Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

ZENTRUM
FÜR
TELEMATIK

Institute for Computer Science VIII - Aerospace Information Technology

Zentrum für Telematik e. V.

Bachelor's thesis

Algorithmic center of mass offset approximation of a spherical air-bearing satellite simulator

Leon Lukaschek

February 2024

First reviewer: Prof. Dr. Marco Schmidt
Julius-Maximilians-Universität of Würzburg

Second reviewer: Prof. Dr.-Ing. Sergio Montenegro
Julius-Maximilians-Universität of Würzburg

Advisor: Vijay Nagalingesh
Smart Small Satellite Systems GmbH

Erklärung - Proclamation

Hiermit versichere ich, Leon Lukaschek, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie sämtliche wörtlichen oder sinngemäßen Übernahmen und Zitate kenntlich gemacht sind.

Diese Arbeit wurde nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt oder eingereicht.

I, Leon Lukaschek, hereby declare that I have independently authored the present work and have not used any sources or aids other than those specified, and that all direct or paraphrased appropriations and quotations are indicated.

This work has not been prepared or submitted, even partially, for any other examination.

Signature: _____

Würzburg, 23.02.2024

Abstract

Currently, at the Zentrum für Telematik e. V. (ZfT) multiple CubeSat-Missions are under development. One example is Telematics Earth Observation Mission (TOM), a CubeSat that is part of the Telematics International Mission (TIM), which will be used perform photogrammetric observations of volcanic ash clouds. Another CubeSat-Mission of interest at ZfT is CloudCT. 10 CubeSats will fly in formation to analyze clouds using computer tomography (CT). Both missions will need a very high level of attitude precision to satisfy their mission requirements. This leads to the need of a robust working Attitude Determination and Control Subsystem (ADCS). Being a challenge to design even for conventional Satellites, it is an even more difficult challenge for CubeSats as their attitude determination and pointing accuracy has to be very precise without having much mass and volume to work with in the confined space of a CubeSat. One way to ensure that the CubeSats ADCS will be working as expected is to approximate the gravity free environment in orbit using a ground based satellite simulator. To evaluate the ADCS hardware and algorithms for missions like TOM and CloudCT, ZfT is using a spherical air-bearing simulator to replicate the frictionless environment. But, the exact space environment can not be replicated on ground due to additional influences acting upon the simulator like gravity and aerodynamic forces. To keep the most influential force on the simulator, gravity, small, the Center of Mass (CoM) must align as close as possible with the Center of Rotation (CoR) of the simulator and device under test. This thesis will investigate batch estimation, filtering as well as a control law based algorithms for the approximation of the CoR to the CoM offset vector. Selected estimation techniques will be evaluated by using a simulation of the attitude simulator as well as by the attitude simulator present at the ZfT. A selection is made based on important characteristics, e.g. the estimation precision and time efficiency.

Zusammenfassung

Derzeit werden am Zentrum für Telematik e. V. (ZfT) mehrere CubeSat-Missionen entwickelt. Ein Beispiel ist die Telematik-Erdbeobachtungsmission (TOM), ein CubeSat, der Teil der Telematik-International-Mission (TIM) ist und zur Verfolgung von Vulkanausbrüchen eingesetzt werden soll. Eine weitere CubeSat-Mission von Interesse am ZfT ist CloudCT. 10 CubeSats werden in Formation fliegen, um Wolken mithilfe der Computertomographie (CT) zu analysieren. Beide Missionen erfordern ein sehr hohes Maß an Lagengenauigkeit, um ihre Missionsanforderungen zu erfüllen. Dies führt zur Notwendigkeit eines robusten Lage-Detektion-und-Kontroll-Subsystems (ADCS). Die Entwicklung dieses Systemes stellt bereits für konventionelle Satelliten eine Herausforderung dar, jedoch ist sie für CubeSats eine noch größere Herausforderung, da ihre Lagenermittlung und Ausrichtung sehr präzise sein muss, gleichzeitig hat ein CubeSat nur ein sehr begrenztes Volumen für Messinstrumente. Ein Weg, um sicherzustellen, dass das ADCS der CubeSats wie erwartet funktioniert, besteht darin, die schwerkraftfreie Umgebung in der Umlaufbahn mithilfe eines bodengestützten Satellitensimulators zu approximieren. Um die ADCS-Hardware und Algorithmen für Missionen wie TOM und CloudCT zu bewerten, verwendet das ZfT einen Luftlager-Simulator, um die reibungslose Umgebung nachzubilden. Die exakte Weltraumumgebung kann jedoch auf dem Boden nicht reproduziert werden, da zusätzliche Einflüsse auf den Simulator wirken, wie Schwerkraft und aerodynamische Kräfte. Um die einflussreichste Störkraft auf den Simulator, die Schwerkraft, gering zu halten, muss der Massenschwerpunkt (CoM) so nah wie möglich am Rotationszentrum (CoR) des Simulators und des Prüfgeräts ausgerichtet werden. Diese Arbeit untersucht Batch-Estimation, Filterung sowie algorithmenbasierte Steuerungsgesetze zur Approximation des Versatzvektors von CoR zu CoM. Ausgewählte Schätzverfahren werden anhand einer Simulation des Lagensimulators sowie des am ZfT vorhandenen Lagensimulators evaluiert. Die Auswahl erfolgt auf der Grundlage wichtiger Merkmale wie der Schätzgenauigkeit und der Zeiteffizienz.

Contents

List of Figures

List of Tables

List of Symbols

Bold lowercase symbols	Vectors
Bold uppercase symbols	Matrices
$\mathbf{A}_{m\times n}$	Matrix **A** with dimensions m × n.
$I_{m\times m}$	Idendity matrix with dimension m × m with ones on main diagonal and zeroes elsewhere.
$\mathbf{x}^T$	Transpose of vector or matrix **x**.
$\mathbf{x}^{-1}$	Inverse of vector or matrix **x**.
φ	Roll angle, i.e., rotation around x-axis.
Θ	Pitch angle, i.e., rotation around y-axis.
ψ	Yaw angle, i.e., rotation around z-axis.
E	Vector of Euler angles.
T	Sampling time
J	Inertia tensor matrix.
q	Quaternion
q	Vector part of Quaternion q.
(**i**, **j**, **k**)	Vectors of an orthogonal system.
×	Vector cross product
[**v**×]	Vector cross product in matrix form.
.	Vector dot product
⊙	Hamiltonian Quaternion product
ω	Angular velocity
H	Angular momentum
M	Torque
r	Unbalance vector
m	Mass in general or mass of the attitude simulator support platform.
m_p	Mass of the Moveable Mass Units.
$\mathbf{J}_i$	Principal moment of inertia around axis i.
Subscript x, y, z	Component x, y, z of quantity.
Subscript i	Quantity related to the inertial frame.
Subscript b	Quantity related to the body frame.
Subscript or Superscript k	Quantity related to time-step or instant k.
$\hat{x}$	Estimate of quantity x.
x^-	A priori state of quantity x.
x^+	A posteriori state of quantity x.

Chapter 1

Introduction

Since the 1950s when the first satellites were launched, the topics of aerospace engineering, satellites and rockets have found their way into the everyday life of almost any human on earth. Without these technologies, life would be very different today. Especially through the space race many new technologies were developed, naming GPS as only one of those technologies. These advancements were made with sometimes taking much risk in the mission's success, resulting in the loss of some missions. However, most of the time the missions involved long test campaigns. With further advancements in the aerospace sector, the testing capabilities and testing devices increased in precision and accuracy.

One of the major aspects of the testing of a satellite is that of the development of its Attitude Determination and Control Subsystem (ADCS). This subsystem is responsible to estimate and change the satellite's attitude in space. The attitude of a satellite can be described as its rotation in space relative to some fixed reference point. For many missions, its success is closely linked to the ability of the ADCS to deliver the needed precision. The magnitude of precision is related to the satellite's overall mission and dictated by the devices and sensors used. As an example, a satellite communicating by a radio dish needs a far less accurate pointing accuracy than a satellite that communicates by optical communication through a laser terminal. Additionally, in recent years CubeSats have found their way into many aerospace missions as they can reduce the missions total cost. CubeSats are small satellites which are measured in so-called U units, one U representing a cube with dimensions of $10 \times 10 \times 10$ cm. An example of such a mission utilizing CubeSats is the Telematics Earth Observation Mission (TOM) mission currently being developed by the Zentrum für Telematik e. V. (ZfT). It comprises three, $3\,U$ CubeSats, which means the CubeSats have dimensions of $30 \times 10 \times 10$ cm. TOMs goal is to fly the three CubeSats in formation so that they are able to measure the spatial extent of volcanic ash clouds by mapping it simultaneously from three different angles. Then, by post-processing and merging, a three-dimensional image is constructed, providing information about the spatial distribution, altitude and more.[1] As CubeSats offer only a very compromised volumetric space, their sensors and actuators are often miniaturized versions of their counterparts used in conventional satellites. With their loss of volume a loss of precision often goes hand in hand, leading to even higher pointing accuracy of the CubeSat.

[1] https://www.telematik-zentrum.de/en/projects/tim-tom/

One of the major challenges of the development of the ADCS is to replicate the environment found in orbit on ground. In orbit, the satellite is exposed to earth's magnetic field, radiation and residual gravitational forces. To be able to develop the ADCS, this environment has to be replicated in the laboratories. For vacuum tests, vacuum chambers can be used, for the earths magnetic field a Helmholtz cage. The micro gravitational environment can be approximated by using a so-called attitude simulator. It is often made up of a spherical air bearing and support structure, onto which the satellite is placed. Through small inlets in the air bearings base pressurized air passes, creating a near frictionless environment. Through this setup, the satellite can rotate about the satellite simulators CoR similar to its rotation in orbit and the ADCS's algorithms and estimation techniques can be developed and tested. However, the low friction allows to copy the low gravity environment found in orbit only to some extent. As the satellites mass distribution is almost never perfectly symmetrical, the satellite will not be in a balanced position by itself. To accomplish such a balanced state, external weights have to be utilized. The placement of the weights change the weight properties of the combined system of the satellite as well as the support structure and the attitude simulator. Goal of this change of mass properties is to move the satellites CoM as close as possible to the systems CoR. This has to be done, as the gravitational force will always act upon a body by its CoM. If this point is offset from the CoR, a torque will act upon the body. In the system of a satellite on an air bearing, this torque will change the angular momentum of the system which in turn falsifies the simulated environment. If the ADCS is developed on such a falsified system, the satellite's mission is at risk as the algorithms for the attitude estimation and attitude changes could produce wrong values or less accurate states. Thus, the torque created due to the CoM to CoR offset vector and acting gravitational forces has to be minimized as much as possible, by a change of mass properties by the balancing masses placement.

