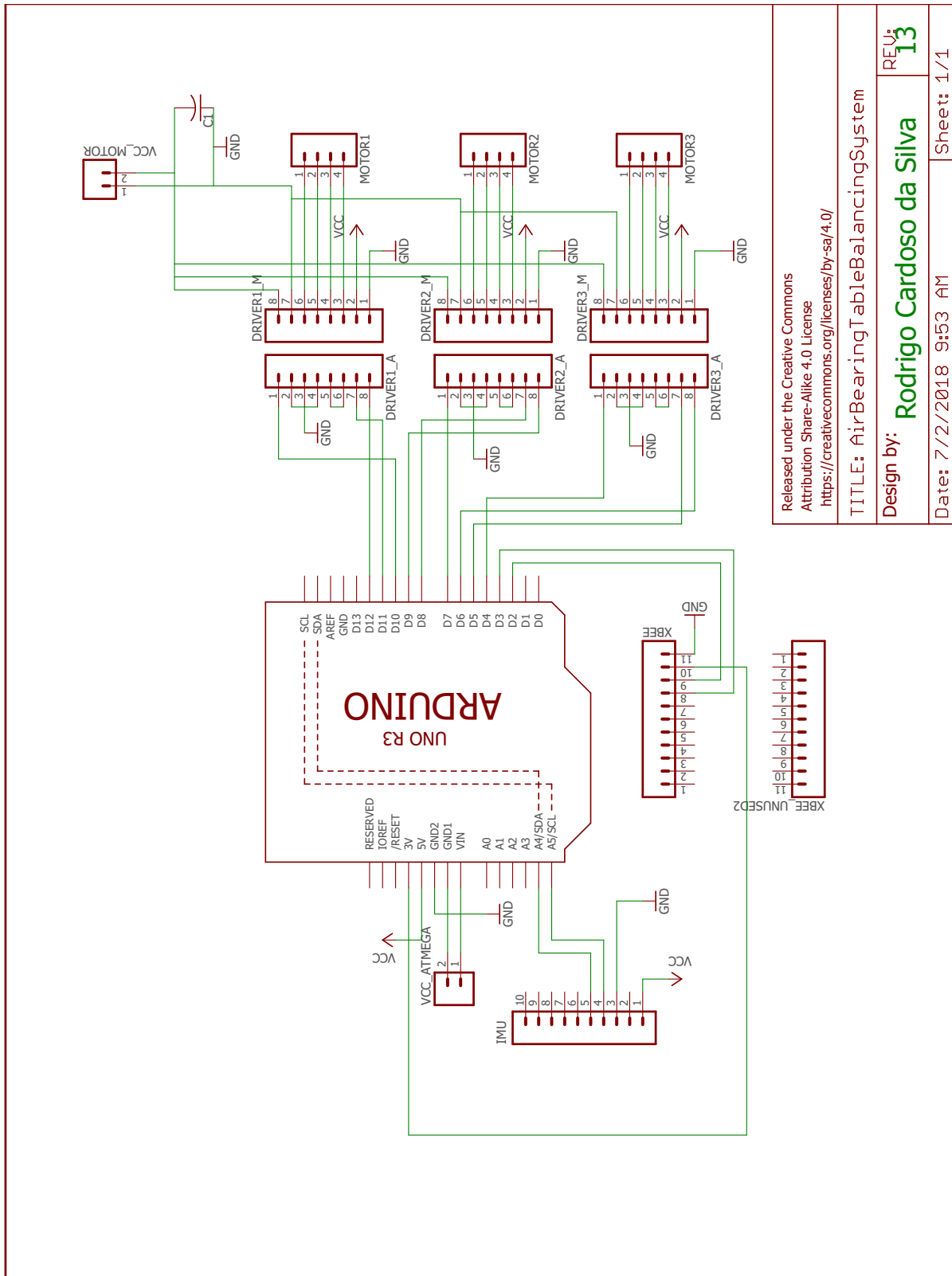


Figure A.3: Electronic board schematics.

# Appendix B

## Euler rates in the ZYX sequence

The Euler rates are the derivatives of the Euler angles and is dependent on the adopted rotation sequence. Based on the procedure shown in [Wie 1998, p. 324] and with the notation used in this work, the body angular velocity vector  $\boldsymbol{\omega}$  may be represented as

$$\boldsymbol{\omega}_i^{i1} = \dot{\psi} \mathbf{i}_z = \dot{\psi} (\mathbf{i1})_z \text{ since } \mathbf{i}_z \text{ equals to } (\mathbf{i1})_z, \quad (\text{B.1})$$

$$\boldsymbol{\omega}_{i1}^{i2} = \dot{\theta} (\mathbf{i1})_y = \dot{\theta} (\mathbf{i2})_y \text{ since } (\mathbf{i1})_y \text{ equals to } (\mathbf{i2})_y \text{ and} \quad (\text{B.2})$$

$$\boldsymbol{\omega}_{(i2)}^b = \dot{\phi} (\mathbf{i2})_x = \dot{\phi} \mathbf{b}_x \text{ since } (\mathbf{i2})_x \text{ equals to } \mathbf{b}_x, \quad (\text{B.3})$$

in which  $\{(\mathbf{i1})_x, (\mathbf{i1})_y, (\mathbf{i1})_z\}$  and  $\{(\mathbf{i2})_x, (\mathbf{i2})_y, (\mathbf{i2})_z\}$  are the versors of the  $\mathbf{i1}$  and  $\mathbf{i2}$  intermediate frames between the inertial  $\mathbf{i}$  and the body  $\mathbf{b}$  frames, respectively, and the  $\boldsymbol{\omega}_x^y$  notation is used to indicate the angular velocity between frames  $x$  and  $y$ . The resulting angular velocity equals to the sum of these three terms, following that

$$\boldsymbol{\omega} = \boldsymbol{\omega}_i^b = \boldsymbol{\omega}_i^{i1} + \boldsymbol{\omega}_{i1}^{i2} + \boldsymbol{\omega}_{(i2)}^b \quad (\text{B.4})$$

$$= \dot{\phi} \mathbf{b}_x + \dot{\theta} (\mathbf{i2})_y + \dot{\psi} (\mathbf{i1})_z, \quad (\text{B.5})$$

which can be rewritten, using the rotation operators  $\mathbf{R}_{x,\phi}$  and  $\mathbf{R}_{y,\theta}$ , as

$$\boldsymbol{\omega}_i^b = \begin{bmatrix} \dot{\phi} & 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{b}_x \\ \mathbf{b}_y \\ \mathbf{b}_z \end{bmatrix}}_{\mathbf{b}} + \begin{bmatrix} 0 & \dot{\theta} & 0 \end{bmatrix} \underbrace{\begin{bmatrix} (\mathbf{i2})_x \\ (\mathbf{i2})_y \\ (\mathbf{i2})_z \end{bmatrix}}_{(\mathbf{i2})} + \begin{bmatrix} 0 & 0 & \dot{\psi} \end{bmatrix} \underbrace{\begin{bmatrix} (\mathbf{i1})_x \\ (\mathbf{i1})_y \\ (\mathbf{i1})_z \end{bmatrix}}_{(\mathbf{i1})} \quad (\text{B.6})$$

$$= \begin{bmatrix} \dot{\phi} & 0 & 0 \end{bmatrix} \mathbf{b} + \begin{bmatrix} 0 & \dot{\theta} & 0 \end{bmatrix} \mathbf{R}_{x,\phi} \mathbf{R}_{y,\theta} \mathbf{b} + \begin{bmatrix} 0 & 0 & \dot{\psi} \end{bmatrix} \mathbf{R}_{x,\phi} \mathbf{b} \quad (\text{B.7})$$

$$= \left( \begin{bmatrix} \dot{\phi} & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \dot{\theta} & 0 \end{bmatrix} \mathbf{R}_{x,\phi} \mathbf{R}_{y,\theta} + \begin{bmatrix} 0 & 0 & \dot{\psi} \end{bmatrix} \mathbf{R}_{x,\phi} \right) \mathbf{b} \quad (\text{B.8})$$

in which it is used the fact that the frame  $(\mathbf{i1})$  is obtained through a rotation of  $\phi$  degrees about the  $\mathbf{b}_x$  axis of the  $\mathbf{b}$  frame and the frame  $(\mathbf{i2})$  is obtained through a rotation of  $\theta$  degrees about the  $(\mathbf{i1})_y$  axis of the  $(\mathbf{i1})$  frame. Considering  $\boldsymbol{\omega}_i^b = [\omega_x \ \omega_y \ \omega_z]^T \mathbf{b}$ , i.e.  $\boldsymbol{\omega}_i^b$  is measured

in the body frame, it follows, after developing the rotation operators and summing the terms, that

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \cos \theta \sin \phi \\ 0 & -\sin \phi & \cos \theta \cos \phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \quad (\text{B.9})$$

which, after isolating the Euler rates, finally gives

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan \theta \sin \phi & \tan \theta \cos \phi \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sec \theta \sin \phi & \sec \theta \cos \phi \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \quad (\text{B.10})$$

Another approach for achieving the Euler rates equations may be seen in [Kuipers et al. 1999, p. 263].

# Appendix C

## The 6-parameter magnetometer calibration

As explained in [Foster and Elkaim 2008], there are mainly five different sources of errors that may affect magnetometer readings. Following the procedure described in [STMicroelectronics 2018], these errors may be grouped in the following calibration equation,

$$\begin{bmatrix} B_{nx} \\ B_{ny} \\ B_{nz} \end{bmatrix} = [\mathbf{C}_m]_{3 \times 3} \underbrace{\begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & \frac{1}{S_z} \end{bmatrix}}_{=\mathbf{C}_S} [\mathbf{C}_{si}]_{3 \times 3} \begin{bmatrix} B_{rx} - O_x \\ B_{ry} - O_y \\ B_{rz} - O_z \end{bmatrix} \quad (\text{C.1})$$

$$= \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} B_{rx} - O_x \\ B_{ry} - O_y \\ B_{rz} - O_z \end{bmatrix} = \mathbf{C}(\mathbf{B}_r - \mathbf{O}), \quad (\text{C.2})$$

in which  $B_{ni}$ ,  $i \in \{x, y, z\}$  are the normalized magnetometer readings,  $B_{ri}$ ,  $i \in \{x, y, z\}$  are the raw magnetometer readings,  $\mathbf{C}_m$  is the misalignment correction matrix,  $\mathbf{C}_{si}$  is the soft iron effect correction matrix,  $S_i$ ,  $i \in \{x, y, z\}$  are the scaling factors and  $O_i$ ,  $i \in \{x, y, z\}$  are the offsets. After performing the calibration methods, the calibration matrix  $\mathbf{C}$  must be obtained, as well as the offsets vector  $\mathbf{O}$ . The goal of the calibration is to obtain unit magnetic field vectors which may be scaled with the local magnetic field intensity.

In this work, it is considered that each of the three magnetic field sensors inside the IMU embedded in the LAICA testbed are perfectly orthogonal within each other, given all of them are inside the same chip, which means that  $\mathbf{C}_m$  is a  $3 \times 3$  identity matrix. The soft iron effect will be considered negligible, which implies that  $\mathbf{C}_{si}$  is also a  $3 \times 3$  identity matrix. For calibrating the magnetometer readings, two kinds of error will be estimated, scaling factors ( $\mathbf{C}_S$  matrix) and offsets (vector  $\mathbf{O}$ ). Also, since the consequence of the hard iron effect is an offset in the readings, it is mathematically indistinguishable of the null shift errors and both



these kinds of error are estimated together.

To perform the calibration, a set of magnetometer readings is acquired while the sensor is rotated in an approximately  $4\pi$  steradian surface. The expected result is, for a sensor suffering from all the five kinds of calibration errors, a tilted ellipsoid, whose equation is given by

$$\begin{aligned} & \frac{(B_{rx} - B_{rx0})^2}{a^2} + \frac{(B_{ry} - B_{ry0})^2}{b^2} + \frac{(B_{rz} - B_{rz0})^2}{c^2} + \dots \\ & \dots + \frac{(B_{rx} - B_{rx0})(B_{ry} - B_{ry0})}{d^2} + \frac{(B_{rx} - B_{rx0})(B_{rz} - B_{rz0})}{e^2} + \frac{(B_{ry} - B_{ry0})(B_{rz} - B_{rz0})}{f^2} = R^2, \end{aligned} \quad (C.3)$$

in which  $B_{rx0}$ ,  $B_{ry0}$  and  $B_{rz0}$  are the coordinates of the center of the ellipsoid,  $a$ ,  $b$  and  $c$  are the semi-axes lengths,  $d$ ,  $e$  and  $f$  are the cross axis effect which makes the ellipsoid tilted and  $R$  is a constant related to the size of the ellipsoid or its radius, when  $a = b = c$ . Since the soft iron effect is considered negligible, Eq. (C.3) may be simplified as

$$\frac{(B_{rx} - B_{rx0})^2}{a^2} + \frac{(B_{ry} - B_{ry0})^2}{b^2} + \frac{(B_{rz} - B_{rz0})^2}{c^2} = R^2. \quad (C.4)$$

To estimate all the unknown parameters in Eq. (C.4), the Least Squares Method (LSM) is used. First, the unknown parameters and the magnetometer readings may be separated as follows

$$\underbrace{B_{rx}^2}_{\mathbf{w}} = \underbrace{\begin{bmatrix} B_{rx} & B_{ry} & B_{rz} & -B_{ry}^2 & -B_{rz}^2 & 1 \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} 2B_{rx0} \\ \frac{a^2}{b^2} 2B_{ry0} \\ \frac{a^2}{c^2} 2B_{rz0} \\ \frac{a^2}{b^2} \\ \frac{a^2}{c^2} \\ a^2 R^2 - B_{rx0}^2 - \frac{a^2}{b^2} B_{ry0}^2 - \frac{a^2}{c^2} B_{rz0}^2 \end{bmatrix}}_{\mathbf{X}}, \quad (C.5)$$

and written, for a single measurement, as

$$\mathbf{w}_{n \times 1} = \mathbf{H}_{n \times 6} \mathbf{X}_{6 \times 1}. \quad (C.6)$$

Augmenting Eq. (C.6) for a set of  $n$  measurements, the solution of the unknown parameters is given by

$$\mathbf{X} = [\mathbf{H}^T \mathbf{H}]^{-1} \mathbf{H}^T \mathbf{w} \quad (C.7)$$

and the calibration parameters may be extracted from the  $\mathbf{X} = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6]^T$  vector

as follows

$$O_x = B_{rx0} = \frac{x_1}{2} \quad (\text{C.8})$$

$$O_y = B_{ry0} = \frac{x_2}{2x_4} \quad (\text{C.9})$$

$$O_z = B_{rz0} = \frac{x_3}{2x_5} \quad (\text{C.10})$$

$$A = a^2 R^2 = x_6 + B_{rx0}^2 + x_4 B_{ry0}^2 + x_5 B_{rz0}^2 \quad (\text{C.11})$$

$$B = \frac{A}{x_4} \quad (\text{C.12})$$

$$C = \frac{A}{x_5} . \quad (\text{C.13})$$

The ellipsoid can be compensated for the offsets - which means recentering the ellipsoid to the (0, 0, 0) origin - making the following transformation

$$B_{rx}^{(1)} = B_{rx} - O_x \quad (\text{C.14})$$

$$B_{ry}^{(1)} = B_{ry} - O_y \quad (\text{C.15})$$

$$B_{rz}^{(1)} = B_{rz} - O_z , \quad (\text{C.16})$$

which turns Eq. (C.4) into

$$\frac{(x^{(1)})^2}{A} + \frac{(y^{(1)})^2}{B} + \frac{(z^{(1)})^2}{C} = 1 . \quad (\text{C.17})$$

Then, determining the scaling factors as

$$S_x = \sqrt{A} \quad (\text{C.18})$$

$$S_y = \sqrt{B} \quad (\text{C.19})$$

$$S_z = \sqrt{C} \quad (\text{C.20})$$

and performing the following transformation

$$B_{nx} = \frac{B_{rx}^{(1)}}{S_x} \quad (\text{C.21})$$

$$B_{ny} = \frac{B_{ry}^{(1)}}{S_y} \quad (\text{C.22})$$

$$B_{nz} = \frac{B_{rz}^{(1)}}{S_z} , \quad (\text{C.23})$$

the magnetometer readings become a unit sphere centered at the origin, as desired, *i.e.*

$$B_{nx}^2 + B_{ny}^2 + B_{nz}^2 = 1 . \quad (\text{C.24})$$

# Appendix D

## Angular momentum about point A - simplified equation

The angular momentum of a rigid body about an arbitrary point A in space may be obtained by summing up the moment of momentum of each particle of this body. Considering discrete particles, the moment of momentum of the  $i^{th}$  particle is given, as shown in Eq.(3.42) by

$$(\mathbf{H}_{A,i}) = \boldsymbol{\rho}_{A,i} \times m_i \mathbf{v}_A + \boldsymbol{\rho}_{A,i} \times m_i (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) , \quad (\text{D.1})$$

which, after integrating for infinitesimal particles all over the body, gives

$$\mathbf{H}_A = \lim_{m_i \rightarrow dm} \sum_i (\mathbf{H}_{A,i}) = \left( \int_m \boldsymbol{\rho}_{A,i} dm \right) \times \mathbf{v}_A + \int_m \boldsymbol{\rho}_{A,i} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) dm . \quad (\text{D.2})$$

Considering that the position of each infinitesimal particle with relation to the Center of Mass (CM or G) may be expressed as

$$\boldsymbol{\rho}_{A,i} = \boldsymbol{\rho}_{G,i} + \boldsymbol{\rho}_{G/A} , \quad (\text{D.3})$$

which means that the distance of the  $i^{th}$  infinitesimal particle to the point A equals the distance of this particle to the Center of Mass (G) plus the distance between the point G with relation to the point A. Considering the point A is fixed to the body, the vector  $\boldsymbol{\rho}_{G/A}$  is constant

in the body frame. Substituting this expression in Eq. D.2 gives

$$\mathbf{H}_A = \left( \int_m \boldsymbol{\rho}_{A,i} dm \right) \times \mathbf{v}_A + \int_m \boldsymbol{\rho}_{A,i} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) dm \quad (\text{D.4})$$

$$= \left( \int_m (\boldsymbol{\rho}_{G,i} + \boldsymbol{\rho}_{G/A}) dm \right) \times \mathbf{v}_A + \int_m \boldsymbol{\rho}_{A,i} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) dm \quad (\text{D.5})$$

$$= \left( \int_m \boldsymbol{\rho}_{G,i} dm \right) \times \mathbf{v}_A + \left( \int_m \boldsymbol{\rho}_{G/A} dm \right) \times \mathbf{v}_A + \int_m \boldsymbol{\rho}_{A,i} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) dm \quad (\text{D.6})$$

$$= m\boldsymbol{\rho}_{G/A} \times \mathbf{v}_A + \underbrace{\int_m \boldsymbol{\rho}_{A,i} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{A,i}) dm}_{(II)} , \quad (\text{D.7})$$

since, by definition,  $(\int_m \boldsymbol{\rho}_{G,i} dm) = 0$  and  $\int_m \boldsymbol{\rho}_{G/A} dm = m\boldsymbol{\rho}_{G/A}$ , since  $\boldsymbol{\rho}_{G/A}$  is a constant vector. The integral in the right-hand side of Eq. (D.7) may be developed as

$$(II) = \int_m (\boldsymbol{\rho}_{G,i} + \boldsymbol{\rho}_{G/A}) \times (\boldsymbol{\omega} \times [\boldsymbol{\rho}_{G,i} + \boldsymbol{\rho}_{G/A}]) dm \quad (\text{D.8})$$

$$= \int_m \boldsymbol{\rho}_G \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_G) dm + \int_m \boldsymbol{\rho}_G dm \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{G/A}) + \dots \quad (\text{D.9})$$

$$\dots + \boldsymbol{\rho}_{G/A} \times \left( \boldsymbol{\omega} \times \int_m \boldsymbol{\rho}_G dm \right) + \boldsymbol{\rho}_{G/A} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{G/A})m \quad (\text{D.10})$$

$$= \mathbf{H}_G + m\boldsymbol{\rho}_{G/A} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{G/A}) , \quad (\text{D.11})$$

since, from Eq. (3.45),  $\mathbf{H}_G = \int_m \boldsymbol{\rho}_G \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_G) dm$ . Substituting (II) in Eq. (D.7) gives

$$\mathbf{H}_A = m\boldsymbol{\rho}_{G/A} \times \mathbf{v}_A + \mathbf{H}_G + m\boldsymbol{\rho}_{G/A} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}_{G/A}) , \quad (\text{D.12})$$

$$= \mathbf{H}_G + m\boldsymbol{\rho}_{G/A} \times (\mathbf{v}_A + \boldsymbol{\omega} \times \boldsymbol{\rho}_{G/A}) . \quad (\text{D.13})$$

From the relative motion between A and G, the velocities of both points are related as

$$\mathbf{v}_G = \mathbf{v}_A + \boldsymbol{\omega} \times \boldsymbol{\rho}_{G/A} , \quad (\text{D.14})$$

which, after substitution in Eq. (D.13), gives

$$\mathbf{H}_A = \boldsymbol{\rho}_{G/A} \times m\mathbf{v}_G + \mathbf{H}_G , \text{ quod erat demonstrandum.} \quad (\text{D.15})$$

# Appendix E

## Quaternion algebra

The following definitions, which describe the algebra of quaternions, were obtained from [Kuipers et al. 1999, Chap. 5].

**Definition E.0.1 Addition.** Being  $p$  and  $q$  two different quaternions and  $p_i$  and  $q_i$ ,  $i \in 1, \dots, 4$  the respective components, its addition is defined as

$$p + q = (p_0 + q_0) + \mathbf{i}(p_1 + q_1) + \mathbf{j}(p_2 + q_2) + \mathbf{k}(p_3 + q_3), \quad (\text{E.1})$$

in which the versors  $\mathbf{i}$ ,  $\mathbf{j}$ ,  $\mathbf{k}$  are treated as pure complex quaternions, i.e.

$$\mathbf{i} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \quad (\text{E.2})$$

$$\mathbf{j} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \quad (\text{E.3})$$

$$\mathbf{k} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}. \quad (\text{E.4})$$

**Definition E.0.2 Equality.** Being  $p$  and  $q$  two different quaternions and  $p_i$  and  $q_i$ ,  $i \in 1, \dots, 4$  the respective components, one can say that  $p$  and  $q$  are equal if and only if the components of  $p$  and  $q$  with same index are equal, i.e. if

$$p_0 = q_0 \quad (\text{E.5})$$

$$p_1 = q_1 \quad (\text{E.6})$$

$$p_2 = q_2 \quad (\text{E.7})$$

$$p_3 = q_3. \quad (\text{E.8})$$

**Definition E.0.3 Conjugate.** Being  $p = p_0 + \mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3$  an arbitrary quaternion, its conjugate is obtained by negating its complex components, i.e.

$$\bar{p} = p_0 - \mathbf{i}p_1 - \mathbf{j}p_2 - \mathbf{k}p_3, \quad (\text{E.9})$$

or, if the quaternion is represented as  $p = p_0 + \mathbf{p}$ , its conjugate is given by

$$p = p_0 - \mathbf{p} , \quad (\text{E.10})$$

in which  $\mathbf{p}$  is called the complex or vector part of the quaternion  $q$ .

**Definition E.0.4 Norm.** Being  $p = p_0 + \mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3$  an arbitrary quaternion, its norm is a scalar given by

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}, \quad \|q\| \in \mathbb{R} . \quad (\text{E.11})$$

The product between two quaternions was already defined throughout the text of Chapter 3. An useful relation is given by the product of a quaternion and its conjugate, which gives

$$\begin{aligned} p\bar{p} &= (p_0 + \mathbf{p})(p_0 - \mathbf{p}) \\ &= p_0^2 - \mathbf{p} \cdot -\mathbf{p} + p_0 - \mathbf{p} + p_0 \mathbf{p} + \mathbf{p} \times -\mathbf{p} \\ &= p_0^2 + \mathbf{p} \cdot \mathbf{p} \\ &= p_0^2 + p_1^2 + p_2^2 + p_3^2 = \|p\|^2 \end{aligned} \quad (\text{E.12})$$

**Definition E.0.5 Inverse.** Being  $p = p_0 + \mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3$  an arbitrary quaternion, its inverse is another quaternion which satisfies

$$p^{-1}p = pp^{-1} = 1 . \quad (\text{E.13})$$

Pre-multiplying and post-multiplying Eq. (E.13) by the conjugate of  $p$ , it follows that

$$p^{-1}p\bar{p} = \bar{p}pp^{-1} = \bar{p} \Rightarrow p^{-1}\|p\|^2 = \bar{p} \Rightarrow p^{-1} = \frac{\bar{p}}{\|p\|^2} , \quad (\text{E.14})$$

in which Eq. (E.12) was used. It is important to notice that, if the quaternion has unit norm, its inverse equals its conjugate, which is an useful relation when working with rotation quaternions.

**Observation:** the fact that the product of two arbitrary quaternions is not commutative justifies why  $p^{-1}p$  and  $pp^{-1}$  Eq. (E.13) cannot be both pre-multiplied or post-multiplied.

# Appendix F

## Quaternion rates

To demonstrate the quaternion rates equation,  $\dot{\mathbf{q}}$ , the limit definition of derivatives may be used, *i.e.*

$$\dot{\mathbf{q}} = \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q}(t + \Delta t) - \mathbf{q}}{\Delta t} \quad (\text{F.1})$$

The unit quaternion  $\mathbf{q}(t + \Delta t)$  is obtained from the initial unit quaternion  $\mathbf{q}$  after performing an infinitesimal rotation. This rotation, using rotation quaternion operators, is given by

$$\mathbf{q}(t + \Delta t) = \mathbf{q}_R \mathbf{q}, \quad (\text{F.2})$$

in which  $\mathbf{q}_R$  is the quaternion rotation operator. This rotation operator is described as an infinitesimal rotation of  $\Delta\alpha$  degrees about the  $\mathbf{u}$  axis, which is a unit vector in the same direction of the angular velocity  $\boldsymbol{\omega}$  of the system, during the interval  $\Delta t$ . In other words,

$$\mathbf{q}_R = \cos\left(\frac{\Delta\alpha}{2}\right) + \mathbf{u} \sin\left(\frac{\Delta\alpha}{2}\right) \approx 1 + \mathbf{u} \frac{\Delta\alpha}{2}, \quad (\text{F.3})$$

in which attention must be given to the fact that, although the argument of the  $\cos$  and  $\sin$  functions is  $\frac{\Delta\alpha}{2}$ , the rotation performed by  $\mathbf{q}_R$  is  $\Delta\alpha$  degrees. Also, the approximations  $\cos \theta \approx 1$ ,  $\theta \ll 1$  and  $\sin \theta \approx \theta$ ,  $\theta \ll 1$  are used. Substituting Eq. (F.3) in Eq. (F.2) and the result in Eq. (F.1) gives

$$\dot{\mathbf{q}} = \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q}(t + \Delta t) - \mathbf{q}}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q}_R \mathbf{q} - \mathbf{q}}{\Delta t} \quad (\text{F.4})$$

$$= \lim_{\Delta t \rightarrow 0} \frac{\left(1 + \mathbf{u} \frac{\Delta\alpha}{2}\right) \mathbf{q} - \mathbf{q}}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{\mathbf{u} \frac{\Delta\alpha}{2} \mathbf{q}}{\Delta t} \quad (\text{F.5})$$

$$= \frac{1}{2} \mathbf{q} \mathbf{u} \lim_{\Delta t \rightarrow 0} \frac{\Delta\alpha}{\Delta t} = \frac{1}{2} \mathbf{q} \mathbf{u} \boldsymbol{\omega} = \frac{1}{2} \mathbf{q} \boldsymbol{\omega}, \quad (\text{F.6})$$

in which  $\omega$  is the scalar angular rate about the direction  $\mathbf{u}$  and  $\boldsymbol{\omega} = \mathbf{u} \omega$  is the angular rate vector of the quaternion  $\mathbf{q}_R$ . To obtain Eq. (F.6) in matrix form, it suffices to perform the quaternion multiplication in matrix form (Eq. (3.13)) between  $\mathbf{q}$  and  $\boldsymbol{\omega} = [0 \ \omega_x \ \omega_y \ \omega_z]^T$ .

# Appendix G

## The Kalman Filter

The discrete-time Kalman Filter is summarized in the steps that follows. The focus in this work is in the usage of the filter and familiarization with the filter notation, whereas the reader may refer to [Simon 2006, p. 123] for further details.

1. The first step in the Kalman filtering approach is to arrange the system model in the state-space form. The most usual path is obtaining a set of continuous-time differential equations describing the system dynamics. These equations may be discretized using any discretization rule, from the simple Euler first order discretization to the fourth order Runge-Kutta method. The discretized dynamic system equations must be arranged in the following format:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \\ \mathbf{y}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k\end{aligned}\tag{G.1}$$

in which  $\mathbf{x}$  is the vector comprising the chosen system states,  $\mathbf{u}$  is the system input,  $\mathbf{y}$  is the system output,  $\mathbf{w}$  is the process noise,  $\mathbf{v}$  is the measurement noise and the  $\mathbf{F}$ ,  $\mathbf{G}$  and  $\mathbf{H}$  matrices are dependent on the system equations and complete the filter design. Also, for the  $\mathbf{w}$  and  $\mathbf{v}$  noises the following properties apply

$$\begin{aligned}E(\mathbf{w}_k\mathbf{w}_j^T) &= \mathbf{Q}_k\delta_{k-j} \\ E(\mathbf{v}_k\mathbf{v}_j^T) &= \mathbf{R}_k\delta_{k-j} \\ E(\mathbf{w}_k\mathbf{v}_j^T) &= \mathbf{0},\end{aligned}\tag{G.2}$$

in which  $\mathbf{Q}$  and  $\mathbf{R}$  are the covariance matrices of the  $\mathbf{w}$  and  $\mathbf{v}$  noises, respectively, and the  $E(\odot)$  operator denotes the expected value. From these equations, some assumptions are made clear:

- The states vector  $\mathbf{x}_{k-1}$  and the system input  $\mathbf{u}_{k-1}$  are linear within the filter equa-



tions.

- Both the  $w$  and  $v$  noises are additive and uncorrelated.

Another essential covariance matrix is the  $\mathbf{P}$  matrix, which represents the covariance of the system states and is used throughout the filter run.

2. To start the filter, the *a posteriori* initial state estimate and initial state covariance matrix must be initialized. Considering the knowledge of the state in the beginning of the process, these variables may be initialized as

$$\begin{aligned}\hat{\mathbf{x}}_0^+ &= E(\mathbf{x}_0) \\ \mathbf{P}_0^+ &= E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)^T]\end{aligned}\quad (\text{G.3})$$

where the hat symbol  $\hat{\phantom{x}}$  indicates an estimated variable.

3. The Kalman gain  $\mathbf{K}$ , the state covariance matrix  $\mathbf{P}$  and the state estimates  $\hat{\mathbf{x}}$  are updated iteratively for  $k = 1, 2, \dots$  according to the following equations

$$\begin{aligned}\mathbf{P}_k^- &= \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ &= \mathbf{P}_k^+\mathbf{H}_k^T\mathbf{R}_k^{-1} \\ \hat{\mathbf{x}}_k^- &= \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1}\mathbf{u}_{k-1} = \textit{a priori state estimate}\end{aligned}\quad (\text{G.4})$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) = \textit{a posteriori state estimate}\quad (\text{G.5})$$

$$\begin{aligned}\mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \\ &= [(\mathbf{P}_k^-)^{-1} + \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k]^{-1} \\ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-\end{aligned}\quad (\text{G.6})$$

in which it is noticed that the  $-$  and  $+$  superscripts are used to indicate the *a priori* and the *a posteriori* values of the referred quantities, *i.e.* the value before and after the measurement update is done, explained in what follows. It is expected that, if the filter runs correctly, the estimated state vector  $\hat{\mathbf{x}}$  will converge to the true state of the system  $\mathbf{x}$  at each instant. It must be noticed that the iterative portion of the Kalman Filter may be divided in two phases: the time update and the measurement update. The time update phase, represented by Eq. (G.4), predicts the propagation in time of the current state using the known system dynamics, whereas the measurement update phase, represented by Eq. (G.5), utilizes the system outputs - usually measurements obtained from the system sensors - to correct the current state.

# Appendix H

## The EKF Algorithm

Maintaining the philosophy adopted in this work for all the filters, only the algorithm of the Extended Kalman Filter (EKF) will be presented, since its concept is already well-known in literature. The reader may refer to [Simon 2006, p. 407] for further details on the filter derivation. In this work, it suffices to understand that the EKF is an approach in which the nonlinear equations of the system are linearized around the current state in order to make the classical Kalman filter applicable. The EKF is summarized as follows.

1. The EKF is one of the filter approaches used with nonlinear systems. After discretizing the system equations, one may end up with a nonlinear system of the following form:

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k)$$

$$\mathbf{w}_k \sim (0, \mathbf{Q}_k)$$

$$\mathbf{v}_k \sim (0, \mathbf{R}_k)$$

in which, considering this general notation, the process noise  $\mathbf{w}$  and the measurement noise  $\mathbf{v}$  may be additive or not. Also, the state equation, given by  $\mathbf{f}_{k-1}$ , and the output equation, given by  $\mathbf{h}_k$  may be nonlinear or not, but are functions of the  $\mathbf{x}$ ,  $\mathbf{u}$ ,  $\mathbf{w}$  and  $\mathbf{v}$  vectors. If at least one of the nonlinear equations,  $\mathbf{f}$  or  $\mathbf{h}$ , are nonlinear, the EKF usage is justified. Both the  $\mathbf{w}$  and  $\mathbf{v}$  noises are assumed to be zero mean Gaussian noises with its covariances given by the  $\mathbf{Q}$  and  $\mathbf{R}$  matrices, respectively.