In this thesis, different algorithms and approaches are investigated that estimate the CoM of a CubeSat on a attitude simulator. This is done with the goal of finding a suitable algorithm, which could be used in the future for the attitude simulator found at the ZfT. Ultimately, this could improve the ADCS's accuracy used for the CubeSats developed and built there.

Chapter 2

Thesis Objective, Structure and Requirements

The objective of this thesis is to identify, present and to implement algorithms that are able to estimate the Center of Rotation (CoR) to Center of Mass (CoM) offset to a high precision. Gathered results could improve the satellite simulator setup at the ZfT which in turn allows a more accurate Attitude Determination and Control Subsystem (ADCS) development. This could ultimately improve the overall mission success of future CubeSat missions at the Zentrum für Telematik e. V. (ZfT).

The structure of this thesis is as follows. Firstly, air bearing satellite simulators and needed theoretical background is introduced in Chapter 3. In Chapter 3.4 different algorithms are presented for the estimation of the CoM offset. The most prominent estimation techniques, a Kalman Filter approach and an adaptive control law are implemented into a simulated attitude satellite simulator, with the Kalman Filter also being validated by utilizing actual data from the satellite simulator present at the ZfT. After that, the accuracy and precision are evaluated in Section 5, and a conclusion is drawn in Chapter 6. Lastly, possible further improvements and future work is discussed in Chapter 7.

In order to be able to compare and evaluate the algorithms, a list of requirements is necessary. Firstly, the accuracy of the estimation should be at least ± 0.1mm as to improve existing estimation techniques used at the ZfT. One important requirement is that the algorithm has to be an online algorithm, meaning that it is capable of processing data in real-time as it is measured. The reason for this is, that the overall testing time should be kept to a minimum to allow a high number of tests in a small amount of time. Thus, the time from taking measurements to having a reliable offset estimate should not exceed 60 seconds. Additionally, the computational effort should be kept as small as possible to ensure timely execution. As the offset approximation algorithm will most likely be implemented on some sort of embedded device in the future, the complexity of the algorithm has to be kept small. If the algorithm is computationally complex, it will take a longer time to be executed which in turn means that it has to be run at a lower frequency making it more susceptible to errors. Lastly, the algorithm should be able to take in new measurements at a frequency of at least 10 Hz. This requirement enables the algorithm to adapt to rapidly changing input information, ensuring responsiveness to dynamic scenarios.

Chapter 3

Literature Review

The following Chapter presents terminology, notation and methods used for the description of the algorithms used to later on in this work. First, air bearing satellite simulators are introduced. Then, different representations of the attitude of bodies in reference frames is discussed. Important for the following simulation of attitude simulators, the dynamic model of such a device is presented. Lastly, different techniques for the estimation of the CoM offset vector are listed.

3.1 Air bearing satellite simulators

The development of a well working ADCS is crucial for the spacecraft's operation in space. As it is not possible to exactly replicate the environment found in earth's orbit such as microgravity, vacuum, radiation and magnetic torque due to the influence of earths magnetic field, simulations of said environment have to be developed to and ensure a well working ADCS. For this purpose, air bearings have been used in different setups and configurations in the past as they can create a close representation of said environment.

An air bearing satellite simulator is a system in which a Cube- or Nanosatellite's dynamics can be tested. In most air bearing satellite simulators a planar structure is mounted onto the air bearing, on which a satellite can be placed. This setup allows the development and testing of the ADCS's attitude control algorithms, if the test environment is able to minimize the gravitational torques and enable a sufficiently frictionless rotation. The attitude simulator found at the ZfT is in a so-called tabletop configuration, as the air bearing is cut in half and onto the flat surface a support structure is mounted. This results in free rotational movement about the Z-axis, while limiting the rotation about the X- and Y-axis to about $\pm 30°$. About 4.5 bar is the working pressure of the air bearing base. One of the most important properties of the air bearing simulator are its total mass and its inertia. By weighing the individual parts of the attitude simulator the total mass can be found to a great accuracy. The simulator support structure at the ZfT weights about 1kg. More difficult to determine is the inertia matrix. There is equipment that is able to determine the inertia matrix to a great precision but as they are very expensive but offer just a single usage non such device is present at the ZfT.[1] An alternative

[1] e.g.: www.resonic.de/products/resonic-g/, www.raptor-scientific.com/products-category/full-mass-

approach is that of using the CAD (Computer Aided Design) model of the CubeSat. Using the CAD tool, the inertia matrix can also be determined and was found to be

$$\mathbf{J} = \begin{bmatrix} 53.6434 & 0.0308 & 0.2156 \\ 0.0308 & 50.2895 & 0.0173 \\ 0.2156 & 0.0173 & 14.481 \end{bmatrix} \cdot 10^{-3} \ \text{kg} \cdot \text{m}^2. \tag{3.1}$$

However, it should be noted that the accuracy of J greatly depends on the accuracy with which the used components are modeled. As the attitude simulator is made up by components from different suppliers, this is difficult to control. The modeling accuracy is also dependent on their material properties, homogeneity, if cables or screws of the individual components are modeled as well as if the flexibility of the structure is considered. Also, as no different approach of determining the inertia matrix was used in this work, J can also not be verified by another measurement.

An overview of existing satellite simulators, their respective platform configurations, hardware specifications and issuing institution can be found in [8, p. 540]

3.1.1 Structural overview of air bearings

Air bearings, in general, have a moving or rotating part and a fixed part. Pressurized air passes through small openings or outlets in the fixed section which creates a thin film of air. The moving part can be set onto the thin film, which works as a kind of lubricant between both parts and enables almost frictionless movement.

There are two main groups into which attitude simulators using air bearings can be divided into: planar systems and spherical systems. Both types have their advantages and disadvantages and different systems offer different rotational and translational Degree of Freedom (DoF).

Planar systems support movement in one plane and rotation about the vector perpendicular to that plane, which means that they have one rotational and two translational DoF. A familiar system using a planar air bearing is that of an air hockey table. Pressurized air passes through small holes in the table which creates a thin film of air. The hockey puck can now move freely and almost frictionless on the table but can only rotate about the vertical axis. In the context of aerospace systems, planar air bearings have been used for simulations of rendezvous and docking in space. As planar systems offer only one rotational DoF, they can only be used to a small degree to develop ADCSs as satellites in orbit almost never rotate about just on axis. [26]

Spherical systems consist of two sections of concentric spheres. Similar to the planar system, pressurized air passes through small openings in the static sphere which creates a thin air film on which the non-fixed sphere can rotate in three rotational DoF. As often additional equipment has to be fixed on top of the rotating sphere, its rotation is rarely unconstrained in all three axes of rotation. A mechanical system that is similar to that of a spherical air bearing is that of a ball-and-socket joint. The advantage of using a spherical air bearing over a ball-and-socket joint is that the friction between both spheres is much smaller. Another similar mechanical system is that of multiple gimbals which can be used for the development of the ADCS as well. The

properties/

major drawback of a system of gimbals is that it introduces the problem of gimbal lock and also that the gimbal dynamics can interact with the payload dynamics. [2]

Spherical systems can be further divided into tabletop-, dumbbell and umbrella-style platforms. The different spherical air bearings and their axes can be seen in Figure 3.3. Both tabletop and umbrella platforms provide 360 degrees of rotational freedom in the yaw axis but are typically restricted to no more than a total of 90 degrees of rotational freedom about the roll and pitch axes, respectively. The difference of both platforms is that of where the support structure is mounted onto the system. In the tabletop platform, the support structure is mounted directly onto a flat face of the bearing. For that, a section of the sphere is cut off. On umbrella platforms, the support structure is mounted onto an extension cylinder or shaft which "grows" out of the sides of the sphere. The name umbrella stems from the fact that often the supporting equipment is mounted outward and down of the support structure, which makes it look like an umbrella. Dumbbell platforms have a fully spherical air bearing from which two "arms" extend outwards onto which the support structure is mounted on. This setup greatly increases the rotational freedom and enables full rotation about the roll and yaw axes but is more complex to design as each side of the dumbbell has to weigh the same to keep the configuration from tipping onto one side. [8, 26, 38]

The fixed part of a spherical air bearing is also called the air bearing cub, while the rotating section is called the air bearing base. An exemplary structural view of a spherical air bearing cup can be seen in Figure 3.1. In Figure 3.2 the spherical air bearing and cup used by the ZfT is shown.

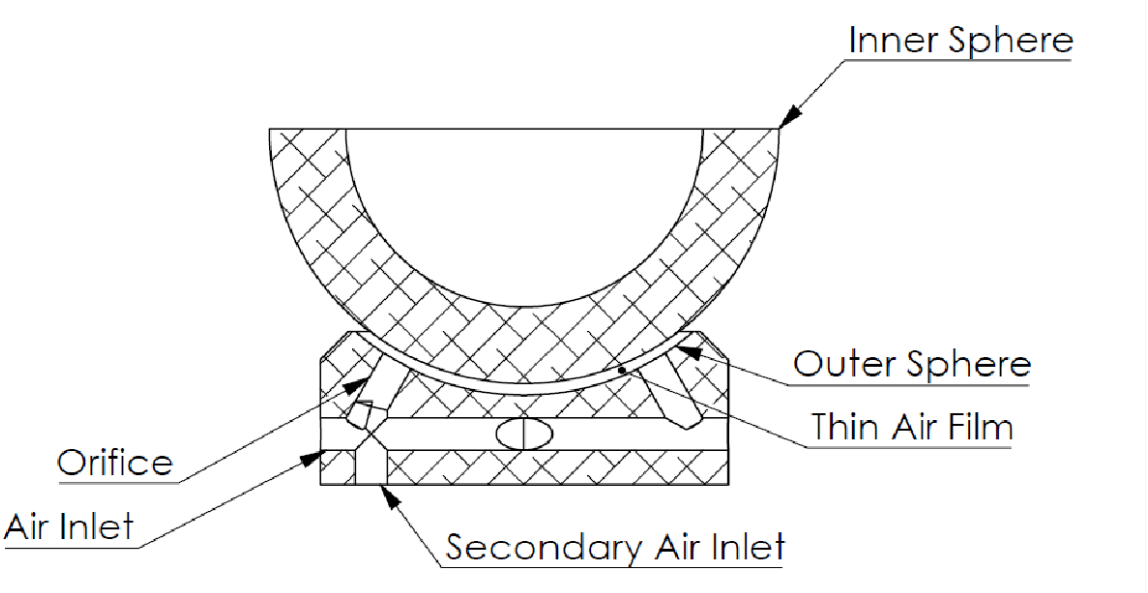
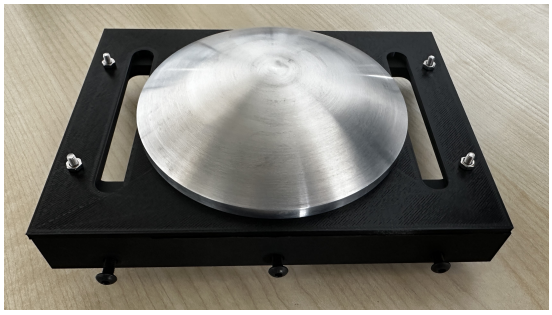
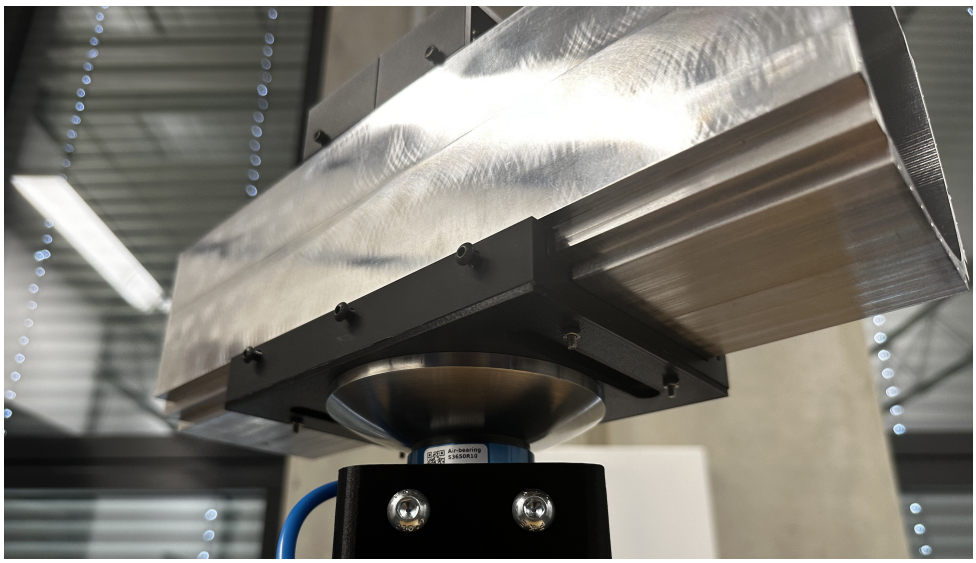