2. Maintaining the strategy adopted in the classical Kalman Filter, reasonable guesses for the initial *a posteriori* state  $\hat{\mathbf{x}}_0$  and the *a posteriori* state covariance matrix  $\mathbf{P}_0^+$  are given by

$$\hat{\mathbf{x}}_0^+ = E(\mathbf{x}_0)$$

$$\mathbf{P}_0^+ = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)^T]$$

3. The following equations must be run iteratively, for  $k = 1, 2, 3, \dots$ :

- Considering the state equation, the following jacobian matrices must be calculated and evaluated at the *a posteriori* previous state  $\mathbf{x}_{k-1}^+$ :

$$\mathbf{F}_{k-1} = \left. \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}} \right|_{\mathbf{x}_{k-1}^+}$$

$$\mathbf{L}_{k-1} = \left. \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{w}} \right|_{\mathbf{x}_{k-1}^+}$$

- With the  $\mathbf{F}_{k-1}$  and  $\mathbf{L}_{k-1}$  jacobians determined, the time update of the previous *a posteriori* state estimate and the previous *a posteriori* estimation-error covariance must be performed as follows:

$$\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{L}_{k-1} \mathbf{Q}_{k-1} \mathbf{L}_{k-1}^T$$

$$\hat{\mathbf{x}}_k^- = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}, 0),$$

in which it is assumed that the process noise  $\mathbf{v}$  is not present. In this phase, the only the known system dynamics are used to predict the propagation in time of the current state.

- In sequence, another two jacobian matrices must be calculated,  $\mathbf{H}_k$  and  $\mathbf{M}_k$ . However, differently from the  $\mathbf{F}_{k-1}$  and  $\mathbf{L}_{k-1}$  jacobians, these are evaluated at the *a priori* current state  $\mathbf{x}_k^-$  as follows:

$$\mathbf{H}_k = \left. \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}} \right|_{\mathbf{x}_k^-}$$

$$\mathbf{M}_k = \left. \frac{\partial \mathbf{h}_k}{\partial \mathbf{v}} \right|_{\mathbf{x}_k^-}$$

- Finally, at the end of the current iteration, the measurement update of the state estimate and estimation-error covariance is performed as follows:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^T)^{-1}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k [\mathbf{y}_k - \mathbf{h}_k(\hat{\mathbf{x}}_k^-, 0)]$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$$

in which the estimation of the current output,  $\hat{\mathbf{y}}_k$ , given by  $\mathbf{h}_k(\hat{\mathbf{x}}_k^-, 0)$ , is evaluated considering that the measurement noise  $\mathbf{v}$  is not present.

# Appendix I

## The Complex Step Differentiation (CSD)

The Complex Step Differentiation (CSD) represents an advantage with respect to the analytic solution of derivatives. When used with the Extended Kalman Filter (EKF), it provides a good generalization for the filter: independently of the system model, the jacobians may be directly estimated numerically. Although it increases the computational effort required to run the filter, it facilitates the filter implementation, which is reasonable when the interest relies only on the offline filter simulation.

The general idea of the CSD method is simplified in this appendix, whereas the reader may refer to [Al-Mohy and Higham 2010, Martins et al. 2001, Martins et al. 2003]. for further details. Starting from a complex function  $f(z) = u(z) + i \cdot v(z)$  of a complex variable  $z = x + i \cdot y$  on the complex plane, if this function is analytic, then the Cauchy-Riemann equations are valid,

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \quad (\text{I.1})$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \quad (\text{I.2})$$

The partial derivatives in Eq. (I.1) may be rewritten using its limit definition. For  $\frac{\partial u}{\partial x}$ , it gives

$$\frac{\partial u}{\partial x} = \lim_{h \rightarrow 0} \frac{v(x + i(y + h)) - v(x + iy)}{h} . \quad (\text{I.3})$$

Considering the focus is to determine the numerical derivative of a real function which describes a physical system, the following assumptions are made:

$$y = 0 \quad (\text{I.4})$$

$$u(x) = f(x) \quad (\text{I.5})$$

$$v(x) = 0 , \quad (\text{I.6})$$

and the limit in Eq. I.3 simplifies to

$$\frac{\partial f}{\partial x} = \frac{\partial u}{\partial x} = \lim_{h \rightarrow 0} \frac{\text{Im}[f(x + ih)]}{h}. \quad (\text{I.7})$$

In other words, approximating the limit in Eq.I.7 for a small  $h$  the ingenious idea behind the CSD method is summarized by the following equation

$$\frac{\partial f}{\partial x} \approx \frac{\text{Im}[f(x + ih)]}{h}, \quad (\text{I.8})$$

which does not use subtractive operations. It is known that subtractive operations incur in cancellation errors which are avoided by the CSD method, since it needs only a division after evaluating the imaginary part of  $f(x + ih)$ .

# Appendix J

## The Unscented Kalman Filter

It is known that the Extended Kalman Filter is widely used in state estimation for nonlinear systems. However, if the system presents strong nonlinearities, the approximation errors caused by the linearization used to propagate the mean and covariance of the current state may not be negligible. In order to avoid this source of error, the Unscented Kalman Filter (UKF) was developed [Julier and Uhlmann 2004]. The basic idea of the UKF is that it is easier to approximate a probability distribution than approximating an arbitrary nonlinear function and this idea culminated with the creation of the Unscented Transformation (UT). Again, the reader may refer to [Simon 2006] for further details, such as the derivation of the UT. In what follows, the UKF is presented algorithmically.

1. As well as in the EKF, the UKF assumes a nonlinear system of equations describing the system dynamics. The general model is replicated:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{f}_{k-1}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \\ \mathbf{w}_k &\sim (0, \mathbf{Q}_k) \\ \mathbf{v}_k &\sim (0, \mathbf{R}_k)\end{aligned}$$

in which it is made the same assumptions of the EKF regarding the process and measurement noises,  $\mathbf{w}$  and  $\mathbf{v}$ .

2. Also, the same filter initialization guesses used in the KF and in the EKF hold for the initial state  $\hat{\mathbf{x}}_0^+$  and state covariance matrix  $\mathbf{P}_0^+$ , which are

$$\begin{aligned}\hat{\mathbf{x}}_0^+ &= E(\mathbf{x}_0) \\ \mathbf{P}_0^+ &= E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)^T]\end{aligned}$$

3. Having the filter initialized, the following steps are run iteratively, for  $k = 1, 2, 3, \dots$ :

- In the **time update** phase, the state and the state covariance matrix are propagated in time using only the previous gathered information and the model dynamics. In the UKF, this phase is divided in four steps, as follows.

- (a) In the first step, the  $2n$  sigma points  $\mathbf{x}_{k-1}^{(i)}$  are determined. The superscript  $(i)$  denotes the  $i^{th}$  sigma point of a total of  $2n$  sigma points, in which  $n$  is the number of states in the state vector  $\mathbf{x}$ . The sigma points are calculated as

$$\begin{aligned}\hat{\mathbf{x}}_{k-1}^{(i)} &= \hat{\mathbf{x}}_{k-1}^+ + \tilde{\mathbf{x}}^{(i)}, \quad i = 1, \dots, 2n \\ \tilde{\mathbf{x}}^{(i)} &= \left( \sqrt{n\mathbf{P}_{k-1}^+} \right)_i^T, \quad i = 1, \dots, n \\ \tilde{\mathbf{x}}^{(n+i)} &= - \left( \sqrt{n\mathbf{P}_{k-1}^+} \right)_i^T, \quad i = 1, \dots, n\end{aligned}$$

in which the subscript  $i$  outside the parenthesis denotes that the  $i^{th}$  column of the matrix inside the parenthesis is taken. It must be noticed that the sigma points are calculated around the previous *a posteriori* estimated state  $\hat{\mathbf{x}}_{k-1}^+$ .

- (b) The sigma points must be transformed in  $\hat{\mathbf{x}}_k^{(i)}$  vectors, *i.e.* propagated in time through the nonlinear state equation  $\mathbf{f}_{k-1}(\cdot)$ , as

$$\hat{\mathbf{x}}_k^{(i)} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_k) \quad (\text{J.1})$$

- (c) The sigma points are combined in order to obtain the *a priori* estimated state  $\hat{\mathbf{x}}_k^-$ , as

$$\hat{\mathbf{x}}_k^- = \frac{1}{2n} \sum_{i=1}^{2n} \hat{\mathbf{x}}_k^{(i)}. \quad (\text{J.2})$$

In this case, a simple arithmetic mean is used. Some studies refer to methodologies which result in optimal weighting functions for combining the  $2n$  sigma points, rather than assigning the same weight to every sigma point.

- (d) Still using the transformed sigma points  $\hat{\mathbf{x}}_k^{(i)}$ , the *a priori* state covariance matrix must be determined as

$$\mathbf{P}_k^- = \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{x}}_k^{(i)} - \hat{\mathbf{x}}_k^-)(\hat{\mathbf{x}}_{k-1}^{(i)} - \hat{\mathbf{x}}_k^-) + \mathbf{Q}_{k-1} \quad (\text{J.3})$$

The (a), (b), (c) and (d) steps combined in order to obtain  $\hat{\mathbf{x}}_k^-$  and  $\mathbf{P}_k^-$  form what is called an **Unscented Transformation (UT)**.

- In the **measurement update** phase, the *a priori* state estimation is updated using the output equation of the system. This phase is divided in four steps, as follows.

- (a) An **optional step** is to recalculate the sigma points, but, instead of using the previous *a posteriori* estimated state  $\hat{\mathbf{x}}_{k-1}^+$ , the current *a priori* estimated

state  $\hat{\mathbf{x}}_k^-$  given by the time update phase must be used, as

$$\begin{aligned}\hat{\mathbf{x}}_k^{(i)} &= \hat{\mathbf{x}}_k^- + \tilde{\mathbf{x}}^{(i)}, \quad i = 1, \dots, 2n \\ \tilde{\mathbf{x}}^{(i)} &= \left( \sqrt{n\mathbf{P}_{k-1}^+} \right)_i^T, \quad i = 1, \dots, n \\ \tilde{\mathbf{x}}^{(n+i)} &= - \left( \sqrt{n\mathbf{P}_{k-1}^+} \right)_i^T, \quad i = 1, \dots, n\end{aligned}$$

Considering or not this step means a trade-off between computational effort and filter performance.

- (b) The nonlinear output equation  $\mathbf{h}_k$  is used to transform the sigma points in  $\hat{\mathbf{y}}_k^{(i)}$  vectors, as

$$\hat{\mathbf{y}}_k^{(i)} = \mathbf{h}_k(\hat{\mathbf{x}}_k^{(i)}) \quad (\text{J.4})$$

- (c) The Unscented Transformation is finished by combining the predicted measurement vectors  $\hat{\mathbf{y}}_k^{(i)}$  to obtain the predicted measurement at current time  $\hat{\mathbf{y}}_k$ , the covariance of the predicted measurement  $\mathbf{P}_y$  and the cross-covariance between  $\hat{\mathbf{x}}_k^-$  and  $\hat{\mathbf{y}}_k$ ,  $\mathbf{P}_{xy}$ , as

$$\hat{\mathbf{y}}_k = \frac{1}{2n} \sum_{i=1}^{2n} \hat{\mathbf{y}}_k^{(i)} \quad (\text{J.5})$$

$$\mathbf{P}_y = \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k)(\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k)^T + \mathbf{R}_k \quad (\text{J.6})$$

$$\mathbf{P}_{xy} = \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{x}}_k^{(i)} - \hat{\mathbf{x}}_k^-)(\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k)^T \quad (\text{J.7})$$

Again, the reader may refer to works in literature regarding optimal weighting functions to calculate these quantities, instead of applying a simple arithmetic mean.

- (d) Finally, the normal KF equations may be used to obtain the *a posteriori* state estimate  $\hat{\mathbf{x}}_k^+$  and  $\mathbf{P}_k^+$  as

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{xy} \mathbf{P}_y^{-1} \\ \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k) \\ \mathbf{P}_k^+ &= \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_y \mathbf{K}_k^T\end{aligned}$$



# Appendix K

## Source codes

### K.1 Dynamic model comparison

```
function ydot = eom1(t,y)
% LAICA Testbed Equations of Motion (Chesi)
% y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]
ydot = zeros(9,1);

%J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];
%J = [0.265 0.1 0.1; 0.1 0.246 0.1; 0.1 0.1 0.427];
J0 = 10*[0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427]; % inertia tensor
%J = [10 -1 -1; -1 10 -1; -1 -1 10];

%Rcm = [-1; -1; -5]*10^-3; % CM offset vector
Rcm = [y(7); y(8); y(9)];
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity
Jaug = [m*(Rcm(2)^2+Rcm(3)^2) -m*Rcm(1)*Rcm(2) -m*Rcm(1)*Rcm(3);
        -m*Rcm(1)*Rcm(2) m*(Rcm(1)^2+Rcm(3)^2) -m*Rcm(2)*Rcm(3);
        -m*Rcm(1)*Rcm(3) -m*Rcm(2)*Rcm(3) m*(Rcm(1)^2+Rcm(2)^2)];
%Jaug=0;
J = J0 + Jaug;

omega = [y(4); y(5); y(6)];
sphi = sin(y(1));
cphi = cos(y(1));
sth = sin(y(2));
cth = cos(y(2));

%% Model equations
% Kinematic equation
% Euler rates
eulerdot = [1 sphi*tan(y(2)) cphi*tan(y(2));
            0 cphi -sphi;
            0 sphi/cth cphi/cth]*omega;
```

```

% Gravity vector in body-frame
gb = [g*sth; -g*sphi*cth; -g*cphi*cth];

% Dynamics equation
omegadot=J\(-cross(omega,J*omega)+cross(Rcm,gb*m));

%% Derivatives (xdot)
% Omega(dot)
ydot(1:3) = eulerdot;
% Euler angles (dot)
ydot(4:6) = omegadot;

```

Listing K.1: Euler equations of motion

```

function ydot = eom2(t,y)
% LAICA Testbed Equations of Motion for the Testbed (Young)
% y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]
%I = [0.265 0 0; 0 0.246 0; 0 0 0.427];
I = 10*[0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427]; % inertia tensor
%I = [10 -1 -1; -1 10 -1; -1 -1 10];

Bd = zeros(3,1); % no damping torque
m=14.307; %Kg
g=9.78; %m/s^2

cphi=cos(y(1));
sphi=sin(y(1));
cth=cos(y(2));
sth=sin(y(2));

A = [ m*y(8)^2+m*y(9)^2+I(1,1) -m*y(7)*y(8)+I(1,2) -m*y(7)*y(9)+I(1,3);
      -m*y(7)*y(8)+I(1,2) m*y(7)^2+m*y(9)^2+I(2,2) -m*y(8)*y(9)+I(2,3);
      -m*y(7)*y(9)+I(1,3) -m*y(8)*y(9)+I(2,3) m*y(7)^2+m*y(8)^2+I(3,3)];
B = [(-2*m*y(8)*y(9)+I(2,3))*y(5)^2+(2*m*y(8)*y(9)-I(2,3))*y(6)^2+(-m*y(7)*y(9)+I(1,3))*y(4)*y(5)+(m
      *y(7)*y(8)-I(1,2))*y(4)*y(6)+(m*y(8)^2-m*y(9)^2-I(2,2)+I(3,3))*y(5)*y(6);
      (2*m*y(7)*y(9)-I(1,3))*y(4)^2+(-2*m*y(7)*y(9)+I(1,3))*y(6)^2+(m*y(8)*y(9)-I(2,3))*y(4)*y(5)+(-m*y
      (7)^2+m*y(9)^2+I(1,1)-I(3,3))*y(4)*y(6)+(-m*y(7)*y(8)+I(1,2))*y(5)*y(6);
      (-2*m*y(7)*y(8)+I(1,2))*y(4)^2+(2*m*y(7)*y(8)-I(1,2))*y(5)^2+(m*y(7)^2-m*y(8)^2-I(1,1)+I(2,2))*y(4)
      *y(5)+(-m*y(8)*y(9)+I(2,3))*y(4)*y(6)+(m*y(7)*y(9)-I(1,3))*y(5)*y(6)];

M1 = [ m*g*(-y(8)*cth*cphi+y(9)*sphi*cth); % Gravitational Torque
      m*g*(y(9)*sth+y(7)*cphi*cth);
      m*g*(-y(7)*sphi*cth-y(8)*sth)];
M2=[-Bd(1)*y(4)^2; % Damping torque
     -Bd(2)*y(5)^2;
     -Bd(3)*y(6)^2];
M=M1+M2;

%% Derivatives
% Euler angles

```

```

ydot(1)=y(4)+sth*sphi/cth*y(5)+sth*cphi/cth*y(6); % phi
ydot(2)=cphi*y(5)-sphi*y(6); % theta
ydot(3)=sphi/cth*y(5)+cphi/cth*y(6); % psi
% Angular accelerations
ydot(4:6)= pinv(A)*(M-B);
% CM Offset variations (CM Offset is constant in the Testbed+RW model)
ydot(7)=0;
ydot(8)=0;
ydot(9)=0;
ydot=ydot';

```

Listing K.2: Young equations of motion

```

function YoungCrossChesi
% Model comparison

tf = 100; % simulation length
dt = 0.1; % simulation step size

%% Testbed characteristics
J = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427]; % inertia tensor
%J = [0.265 0 0; 0 0.246 0; 0 0 0.427];
%J = [10 -1 -1; -1 10 -1; -1 -1 10];

Rcm = [-1; -1; -5]*10^-0; % CM offset vector
m = 14.307; % mass of the testbed

%% Initial values of state variables
x = [0; 0; 0; J(1,1); J(2,2); J(3,3); -J(1,2); -J(2,3); -J(1,3); m*Rcm(1); m*Rcm(2); m*Rcm(3)]; % initial state
x2 = [0; 0; 0; J(1,1); J(2,2); J(3,3); -J(1,2); -J(2,3); -J(1,3); m*Rcm(1); m*Rcm(2); m*Rcm(3)]; % initial state

% Initialize arrays for later plotting
xArray = x;
x2Array = x2;
tArray = 0;
y = [0 0 0 0 0 Rcm(1) Rcm(2) Rcm(3)];
y2 = [0 0 0 0 0 Rcm(1) Rcm(2) Rcm(3)];

dtPlot = dt; % how often to plot output data

tic;
for t = dt : dt : tf
    if mod(t,1)==0
        t
    end
    tc = round(t/dt); % time counter
    %% Simulate the system.
    % Integrate
    [t1, y_dt] = ode23('eom1 ', [0 dt], y(:), 1e-5);
    y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]

```

```

[t2, y2_dt] = ode23('eom2', [0 dt], y2(:), 1e-5);
y2(:)=y2_dt(size(y2_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]

%% Observation: The state vector is x = [x1 x2]
% x1 = [omegax omegay omegaz]^T
% x2 = [Jx Jy Jz Jxy Jxz Jyz mRx mRy mRz]^T
x(1:3) = y(4:6); % x(4:12) is already set (see beginning of code)
x2(1:3) = y2(4:6); % x(4:12) is already set (see beginning of code)

%% Save data for plotting.
xArray = [xArray x];
x2Array = [x2Array x2];
tArray = [tArray t];
end

% Plot results
close all
t = 0 : dtPlot : tf;

figure;
plot(t, xArray(1:3,:)-x2Array(1:3,:));
set(gca, 'FontSize', 12); set(gcf, 'Color', 'White');
xlabel('Seconds'); ylabel('Error (\omega_1 - \omega_2) (in rad/s)');

figure;
plot(t, xArray(1:3,:));
hold on
plot(t, x2Array(1:3,:));
time=clock;
%save(['EKF-Data-day-25_' num2str(time(4)) '-' num2str(time(5)) '.mat']);
disp(['Elapsed time = ', num2str(toc), ' seconds']);

```

Listing K.3: Model comparison code for Young and Euler equations of motion

## K.2 Kalman Filter

```

function KalmanFilterForBalancing
% Kalman filter for estimating the unbalance vector r.
% Estimate the 3x1 unknown parameter vector r = [rx ry rz]
% Observation: to change the platform inertia, modify the value of
% J0 in the testbed dynamics file "eom1.m"
close all
clear all
clc
tf = 100; % simulation length
T = 0.1; % sampling time

%% Testbed characteristics

```

```

%%J = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427]; % full inertia
J = [0.265 0 0; 0 0.246 0; 0 0 0.427]; % diagonal inertia
Rcm = [-1; -1; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity

%% Matrices setup
R = [0.01^2 0 0; 0 0.01^2 0; 0 0 0.01^2]; % process noise matrix
Q = diag([0.000005 0.000005 0.000005 (0.1*Rcm(1))^2 (0.1*Rcm(2))^2 (0.1*Rcm(2))^2]); % measurement
noise matrix
H = [eye(3) zeros(3)]; % measurement matrix

%% Initial values
P_post = zeros(6);
x = [0; 0; 0; Rcm(1); Rcm(2); Rcm(3)]; % initial true state
xhat_post = [zeros(3,1); x(4:6)]; % initial state estimate
z = H * x + sqrt(R) * [randn; randn; randn]; % initial measurement

%% Initialize arrays for later plotting
PArray = P_post;
zArray = z;
xArray = x;
tArray = 0;
xhatArray = xhat_post;
y = [0 0 0 0 0 Rcm(1) Rcm(2) Rcm(3)];
yArray = y';
phi = y(1);
theta = y(2);
psi = y(3);

tic;
chi2lim = chi2inv(0.95,length(z)) % chi-squared test limit
pause(3)
d2 = 0;
for t = T : T : tf
    k = round(t/T)+1; % MATLAB initial vector position is 1
    %% Simulate the system.
    % Integrate
    [~, y_dt] = ode23('eom1', [0 T], y(:), 1e-5);
    y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]
    % The intermediate values of the integration with ode23 are transparent
    % for the x state vector.
    phi(k) = y(1);
    theta(k) = y(2);
    psi(k) = y(3);
    Phi_matrix = [0 -m*g*T/(2*J(1,1))*(cos(phi(k))*cos(theta(k))+cos(phi(k-1))*cos(theta(k-1))) m*g*T/(2*J
    (1,1))*(sin(phi(k))*cos(theta(k))+sin(phi(k-1))*cos(theta(k-1)));
    m*g*T/(2*J(2,2))*(cos(phi(k))*cos(theta(k))+cos(phi(k-1))*cos(theta(k-1))) 0 m*g*T/(2*J(2,2))*(sin(
    theta(k))+sin(theta(k-1)));

```

```

-m*g*T/(2*J(3,3))*(sin(phi(k))*cos(theta(k))+sin(phi(k-1))*cos(theta(k-1))) -m*g*T/(2*J(3,3))*(sin(
theta(k))+sin(theta(k-1))) 0];
%% Observation: The state vector is x = [wx wy wz rx ry rz]
F = [eye(3) Phi_matrix;
zeros(3) eye(3)];
x = y(4:9)' + sqrt(Q) * [randn; randn; randn; randn; randn; randn];
z = H * x + sqrt(R) * [randn; randn; randn]; % size(H)=[3 3]; size(x) = [6 1];

%% Simulate the Kalman filter.
P_pri = F * P_post * F' + Q;
K = P_pri * H' * inv(H * P_pri * H' + R);
% P[6x6], H[3x6], R[3x3]
%% Prediction phase
xhat_pri = F * xhat_post;
% Innovation squared (d2) a priori:
d2(k) = (z - H * xhat_pri)' * inv(H * P_pri * H' + R) * (z - H * xhat_pri);
% The first innovation term is calculated between k=1 and k=2
%% Correction phase
xhat_post = xhat_pri + K * (z - H * xhat_pri);
P_post = (eye(6) - K * H) * P_pri * (eye(6) - K * H)' + K * R * K';

%% Save data for plotting.
PArray(:,k) = P_post;
xArray = [xArray x];
xhatArray = [xhatArray xhat_post];
yArray = [yArray y'];
zArray = [zArray z];
tArray = [tArray t];
end

for i = 1:size(PArray,3)
    vwx_est(i) = PArray(1,1,i);
    vwy_est(i) = PArray(2,2,i);
    vwz_est(i) = PArray(3,3,i);
    vrx_est(i) = PArray(4,4,i);
    vry_est(i) = PArray(5,5,i);
    vrz_est(i) = PArray(6,6,i);
end

count = 0;
for i = 1:length(d2)
    if d2(i) > chi2lim
        count = count + 1;
    end
end

Performance = 1 - count/length(d2);
display(sprintf('Percentage of innovation samples inside Chi-Square margin: %.2f
%%', 100*Performance))

```

```

%% Plot results
subplot(321);
plot(tArray,xArray(1,:)-xhatArray(1:),'k'); hold on;
plot(tArray,+3*sqrt(vwx_est),'k:'); plot(tArray,-3*sqrt(vwx_est),'k:');
xlabel('t [s]'); ylabel('\omega_x [rad/s]'); title('Estimation error in \omega_x and
3\sigma intervals');
axis([0 tArray(end) -1.3*max(3*sqrt(vwx_est(round(end/2):end))) 1.3*max(3*sqrt(vwx_est(round(end/2):end
)))]
subplot(322);
plot(tArray,xArray(2,:)-xhatArray(2:),'k'); hold on;
plot(tArray,+3*sqrt(vvy_est),'k:'); plot(tArray,-3*sqrt(vvy_est),'k:');
xlabel('t [s]'); ylabel('\omega_y [rad/s]'); title('Estimation error in \omega_y and
3\sigma intervals');
axis([0 tArray(end) -1.3*max(3*sqrt(vvy_est(round(end/2):end))) 1.3*max(3*sqrt(vvy_est(round(end/2):end
)))]
subplot(323);
plot(tArray,xArray(3,:)-xhatArray(3:),'k'); hold on;
plot(tArray,+3*sqrt(vwz_est),'k:'); plot(tArray,-3*sqrt(vwz_est),'k:');
xlabel('t [s]'); ylabel('\omega_z [rad/s]'); title('Estimation error in \omega_z and
3\sigma intervals');
axis([0 tArray(end) -1.3*max(3*sqrt(vwz_est(round(end/2):end))) 1.3*max(3*sqrt(vwz_est(round(end/2):end
)))]
subplot(324);
plot(tArray,xArray(4,:)-xhatArray(4:),'k'); hold on;
plot(tArray,+3*sqrt(vrx_est),'k:'); plot(tArray,-3*sqrt(vrx_est),'k:');
xlabel('t [s]'); ylabel('r_x [m]'); title('Estimation error in r_x and 3\sigma
intervals');
axis([0 tArray(end) -1.1*max(3*sqrt(vrx_est(round(end/2):end))) 1.1*max(3*sqrt(vrx_est(round(end/2):end
))]
subplot(325);
plot(tArray,xArray(5,:)-xhatArray(5:),'k'); hold on;
plot(tArray,+3*sqrt(vry_est),'k:'); plot(tArray,-3*sqrt(vry_est),'k:');
xlabel('t [s]'); ylabel('r_y [m]'); title('Estimation error in r_y and 3\sigma
intervals');
axis([0 tArray(end) -1.1*max(3*sqrt(vry_est(round(end/2):end))) 1.1*max(3*sqrt(vry_est(round(end/2):end
))]
subplot(326);
plot(tArray,xArray(6,:)-xhatArray(6:),'k'); hold on;
plot(tArray,+3*sqrt(vrz_est),'k:'); plot(tArray,-3*sqrt(vrz_est),'k:');
xlabel('t [s]'); ylabel('r_z [m]'); title('Estimation error in r_z and 3\sigma
intervals');
axis([0 tArray(end) -1.1*max(3*sqrt(vrz_est(round(end/2):end))) 1.1*max(3*sqrt(vrz_est(round(end/2):end
))]

figure;
size(tArray)
size(d2)
plot(tArray, d2, 'k',[tArray(1) tArray(end)], chi2lim*[1 1], 'k--'); hold on; xlabel('t [s]'); ylabel('d

```

```

    ^2(k) [m] ');
title('KF: Comparison between Inovation d^2 and Chi-square limit')
legend('d^2', 'Chi-square limit')
axis([0 tArray(end) 0 max(d2)])

figure;
plot(tArray, xhatArray(4,:), 'k');
hold on
plot(tArray, Rcm(1)*ones(size(tArray)), 'k--');
legend('r_x estimated', 'r_x true')
title('r_x estimation when J_{ij}\approx J_{ii}')
xlabel('Time (s)')
ylabel('r_x (m)')
axis([0 max(tArray) min([xhatArray(4,:) xArray(4,:)] max([xhatArray(4,:) xArray(4,:)])])

figure;
plot(tArray, xhatArray(1,:));
hold on;
plot(tArray, xArray(1,:));
hold on
plot(tArray, yArray(4,:));
hold on
plot(tArray, zArray(1,:));
legend('\omega_{x-est}', '\omega_{x-processnoise}', '\omega_{x-true}', '\omega_{x-measured}')

figure;
plot(tArray, xhatArray(4,:));
title('r_x estimated')

set(gca, 'FontSize', 12); set(gcf, 'Color', 'White');
display(sprintf('Estimated r_x = %.10f mm', 10^3*xhat_post(4,end)));
display(sprintf('True r_x = %.10f mm', 10^3*Rcm(1)));
display(sprintf('Estimated r_y = %.10f mm', 10^3*xhat_post(5,end)));
display(sprintf('True r_y = %.10f mm', 10^3*Rcm(2)));
display(sprintf('Estimated r_z = %.10f mm', 10^3*xhat_post(6,end)));
display(sprintf('True r_z = %.10f mm', 10^3*Rcm(3)));
disp(['Elapsed time = ', num2str(toc), ' seconds']);

```

Listing K.4: Kalman Filter

```

function KalmanFilterForBalancing_graphs
% Graphs showing how KF estimates for the unbalance vector
% reacts when specific conditions are tested (e.g. high angular
% velocities, non-diagonal inertia)
% Estimates the 3x1 unknown parameter vector r = [rx ry rz]
% Observation: to change the platform inertia, modify the value of
% J0 in the testbed dynamics file "eom1.m"
close all
clear all

```



```

clc
tf = 50; % simulation length
T = 0.1; % sampling time

%% Testbed characteristics
J = [0.265 0 0; 0 0.246 0; 0 0 0.427]; % change in "eom1.m" also
Rcm = [-1; -1; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity

%% Matrices setup
Q = diag([0.0005 0.0005 0.0005 (0.1*Rcm(1))^2 (0.1*Rcm(2))^2 (0.1*Rcm(3))^2]);
R = [0.05^2 0 0; 0 0.05^2 0; 0 0 0.05^2]; % covariance of measurement noise
H = [eye(3) zeros(3)]; % measurement matrix

%% Initial values
P_post = zeros(6);
x = [0; 0; 0; Rcm(1); Rcm(2); Rcm(3)]; % initial state
xhat_post = [0; 0; 0; 3*1e-3; 0; 0]; % initial state estimate
z = H * x + sqrt(R) * [randn; randn; randn]; % initial measurement

% Initialize arrays for later plotting
PArray = P_post;
zArray = z;
xArray = x;
tArray = 0;
xhatArray = xhat_post;
y = [0 0 0 0 0 Rcm(1) Rcm(2) Rcm(3)];
yArray = y';
phi = y(1);
theta = y(2);
psi = y(3);

tic;
d2 = 0;
for test_number = 1:3
    for t = T : T : tf
        k = round(t/T)+1; % MATLAB initial vector position is 1
        %% Simulate the system.
        % Integrate
        [~, y_dt] = ode23('eom1', [0 T], y(:), 1e-5);
        y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]
        % The intermediate values of the integration with ode23 are transparent
        % for the x state vector.
        phi(k) = y(1);
        theta(k) = y(2);
        psi(k) = y(3);

        Phi_matrix = [0 -m*g*T/(2*J(1,1))*(cos(phi(k))*cos(theta(k))+cos(phi(k-1))*cos(theta(k-1))) m*g*T/(2

```

```

*J(1,1))*(sin(phi(k))*cos(theta(k))+sin(phi(k-1))*cos(theta(k-1)));
    m*g*T/(2*J(2,2))*(cos(phi(k))*cos(theta(k))+cos(phi(k-1))*cos(theta(k-1))) 0 m*g*T/(2*J(2,2))*(sin
(theta(k))+sin(theta(k-1)));
    -m*g*T/(2*J(3,3))*(sin(phi(k))*cos(theta(k))+sin(phi(k-1))*cos(theta(k-1))) -m*g*T/(2*J(3,3))*(sin(
theta(k))+sin(theta(k-1))) 0];
%% Observation: The state vector is x = [wx wy wz rx ry rz]
F = [eye(3) Phi_matrix;
     zeros(3) eye(3)];
x = y(4:9)' + sqrt(Q) * [randn; randn; randn; randn; randn; randn];
z = H * x + sqrt(R) * [randn; randn; randn]; % size(H)=[3 3]; size(x) = [6 1];

%% Simulate the Kalman filter.
P_pri = F * P_post * F' + Q;
K = P_pri * H' * inv(H * P_pri * H' + R);
% P[6x6], H[3x6], R[3x3]
%% Prediction phase
xhat_pri = F * xhat_post;
% Innovation squared (d2) a priori
d2(k) = (z - H * xhat_pri)' * inv(H * P_pri * H' + R) * (z - H * xhat_pri);
% The first innovation term is calculated between k=1 and k=2
%% Correction phase
xhat_post = xhat_pri + K * (z - H * xhat_pri);
P_post = (eye(6) - K * H) * P_pri * (eye(6) - K * H)' + K * R * K';

%% Save data for plotting.
PArray(:,:,k) = P_post;
xArray = [xArray x];
xhatArray = [xhatArray xhat_post];
yArray = [yArray y'];
zArray = [zArray z];
tArray = [tArray t];
end

subplot(3,1,test_number)
plot(tArray, xhatArray(4,:), 'k ')
hold on
plot(tArray, x(4)*ones(size(tArray)), 'k--')
if test_number == 1
    title('r_x component estimation under various conditions')
end
legend('r_x estimated', 'r_x true')
ylabel('r_x (m) ')
axis([0 50 min([xhatArray(4,:) xArray(4,:)] max([xhatArray(4,:) xArray(4,:)]))]
if test_number == 2
    axis([0 50 -30*1e-3 max([xhatArray(4,:) xArray(4,:)]))]
end
if test_number == 3
    axis([0 50 -30*1e-4 10*1e-4])
end
end