Figure 3.1: Structural view of an air bearing cup. [38]

(a) Air bearing cup.

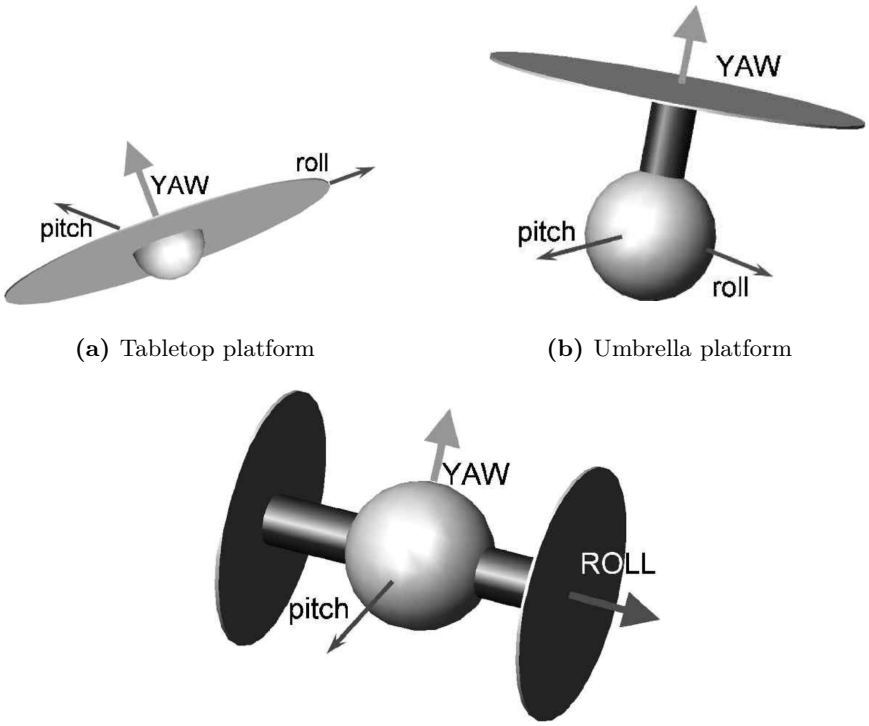


(b) Air bearing base connected to support structure.



(c) CubeSat mass model on attitude simulator.

Figure 3.2: Attitude air bearing simulator cup (a) and base (b) at the ZfT. (c) Shows the mass model for the TOM CubeSat placed ontop of the air bearing. Partly visible on top is the support equipment mounted for the optical tracking of the CubeSat. Image Credit: ZfT.



(a) Tabletop platform

(b) Umbrella platform

(c) Dumbbell platform

Figure 3.3: Comparison of spherical air bearings. [26]

3.2 Attitude Representation

The orientation of a spacecraft in space in relation to a point in space is described as the attitude of a spacecraft. Or, as stated in [29, p. 318]:

«The attitude of a body is thought of as a coordinate transformation that transforms a defined set of reference coordinates into the body coordinates of the spacecraft.»

In general, there are different ways of describing the attitude of a rigid body such as Euler angles and quaternions. Before one may talk about attitude representation, reference frames and rotation matrices have to be introduced first.

3.2.1 Reference Frames

A reference frame is defined by the location of its origin and the orientation of its coordinate axes, which are mutually perpendicular to each another. In three dimensions, multiple reference frames are of interest to the representation of the attitude of an object but only two of them, a body-fixed reference frame and an inertial reference frame are used in this work.

The spacecraft body frame is defined by three Cartesian axes and an origin at a specified point in the spacecraft body. It is used to align the components during spacecraft assembly and to keep track of the spacecraft's components locations after launch. Due to the very high forces

during launch and due to thermal deformation in orbit, components might shift and change their location. Some other components, such as solar panels and instruments on gimbals change their location and orientation on purpose. Often, a rigid navigational base is selected as the origin of the body frame which includes the most important attitude sensors and payload instruments. In a body-fixed frame, its orientation and position is constant in relation to the moving body. [21]

Similar to the body reference frame, an inertial reference frame is defined by an origin and three perpendicular axes. Additionally, an inertial reference frame is a frame in which Newton's laws of motion are valid. This means, that an inertial frame is not accelerating or decelerating. Also, a frame moving at constant velocity and without rotation in relation to another inertial frame is inertial as well. [34]

In this work, the inertial frame is represented by the $(\mathbf{x}_\mathrm{i}, \mathbf{y}_\mathrm{i}, \mathbf{z}_\mathrm{i})$ axes, while the body frame is represented by the $(\mathbf{x}_\mathrm{b}, \mathbf{y}_\mathrm{b}, \mathbf{z}_\mathrm{b})$ axes. Since the air bearing setup used in this work has a precisely known CoR, it is used as the center of origin for both reference frames, the body-fixed frame and the inertial frame. The orientation of the body-fixed coordinate system with respect to the inertial coordinate system defines the attitude of the spacecraft simulator. The $\mathbf{x}_\mathrm{i}$ and $\mathbf{y}_\mathrm{i}$ vectors form a plane parallel to the laboratory ground, and the $\mathbf{z}_\mathrm{i}$ vector is perpendicular on the $\mathbf{x}_\mathrm{i}$ and $\mathbf{y}_\mathrm{i}$ vectors, facing the direction of the zenith. Figure 4.8 presents this reference frame. From this follows the gravity vector as $\mathbf{g} = [0, 0, -\mathrm{g}]^\mathrm{T}$. The value of g is selected to be 9.80999 $\frac{\mathrm{m}}{\mathrm{s}^2}$, which is the local gravitational acceleration at Würzburg according to [18], using the latitude 49.79762631° and an altitude of 171.77 m which were taken from [20].

As the position of the satellite is fixed to the attitude simulator, which is stationary in the laboratory, all transformations in this work are so-called *passive transformations*. In general, geometric transformations can be distinguished into two groups, *active* or *alibi transformations* and *passive* or *alias transformations*. For active transformations, an object itself changes position in the coordinate system. In contrast, a passive transformation leaves objects fixed in space but the reference frame itself changes position and orientation relative to another reference frame. [28]

3.2.2 Direction Cosine Matrix

The current and the next two Chapters are based closely on [29, Appendix A] and [9]. Additionally, from now on, vectors are going to be represented as bold lowercase letters and matrices by bold uppercase letters.

The most basic transformation of one frame into another is based on the direction cosine matrix which uses a 3×3 matrix, $\mathbf{D}$, to represent the linear transformation mapping from one coordinate frame to another rotated coordinate frame. It forms a mapping describing how much each axis lines up with the axes of another reference frame. The name direction *cosine* matrix stems from the fact that the scalar product between two vectors is build using the cosine function,

$$\vec{\mathbf{a}} \cdot \vec{\mathbf{b}} = |\vec{\mathbf{a}}||\vec{\mathbf{b}}|\cos \sphericalangle(\vec{\mathbf{a}}, \vec{\mathbf{b}}).$$

To aid legibility, the arrows above vectors are neglected in this thesis, vectors are represented as lowercase bold letters. Each column of the direction cosine matrix $\mathbf{D}$ represents the unit vector

in the inertial reference system projected along the body reference frame axes. The direction cosine matrix **D** is calculated as

$$\mathbf{D}_{\mathrm{i}}^{\mathrm{b}} = \begin{bmatrix} \cos(\Theta_{\mathrm{x_i,x_b}}) & \cos(\Theta_{\mathrm{y_i,x_b}}) & \cos(\Theta_{\mathrm{z_i,x_b}}) \\ \cos(\Theta_{\mathrm{x_i,y_b}}) & \cos(\Theta_{\mathrm{y_i,y_b}}) & \cos(\Theta_{\mathrm{z_i,y_b}}) \\ \cos(\Theta_{\mathrm{x_i,z_b}}) & \cos(\Theta_{\mathrm{y_i,z_b}}) & \cos(\Theta_{\mathrm{z_i,z_b}}) \end{bmatrix} \tag{3.2}$$

in which $\Theta_{\mathrm{x_i,x_b}}$ is the angle between the inertial and body x-axis. Similar statements can be produced for the other Θ angles.