```

```

pause(1)
if test_number == 1
    clear z y x xhat_post P_post xArray xhatArray yArray zArray tArray
    P_post = zeros(6);
    x = [0; 0; 0; 20*Rcm(1); 20*Rcm(2); 20*Rcm(3)]; % initial state
    xhat_post = 0 * x; % initial state estimate
    z = H * x + sqrt(R) * [randn; randn; randn]; % initial measurement
    % Initialize arrays for later plotting
    PArray = P_post;
    zArray = z;
    xArray = x;
    tArray = 0;
    xhatArray = xhat_post;
    y = [0 0 0 0 0 20*Rcm(1) 20*Rcm(2) 20*Rcm(3)];
    yArray = y';
    phi = y(1);
    theta = y(2);
    psi = y(3);
else
    clear z y x xhat_post P_post xArray xhatArray yArray zArray tArray
    P_post = zeros(6);
    x = [1; 1; 1; Rcm(1); Rcm(2); Rcm(3)]; % initial state
    xhat_post = 0 * x; % initial state estimate
    z = H * x + sqrt(R) * [randn; randn; randn]; % initial measurement
    % Initialize arrays for later plotting
    PArray = P_post;
    zArray = z;
    xArray = x;
    tArray = 0;
    xhatArray = xhat_post;
    y = [0 0 0 1 1 1 Rcm(1) Rcm(2) Rcm(3)];
    yArray = y';
    phi = y(1);
    theta = y(2);
    psi = y(3);
end
end
xlabel('Time (s)')
disp(['Elapsed time = ', num2str(toc), ' seconds']);

```

Listing K.5: Kalman Filter performance under specific conditions

## K.3 Extended Kalman Filter

```

function EKFForBalancing_3D
% 6-state Extended Kalman filter for unbalance vector estimation.
% In this case the EKF jacobians are determined analytically.

tf = 100; % simulation length

```

```

dt = 0.1; % simulation step size
n = 6; % number of states

%% Testbed characteristics
% (see eom_ekf3d.m)
J = [0.265 0 0; 0 0.246 0; 0 0 0.427];
Jx = J(1,1); Jy = J(2,2); Jz = J(3,3);
Rcm = [-1; -1; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity

%% Filter matrices
R = [0.05^2 0 0; 0 0.05^2 0; 0 0 0.05^2];
%R = eye(3);
H = [eye(3) zeros(3)]; % measurement matrix (OK)

%% Initialization of variables
xstate = [0; 0; 0; Rcm(1); Rcm(2); Rcm(3)]; % initial state
xhatekf = [zeros(3,1); 2*xstate(4:6)]; % initial UKF state estimate
z = H * xstate + sqrt(R) * [randn; randn; randn];
Q = diag([0.000005 0.000005 0.000005 (0.01*Rcm(1))^2 (0.01*Rcm(2))^2 (0.01*Rcm(2))^2]);
% change the initial angular velocity of the simulation in y(4:6)
y = [0 0 0 0 0 Rcm(1) Rcm(2) Rcm(3)];

%% Initialize arrays for later plotting
yArray = y';
zArray = z;
tArray = 0;
xArray = xstate;
dtPlot = dt; % how often to plot output data
xhatekfArray = xhatekf;
P = 1e-5*eye(6);
Pekf = P;
Pekfarray = diag(Pekf);
chi2lim = chi2inv(0.95,3);
d2 = 0;
tic;
for t = dt : dt : tf
    k = round(t/dt)+1;
    tc=round(t/dt); % Time counter
    %% Simulate the system.
    % Integrate
    [~, y_dt] = ode23('eom1', [0 dt], y(:), 1e-5);
    y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]
    xstate(1:6) = y(4:9)' + sqrt(Q) * [randn; randn; randn; randn; randn; randn]; % xstate is a 12x1 column
    vector
    z = H * xstate + sqrt(R) * [randn; randn; randn]; % Sensor simulation with noise

    %% Simulate the Extended Kalman filter.

```

```

%% Step 1: the model must be written in the form
%  $x_k = f_{k-1}(x_{k-1}, u_{k-1}, t_{k-1}) + w_{k-1}$  (additive noise model)
%  $y_k = h(x_k, t_k) + v_k$ 
%  $w_k \sim (0, Q_k)$ 
%  $v_k \sim (0, R_k)$ 

%% Step 2: The EKF must be initialized ( $\hat{x}_0$  posteriori and  $P_0$  posteriori)
% Done
%% Step 3: repeat iteratively the following steps
% (a) compute the f jacobians relative to x and w
sth = sin(y(2)); cth = cos(y(2));
sphi = sin(y(1)); cphi = cos(y(1));
gb = [g*sth; -g*sphi*cth; -g*cphi*cth];
F = [1 xhatekf(3)*((Jy-Jz)/Jx)*dt xhatekf(2)*((Jy-Jz)/Jx)*dt 0 (1/Jx)*m*gb(3)*dt -(1/Jx)*m*gb(2)*dt;
     xhatekf(3)*((Jz-Jx)/Jy)*dt 1 xhatekf(1)*((Jz-Jx)/Jy)*dt -(1/Jy)*m*gb(3)*dt 0 (1/Jy)*m*gb(1)*dt;
     xhatekf(2)*((Jx-Jy)/Jz)*dt xhatekf(1)*((Jx-Jy)/Jz)*dt 1 (1/Jz)*m*gb(2)*dt -(1/Jz)*m*gb(1)*dt 0;
     zeros(3) eye(3)];
L = eye(6);
% (b) Time update
P = F * P * F' + L * Q * L';
xhatekf = [xhatekf(1)+(xhatekf(2)*xhatekf(3)*(Jy-Jz)/Jx + m/Jx*(gb(3)*xhatekf(5)-gb(2)*xhatekf(6)))*dt;
          xhatekf(2)+(xhatekf(1)*xhatekf(3)*(Jz-Jx)/Jy + m/Jy*(-gb(3)*xhatekf(4)+gb(1)*xhatekf(6)))*dt;
          xhatekf(3)+(xhatekf(1)*xhatekf(2)*(Jx-Jy)/Jz + m/Jz*(gb(2)*xhatekf(4)-gb(1)*xhatekf(5)))*dt;
          xhatekf(4);
          xhatekf(5);
          xhatekf(6)];
% inovation = d = zk - Hk*xkhat (xkhat a priori)
d2(k) = (z - H * xhatekf)' * inv(H * P * H' + R) * (z - H * xhatekf); %xhatekf and P * H' -> a priori

% (c) compute the h jacobians relative to x and v
H = [eye(3) zeros(3)]; % the measurement equation is linear
M = eye(3); % v is an additive noise

% (d) Measurement update
K = P * H' * inv(H * P * H' + M * R * M');
xhatekf = xhatekf + K * (z - H * xhatekf);
P = (eye(6) - K * H) * P;

%% Save data for plotting.
yArray(:, tc+1) = y';
PArray(:, :, k) = P;
xArray(:, tc+1) = xstate;
zArray(:, tc+1) = z;
xhatekfArray(:, tc+1) = xhatekf;
Pekfarray(:, tc+1) = diag(P);
tArray = [tArray t];
end
rx_hat_mean = mean(xhatekfArray(4, round(end/2):end))
close all

```

```

t = 0 : dtPlot : tf;
for i = 1:size(PArray,3)
    vwx_est(i) = PArray(1,1,i);
    vwy_est(i) = PArray(2,2,i);
    vwz_est(i) = PArray(3,3,i);
    vrx_est(i) = PArray(4,4,i);
    vry_est(i) = PArray(5,5,i);
    vrz_est(i) = PArray(6,6,i);
end
count = 0;
for i = 1:length(d2)
    if d2(i) > chi2lim
        count = count + 1;
    end
end
Performance = 1 - count/length(d2);
display(sprintf('Percentage of inovation samples inside Chi-Square margin: %.2f
%%',100*Performance))

%% Plot results
figure;
subplot(321);
title('3\sigma intervals in 6-state EKF')
plot(tArray,xArray(1,:)-xhatekfArray(1:),'k'); hold on;
plot(tArray,+3*sqrt(vwx_est),'k:'); plot(tArray,-3*sqrt(vwx_est),'k:');
xlabel('Time [s]'); ylabel('\omega_x [rad/s]'); title('Estimation error in \omega_x
and 3\sigma intervals');
axis([0 100 -0.1 0.1])
subplot(322);
plot(tArray,xArray(2,:)-xhatekfArray(2:),'k'); hold on;
plot(tArray,+3*sqrt(vwy_est),'k:'); plot(tArray,-3*sqrt(vwy_est),'k:');
xlabel('Time [s]'); ylabel('\omega_y [rad/s]'); title('Estimation error in \omega_y
and 3\sigma intervals');
axis([0 100 -0.1 0.1])
subplot(323);
plot(tArray,xArray(3,:)-xhatekfArray(3:),'k'); hold on;
plot(tArray,+3*sqrt(vwz_est),'k:'); plot(tArray,-3*sqrt(vwz_est),'k:');
xlabel('Time [s]'); ylabel('\omega_z [rad/s]'); title('Estimation error in \omega_z
and 3\sigma intervals');
axis([0 100 -0.1 0.1])
subplot(324);
plot(tArray,xArray(4,:)-xhatekfArray(4:),'k'); hold on;
plot(tArray,+3*sqrt(vrx_est),'k:'); plot(tArray,-3*sqrt(vrx_est),'k:');
xlabel('Time [s]'); ylabel('r_x [m]'); title('Estimation error in r_x and 3\sigma
intervals');
axis([0 100 -4*1e-4 4*1e-4])
subplot(325);
plot(tArray,xArray(5,:)-xhatekfArray(5:),'k'); hold on;
plot(tArray,+3*sqrt(vry_est),'k:'); plot(tArray,-3*sqrt(vry_est),'k:');

```

```

xlabel('Time [s]'); ylabel('r_y [m]'); title('Estimation error in r_y and 3\sigma
intervals');
axis([0 100 -4*1e-4 4*1e-4])
subplot(326);
plot(tArray,xArray(6,:)-xhatekfArray(6:),'k'); hold on;
plot(tArray,+3*sqrt(vrz_est),'k:'); plot(tArray,-3*sqrt(vrz_est),'k:');
xlabel('Time [s]'); ylabel('r_z [m]'); title('Estimation error in r_z and 3\sigma
intervals');
axis([0 100 -4*1e-4 4*1e-4])

figure;
plot(t, xhatekfArray(4,:), 'k', 'LineWidth', 2); hold on;
plot(t, xhatekfArray(5,:), 'k'); hold on;
plot(t, xhatekfArray(6,:), 'k:'); hold on;
plot(t, Rcm(1)*ones(size(t,2)), 'k--'); hold on;
plot(t, Rcm(3)*ones(size(t,2)), 'k--');
title('Unbalance vector components estimation in 6-state EKF')
set(gca,'FontSize',12); set(gcf,'Color','White');
legend('r_x estimate (m)','r_y estimate (m)','r_z estimate (m)','r_x true (m)
'),'r_y true (m)','r_z true (m)')
xlabel('Time [s]'); ylabel('r [m]');
axis([0 5 -0.013 0.005])

figure;
plot(tArray, d2,'k',[tArray(1) tArray(end)], chi2lim*[1 1],'k--'); hold on; xlabel('Time [s]'); ylabel(
'd^2(k) [m]');
legend('d^2', 'Chi-square limit')
title('Chi-squared test in 6-state EKF')
axis([0 100 0 30])

display(sprintf('Estimated r_x = %.10f mm', 10^3*xhatekf(4,end)));
display(sprintf('True r_x = %.10f mm', 10^3*Rcm(1)));
display(sprintf('Estimated r_y = %.10f mm', 10^3*xhatekf(5,end)));
display(sprintf('True r_y = %.10f mm', 10^3*Rcm(2)));
display(sprintf('Estimated r_z = %.10f mm', 10^3*xhatekf(6,end)));
display(sprintf('True r_z = %.10f mm', 10^3*Rcm(3)));
disp(['Elapsed time = ', num2str(toc), ' seconds']);

```

Listing K.6: 6-state Extended Kalman Filter (analytic jacobians)

```

function EKFForBalancing_RW_Xu
% 12-state Extended Kalman filter for unbalance vector estimation.
% In this case the EKF jacobians are determined numerically by using
% the Complex Step Differentiation (CSD) method.
% Observations:
% 1 - this method requires an active control torque (e.g. reaction wheels)
% 2 - the "jacobiancsd" function is available at
% https://www.mathworks.com/matlabcentral/fileexchange/18176-complex-step-jacobian
% 3 - This EKF estimates the 9x1 unknown parameter vector x2=[Jij mRi]', i=1:3, j=1:3, ji=ij
% 4 - Adjust xstate and y before the loop! (initial condition)

```

```

tf = 300; % simulation length
dt = 0.1; % simulation step size

%% Testbed characteristics
J = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427]; % inertia tensor
%J = [0.265 0 0; 0 0.246 0; 0 0 0.427];
Rcm = [-1; -1; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity

Q = diag([0.000005 0.000005 0.000005 1e-6*ones(1,3) 1e-7*ones(1,3) (0.01*m*Rcm(1))^2 (0.01*m*Rcm
(2))^2 (0.01*m*Rcm(2))^2]); % process noise covariance
R = [0.005^2 0 0; 0 0.005^2 0; 0 0 0.005^2];
H = [eye(3) zeros(3,9)]; % measurement matrix
P = 1e-4*eye(12);
xstate = [0; 0; 0; J(1,1); J(2,2); J(3,3); J(1,2); J(1,3); J(2,3); m*Rcm(1); m*Rcm(2); m*Rcm(3)]; % initial
state
xhatekf = [1; 1; 1; 2*xstate(4:12)]; % initial state estimate
u = [0; 0; 0];
z = H * xstate + sqrt(R) * [randn; randn; randn];

% Initialize arrays for later plotting
zArray = z;
xArray = xstate;
xhatekfArray = xhatekf;
uArray = u;
PArray = P;
tArray = 0;
dtPlot = dt; % how often to plot output data
y = [0 0 0 0 0 0 Rcm(1) Rcm(2) Rcm(3) 0 0 0];

%% Configure the input signal
uT = 1; % square wave period
A = 1*[0.05; 0.05; 0.05]; % in N.m, the amplitude of the torque signal

chi2lim = chi2inv(0.95,3);
d2 = 0;
tic;
for t = dt : dt : tf
    tc=round(t/dt); % Time counter
    k = round(t/dt)+1;
    %% Simulate the system.
    % Input signal value
    if mod(t, uT) < uT/2
        %u = - A - cross(omega, A*mod(t, uT));
        y(10:12) = A;
    else
        %u = A + cross(omega, A*mod(t, uT));
        y(10:12) = -A;
    end
end

```



```

end

%% Simulate the system.
% Integrate
[~, y_dt] = ode23('eom_ekf3d_aug', [dt*(tc) dt*(tc+1)], y(:), 1e-5);
y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz, hdotx, hdoty, hdotz]
% Gravity vector in body-frame
sth = sin(y(2)); cth = cos(y(2));
sphi = sin(y(1)); cphi = cos(y(1));
gb = [g*sth; -g*sphi*cth; -g*cphi*cth];
xstate(1:3) = y(4:6);
xstate(1:12) = xstate(1:12) + sqrt(Q) * [randn; randn; randn; randn; randn; randn; randn; randn; randn;
    randn; randn; randn]; % xstate is a 12x1 column vector
z = H * xstate + sqrt(R) * [randn; randn; randn]; % size(H)=[3 12]; size(x) = [12 1];

%% Simulate the Kalman filter.
% Input signal value ESTIMATE (uhat - it is dependent on xhatekf(1:3)!)
if mod(t, uT) < uT/2
    uhat = - A - cross(xhatekf(1:3), A*mod(t, uT));
else
    uhat = A + cross(xhatekf(1:3), A*mod(t, uT));
end
% (a) compute the f jacobians relative to x and w
L = eye(12);
% dF/domega jacobian
omegahat = xhatekf(1:3);
Jhat=[xhatekf(4) -xhatekf(7) -xhatekf(8);
-xhatekf(7) xhatekf(5) -xhatekf(9);
-xhatekf(8) -xhatekf(9) xhatekf(6)];
mRhat = xhatekf(10:12);
rhat = xhatekf(10:12)/m;
Jaug = [m*(rhat(2)^2+rhat(3)^2) -m*rhat(1)*rhat(2) -m*rhat(1)*rhat(3);
-m*rhat(1)*rhat(2) m*(rhat(1)^2+rhat(3)^2) -m*rhat(2)*rhat(3);
-m*rhat(1)*rhat(3) -m*rhat(2)*rhat(3) m*(rhat(1)^2+rhat(2)^2)];
if mod(t, uT) < uT/2
    f1=@(param1)inv(Jhat+Jaug)*(-cross(param1,(Jhat+Jaug)*param1)-cross(gb,xhatekf(10:12))+(- A -
    cross(param1, A*mod(t, uT))));
else
    f1=@(param1)inv(Jhat+Jaug)*(-cross(param1,(Jhat+Jaug)*param1)-cross(gb,xhatekf(10:12))+(A +
    cross(param1, A*mod(t, uT))));
end
[F1,omegadothat]=jacobiancsd(f1,xhatekf(1:3));
% dF/dJ jacobian
f2=@(param2)inv([param2(1) -param2(4) -param2(5);
-param2(4) param2(2) -param2(6);
-param2(5) -param2(6) param2(3)]+Jaug)*(-cross(omegahat,[param2(1) -param2(4) -param2(5);
-param2(4) param2(2) -param2(6);
-param2(5) -param2(6) param2(3)]+Jaug)*omegahat)-cross(gb,
mRhat)+uhat);

```

```

[F2,~]=jacobiancsd(f2,xhatekf(4:9));
% dF/d(mR) jacobian
f3=@(param3)inv(Jhat+[m*((param3(2)/m)^2+(param3(3)/m)^2) -m*(param3(1)/m)*(param3(2)/m) -m*(
param3(1)/m)*(param3(3)/m);
    -m*(param3(1)/m)*(param3(2)/m) m*((param3(1)/m)^2+(param3(3)/m)^2) -m*(param3(2)/
m)*(param3(3)/m);
    -m*(param3(1)/m)*(param3(3)/m) -m*(param3(2)/m)*(param3(3)/m) m*((param3(1)/m)
^2+(param3(2)/m)^2)]*...
(-cross(omegahat,(Jhat+[m*((param3(2)/m)^2+(param3(3)/m)^2) -m*(param3(1)/m)*(param3(2)/m) -m
*(param3(1)/m)*(param3(3)/m);
    -m*(param3(1)/m)*(param3(2)/m) m*((param3(1)/m)^2+(param3(3)/m)^2) -m*(param3(2)/
m)*(param3(3)/m);
    -m*(param3(1)/m)*(param3(3)/m) -m*(param3(2)/m)*(param3(3)/m) m*((param3(1)/m)
^2+(param3(2)/m)^2))*omegahat-cross(gb,param3)+uhat);
[F3,~]=jacobiancsd(f3,xhatekf(10:12));
F=[F1 F2 F3; zeros(9,12)] * dt; % multiply by dt!!!
F=F+eye(12);

% (b) Time update
P = F * P * F' + L * Q * L';
xhatekfdot = [omegadot; zeros(6,1); zeros(3,1)]; % Jhatdot and mRdot equals zero!
xhatekf = xhatekf + xhatekfdot * dt;

% inovation = d = zk - Hk*xkhat (xkhat a priori)
%d2(k) = (z - H * xhatekf)' * inv(H * P * H' + R) * (z - H * xhatekf); %xhatekf and Pkf -> a priori
d2(k) = (z - H * xhatekf)' * inv(H * P * H' + R) * (z - H * xhatekf); %xhatekf and Pkf -> a priori
% (c) compute the h jacobians relative to x and v
H = [eye(3) zeros(3,9)]; % the measurement equation is linear
M = eye(3); % v is an additive noise
% (d) Measurement update
K = P * H' * inv(H * P * H' + M * R * M');
xhatekf = xhatekf + K * (z - H * xhatekf);
P = (eye(12) - K * H) * P;
%P = (eye(12)-K*H) * P * (eye(12)-K*H)' + K * R * K';

%% Save data for plotting.
uArray = [uArray u];
xArray = [xArray xstate];
xhatekfArray = [xhatekfArray xhatekf];
zArray = [zArray z];
PArray(:, :, k) = P;
tPlot = t;
tArray = [tArray t];
end

for i = 1:size(PArray,3)
    vwx_est(i) = PArray(1,1,i);
    vwz_est(i) = PArray(2,2,i);
    vwz_est(i) = PArray(3,3,i);

```

```

vJx_est(i) = PArray(4,4,i);
vJy_est(i) = PArray(5,5,i);
vJz_est(i) = PArray(6,6,i);
vJxy_est(i) = PArray(7,7,i);
vJxz_est(i) = PArray(8,8,i);
vJyz_est(i) = PArray(9,9,i);
vmRx_est(i) = PArray(10,10,i);
vmRy_est(i) = PArray(11,11,i);
vmRz_est(i) = PArray(12,12,i);
end

% Plot results
close all
t = 0 : dtPlot : tf;

count = 0;
for i = 1:length(d2)
    if d2(i) > chi2lim
        count = count + 1;
    end
end

Performance = 1 - count/length(d2);
display(sprintf('Percentage of inovation samples inside Chi-Square margin: %.2f
%% ',100*Performance))

figure;
plot(tArray, d2, 'k', [tArray(1) tArray(end)], chi2lim*[1 1], 'k-'); hold on; xlabel('Time [s]'); ylabel('
d^2 (k) [m]');
title('Chi-squared test in 12-state EKF')
axis([0 tArray(end) 0 30])

figure;
% Plot results
subplot(621);
title('3\sigma intervals in 12-state EKF')
plot(tArray,xArray(1,:)-xhatekfArray(1,:),'k'); hold on;
plot(tArray,+3*sqrt(vwx_est),'k:'); plot(tArray,-3*sqrt(vwx_est),'k:');
xlabel('Time [s]'); ylabel('\omega_x [rad/s]'); title('Estimation error in \omega_x
and 3\sigma intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(622);
plot(tArray,xArray(2,:)-xhatekfArray(2,:),'k'); hold on;
plot(tArray,+3*sqrt(vwy_est),'k:'); plot(tArray,-3*sqrt(vwy_est),'k:');
xlabel('Time [s]'); ylabel('\omega_y [rad/s]'); title('Estimation error in \omega_y
and 3\sigma intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(623);
plot(tArray,xArray(3,:)-xhatekfArray(3,:),'k'); hold on;
plot(tArray,+3*sqrt(vwz_est),'k:'); plot(tArray,-3*sqrt(vwz_est),'k:');

```

```

xlabel('Time [s]'); ylabel('\omega_z [rad/s]'); title('Estimation error in \omega_z
and 3\sigma intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(624);
plot(tArray,J(1,1)*ones(size(xArray(4,:)))-xhatekfArray(4:),'k'); hold on;
plot(tArray,+3*sqrt(vJx_est),'k:'); plot(tArray,-3*sqrt(vJx_est),'k:');
xlabel('Time [s]'); ylabel('J_x [m]'); title('Estimation error in J_x and 3\sigma
intervals');
axis([0 tArray(end) -0.04 0.04])
subplot(625);
plot(tArray,J(2,2)*ones(size(xArray(5,:)))-xhatekfArray(5:),'k'); hold on;
plot(tArray,+3*sqrt(vJy_est),'k:'); plot(tArray,-3*sqrt(vJy_est),'k:');
xlabel('Time [s]'); ylabel('J_y [m]'); title('Estimation error in J_y and 3\sigma
intervals');
axis([0 tArray(end) -0.04 0.04])
subplot(626);
plot(tArray,J(3,3)*ones(size(xArray(6,:)))-xhatekfArray(6:),'k'); hold on;
plot(tArray,+3*sqrt(vJz_est),'k:'); plot(tArray,-3*sqrt(vJz_est),'k:');
xlabel('Time [s]'); ylabel('J_z [m]'); title('Estimation error in J_z and 3\sigma
intervals');
axis([0 tArray(end) -0.06 0.06])
subplot(627);
plot(tArray,-J(1,2)*ones(size(xArray(7,:)))-xhatekfArray(7:),'k'); hold on;
plot(tArray,+3*sqrt(vJxy_est),'k:'); plot(tArray,-3*sqrt(vJxy_est),'k:');
xlabel('Time [s]'); ylabel('J_{xy}'); title('Estimation error in J_{xy} and 3\sigma
intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(628);
plot(tArray,-J(1,3)*ones(size(xArray(8,:)))-xhatekfArray(8:),'k'); hold on;
plot(tArray,+3*sqrt(vJxz_est),'k:'); plot(tArray,-3*sqrt(vJxz_est),'k:');
xlabel('Time [s]'); ylabel('J_{xz}'); title('Estimation error in J_{xz} and 3\sigma
intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(629);
plot(tArray,-J(2,3)*ones(size(xArray(9,:)))-xhatekfArray(9:),'k'); hold on;
plot(tArray,+3*sqrt(vJyz_est),'k:'); plot(tArray,-3*sqrt(vJyz_est),'k:');
xlabel('Time [s]'); ylabel('J_{yz}'); title('Estimation error in J_{yz} and 3\sigma
intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(6,2,10);
plot(tArray,m*Rcm(1)*ones(size(xArray(10,:)))-xhatekfArray(10:),'k'); hold on;
plot(tArray,+3*sqrt(vmRx_est),'k:'); plot(tArray,-3*sqrt(vmRx_est),'k:');
xlabel('Time [s]'); ylabel('mR_x'); title('Estimation error in mR_x and 3\sigma
intervals');
axis([0 tArray(end) -3*1e-3 3*1e-3])
subplot(6,2,11);
plot(tArray,m*Rcm(2)*ones(size(xArray(11,:)))-xhatekfArray(11:),'k'); hold on;
plot(tArray,+3*sqrt(vmRy_est),'k:'); plot(tArray,-3*sqrt(vmRy_est),'k:');
xlabel('Time [s]'); ylabel('mR_y'); title('Estimation error in mR_y and 3\sigma

```

```

    intervals');
axis([0 tArray(end) -3*1e-3 3*1e-3])
subplot(6,2,12);
plot(tArray,m*Rcm(3)*ones(size(xArray(12,:)))-xhatekfArray(12,:), 'k'); hold on;
plot(tArray,+3*sqrt(vmRz_est), 'k: '); plot(tArray,-3*sqrt(vmRz_est), 'k: ');
xlabel('Time [s]'); ylabel('mR_z'); title('Estimation error in mR_z and 3\sigma
intervals');
axis([0 tArray(end) -7*1e-3 7*1e-3])

figure;
plot(t, xArray(1:3,:));
hold on
plot(t, xhatekfArray(1:3,:));
set(gca, 'FontSize',12); set(gcf, 'Color', 'White');
xlabel('Seconds'); ylabel('\omega (in rad/s)');

figure;
subplot(1,1,1)
title('Convergence of the parameters in 12-state EKF (normal angular rates)')
)
subplot(2,2,2)
plot([0 t(end)], [J(1,1) J(1,1); J(2,2) J(2,2); J(3,3) J(3,3)], 'k--')
hold on
plot(t, xhatekfArray(4,:), 'k: '); hold on;
plot(t, xhatekfArray(5,:), 'k'); hold on;
plot(t, xhatekfArray(6,:), 'k', 'LineWidth',3);
set(gca, 'FontSize',12); set(gcf, 'Color', 'White');
xlabel('Seconds'); ylabel('Principal Inertia Moments');
legend('Jx', 'Jy', 'Jz', 'Jxhat', 'Jyhat', 'Jzhat')
subplot(2,2,4)
plot([0 t(end)], [-J(1,2) -J(1,2); -J(1,3) -J(1,3); -J(2,3) -J(2,3)], 'k--')
hold on
plot(t, xhatekfArray(7,:), 'k: '); hold on;
plot(t, xhatekfArray(8,:), 'k'); hold on;
plot(t, xhatekfArray(9,:), 'k', 'LineWidth',3);
set(gca, 'FontSize',12); set(gcf, 'Color', 'White');
xlabel('Seconds'); ylabel('Inertia Products');
legend('Jxy', 'Jxz', 'Jyz', 'Jxyhat', 'Jxzhat', 'Jyzhat')
subplot(2,2,[1 3])
plot([0 t(end)], [m*Rcm(1) m*Rcm(1); m*Rcm(2) m*Rcm(2); m*Rcm(3) m*Rcm(3)];, 'k--')
hold on
plot(t, xhatekfArray(10,:), 'k: '); hold on;
plot(t, xhatekfArray(11,:), 'k'); hold on;
plot(t, xhatekfArray(12,:), 'k', 'LineWidth',3);
set(gca, 'FontSize',12); set(gcf, 'Color', 'White');
xlabel('Seconds'); ylabel('mR');
legend('mRx', 'mRy', 'mRz', 'mRxhat', 'mRyhat', 'mRzhat')

display(m*Rcm)

```

```

time=clock;
save(['EKF-Data-day-25_' num2str(time(4)) '-' num2str(time(5)) '.mat']);
disp(['Final estimation error = ', num2str(xArray(4,end)-xhatekfArray(4,end))]);
disp(['Elapsed time = ', num2str(toc), ' seconds!']);

```

Listing K.7: 12-state Extended Kalman Filter (jacobians calculated through Complex Step Differentiation)

```

function ydot = eom_ekf3d_aug(t,y)
% This dynamic model is augmented for
% considering the input torque given
% by hdotx, hdoty and hdotz.
% y = [phi, theta, psi, wx, wy, wz, rx, ry, rz, hdotx, hdoty, hdotz]
ydot = zeros(12,1);
uT = 1;% square wave period
% state-space dynamics: xdot = f(x) + d(t), d(t) process noise.
% output equation: y = H*x + v(t), v(t) measurement noise.
Rcm = [y(7); y(8); y(9)];
%J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];
J0 = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427];
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity
Jaug = [m*(Rcm(2)^2+Rcm(3)^2) -m*Rcm(1)*Rcm(2) -m*Rcm(1)*Rcm(3);
        -m*Rcm(1)*Rcm(2) m*(Rcm(1)^2+Rcm(3)^2) -m*Rcm(2)*Rcm(3);
        -m*Rcm(1)*Rcm(3) -m*Rcm(2)*Rcm(3) m*(Rcm(1)^2+Rcm(2)^2)];
J = J0 + Jaug;
%J = J0;
hdot = [y(10); y(11); y(12)];
omega = [y(4); y(5); y(6)];
sphi = sin(y(1));
cphi = cos(y(1));
sth = sin(y(2));
cth = cos(y(2));

%% Model equations
% Kinematic equation
% Euler rates
eulerdot = [1 sphi*tan(y(2)) cphi*tan(y(2));
            0 cphi -sphi;
            0 sphi/cth cphi/cth]*omega;
% Gravity vector in body-frame
gb = [g*sth; -g*sphi*cth; -g*cphi*cth];
% Dynamics equation
omegadot=J\(-hdot-cross(omega,hdot*mod(t, uT))-cross(omega,J*omega)+cross(Rcm,gb*m)); % hdot is
assumed constant

%% Derivatives (xdot)
% Omega(dot)
ydot(1:3) = eulerdot;
% Euler angles (dot)

```

```
ydot(4:6) = omegadot;
```

Listing K.8: Dynamic model used in the 12-state EKF

## K.4 Unscented Kalman Filter

```
function UKFForBalancing_3D_quat
% Unscented Kalman filter for estimating the unbalance
% vector of the platform.

tf = 50; % simulation length
dt = 0.01; % simulation step size
n=6; % number of states

%% Testbed characteristics
% (see eom_ukf_quat.m)
%J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];
J0 = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427];
Rcm = 20*[-1; -3; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity

%% Filter matrices
R = 1*diag([0.0026^2 0.0049^2 0.0031^2]); % covariance of measurement noise (OK)
H = [eye(3) zeros(3)]; % measurement matrix (OK)

%% Initialization of variables
xstate = [0.2*ones(3,1); Rcm(1); Rcm(2); Rcm(3)]; % initial state
xhatukf = 2 * xstate; % initial UKF state estimate
z = H * xstate + sqrt(R) * [randn; randn; randn];
W = ones(2*n,1)/(2*n); % UKF weights (in this case, equal weights for all sigma points)
Q = 1*diag([0.05 0.05 0.05 1e-7*ones(1,3)]); % process noise covariance
y = [1 0 0 0 10 10 10 Rcm(1) Rcm(2) Rcm(3)];

%% Initialize arrays for later plotting
yArray = y';
zArray = z;
tArray = 0;
zhatArray = z;
xArray = xstate;
dtPlot = dt; % how often to plot output data
xhatukfArray = xhatukf;
P = 1e-5*eye(6);
Pukf = P;
Parray = diag(P);
Pukfarray = diag(Pukf);
chi2lim = chi2inv(0.95,3);
d2 = 0;
tic;
```

```

for t = dt : dt : tf
    k = round(t/dt)+1;
    tc=round(t/dt); % Time counter
    %% Simulate the system.
    % Integrate
    [~, y_dt] = ode23('eom_ukf_quat', [0 dt], y(:), 1e-5);
    y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rx, ry, rz]
    xstate(1:6) = y(5:10)' + sqrt(Q) * [randn; randn; randn; randn; randn; randn]; % xstate is a 12x1 column
        vector
    % Sensor simulation with noise
    z = H * xstate + sqrt(R) * [randn; randn; randn]; % size(H)=[3 4]; size(x) = [4 1];

    %% Simulate the Unscented Kalman filter.
    %% Step 1: the model must be written in the form
    %  $x_{k+1} = f(x_k, u_k, t_k) + w_k$  (additive noise model)
    %  $y_k = h(x_k, t_k) + v_k$ 
    %  $w_k \sim (0, Q_k)$ 
    %  $v_k \sim (0, R_k)$ 

    %% Step 2: The UKF must be initialized ( $\hat{x}_{0|0}$  and  $P_{0|0}$ )
    % Done

    %% Step 3: Time update
    [root,~] = chol(n*Pukf); %n*P
    for i = 1 : n %1:n
        sigma(:,i) = xhatukf + root(i,:);
        sigma(:,i+n) = xhatukf - root(i,:);
    end
    for i = 1 : 2*n %1:2n
        xbreve(:,i) = sigma(:,i);
    end

    %% Step 3(b): Use the known f(.) non linear function to transform the
    % sigma points into  $\hat{x}_k(i)$  vectors.
    for i = 1:2*n
        xbrevedot = eom_ukf_quat(t,[y(1:4)'; xbreve(1:6,i)]);
        xbreve(:,i) = xbreve(:,i) + [xbrevedot(5:7); 0; 0; 0] * dt;
    end

    %% Step 3(c): Combine the  $\hat{x}_k(i)$  vectors to obtain the a priori state
    % estimate at time k.
    xhatukf = zeros(n,1);
    for i = 1 : 2*n % i=1:2n
        xhatukf = xhatukf + W(i) * xbreve(:,i); % W is the wheight function (see beginning of file)
    end

    %% Step 3(d): Estimate a priori error covariance.
    Pukf = zeros(n,n);
    for i = 1 : 2*n %i=1:2n