As an example, vector $\mathbf{v}_{\mathrm{i}}$ in the inertial frame can be described in the body frame as $\mathbf{v}_{\mathrm{b}}$ using **D** as

$$\mathbf{v}_{\mathrm{b}} = \mathbf{D}_{\mathrm{i}}^{\mathrm{b}} \mathbf{v}_{\mathrm{i}}. \tag{3.3}$$

3.2.3 Euler Angle Rotations

A rotation in $\mathbb{R}^3$ can be seen as the transformation

$$\mathbf{v}_{\mathrm{a}} = \mathbf{R}\mathbf{v}_{\mathrm{b}}, \tag{3.4}$$

in which **R** is a 3×3 transformation matrix that maps vector $\mathbf{v}_{\mathrm{b}}$ to the rotated vector $\mathbf{v}_{\mathrm{a}}$ without undergoing any translation, $\mathbf{R} : \mathbf{v}_{\mathrm{b}} \mapsto \mathbf{v}_{\mathrm{a}}$. One example of such a rotation matrix was already presented in the preceding Chapter as matrix $\mathbf{D}_{\mathrm{i}}^{\mathrm{b}}$. If the rotation matrix is an orthogonal matrix, which means that its rows and columns are unit vectors (length of vector is 1), the rotation reversal can be accomplished by multiplying the rotated vector $\mathbf{v}_{\mathrm{a}}$ with the transpose of the rotation matrix **R**,

$$\mathbf{v}_{\mathrm{b}} = \mathbf{R}^{\mathrm{T}} \mathbf{v}_{\mathrm{a}}. \tag{3.5}$$

Euler Angle rotation is defined as three successive angular rotations about the three orthogonal frame axes, as stated in the Theorem of Euler Angles.

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 leads to a total sum of twelve possible rotation sequences, which can be divided into two groups: successive rotations about each axis of the reference frame or a rotation about the same axis as the first and third rotation with a rotation about a second axis as the second step in the rotation sequence.

Suppose we have a reference frame with axes **i**, **j** and **k**. All possible successive rotations are *i-j-k, i-k-j, j-i-k, j-k-i, k-i-j* and *k-j-i*. Meanwhile, all sequences using the same first and third axes are *i-j-i, i-k-i, j-i-j, j-k-j, k-i-k* and *k-j-k*.

Which actual sequence is used depends on the situation or system model faced. When working in the context of aerospace projects, Euler angles are most often used in a *Z-Y-X* sequence. The rotation about the X-axis is then called roll, the rotation about the Y-axis pitch and the rotation about the Z-axis yaw.

To aid legibility, cos is denoted as c and sin as s in the following matrices. The rotations about the X, Y and Z axes are give by the following direction cosine matrices,

$$\mathbf{R}_{\mathrm{X},\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\varphi & \mathrm{s}_\varphi \\ 0 & -\mathrm{s}_\varphi & \mathrm{c}_\varphi \end{bmatrix} \tag{3.6}$$

$$\mathbf{R}_{\mathrm{Y},\Theta} = \begin{bmatrix} \mathrm{c}_\Theta & 0 & -\mathrm{s}_\Theta \\ 0 & 1 & 0 \\ \mathrm{s}_\Theta & 0 & \mathrm{c}_\Theta \end{bmatrix} \tag{3.7}$$

$$\mathbf{R}_{\mathrm{Z},\psi} = \begin{bmatrix} \mathrm{c}_\psi & \mathrm{s}_\psi & 0 \\ -\mathrm{s}_\psi & \mathrm{c}_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

with respective angles of rotation of φ, Θ and ψ degree or rad, with the rotation being performed counterclockwise. By combining the rotations about the respective axes as their angles together into one vector, the attitude can be represented as Euler Angles $\mathbf{E} = [\varphi, \Theta, \psi]^{\mathrm{T}}$. The mathematical derivation of (3.6), (3.7) and (3.8) can be made by geometric relations between vectors in a reference frame and the reference frame itself.

The relation between the coordinates of a vector in the body and inertial frame for the Z-Y-X sequence is given by $\mathbf{v}_\mathrm{b} = \left(\mathbf{R}_{\mathrm{X},\varphi}\mathbf{R}_{\mathrm{Y},\Theta}\mathbf{R}_{\mathrm{Z},\psi}\right)\mathbf{v}_\mathrm{i}$.

The sequence $\mathbf{R}_{\mathrm{X},\varphi}\mathbf{R}_{\mathrm{Y},\Theta}\mathbf{R}_{\mathrm{Z},\psi}$ can be simplified to a single matrix $\mathbf{R}$,

$$\begin{aligned} \mathbf{R} = \mathbf{R}_{\mathrm{X},\varphi}\mathbf{R}_{\mathrm{Y},\Theta}\mathbf{R}_{\mathrm{Z},\psi} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\varphi & \mathrm{s}_\varphi \\ 0 & -\mathrm{s}_\varphi & \mathrm{c}_\varphi \end{bmatrix} \begin{bmatrix} \mathrm{c}_\Theta & 0 & -\mathrm{s}_\Theta \\ 0 & 1 & 0 \\ \mathrm{s}_\Theta & 0 & \mathrm{c}_\Theta \end{bmatrix} \begin{bmatrix} \mathrm{c}_\psi & \mathrm{s}_\psi & 0 \\ -\mathrm{s}_\psi & \mathrm{c}_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{c}_\psi\mathrm{c}_\Theta & \mathrm{s}_\psi\mathrm{c}_\Theta & -\mathrm{s}_\Theta \\ -\mathrm{s}_\psi\mathrm{c}_\varphi + \mathrm{c}_\psi\mathrm{s}_\Theta\mathrm{s}_\varphi & \mathrm{c}_\psi\mathrm{c}_\varphi + \mathrm{s}_\psi\mathrm{s}_\Theta\mathrm{s}_\psi & \mathrm{c}_\Theta\mathrm{s}_\varphi \\ \mathrm{s}_\psi\mathrm{s}_\varphi + \mathrm{c}_\psi\mathrm{s}_\Theta\mathrm{c}_\varphi & -\mathrm{c}_\psi\mathrm{s}_\varphi + \mathrm{s}_\psi\mathrm{s}_\Theta\mathrm{c}_\varphi & \mathrm{c}_\Theta\mathrm{c}_\varphi \end{bmatrix} = \mathbf{R}_\mathrm{b}^\mathrm{i} \end{aligned} \tag{3.9}$$

$\mathbf{R}_\mathrm{b}^\mathrm{i}$ is the rotation matrix that transforms a vector from the inertial frame to the body frame.

The derivatives of the Euler angles will play an important role in the following Chapters and is accomplished by

$$\begin{aligned} \mathbf{E}_{\mathrm{k}+1} &= \mathbf{E}_\mathrm{k} + \dot{\mathbf{E}} \cdot T \\ \begin{bmatrix} \varphi_{\mathrm{k}+1} \\ \Theta_{\mathrm{k}+1} \\ \psi_{\mathrm{k}+1} \end{bmatrix} &= \begin{bmatrix} \varphi_\mathrm{k} \\ \Theta_\mathrm{k} \\ \psi_\mathrm{k} \end{bmatrix} + \begin{bmatrix} \dot{\varphi_\mathrm{k}} \\ \dot{\Theta_\mathrm{k}} \\ \dot{\psi_\mathrm{k}} \end{bmatrix} \cdot T \end{aligned} \tag{3.10}$$

in which T is the sampling time and $\mathbf{E} = [\varphi, \Theta, \psi]^{\mathrm{T}}$ the attitude as Euler Angles . For the modeling the rotation of a rigid body, which will be used for the simulation of the air bearing

satellite simulator in Chapter 4.1, it is necessary to propagate the Euler angles in time. The derivative of the Euler angles are given by

$$\dot{\mathbf{E}} = \begin{bmatrix} \dot{\varphi} \\ \dot{\Theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\Theta\sin\varphi & \tan\Theta\cos\varphi \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sec\Theta\sin\varphi & \sec\Theta\cos\varphi \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{3.11}$$

Both, the propagation of the Euler angles and the derivative of the Euler angles are dependent on the rotation sequence. In this work, only the zyx sequence will be used. Equation (3.11) can be developed from

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\Theta \\ 0 & \cos\varphi & \cos\Theta\sin\varphi \\ 0 & -\sin\varphi & \cos\Theta\cos\varphi \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\Theta} \\ \dot{\psi} \end{bmatrix}. \tag{3.12}$$

$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$ represents the bodies angular velocity vector. A full derivation of the derivative of the Euler angles can be found in [30, Appendix B].

One major drawback of using Euler angles for the representation of attitude is that there are some angles for which the Euler rates are not defined. This results from the presence of the tan and sec $= \cos^{-1}$ trigonometric functions in (3.11), which are not defined for values $\Theta = \pm\frac{\pi}{2}$. This singularity can be avoided when using more sophisticated filters, but results in more complex algorithms and higher computational efforts. Another drawback is that Euler angles are based on direction cosine matrices, which are always represented by 9 elements in $\mathbb{R}^3$. As can be seen in the following Section 3.2.4, quaternions offer an attitude representation using only 4 elements.

3.2.4 Quaternions

A quaternion is a 4-tuple which defines an element in $\mathbb{R}^4$ and is written as $\mathrm{q} = (\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3)$ where q_0, q_1, q_2 and q_3 are real numbers or scalars.

An alternative representation of a quaternion is to define it as a scalar part q_0 and a vector part, $\mathbf{q}$. $\mathbf{q}$ is a vector in $\mathbb{R}^3$, namely $\mathbf{q} = \mathbf{i}\mathrm{q}_1 + \mathbf{j}\mathrm{q}_2 + \mathbf{k}\mathrm{q}_3$. $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are the orthonormal basis for $\mathbb{R}^3$, $\mathbf{i} = (1,0,0)$ $\mathbf{j} = (0,1,0)$ $\mathbf{k} = (0,0,1)$. This leads to the second definition of a quaternion, $\mathrm{q} = \mathrm{q}_0 + \mathbf{q} = \mathrm{q}_0 + \mathbf{i}\mathrm{q}_1 + \mathbf{j}\mathrm{q}_2 + \mathbf{k}\mathrm{q}_3$.

From the second definition follows the rule developed by William Rowan Hamilton, who had developed quaternions alongside Olinde Rodrigues in the 19th century, that

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{3.13}$$

Quaternions have multiple properties, which can be seen in [30, Appendix E] and more information about Quaternion operations and quaternion Algebra can be seen in [14, 19]. As it will often be used in the following chapters, emphasis is being given to the multiplication of quaternions. Defining a quaternion q as $\mathrm{q} = \mathrm{q}_0 + \mathbf{q}, \mathrm{q}_0 \in \mathbb{R}^3$ and $\mathbf{q} = \mathbf{i}\mathrm{q}_1 + \mathbf{j}\mathrm{q}_2 + \mathbf{k}\mathrm{q}_3$ and

considering (3.13), the product between two quaternions, p and q, can be calculated as

$$\begin{aligned} p \odot q =& p_0 q_0 - (p_1 q_1 + p_2 q_2 + p_3 q_3) + \\ &+ p_0(\mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3) + q_0(\mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3) + \\ &+ \mathbf{i}(p_2 q_3 - p_3 q_2) + \mathbf{j}(p_3 q_1 - p_1 q_3) + \mathbf{k}(p_1 q_2 - p_2 q_1). \end{aligned} \tag{3.14}$$

The symbol $\odot$ is used in this work for the Hamiltonian quaternion product. The standard dot and cross products of vectors in $\mathbb{R}^3$ can be used to write the quaternion multiplication in the following alternative way

$$p \odot q = p_0 q_0 - \mathbf{p} \cdot \mathbf{q} + p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \times \mathbf{q}, \tag{3.15}$$

which can also be written in matrix form as

$$p \odot q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & -p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \tag{3.16}$$

It should be noted that quaternion multiplications are not commutative.

The quaternions defined in this chapter can only represent a certain attitude but can not be used to express actual rotations. For that, unit quaternions are necessary. A unit quaternion is a quaternion whose norm is equal to one, the norm of a quaternion being its length. In literature, unit quaternions are sometimes also called *Euler–Rodrigues symmetric parameters*, *Euler symmetric parameters* or *Quaternions of rotation*. Unit quaternions can be used for performing rotations in $\mathbb{R}^3$ similarly as rotation matrices can be used for rotations in $\mathbb{R}^3$. A rotation can be represented using a rotation quaternion, which can be defined as follows

Being $q = q_0 + \mathbf{q}$ an unit quaternion with $||q||= 1$, $||q||$ being the norm of quaternion q, it is also called a rotation quaternion if it can be written as

$$q = \cos\left(\frac{\Theta}{2}\right) + \mathbf{u}\sin\left(\frac{\Theta}{2}\right),\ \mathbf{u} \in \mathbb{R}^3, -\pi < \Theta < \pi \tag{3.17}$$

in which $\mathbf{u}$ is the rotation axis for a given rotation angle Θ. It's important to highlight that the rotation parameter accounts for only half of the intended rotation angle. [14]

From (3.17) it can be deducted, that for a vector $\mathbf{v} \in \mathbb{R}^3$, if rotated about an angle Θ, we receive a vector $\mathbf{w} \in \mathbb{R}^3$ by evaluating

$$\mathbf{w} = \bar{q}\mathbf{v}q = \mathbf{Q}_i^b \mathbf{v}, \tag{3.18}$$

where $q = \cos\left(\frac{\Theta}{2}\right) + \mathbf{u}\sin\left(\frac{\Theta}{2}\right)$ is the rotation Quaternion and $\bar{q}$ is the conjugate of q, $\bar{q} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_2\mathbf{k}$.

$\mathbf{Q}_i^b$ is given by

$$\mathbf{Q}_i^b = \left[\begin{array}{c:c:c} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ \hdashline 2q_1q_2 - 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 + 2q_0q_1 \\ \hdashline 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{array}\right]. \tag{3.19}$$

The quaternion rates equation, or the equation used to propagate the state of a quaternion in time, is given by

$$\begin{aligned}\frac{\mathrm{dq}}{\mathrm{dt}} &= \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \\ &= \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}\end{aligned} \tag{3.20}$$

and is explained in great detail in [19, p. 263].

A quaternion multiplication can be used to express a series of attitude transformations. Let's say, an attitude transformation is performed by using first unit quaternion q and a successive attitude transformation is performed by using unit Quaternion p. The overall attitude transformation is then given by $\mathrm{q} \odot \mathrm{p}$ using the quaternion multiplication defined earlier in this section.

The usage of quaternions for attitude representation has some strong advantages compared to rotation matrices. Firstly, they require only 4 instead of 9 elements for its representation. Secondly, quaternions do not have a singularity as Euler angles do have when propagating in time as can be seen when comparing Equations (3.11) and (3.20). What can be also noted is that the propagation in time of quaternions does need less computational effort, as it only involves multiplications of scalar values. The time propagation of Euler angles in comparison is more computational expensive, as it needs multiple sin, cos and tan values. In total, the attitude representation using quaternions needs 16 multiplications and 12 addition. In contrast, the attitude representation using Euler angles needs 27 multiplications and 18 additions. [22, 28] One major drawback of using quaternions is that they do not offer an intuitive meaning as Euler angles do. To solve this, one can convert a quaternion to its Euler angle when needed. The derivation for this can be found in [5] and is given by

$$\begin{bmatrix} \varphi \\ \Theta \\ \psi \end{bmatrix} = \begin{bmatrix} \mathrm{atan2}(2(q_0q_1 + q_2q_3), 1{-}2(q_1^2 + q_2^2)) \\ -\frac{\pi}{2} + 2\mathrm{atan2}(\sqrt{1 + 2(q_0q_2{-}q_1q_3)}, \sqrt{1 - 2(q_0q_2{-}q_1q_3)}) \\ \mathrm{atan2}(2(q_0q_3 + q_1q_2), 1{-}2(q_2^2 + q_3^2)) \end{bmatrix}. \tag{3.21}$$

The derivation for the Euler Angles to quaternion conversion will not be used in this work and thus not explained here, but it can be found in [4, 13].

3.3 Dynamic modeling

To be able to implement different algorithms and controllers for the determination of the Center of Rotation (CoR) to Center of Mass (CoM) offset, it is necessary to model the system's kinematics accurately. The dynamic model used in this work translates torques acted upon the satellite simulator into angular velocities and accelerations of it. As the simulator is fixed about the CoR, no translations have to be taken into account.

Some of the most used models are based upon research conducted by [40] and [7]. As presented in [30, Section 3.2, Figure 3.9], the modeling differences of the angular velocities between the models are in the magnitude of $10^{-5}\,\frac{\text{rad}}{\text{s}}$.

In this work, the dynamic model of [7] will be used as it was adopted by [17]. It is based on the so-called Euler Equations of Motion (EEoM),

$$\begin{aligned} \frac{d\mathbf{H}_0}{dT} &= \mathbf{M}_0, \\ \dot{\mathbf{H}}_0 &= \dot{\mathbf{H}}_0 + \omega \times \mathbf{H}_0 = \mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega, \\ \mathbf{M}_0 &= \mathbf{r} \times \mathbf{F}_\text{g} = \mathbf{r} \times \text{m}\mathbf{g}. \end{aligned} \tag{3.22}$$

This set of equations defines the motion of the satellite simulator about its CoR. In this equation, $\mathbf{H}_0$ is the angular momentum, $\mathbf{M}_0$ is the applied torque to the simulator, ω is the angular velocity of the simulator, $\mathbf{J}$ is the inertia tensor about the CoR of the testbed. Finally, $\mathbf{F}_\text{G}$ is the gravitational force acting upon the satellite simulator, and $\mathbf{r}$ is the unbalance vector or the offset vector from CoR to CoM, upon which the gravitational force will act. In short, in the model defined by Equations (3.22), the only external force acting upon the satellite simulator is the gravitational force. This force, $\mathbf{F}_\text{G}$, creates a torque offset form the CoR by $\mathbf{r}$, which is the CoM. Due to the applied torque, the angular momentum of the satellite simulator is not constant but changes over time. The total torque acted upon the simulator, $\mathbf{M}_0$, can be separated into

$$\mathbf{M}_0 = \mathbf{M}_\text{aero} + \mathbf{M}_\text{tor} + \mathbf{M}_\text{G} \tag{3.23}$$

in which $\mathbf{M}_\text{aero}$ is the aerodynamic torque, $\mathbf{M}_\text{tor}$ is the torque generated by internal momentum exchangers (e.g. reaction wheels, moment control gyroscopes) and $\mathbf{M}_\text{G}$ the torque due to gravity. Of these components, only $\mathbf{M}_\text{G}$ is included into the simulation as $\mathbf{M}_\text{aero}$ is way less significant than $\mathbf{M}_\text{G}$ and as the TOM CubeSat will not use any internal momentum exchange actuators. $\mathbf{M}_\text{G}$ can be calculated as follows

$$\mathbf{M}_\text{G} = \mathbf{r} \times \underbrace{\text{m}\mathbf{g}_\text{b}}_{\mathbf{F}_\text{G}} = \text{mg} \begin{bmatrix} -\text{r}_\text{y} \cos\varphi \cos\Theta + \text{r}_\text{z} \sin\varphi \cos\Theta \\ \text{r}_\text{x} \cos\varphi \cos\Theta + \text{r}_\text{z} \sin\Theta \\ -\text{r}_\text{x} \sin\varphi \cos\Theta - \text{r}_\text{y} \sin\Theta \end{bmatrix} \tag{3.24}$$

in which $\mathbf{g}_\text{b}$ is the gravity vector in the body frame.

According to [40], the derivative of the angular velocity of the satellite simulator is given by

$$\dot{\omega} = \begin{bmatrix} \frac{\text{mg}}{\text{J}_\text{x}} \left(-\text{r}_\text{y} \cos\varphi \cos\Theta + \text{r}_\text{z} \sin\varphi \cos\Theta\right) \\ \frac{\text{mg}}{\text{J}_\text{y}} \left(\text{r}_\text{x} \cos\varphi \cos\Theta + \text{r}_\text{z} \sin\Theta\right) \\ \frac{\text{mg}}{\text{J}_\text{z}} \left(-\text{r}_\text{x} \sin\varphi \cos\Theta - \text{r}_\text{y} \sin\Theta\right) \end{bmatrix}. \tag{3.25}$$

As the inertia tensor given by the CAD model is given in relation to the CoM, but the EEoM needs the inertia tensor about the CoR, this value has to be augmented to be useable in the EEoM. In other words, the following Equation can be used to express the relationship between the inertia about the CoR ($\mathbf{J}_0$) and the inertia about the CoM ($\mathbf{J}_\text{G}$):

$$\mathbf{J}_0 = \mathbf{J}_\text{G} + \mathbf{J}_+ \tag{3.26}$$

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

$$\mathbf{J}_+ = \begin{bmatrix} m(y_G^2 + z_G^2) & -mx_G y_G & -mx_G z_G \\ -mx_G y_G & m(x_G^2 + z_G^2) & -mz_G y_G \\ -mx_G z_G & -mz_G y_G & m(x_G^2 + y_G^2) \end{bmatrix}. \tag{3.27}$$

This is based upon the parallel axis theorem, which allows the mapping of the inertia about the CoM to an arbitrary point. [1]

The dynamic model implemented into a python simulation will be explained in detail in Section 4.1. The resulting attitude and angular velocities for a simulation time of 100 seconds with simulation parameters taken from the TOM CubeSat can be seen in Figure 4.2.