```



```

    Pukf = Pukf + W(i) * (xbreve(:,i) - xhatukf) * (xbreve(:,i) - xhatukf)';
end
Pukf = Pukf + Q;
d2(k) = (z - H * xhatukf)' * inv(H * Pukf * H' + R) * (z - H * xhatukf); %xhatukf and Pukf -> a priori

%% Step 4: Measurement update
%% Step 4(a): Choose sigma points x_k(i).
% Start of optional step (comment lines below if you want)
[root,~] = chol(n*Pukf); %n=15
for i = 1 : n %1:n
    sigma(:,i) = xhatukf + root(i,:);
    sigma(:,i+n) = xhatukf - root(i,:);
end
for i = 1 : 2*n %1:2n
    xbreve(:,i) = sigma(:,i);
end
% End of optional step (comment lines above if you want)

%% Step 4(b): Apply the nonlinear measurement equation to the sigma
% points. In our case, the measurement equation is linear.
for i = 1 : 2*n %i=1:2n
    zukf(:,i) = H*xbreve(:,i);
end

%% Step 4(c): Combine the yhat_k(i) vectors to obtain predicted
% measurement at time k.
zhat = zeros(3,1);
for i = 1 : 2*n %i=1:2n
    zhat = zhat + W(i) * zukf(:,i);
end

%% Step 4(d): Estimate the covariance of the predicted measurement.
% Obs.: Rk has to be added.
% and Step 4(e): estimate the cross covariance between xhat_k_priori
% and yhat_k
Py = zeros(3,3);
Pxy = zeros(n,3);
for i = 1 : 2*n %i=1:2n
    Py = Py + W(i) * (zukf(:,i) - zhat) * (zukf(:,i) - zhat)';
    Pxy = Pxy + W(i) * (xbreve(:,i) - xhatukf) * (zukf(:,i) - zhat)';
end
Py = Py + R;

%% Step 4(f): The measurement update may be performed using the standard
% Kalman filter equations.
Kukf = Pxy * inv(Py);
xhatukf = xhatukf + Kukf * (z - zhat);
Pukf = Pukf - Kukf * Py * Kukf';

```

```

%% Save data for plotting.
yArray(:, tc+1) = y';
PArray(:,k) = Pukf;
xArray(:,tc+1) = xstate;
zArray(:,tc+1) = z;
zhatArray(:,tc+1) = zhat;
xhatukfArray(:,tc+1) = xhatukf;
Parray(:,tc+1) = diag(P);
Pukfarray(:,tc+1) = diag(Pukf);
tArray = [tArray t];
end
rx_hat_mean = mean(xhatukfArray(4,round(end/2):end))
% Plot results
close all
t = 0 : dtPlot : tf;

for i = 1:size(PArray,3)
    vwx_est(i) = PArray(1,1,i);
    vwy_est(i) = PArray(2,2,i);
    vwz_est(i) = PArray(3,3,i);
    vrx_est(i) = PArray(4,4,i);
    vry_est(i) = PArray(4,4,i);
    vrz_est(i) = PArray(4,4,i);
end

count = 0;
for i = 1:length(d2)
    if d2(i) > chi2lim
        count = count + 1;
    end
end

Performance = 1 - count/length(d2);
display(sprintf('Percentage of inovation samples inside Chi-Square margin: %.2f
%% ',100*Performance))

figure;
plot(tArray, yArray(5:7,:));
set(gca, 'FontSize',12); set(gcf, 'Color','White');
xlabel('Seconds'); ylabel('\omega (in rad/s)');
title('y (1) ');

figure;
% Plot results
subplot(421);
plot(tArray,xArray(1,:)-xhatukfArray(1,:), 'k '); hold on;
plot(tArray,+3*sqrt(vwx_est), 'k '); plot(tArray,-3*sqrt(vwx_est), 'k ');
xlabel('t [s]'); ylabel('\omega_x [rad/s]'); title('Estimation error in \omega_x and
3\sigma intervals');

```

```

axis([0 tArray(end) -0.02 0.02])
subplot(422);
plot(tArray,xArray(2,:)-xhatukfArray(2,:), 'k '); hold on;
plot(tArray,+3*sqrt(vwy_est), 'k: '); plot(tArray,-3*sqrt(vwy_est), 'k: ');
xlabel('t [s]'); ylabel('\omega_y [rad/s]'); title('Estimation error in \omega_y and
3\sigma intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(423);
plot(tArray,xArray(3,:)-xhatukfArray(3,:), 'k '); hold on;
plot(tArray,+3*sqrt(vwz_est), 'k: '); plot(tArray,-3*sqrt(vwz_est), 'k: ');
xlabel('t [s]'); ylabel('\omega_z [rad/s]'); title('Estimation error in \omega_z and
3\sigma intervals');
axis([0 tArray(end) -0.02 0.02])
subplot(424);
plot(tArray,xArray(4,:)-xhatukfArray(4,:), 'k '); hold on;
plot(tArray,+3*sqrt(vrz_est), 'k: '); plot(tArray,-3*sqrt(vrz_est), 'k: ');
xlabel('t [s]'); ylabel('r_x [m]'); title('Estimation error in r_x and 3\sigma
intervals');
%axis([0 tArray(end) -1e-3 1e-3])
subplot(425);
plot(tArray,xArray(5,:)-xhatukfArray(5,:), 'k '); hold on;
plot(tArray,+3*sqrt(vrz_est), 'k: '); plot(tArray,-3*sqrt(vrz_est), 'k: ');
xlabel('t [s]'); ylabel('r_y [m]'); title('Estimation error in r_y and 3\sigma
intervals');
%axis([0 tArray(end) -1e-3 1e-3])
subplot(426);
plot(tArray,xArray(6,:)-xhatukfArray(6,:), 'k '); hold on;
plot(tArray,+3*sqrt(vrz_est), 'k: '); plot(tArray,-3*sqrt(vrz_est), 'k: ');
xlabel('t [s]'); ylabel('r_z [m]'); title('Estimation error in r_z and 3\sigma
intervals');
%axis([0 tArray(end) -1e-3 1e-3])
subplot(4,2,[7 8])
plot(tArray, d2, 'k', [tArray(1) tArray(end)], chi2lim*[1 1], 'k--'); hold on; xlabel('t [s]'); ylabel('d
^2(k) [m]');
axis([0 tArray(end) 0 30])
title('Chi-squared test')

figure;
plot([0 t(end)], [Rcm(1) Rcm(1)], 'k--'); hold on;
plot([0 t(end)], [Rcm(2) Rcm(2)], 'k--'); hold on;
plot([0 t(end)], [Rcm(3) Rcm(3)], 'k--'); hold on;
hold on
plot(t, xhatukfArray(4,:), 'k ');
hold on
plot(t, xhatukfArray(5,:), 'k: ');
hold on
plot(t, xhatukfArray(6,:), 'k', 'LineWidth', 3);
set(gca, 'FontSize', 12); set(gcf, 'Color', 'White');
xlabel('Time [s]'); ylabel('r [m]');

```

```

title('UKF: unbalance vector components estimation')
legend('r_x', 'r_y', 'r_z', 'r_xhat', 'r_yhat', 'r_zhat')
%axis([0 t(end) -0.015 0.005])

scrsz = get(groot, 'ScreenSize');
figure('OuterPosition',[0 scrsz(4)/2 scrsz(3)/3 scrsz(4)/2])
%[(distance of left border from left margin of screen);
% (distance of bottom border from bottom of screen);
% width; height]'.
plot(t, xArray(1:3,:));
set(gca, 'FontSize',12); set(gcf, 'Color','White');
xlabel('Seconds'); ylabel('\omega (in rad/s)');

figure('OuterPosition',[scrsz(3)/3 scrsz(4)/2 scrsz(3)/3 scrsz(4)/2])
plot(t, xhatukfArray(1:3,:));
set(gca, 'FontSize',12); set(gcf, 'Color','White');
legend('\omega_x (rad/s)', '\omega_y (rad/s)', '\omega_z (rad/s)')
xlabel('Seconds'); ylabel('xhat');

figure('OuterPosition',[scrsz(3)/3 0 scrsz(3)/3 scrsz(4)/2])
plot(t, xhatukfArray(4,:));
set(gca, 'FontSize',12); set(gcf, 'Color','White');
legend('r_x (m)')
xlabel('Seconds'); ylabel('xhat');

figure('OuterPosition',[2*scrsz(3)/3 scrsz(4)/2 scrsz(3)/3 scrsz(4)/2])
plot(t, zArray(1:3,:));
hold on
plot(t, zhatArray(1:3,:));
legend('z', 'zhat')
set(gca, 'FontSize',12); set(gcf, 'Color','White');
xlabel('Seconds'); ylabel('\omega (in rad/s)');

figure('OuterPosition',[0 0 scrsz(3)/3 scrsz(4)/2])
plot(t, Pukfarray(4,:));
set(gca, 'FontSize',12); set(gcf, 'Color','White');
xlabel('Seconds'); ylabel('P_{44} (in rad/s)');

figure;
plot(tArray, d2, 'b', [tArray(1) tArray(end)], chi2lim*[1 1], 'r'); hold on; xlabel('t [s]'); ylabel('d^2 (k) [m]');

%% FFT analysis
figure;
fs = 1/dt;
nsamples = length(xhatukfArray(1,:));
freq = (0:nsamples-1)*(fs/nsamples);
freq0 = (-nsamples/2:nsamples/2-1)*(fs/nsamples);
y1 = fft(xhatukfArray(1,:)); power1 = abs(y1).^2/nsamples; y10 = fftshift(y1); power10 = abs(y10).^2/

```

```

    nsamples;
y2 = fft(xhatukfArray(2,:)); power2 = abs(y2).^2/nsamples; y20 = fftshift(y2); power20 = abs(y20).^2/
    nsamples;
y3 = fft(xhatukfArray(3,:)); power3 = abs(y3).^2/nsamples; y30 = fftshift(y3); power30 = abs(y30).^2/
    nsamples;
%plot(freq,power1); hold on; plot(freq,power2); hold on; plot(freq,power3)
plot(freq0,power10); hold on; plot(freq0,power20); hold on; plot(freq0,power30)
xlabel('Frequency')
ylabel('Power')

display(sprintf('Estimated r_x = %.10f mm', 10^3*xhatukf(4,end)));
display(sprintf('True r_x = %.10f mm', 10^3*Rcm(1)));
display(sprintf('Estimated r_y = %.10f mm', 10^3*xhatukf(5,end)));
display(sprintf('True r_y = %.10f mm', 10^3*Rcm(2)));
display(sprintf('Estimated r_z = %.10f mm', 10^3*xhatukf(6,end)));
display(sprintf('True r_z = %.10f mm', 10^3*Rcm(3)));
disp(['Elapsed time = ', num2str(toc), ' seconds!']);

```

Listing K.9: 6-state Unscented Kalman Filter

```

function ydot = eom_ukf_quat(t,y)
% y = [q1, q2, q3, q4, wx, wy, wz, rx, ry, rz]
ydot = zeros(10,1);
q = [y(1); y(2); y(3); y(4)];
omega = [y(5); y(6); y(7)];
Rcm = [y(8); y(9); y(10)];

% Testbed characteristics
J0 = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427];
%J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];
%J0 = eye(3);
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity
Jaug = [m*(Rcm(2)^2+Rcm(3)^2) -m*Rcm(1)*Rcm(2) -m*Rcm(1)*Rcm(3);
        -m*Rcm(1)*Rcm(2) m*(Rcm(1)^2+Rcm(3)^2) -m*Rcm(2)*Rcm(3);
        -m*Rcm(1)*Rcm(3) -m*Rcm(2)*Rcm(3) m*(Rcm(1)^2+Rcm(2)^2)];
%Jaug=0;
J = J0 + Jaug;

%% Model equations
% Kinematic equation
% Quaternion rates
qdot = 0.5*[0 -omega(1) -omega(2) -omega(3);
            omega(1) 0 omega(3) -omega(2);
            omega(2) -omega(3) 0 omega(1);
            omega(3) omega(2) -omega(1) 0]*q;

% Gravity vector in body-frame
Rbi = [2*q(1)^2-1+2*q(2)^2    2*q(2)*q(3)-2*q(1)*q(4)    2*q(2)*q(4)+2*q(1)*q(3);
        2*q(2)*q(3)+2*q(1)*q(4)    2*q(1)^2-1+2*q(3)^2    2*q(3)*q(4)-2*q(1)*q(2);

```

```

    2*q(2)*q(4)-2*q(1)*q(3)  2*q(3)*q(4)+2*q(1)*q(2)  2*q(1)^2-1+2*q(4)^2];
    gb = Rbi' * [0; 0; -g];

    %% Dynamics equation
    omegadot=J\(-cross(omega,J*omega)+cross(Rcm,gb*m));

%% Derivatives (xdot)
% qdot
ydot(1:4) = qdot;
% Euler angles (dot)
ydot(5:7) = omegadot;

```

Listing K.10: Equations of motion used in the 6-state UKF

## K.5 Adaptive Control Simulation

```

function ChesiBalancing
% Adaptive control method for balancing the transverse plane
% components of the unbalance vector developed by Chesi.

tf = 30; % simulation length
dt = 0.01; % simulation step size

%% Testbed characteristics
% (see eom_chesi.m)
%J = [0.265 0 0; 0 0.246 0; 0 0 0.427];
J = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427];
Rcm = [-1; -2; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed, in kg
m_mmu = 0.7; % mass of one Movable Mass Unit (MMU), in kg
g = 9.78; % in m/s^2, local gravity
kp = 0.5; % Gain constant in the controller
v_mmu = 0.001; % velocity of the MMUs (1mm/s approx.)

%% Initialize arrays for later plotting
tArray = 0;
omegaArray = zeros(3,1);
r_mmusArray = zeros(3,1);
ctrltorqueArray = zeros(3,1);
r_estArray = zeros(3,1);

%% Initialization of model initial conditions
y = [[1 zeros(1,3)] [0 0 0] Rcm' zeros(1,3) zeros(1,3)];

tic;
for t = dt : dt : tf
    tc=round(t/dt); % Time counter

    %% Simulation of the system model

```

```

[~, y_dt] = ode23('eom_chesi', [0 dt], y(:), 1e-5);
y(:)=y_dt(size(y_dt,1),:); % y = [q omega r r_est, r_mmu]
%% MMUs operation
omega = [y(5); y(6); y(7)];
% gb cross operator at end of time step
q = [y(1); y(2); y(3); y(4)];
Rbi = [2*q(1)^2-1+2*q(2)^2    2*q(2)*q(3)-2*q(1)*q(4)    2*q(2)*q(4)+2*q(1)*q(3);
       2*q(2)*q(3)+2*q(1)*q(4) 2*q(1)^2-1+2*q(3)^2    2*q(3)*q(4)-2*q(1)*q(2);
       2*q(2)*q(4)-2*q(1)*q(3) 2*q(3)*q(4)+2*q(1)*q(2) 2*q(1)^2-1+2*q(4)^2];
gb = Rbi * [0; 0; -g];
gbcross = [0 -gb(3) gb(2);
           gb(3) 0 -gb(1);
           -gb(2) gb(1) 0];
% control torque at end of time step
r_est = [y(11); y(12); y(13)];
Phi = -m*gbcross;
P = eye(3) - gb*gb'/norm(gb)^2;
omegap = P*omega;
control_torque = -Phi*r_est-kp*omegap;

% Determine the new r_mmus vector. Dynamics of the MMUs are not considered,
% which means r_mmus changes instantaneously
r_mmus = gbcross*control_torque/(norm(gb)^2*m_mmu);
for mmu_counter=1:3
    y(13+mmu_counter) = r_mmus(mmu_counter);
end

%% Save data for plotting.
%xArray(:,tc+1) = xstate;
tArray = [tArray t];
omegaArray = [omegaArray omega];
r_mmusArray = [r_mmusArray r_mmus];
r_estArray = [r_estArray r_est];
ctrltorqueArray = [ctrltorqueArray control_torque];
end

%% Plot results
figure;
plot(tArray, omegaArray(1,:));
hold on
plot(tArray, omegaArray(2,:));
hold on
plot(tArray, omegaArray(3,:));
title('Angular velocities of the testbed')
legend('\omega_x', '\omega_y', '\omega_z')
xlabel('Time (s)')
ylabel('Angular velocities [rad/s]')
figure;