3.4 Batch estimation, filtering and adaptive control for state estimation

There are multiple options for determining the CoR to CoM offset vector. [17] categorized the different approaches into the following groups: Batch estimation techniques, Filtering techniques and active control techniques. There is also the possibility of manually balancing the satellite simulator, but this is a rather time-consuming task, and its precision is much dependent on the expertise of the person conducting the manual balancing. Also, the minimum recorded torque value after manual balancing found in the conducted research was that of about 0.01 Nm [24, 25]. Meanwhile, the gravitational torque acting upon a CubeSat in Low Earth Orbit ranges from about 10^{-4} Nm [11] down to 10^{-6} Nm [3], depending on the specific CubeSat. Thus, manual balancing does not meet the precision needed to accurately simulate the environment in Low Earth Orbit.

Batch estimation algorithms such as the Least Square Method (LSM) or the torque method work iteratively. Filter methods employ methods such as the Kalman filter or one of its derivatives to approximate the offset vector. Finally, active control techniques such as a non-linear controller approximate the offset vector at the same time as they correct it.

In the following sections, algorithms used later in the simulation will be presented alongside some other approaches of interest.

3.4.1 Batch estimation techniques

Batch estimation techniques estimate the offset vector either in a single step or iteratively. In both cases, data is recorded and only after the data recording is ended the measurements are processed resulting in two-step techniques: sampling and estimating. The sampling is done to be able to solve a linear system describing the physical system present - with known parameters from the measurements - together with the unknown, to be estimated parameters. This two-step process of first measuring the data and then processing it later is the per se definition of an offline algorithm. Thus, these estimation methods, either by processing the data in one single step or iteratively, are both offline algorithms. [3]

As one of the requirements stated in chapter 2 was that the algorithm has to be an online algorithm as to keep the total testing time low, the batch estimation techniques can not be

considered further to estimate the CoR to CoM offset vector. However, as they are important to discuss because they lay the foundation for the online algorithms presented in sections 3.4.2 and 3.4.3, they are still presented.

3.4.1.1 Least Square Method

Since equation (3.25) directly relates the CoM offset with other known values such as the angular velocity, which can be measured by an Inertial Measurement Unit (IMU) or other device, and the Euler Angles, which can also be calculated by using data from the IMU or an alternate instrument, it can be used to approximate the CoM offset vector. As the IMU or similar hardware is used intensely for this approach, their measurement noises can not be ignored. One possibility is to use the LSM, which uses many data sets to reduce the error in the estimation. In this way, Least Square Method (LSM) is used to minimize the mean square error of the measured data and is able to estimate the true offset value. [8, 27]

The first step is to discretize equation (3.25), to be able to separate the offset vector $\mathbf{r} = [\mathrm{r_x}, \mathrm{r_y}, \mathrm{r_z}]^\mathrm{T}$.

$$\begin{gathered} \Delta\Omega = \Phi\mathbf{r} \\ \begin{bmatrix} \Delta\omega_\mathrm{x}^\mathrm{k} \\ \Delta\omega_\mathrm{y}^\mathrm{k} \\ \Delta\omega_\mathrm{z}^\mathrm{k} \end{bmatrix} = \begin{bmatrix} 0 & \Phi_{12} & \Phi_{13} \\ \Phi_{21} & 0 & \Phi_{23} \\ \Phi_{31} & \Phi_{32} & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r_x} \\ \mathrm{r_y} \\ \mathrm{r_z} \end{bmatrix} \end{gathered} \tag{3.28}$$

The subscript denotes the time step at which the measurements were taken. The values for $\Phi\_\_$ are given as following

$$\begin{aligned} \Phi_{12} &= \frac{-\mathrm{mg}\Delta\mathrm{t}}{2\mathrm{J_{xx}}} \left((\cos\varphi\cos\Theta)_2 + (\cos\varphi\cos\Theta)_1\right) \\ \Phi_{13} &= \frac{\mathrm{mg}\Delta\mathrm{t}}{2\mathrm{J_{xx}}} \left((\sin\varphi\cos\Theta)_2 + (\sin\varphi\cos\Theta)_1\right) \\ \Phi_{21} &= \frac{-\mathrm{mg}\Delta\mathrm{t}}{2\mathrm{J_{yy}}} \left((\cos\varphi\cos\Theta)_2 + (\cos\varphi\cos\Theta)_1\right) \\ \Phi_{23} &= \frac{\mathrm{mg}\Delta\mathrm{t}}{2\mathrm{J_{yy}}} \left((\sin\Theta)_2 + (\sin\Theta)_1\right) \\ \Phi_{31} &= \frac{-\mathrm{mg}\Delta\mathrm{t}}{2\mathrm{J_{zz}}} \left((\sin\varphi\cos\Theta)_2 + (\sin\varphi\cos\Theta)_1\right) \\ \Phi_{32} &= \frac{-\mathrm{mg}\Delta\mathrm{t}}{2\mathrm{J_{yy}}} \left((\sin\Theta)_2 + (\sin\Theta)_1\right) \end{aligned} \tag{3.29}$$

Equation (3.29) can be derived from equation (3.25) by integrating over small timesteps as shown in [39]. One can then over-sample equation (3.28) and then calculate the offset vector as

$$\mathbf{r} = \left[\Phi_\mathrm{aug}^\mathrm{T}\Phi_\mathrm{aug}\right]^{-1}\Phi_\mathrm{aug}^\mathrm{T}\Delta\Omega_\mathrm{aug} \tag{3.30}$$

in which the subscript *aug* means that additional rows are added to the vectors and matrices in equation (3.28) to achieve the oversampling. [31] states, that 500 measurements should

be sufficient to approximate the CoM offset vector using the LSM method. There it is also stated, that the total time until a satisfactory value was found took about 50 seconds of taking measurements and calculating the approximation. As LSM is an offline algorithm, it can not be considered for the offset vector estimate as one of the main requirement for the estimation algorithm was stated in chapter 2 to being an online algorithm. Other negative factors are, that before the approximation can start, many measurements have to be taken and saved, which could be difficult for the small memory capacity of most CubeSats and a sufficiently fast-working CPU and On-board computer (OBC) is needed so that the CubeSat is not deadlocked while calculating the offset vector.

3.4.1.2 Torque method

Another batch estimation technique is that of the torque method. This method has the advantage that it is not only able to approximate the CoM offset vector, but can simultaneously approximate the inertia tensor of the satellite simulator as well. To do this, the dynamic model in equation (3.22) in arranged in such a way that both the CoM and the inertia tensor can be estimated. The major drawback of this approach is that it relies on the CubeSat having integrated momentum exchange devices such as reaction wheels present. For TOM, this is not the case for the development model used at the time of writing this thesis. This is the reason why this approach will not be considered for the simulation, but should be investigated further at a later time. Following, a short overview of this approach is given based on [8].

Firstly, equations (3.22) are modified so that the torque generated by the momentum exchange devices $\mathbf{M}_*$ is included,

$$\dot{\mathbf{H}}_0 = \mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega + \dot{\mathbf{M}}_* + \omega \times \mathbf{M}_*. \tag{3.31}$$

This equation can then be written as

$$\mathbf{J}\dot{\omega} + \omega \times \mathbf{J} = \dot{\mathbf{M}}_* - \omega \times \mathbf{M}_* - [\mathbf{g}\times]\mathrm{m}\mathbf{r} \tag{3.32}$$

by equating the first and third equation of (3.22) and using the matrix convention $[\mathbf{v}\times]$ as

$$[\mathbf{v}\times] = \begin{bmatrix} 0 & -\mathrm{v}_3 & \mathrm{v}_2 \\ \mathrm{v}_3 & 0 & -\mathrm{v}_1 \\ -\mathrm{v}_2 & \mathrm{v}_1 & 0 \end{bmatrix} \text{ for any given vector } \mathbf{v} = [\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3]^{\mathrm{T}}. \tag{3.33}$$

Now, equation (3.32) can be written as

$$\left[\dot{\Omega} + \omega \times \Omega\,[\mathbf{g}\times]\right] \begin{bmatrix} \tilde{\mathbf{J}} \\ \mathrm{m}\mathbf{r} \end{bmatrix} = -\dot{\mathbf{M}}_* - \omega \times \mathbf{M}_* \tag{3.34}$$

for which the terms Ω, $\tilde{\mathbf{J}}$ and $[\mathbf{g}\times]$ are given as

$$\Omega = \begin{bmatrix} \omega_1 & 0 & 0 & -\omega_2 & -\omega_3 & 0 \\ 0 & \omega_2 & 0 & -\omega_1 & 0 & -\omega_3 \\ 0 & 0 & \omega_3 & 0 & -\omega_1 & -\omega_2 \end{bmatrix}, \tag{3.35}$$

$$\tilde{\mathbf{J}} = \left[\mathrm{J_x}\ \mathrm{J_y}\ \mathrm{J_z}\ \mathrm{J_{xy}}\ \mathrm{J_{xz}}\ \mathrm{J_{yz}}\right]^{\mathrm{T}}, \tag{3.36}$$

$$[\mathbf{g}\times] = \begin{bmatrix} 0 & \cos\varphi\cos\Theta & \sin\varphi\cos\Theta \\ \cos\varphi\cos\Theta & 0 & \sin\Theta \\ -\sin\Phi\cos\Theta & -\sin\Theta & 0 \end{bmatrix}. \tag{3.37}$$

Equation (3.34) can then be written as

$$\underbrace{\left[\Omega + \int_{\mathrm{t_0}}^{\mathrm{t}} \omega \times \Omega \ dT\right]}_{\Phi} \underbrace{\begin{bmatrix}\tilde{\mathbf{J}} \\ \mathrm{m}\mathbf{r}\end{bmatrix}}_{\mathbf{x}} = \underbrace{-\mathbf{M}_* - \int_{\mathrm{t_0}}^{\mathrm{t}} \omega \times \mathbf{M}_* \ dT}_{\mathbf{S}} \tag{3.38}$$

in which the integration over small time-steps of the angular rates is a measure to reduce the measurement noise.

To obtain vector $\mathbf{x}$, which contains the to be estimated offset vector $\mathbf{r}$, equation (3.38) can be solved using the least squares approximation method in similar manner to equation (3.30)

$$\mathbf{x} = \left(\Phi_{\mathrm{aug}}^{\mathrm{T}}\Phi_{\mathrm{aug}}\right)^{-1}\Phi_{\mathrm{aug}}^{\mathrm{T}}\,\mathbf{S}_{\mathrm{aug}}. \tag{3.39}$$

3.4.2 Filtering methods

A different approach that is also based on iterative or recursive estimation is that of using a Kalman Filter. In contrast to the batch estimation techniques presented in the preceding section, the filter methods are able to analyze data as it is recorded. There are many extensions of the basic Kalman Filter, such as the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) or Constrained Kalman Filter (CKF). For the estimation of the CoM offset vector, a basic Kalman Filter and for the vertical component of the adaptive control technique an UKF will be considered.