```

```

plot(tArray, r_mmusArray(1,:));
hold on
plot(tArray, r_mmusArray(2,:));
hold on
plot(tArray, r_mmusArray(3,:));
title('Balance masses positions')
legend('r_{mmux}', 'r_{mmuy}', 'r_{mmuz}')
xlabel('Time (s)')
ylabel('Positions [m]')

figure;
plot(tArray, ctrltorqueArray(1,:));
hold on
plot(tArray, ctrltorqueArray(2,:));
hold on
plot(tArray, ctrltorqueArray(3,:));
title('Control torque components')
legend('\tau_{x}', '\tau_{y}', '\tau_{z}')
xlabel('Time (s)')
ylabel('Torques [N\cdot m]')

figure;
plot(tArray, r_estArray(1,:));
hold on
plot(tArray, r_estArray(2,:));
hold on
plot(tArray, r_estArray(3,:));
title('Estimated unbalance vector components')
legend('r_{x-estimated}', 'r_{y-estimated}', 'r_{z-estimated}')
xlabel('Time (s)')
ylabel('Length [m]')
disp(['Elapsed time = ', num2str(toc), ' seconds']);
end

```

Listing K.11: Adaptive Control scheme

```

function ydot = eom_chesi(t,y)
% Equations of motion developed in the work of Chesi.
% y = [q1, q2, q3, q4, wx, wy, wz, rx, ry, rz, rx_est, ry_est, rz_est, r_mmux, r_mmuy, r_mmuz]
kp = 0.5; % Control gain
ydot = zeros(16,1);
dt = 0.01;
v_mmu = 0.001; % 1mm/s

q = [y(1); y(2); y(3); y(4)];
omega = [y(5); y(6); y(7)];
Rcm = [y(8); y(9); y(10)];
r_est = [y(11); y(12); y(13)];
MMUx = [y(14); -0.22; 0];
MMUy = [0.22; y(15); 0];

```



```

MMUz = [-0.22; 0; y(16)];

% Testbed characteristics
J0 = [0.265 -0.014 -0.035; -0.014 0.246 -0.018; -0.035 -0.018 0.427];
%J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];
%J0 = eye(3);
m = 14.307; % mass of the testbed
m_mmu = 0.7; % mass of movable part of a MMU
g = 9.78; % in m/s^2, local gravity
Jaug = [m*(Rcm(2)^2+Rcm(3)^2) -m*Rcm(1)*Rcm(2) -m*Rcm(1)*Rcm(3);
        -m*Rcm(1)*Rcm(2) m*(Rcm(1)^2+Rcm(3)^2) -m*Rcm(2)*Rcm(3);
        -m*Rcm(1)*Rcm(3) -m*Rcm(2)*Rcm(3) m*(Rcm(1)^2+Rcm(2)^2)];
%Jaug=0;
J = J0 + Jaug;

% Update of inertia tensor
MMUxCross = [0 -MMUx(3) MMUx(2);
             MMUx(3) 0 -MMUx(1);
             -MMUx(2) MMUx(1) 0];
MMUyCross = [0 -MMUy(3) MMUy(2);
             MMUy(3) 0 -MMUy(1);
             -MMUy(2) MMUy(1) 0];
MMUzCross = [0 -MMUz(3) MMUz(2);
             MMUz(3) 0 -MMUz(1);
             -MMUz(2) MMUz(1) 0];
J = J - m_mmu * (MMUxCross*MMUxCross + MMUyCross*MMUyCross + MMUzCross*MMUzCross);

%% Model equations
% Kinematic equation
% Quaternion rates
qdot = 0.5*[0 -omega(1) -omega(2) -omega(3);
           omega(1) 0 omega(3) -omega(2);
           omega(2) -omega(3) 0 omega(1);
           omega(3) omega(2) -omega(1) 0]*q;

% Gravity vector in body-frame
Rbi = [2*q(1)^2-1+2*q(2)^2 2*q(2)*q(3)-2*q(1)*q(4) 2*q(2)*q(4)+2*q(1)*q(3);
       2*q(2)*q(3)+2*q(1)*q(4) 2*q(1)^2-1+2*q(3)^2 2*q(3)*q(4)-2*q(1)*q(2);
       2*q(2)*q(4)-2*q(1)*q(3) 2*q(3)*q(4)+2*q(1)*q(2) 2*q(1)^2-1+2*q(4)^2];
gb = Rbi' * [0; 0; -g];

% Projection operator
P = eye(3) - gb*gb'/norm(gb)^2;
omegap = P*omega;
gbcross = [0 -gb(3) gb(2);
          gb(3) 0 -gb(1);
          -gb(2) gb(1) 0];
Phi = -m*gbcross;
control_torque = -Phi*r_est-kp*omegap;

```

```

r_mmu = gbcross*control_torque/(norm(gb)^2*m_mmu);
control_torque = m_mmu * cross(r_mmu,gb);

% Adaptive control law for the estimated unbalance vector
r_est_dot = Phi*omega;

%% Dynamics equation
omegadot=J\(-cross(omega,J*omega)+cross(Rcm,gb*m)+control_torque);

%% Derivatives (xdot)
% qdot
ydot(1:4) = qdot;
% Euler angles (dot)
ydot(5:7) = omegadot;
% unbalance vector derivative - the vector does not change dynamically
ydot(8:10) = 0;
% adaptive control law for estimated unbalance vector
ydot(11:13) = r_est_dot;
% mmu positions derivative
ydot(14:16) = 0;

```

Listing K.12: Equations of motion used in ChesiBalancing

```

function UKFForBalancing_Chesi
% 4-state UKF used in the second phase of the hybrid
% balancing scheme for balancing the rz component.
tf = 100; % simulation length
dt = 0.1; % simulation step size
n=4; % number of states

%% Testbed characteristics
% (see eom_ukf.m)
J = [0.265 0 0; 0 0.246 0; 0 0 0.427];
Rcm = [0; 0; -5]*10^-3; % CM offset vector
m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity

%% Filter matrices
R = [0.005^2 0 0; 0 0.005^2 0; 0 0 0.005^2];
H = [eye(3) zeros(3,1)]; % measurement matrix

%% Initialization of variables
xstate = [0; 0; 0; Rcm(3)]; % initial state
xhatukf = 2 * xstate; % initial UKF state estimate
z = H * xstate + sqrt(R) * [randn; randn; randn];
W = ones(2*n,1)/(2*n); % UKF weights (in this case, equal weights for all sigma points)
Q = diag([0.00005 0.00005 0.00005 (0.1*Rcm(3))^2]);
y = [pi/6 pi/6 0 0 0 0 Rcm(3)];

%% Initialize arrays for later plotting

```

```

yArray = y';
zArray = z;
tArray = 0;
zhatArray = z;
xArray = xstate;
dtPlot = dt; % how often to plot output data
xhatukfArray = xhatukf;
P = 1e-5*eye(4);
Pukf = P;
Parray = diag(P);
Pukfarray = diag(Pukf);
chi2lim = chi2inv(0.95,length(xstate));
d2 = 0;
tic;
for t = dt : dt : tf
    k = round(t/dt)+1;
    tc=round(t/dt); % Time counter
    %% Simulate the system.
    % Integrate
    [~, y_dt] = ode23('eom_ukf', [0 dt], y(:), 1e-5);
    y(:)=y_dt(size(y_dt,1),:); % y = [phi, theta, psi, wx, wy, wz, rz]
    xstate(1:4) = y(4:7)' + sqrt(Q) * [randn; randn; randn; randn]; % xstate is a 12x1 column vector
    % Sensor simulation with noise
    z = H * xstate + sqrt(R) * [randn; randn; randn];
    %% Simulate the Unscented Kalman filter.
    %% Step 1: the model must be written in the form
    %  $x_{k+1} = f(x_k, u_k, t_k) + w_k$  (additive noise model)
    %  $y_k = h(x_k, t_k) + v_k$ 
    %  $w_k \sim (0, Q_k)$ 
    %  $v_k \sim (0, R_k)$ 

    %% Step 2: The UKF must be initialized (x0hat_posteriori and P0_posteriori)
    % Done
    %% Step 3: Time update
    [root,~] = chol(n*Pukf); %n*P
    for i = 1 : n %1:n
        sigma(:,i) = xhatukf + root(i,:);
        sigma(:,i+n) = xhatukf - root(i,:);
    end
    for i = 1 : 2*n %1:2n
        xbreve(:,i) = sigma(:,i);
    end

    %% Step 3(b): Use the known f(.) non linear function to transform the
    % sigma points into xhat_k(i) vectors.
    for i = 1:2*n
        % x = [omegax, omegay, omegaz, Rcm(3)]
        % y = [phi, theta, psi, wx, wy, wz, rz]
        xbrevedot = eom_ukf(t,[y(1:3)'; xbreve(1:4,i)]); %first input to UKF_laicatestbed_Chesi2015 is

```

```

    irrelevant (time t)
    xbreve(:,i) = xbreve(:,i) + [xbrevedot(4:6); 0] * dt;
end

%% Step 3(c): Combine the xhat_k(i) vectors to obtain the a priori state
% estimate at time k.
xhatukf = zeros(n,1);
for i = 1 : 2*n % i=1:2n
    xhatukf = xhatukf + W(i) * xbreve(:,i); % W is the wheight function (see beginning of file)
end

%% Step 3(d): Estimate a priori error covariance.
Pukf = zeros(n,n);
for i = 1 : 2*n %i=1:2n
    Pukf = Pukf + W(i) * (xbreve(:,i) - xhatukf) * (xbreve(:,i) - xhatukf)';
end
Pukf = Pukf + Q;
d2(k) = (z - H * xhatukf)' * inv(H * Pukf * H' + R) * (z - H * xhatukf); %xhatukf and Pukf -> a priori

%% Step 4: Measurement update
%% Step 4(a): Choose sigma points x_k(i).
% Start of optional step (comment lines below if you want)
[root,~] = chol(n*Pukf); %n=15
for i = 1 : n %1:n
    sigma(:,i) = xhatukf + root(i,:);
    sigma(:,i+n) = xhatukf - root(i,:);
end
for i = 1 : 2*n %1:2n
    xbreve(:,i) = sigma(:,i);
end
% End of optional step (comment lines above if you want)

%% Step 4(b): Apply the nonlinear measurement equation to the sigma
% points. In our case, the measurement equation is linear.
for i = 1 : 2*n %i=1:2n
    zukf(:,i) = H*xbreve(:,i);
end

%% Step 4(c): Combine the yhat_k(i) vectors to obtain predicted
% measurement at time k.
zhat = zeros(3,1);
for i = 1 : 2*n %i=1:2n
    zhat = zhat + W(i) * zukf(:,i);
end

%% Step 4(d): Estimate the covariance of the predicted measurement.
% Obs.: Rk has to be added.
% and Step 4(e): estimate the cross covariance between xhat_k_priori
% and yhat_k

```

```

Py = zeros(3,3);
Pxy = zeros(n,3);
for i = 1 : 2*n %i=1:2n
    Py = Py + W(i) * (zukf(:,i) - zhat) * (zukf(:,i) - zhat)';
    Pxy = Pxy + W(i) * (xbreve(:,i) - xhatukf) * (zukf(:,i) - zhat)';
end
Py = Py + R;

%% Step 4(f): The measurement update may be performed using the standard
% Kalman filter equations.
Kukf = Pxy * inv(Py);
xhatukf = xhatukf + Kukf * (z - zhat);
Pukf = Pukf - Kukf * Py * Kukf';

%% Save data for plotting.
yArray(:,k) = y';
PArray(:,k) = Pukf;
xArray(:,tc+1) = xstate;
zArray(:,tc+1) = z;
zhatArray(:,tc+1) = zhat;
xhatukfArray(:,tc+1) = xhatukf;
Parray(:,tc+1) = diag(P);
Pukfarray(:,tc+1) = diag(Pukf);
tArray = [tArray t];
end
rx_hat_mean = mean(xhatukfArray(4,round(end/2):end))
% Plot results
close all
t = 0 : dtPlot : tf;
for i = 1:size(PArray,3)
    vwx_est(i) = PArray(1,1,i);
    vwy_est(i) = PArray(2,2,i);
    vwz_est(i) = PArray(3,3,i);
    vrz_est(i) = PArray(4,4,i);
end
count = 0;
for i = 1:length(d2)
    if d2(i) > chi2lim
        count = count + 1;
    end
end
Performance = 1 - count/length(d2);
display(sprintf('Percentage of innovation samples inside Chi-Square margin: %.2f
%% ',100*Performance))

figure;
plot(tArray, yArray(1:3,:));
set(gcf, 'FontSize',12); set(gcf, 'Color', 'White');
xlabel('Seconds'); ylabel('Attitude [rad] ');

```

```

title('y(1:3) - rpy')

% Plot results
figure;
subplot(421);
plot(tArray,xArray(1,)-xhatukfArray(1,),'k'); hold on;
plot(tArray,+3*sqrt(vwx_est),'k:'); plot(tArray,-3*sqrt(vwx_est),'k:');
xlabel('t [s]'); ylabel('\omega_x [rad/s]'); title('Estimation error in \omega_x and
3\sigma intervals');
subplot(422);
plot(tArray,xArray(2,)-xhatukfArray(2,),'k'); hold on;
plot(tArray,+3*sqrt(vwy_est),'k:'); plot(tArray,-3*sqrt(vwy_est),'k:');
xlabel('t [s]'); ylabel('\omega_y [rad/s]'); title('Estimation error in \omega_y and
3\sigma intervals');
subplot(423);
plot(tArray,xArray(3,)-xhatukfArray(3,),'k'); hold on;
plot(tArray,+3*sqrt(vwz_est),'k:'); plot(tArray,-3*sqrt(vwz_est),'k:');
xlabel('t [s]'); ylabel('\omega_z [rad/s]'); title('Estimation error in \omega_z and
3\sigma intervals');
subplot(424);
plot(tArray,xArray(4,)-xhatukfArray(4,),'k'); hold on;
plot(tArray,+3*sqrt(vrz_est),'k:'); plot(tArray,-3*sqrt(vrz_est),'k:');
xlabel('t [s]'); ylabel('r_z [m]'); title('Estimation error in r_z and 3\sigma
intervals');
subplot(4,2,[5 6]);
plot(tArray, d2, 'k',[tArray(1) tArray(end)], chi2lim*[1 1], 'k'); hold on; ylabel('d^2(k)');
title('Chi-squared test');
subplot(4,2,[7 8]);
plot(t, xhatukfArray(4,),'k'); hold on; plot(t, sqrt(Pukfarray(4,)), 'k');
xlabel('t [s]'); ylabel('r_z [m]'); legend('r_zhat','P_{44}^{1/2}')
title('Estimation of r_z and corresponding deviation');

scrsz = get(groot,'ScreenSize');
figure('OuterPosition',[0 scrsz(4)/2 scrsz(3)/3 scrsz(4)/2])
%[(distance of left border from left margin of screen);
% (distance of bottom border from bottom of screen);
% width; height]'.
plot(t, xArray(1:3,:));
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Seconds'); ylabel('\omega (in rad/s)');
disp(['Elapsed time = ', num2str(toc), ' seconds']);

```

Listing K.13: 4-state UKF used in the second phase of the hybrid balancing scheme

```

function ydot = eom_ukf(t,y)
% y = [phi, theta, psi, wx, wy, wz, rz]
ydot = zeros(7,1);
Rcm = [0 0 y(7)];
%J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];
J0 = [0.265 0 0; 0 0.246 0; 0 0 0.427];

```

```

m = 14.307; % mass of the testbed
g = 9.78; % in m/s^2, local gravity
Jaug = [m*(Rcm(2)^2+Rcm(3)^2) -m*Rcm(1)*Rcm(2) -m*Rcm(1)*Rcm(3);
        -m*Rcm(1)*Rcm(2) m*(Rcm(1)^2+Rcm(3)^2) -m*Rcm(2)*Rcm(3);
        -m*Rcm(1)*Rcm(3) -m*Rcm(2)*Rcm(3) m*(Rcm(1)^2+Rcm(2)^2)];
J = J0 + Jaug;
omega = [y(4); y(5); y(6)];
sphi = sin(y(1));
cphi = cos(y(1));
sth = sin(y(2));
cth = cos(y(2));
%% Model equations
% Kinematic equation
% Euler rates
eulerdot = [1 sphi*tan(y(2)) cphi*tan(y(2));
            0 cphi -sphi;
            0 sphi/cth cphi/cth]*omega;
% Gravity vector in body-frame
%gb = [g*sth; -g*sphi*cth; -g*cphi*cth];
% Dynamics equation
omegadot=J\(-cross(omega,J*omega)+ [g*sphi*cth*m*y(7); g*sth*m*y(7); 0]);
%% Derivatives (xdot)
% Omega(dot)
ydot(1:3) = eulerdot;
% Euler angles (dot)
ydot(4:6) = omegadot;

```

Listing K.14: Equations of motion used in the 4-state UKF