3.4.2.1 Kalman Filter

Following, a brief overview of the (basic) Kalman Filter equations and notation is given based on [37] and [30, Appendix G] as they will be used in the simulation later on. For a more in-depth introduction, [32] could be considered.

The Kalman Filter is a set of mathematical equations which provide an efficient recursive solution of the least square method. It was first published in 1960 by R. E. Kalman [16]. A process is estimated by the Kalman Filter by a kind of feedback controller. First, the filter estimates the process state at a certain timestep. Then, feedback is added in the form of - to some extent - noisy measurements of some of the state variables. Because of this separation, the equations of the Kalman Filter can be divided into *time update* and *measurement update* equations or phases. The *time update* equations project the current state forward in time to obtain a so-called *a priori* estimate for the next time step. This is why they are also sometimes referred to as *predictor* equations. The *measurement update* equations are used as feedback,

they incorporate a new measurement to be able to estimate an improved *a posteriori* estimate. Together, they form a *predictor-corrector* algorithm.

Firstly, the state to be estimated has to be described in the following format,

$$\mathbf{x}_k = \mathbf{F}_{k-1}\,\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\,\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{3.40}$$

$$\mathbf{y}_k = \mathbf{H}_k\,\mathbf{x}_k + \mathbf{v}_k\,. \tag{3.41}$$

$\mathbf{x}_k$ describes the system state at timestep k, while $\mathbf{y}_k$ is the measurement at timestep k. $\mathbf{u}$ is the system input, $\mathbf{w}$ is the process noise, $\mathbf{v}$ the measurement noise and matrices $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ are dependent on the specific system equations and complete the filter design. The noises, $\mathbf{w}$ and $\mathbf{v}$ are assumed to be independent of each other and can be modeled as white-noise, normal probability distributions functions.

Matrix $\mathbf{F}$ relates the state at timestep *k-1* to the next state k. Later, the covariance matrices $\mathbf{Q}$ of $\mathbf{w}$ and $\mathbf{R}$ of $\mathbf{v}$ will be used as well as the system state covariance matrix $\mathbf{P}$.

With the system modeled by equations (3.40) and (3.41), the *a posteriori*, or initial state estimate and initial state covariance matrix must be initialized. For the Kalman Filter implementation in the simulation the initial state is defined by the simulation's initial values. For an implementation on an embedded system, the initial values should be derived from initial sensor readings. Generally, the variables should be initialized as

$$\begin{aligned}\hat{\mathbf{x}}_0^+ &= \mathrm{E}(\mathbf{x}_0)\\ \mathbf{P}_0^+ &= \mathrm{E}\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)(\mathbf{x}_0 - \hat{\mathbf{x}_0}^+)^{\mathrm{T}}\right].\end{aligned} \tag{3.42}$$

The function E(_) denotes the expected value and the hat symbol $\widehat{\ }$ indicates the estimated variable. The superscripts $\_^-$ denotes the *a priori* and $\_^+$ denotes the *a posteriori* values of the referred quantities. As an example, $\_^-$ refers to a value before a measurement update and $\_^+$ to a value after a measurement update.

The a priori state estimate is given as

$$\hat{\mathbf{x}}_k^- = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \tag{3.43}$$

and the a posteriori state estimate as

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-\right). \tag{3.44}$$

$\mathbf{K}_k$ is the so-called Kalman Filter gain. It adjusts the predicted state based on the difference between the predicted measurement and the actual measurement. The larger the Kalman gain is, the more weight is given to the measurement. It is computed to a larger number, if the measurements are relatively consistent and therefore have a lower uncertainty. Vice versa, the smaller the Kalman gain is, the more the prediction is trusted. The gain can be computed as

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^-\,\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\,\mathbf{P}_k^-\,\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1}\\ &= \mathbf{P}_k^+\,\mathbf{H}_k^{\mathrm{T}}\,\mathbf{R}_k^{-1}.\end{aligned} \tag{3.45}$$

$\mathbf{P}_k^-$ and $\mathbf{P}_k^+$, the a priori and a posteriori error covariance matrices, are given as

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\,\mathbf{P}_{k-1}^+\,\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \tag{3.46}$$

and

$$\begin{aligned}\mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_k^-\,(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\,\mathbf{R}_k\,\mathbf{K}_k^T \\ &= \left[(\mathbf{P}_k^-)^{-1} + \mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{H}_k\right]^{-1} \\ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_k^- .\end{aligned} \tag{3.47}$$

To sum the Kalman Filter equations up, the first step is to compute the Kalman Filter gain as given in equation (3.45). Then, a measurement is taken, and the measurement is transformed into the system state by equation (3.41). After that, an a posteriori state estimate is calculated by using equation (3.44). Finally, the a posteriori state error covariance is obtained by equation (3.47). For the next Kalman Filter iteration, the a posteriori state estimate transforms into the new a priori state estimate. This recursive approach is one of the advantages of using a Kalman Filter, as data can be taken and the state can be updated while a dynamic process is going on. One drawback of using the basic Kalman Filter is that it works only for linear systems. For non-linear systems, an EKF or UKF should be used instead. The Kalman Filter implementation to approximate the CoM used in the satellite simulator simulation will be explained in more detail in section 4.2.

3.4.2.2 Unscented Kalman Filter

As stated in section 3.4.2.1, one major drawback of using the basic Kalman Filter is that it only works well for linear systems. For non-linear systems, an EKF or UKF should be used instead, as they also work for non-linear systems.

In the EKF the non-linear equations of the system are linearized around the current state in order to be able to use the Kalman Filter equations by using first order Taylor approximations [37]. For the UKF, this is achieved not by linearization around the current state, but by estimation of a probability distribution of multiple possible estimated states that are generated together with a weight and then the weighted mean of all possible estimated states is calculated. In simpler terms, the UKF generates a set of so-called sigma points and then builds the average of all sigma points to estimate the next state. To calculate the statistics and estimated state of a variable, the so called Unscented Transformation (UT) is used, which was developed by Simon Julier and Jeffrey Uhlmann in 2004 [15].

In this work, the UKF is chosen as it promises better non-linearity handling as the EKFs linearization. The EKF utilizes a first order Taylor approximation could lead to bigger expected inaccuracies when compared to the sampling technique of the UKF [10, 33].

Following a very brief overview of the most important UKF equations and notation is given based on [15, 35, 36] and [30, Appendix J].

Firstly, the general model of the UKF is given by the following non-linear equations,

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_K \tag{3.48}$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \tag{3.49}$$

For this general case, the process noise $\mathbf{w}_k$ and the measurement noise $\mathbf{v}_k$ might be additive or not. They both are assumed to being zero mean Gaussian noises with its covariance given by the matrices $\mathbf{Q}$ and $\mathbf{R}$, respectively. This means, that the noise probability distribution follows a Gaussian distribution and centers around a mean of zero on average.

The initial a posteriori state and initial a posteriori state covariance matrix are given similarly to the initial values of the basic Kalman Filter in equation (3.42) as

$$\begin{aligned} \hat{\mathbf{x}}_0^+ &= \mathrm{E}(\mathbf{x}_0) \\ \mathbf{P}_0^+ &= \mathrm{E}\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)(\mathbf{x}_0 - \hat{\mathbf{x}}_0^+)^{\mathrm{T}}\right] \end{aligned} \tag{3.50}$$

In the *time update* phase, the propagation of the system state and system state covariance takes place using only previously collected information and the system dynamics. This step is divided into four sub-steps **A**-**D**. Together, they form the UT.

A. In the first step, the sigma points $\mathbf{x}_{k-1}^{(i)}$ are generated. The superscript (i) denotes the ith sigma point of all 2n sigma points generated, n being the number of states in the state vector $\mathbf{x}$. The choice of the amount of sigma points depends on the system present. In the original design of the UKF by Uhlmann, 2n sigma points were created. Different implementations use different amount of sigma points. The sigma points are calculated as

$$\begin{aligned} \hat{\mathbf{x}}_{k-1}^{(i)} &= \hat{\mathbf{x}}_{k-1}^+ + \tilde{\mathbf{x}}^{(i)},\ i = 1, ..., 2n \\ \tilde{\mathbf{x}}^{(i)} &= \left(\sqrt{\frac{n}{1 - W^{(i)}}\mathbf{P}_{k-1}^+}\right)^{\mathrm{T}},\ i = 1, ..., n \\ \tilde{\mathbf{x}}^{(n+i)} &= -\left(\sqrt{\frac{n}{1 - W^{(i)}}\mathbf{P}_{k-1}^+}\right)^{\mathrm{T}},\ i = 1, ..., n \end{aligned} \tag{3.51}$$

around the previous a posteriori estimated state $\hat{\mathbf{x}}_{k-1}^+$. The weight factor $W^{(i)}$ is calculated as

$$W^{(i)} = \frac{1 - W}{2n}, \text{ for all } i = 1, ..., 2n \tag{3.52}$$

and meets this criterion

$$\sum_{i=0}^{2n} W^{(i)} = 1. \tag{3.53}$$

The weight $W^{(i)}$ ranges from $-1 \leq W^{(i)} \leq 1$. It describes the position of the sigma points. Weights smaller than zero tend to be closer to the origin while weights greater than zero are more likely to be further from the origin. In different UKF implementations, the weight might be individually calculated or the same for all sigma points, depending on the general filter design.

B. After generating the sigma points, they have to be propagated in time which turns them into $\hat{\mathbf{x}}^{(i)}$ vectors. This is done by using the non-linear function $\mathbf{f}_{k-1}()$,

$$\hat{\mathbf{x}}_k^{(i)} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_k) \tag{3.54}$$

C. Now, the individual sigma points are combined into a single a priori estimated state $\hat{\mathbf{x}}_k^-$,

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2n} W^{(i)}\, \hat{\mathbf{x}}_k^{(i)}\,. \tag{3.55}$$

D. Alongside the a priori state estimate, the a priori state covariance $\mathbf{P}_k^-$ matrix must be determined as well:

$$\mathbf{P}_k^- = \sum_{i=0}^{2n} W^{(i)}(\hat{\mathbf{x}}_k^{(i)} - \hat{\mathbf{x}}_k^+)(\hat{\mathbf{x}}_{k-1}^{(i)} - \hat{\mathbf{x}}_k^+)^T + \mathbf{Q}_{k-1} \tag{3.56}$$

The second phase is the *measurement update* phase in which the a priori state estimation is updated using the output equation of the system. It is again divided into four sub-steps **A** through **D**.

A. The sigma points are first recalculated using the a priori estimated state $\hat{\mathbf{x}}_{k-1}^+$ given by the time update phase instead of using the a posteriori estimated state $\hat{\mathbf{x}}_{k-1}^-$ as done before in step A of the first phase.

$$\begin{aligned}
\mathbf{x}_{k-1}^{(i)} &= \mathbf{x}_{k-1}^- + \tilde{\mathbf{x}}^{(i)},\ i = 1, ..., 2n \\
\tilde{\mathbf{x}}^{(i)} &= \left(\sqrt{\frac{n}{1 - W^{(i)}}\mathbf{P}_{k-1}^+}\right)^T,\ i = 1, ..., n \\
\tilde{\mathbf{x}}^{(n+i)} &= -\left(\sqrt{\frac{n}{1 - W^{(i)}}\mathbf{P}_{k-1}^+}\right)^T,\ i = 1, ..., n
\end{aligned} \tag{3.57}$$

B. The non-linear output equation $\mathbf{h}_k()$ is used to transform the sigma points into $\hat{\mathbf{y}}_k^{(i)}$ vectors as given by

$$\hat{\mathbf{y}}_k^{(i)} = \mathbf{h}_k(\hat{\mathbf{x}}_k^{(i)}). \tag{3.58}$$

C. The predicted vectors $\hat{\mathbf{y}}_k^{(i)}$ are then combined into a predicted measurement at time-step k alongside the covariance of the predicted measurement $\mathbf{P}_y$ and the cross-variance between $\hat{\mathbf{x}}_k^-$ and $\hat{y}_k$, $\mathbf{P}_{xy}$, as

$$\hat{\mathbf{y}} = \sum_{i=0}^{2n} W^{(i)}\hat{\mathbf{y}}_k^{(i)} \tag{3.59}$$

$$\mathbf{P}_y = (\hat{\mathbf{y}}^{(i)} - \hat{\mathbf{y}}_k)(\hat{\mathbf{y}}^{(i)} - \hat{\mathbf{y}}_k)^T + \mathbf{R}_k \tag{3.60}$$

$$\mathbf{P}_{xy} = (\hat{\mathbf{x}}^{(i)} - \hat{\mathbf{x}}_k)(\hat{\mathbf{y}}^{(i)} - \hat{\mathbf{y}}_k)^T \tag{3.61}$$

D. Finally, the basic Kalman Filter equation are used to obtain the a posteriori state estimate $\hat{\mathbf{x}}_k^+$ and $\mathbf{P}_k^+$,

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_{xy}\mathbf{P}_y^{-1} \\
\hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k) \\
\mathbf{P}_k^+ &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{P}_y\,\mathbf{K}_k^T
\end{aligned} \tag{3.62}$$

3.4.3 Adaptive control algorithm

Different from the approaches shown in sections 3.4.2.1 and 3.4.2.2 in which the basic Kalman Filter and Unscented Kalman Filter were presented, is the approach of an adaptive control algorithm. This technique is based on work done by [7]. In it, the proposed algorithm utilizes three sliding masses, which are controlled by actuators. The masses are moved to change the mass properties of the simulator in such a way that it eliminates the offset between the CoR and the CoM. Each mass moves along one of the body axes. One problem of that approach is that the torque generated by the balance masses is physically restricted in the direction perpendicular to the gravity field. One solution to this problem is to use a two-step control system. The first step compensates the offset in the transversal directions, meaning the directions perpendicular to the gravity field. The second step utilizes a Kalman Filter to estimate the remaining offset in the Z-axis and can then be compensated after.

One major advantage of this balancing procedure is that the offset vector is iteratively estimated and corrected at the same time, offering a time-efficient approach. [7] serves as the primary source used for the information presented in the subsequent chapters with some equations taken from [30] as well.

3.4.3.1 Extended System Dynamic Model

The first step is the transversal imbalance compensation. It is based on the law of conservation of angular momentum. Ideally, the spacecraft simulator is perfectly balanced and thus no external torques acts upon the satellite simulator, the derivative of the angular momentum is zero. But if there is an external torque acting on the satellite simulator, the derivative of the angular momentum is not zero. In the system described earlier, the only external force acting upon the satellite simulator is the gravitational force and a torque is generated due to the offset between the CoR and CoM. By nullifying the torque, the offset vector is nullified as well.

In section 3.3, the dynamic model not experiencing any external torques besides the one generated by the gravitational force and the CoR to CoM offset vector was given by the EEoM (3.22), as

$$\frac{d\mathbf{H}_0}{dT} = \dot{\mathbf{H}}_0 + \omega \times \mathbf{H_0} = \mathbf{r} \times \mathrm{m}\, \mathbf{g}_\mathrm{b}\,.$$

It is assumed that the three Moveable Mass Units (MMUs) are aligned with the axes of principal inertia and have a mass of $\mathrm{m_p}$ each. As the MMUs affect the inertia of the satellite simulator, its inertia matrix has to be recalculated each time at least one of the moveable masses moves,

$$\mathbf{J} = \mathbf{J}_0 - \mathrm{m_p} \sum_\mathrm{i} [\mathbf{r}_\mathrm{i}\times]\,[\mathbf{r}_\mathrm{i}\times] \;\text{ for } \mathrm{i} = \mathrm{x,y,z} \tag{3.63}$$

with $\mathbf{J}_0$ being the inertia matrix before the balancing procedure and $\mathbf{r}_\mathrm{i}$ represents the position of the i-th balance mass. As the masses $\mathrm{m_p}$ can be assumed to being relatively small compared to the total mass of the satellite simulator, the distance traveled per iteration being small as well as their velocity, the motion of the MMUs itself is assumed not to affect the angular momentum. Only the change in position is considered as an impact to the inertia $\mathbf{J}$.

The adaptive control algorithm as proposed by [7] uses quaternions for the attitude representation and the gravity vector in the inertia reference frame can thus be transformed into the

body reference frame by transformation matrix $\mathbf{Q}_{\mathrm{i}}^{\mathrm{b}}$ (3.19),

$$\mathbf{g}_{\mathrm{b}} = \mathbf{Q}_{\mathrm{i}}^{\mathrm{b}} \mathbf{g}_{\mathrm{i}}, \text{ with } \mathbf{g}_{\mathrm{i}} = [0, 0, -\mathrm{g}]^{\mathrm{T}} . \tag{3.64}$$

It should be noted that the notation in this work is different from the notation found in [7], as in this thesis a Quaternion is defined as $\mathbf{q} = [\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3]^{\mathrm{T}}$ and not as $\mathbf{q} = [\mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3, \mathrm{q}_4]^{\mathrm{T}}$.

3.4.3.2 Transverse plane compensation

As stated before, the control technique designed by [7] is based on the conservation of angular momentum. If no external forces act upon the simulator, the derivate of the angular momentum is zero. However, if there is an offset between the CoR and CoM, the total angular momentum of the satellite simulator changes over time, its derivative is not zero. The purpose of the algorithm is to move the MMUs in such a way, that the offset vector gets nullified and no external torque is generated and ultimately the derivative of the angular momentum reaches zero.

The torque generated by the MMUs is given as

$$\mathbf{M}_{\mathrm{r}} = \mathrm{m}_{\mathrm{p}} \sum_{\mathrm{i}} \mathbf{r}_{\mathrm{i}} \times \mathbf{g}_{\mathrm{b}} \text{ for } \mathrm{i} = \mathrm{x}, \mathrm{y}, \mathrm{z} \tag{3.65}$$

where $\mathbf{r}_{\mathrm{i}}$ is again the *i*-th balance masses position with respect to the CoR. $\mathbf{M}_{\mathrm{r}}$ is the quantity to be determined as we can calculate the MMUs positions from it. Equation (3.65) can be added to (3.22),

$$\frac{d\mathbf{H}_0}{dT} = \dot{\mathbf{H}}_0 + \omega \times \mathbf{H_0} = \mathbf{r} \times \mathrm{m}\, \mathbf{g} + \mathbf{M}_{\mathrm{r}} . \tag{3.66}$$

The control law designed in [7] is given as

$$\mathbf{M}_{\mathrm{r}} = -\,\Phi\, \hat{\Theta} - \mathrm{k}_{\mathrm{p}} \omega_{\mathrm{p}}, \tag{3.67}$$

in which Φ is function of the attitude $\Phi(\mathrm{q}) = -\mathrm{m}\, \mathbf{g}_{\mathrm{b}} \times$ and $\hat{\Theta}$ is the estimate of the unknown unbalance vector. Additional, ω_{p} is the component of the system angular velocity ω orthogonal to $\mathbf{g}_{\mathrm{b}}$. Lastly k_{p}, $0 < \mathrm{k}_{\mathrm{p}} \leq 1$, is a positive scalar value. ω_{p} can be calculated from ω by

$$\omega_{\mathrm{P}} = \mathrm{P}_{\mathrm{p}}(\mathrm{q})\omega, \tag{3.68}$$

$$\mathrm{P}(\mathrm{q}) = \left[\mathbf{I} - \frac{\mathbf{g}_{\mathrm{b}} \mathbf{g}_{\mathrm{b}}^{\mathrm{T}}}{||\mathbf{g}_{\mathrm{b}}||^2} \right] \tag{3.69}$$

in which q is the quaternion describing the bodies attitude. To update the estimate of the unknown unbalance vector $\hat{\Theta}$, its derivation is calculated between iterations by

$$\dot{\hat{\Theta}} = \Phi^{\mathrm{T}}\, \omega . \tag{3.70}$$

The derivation of equation (3.67) can be found in [7, Section II, Paragraph B] alongside a Lyapunov function, which derivative was shown to be negative semidefinite. From this follows that the control system is Lyapunov stable. This means that the control system will not show uncontrollable oscillations or diverge from the desired behavior over time but converges to an

equilibrium state [12]. For a more in depth derivation of the presented control law, [7] can be considered.

Now that it is proven that the designed control law accomplishes its designed task, the next step is to calculate the $\mathbf{r}_\mathrm{i}$ vectors from equation (3.65). For that, the vectors $\mathbf{r}_\mathrm{x}, \mathbf{r}_\mathrm{y}, \mathbf{r}_\mathrm{z}$ are grouped into a single vector $\mathbf{r}_\mathrm{mmus} = \left[\mathrm{r_{x,x}}, \mathrm{r_{y,y}}, \mathrm{r_{z,z}}\right]^\mathrm{T}$. This is possible due to the fact that in the presented system it is assumed that the MMUs move perfectly along only one of the body axes each, e.g. $\mathbf{r_x} = [\mathrm{r_{x,x}}, 0, 0]^\mathrm{T}$.

The following equation is derived from equation (3.65) using the newly defined $\mathbf{r}_\mathrm{mmus}$,

$$\mathbf{M}_\mathrm{r} = \mathrm{m_p} \left(- \mathbf{g}_\mathrm{b} \times \mathbf{r}_\mathrm{mmus}\right). \tag{3.71}$$

and can be rearranged to give a solution for $\mathbf{r}_\mathrm{mmus}$. In [6] this was proven to be

$$\mathbf{r}_\mathrm{mmus} = \frac{\mathbf{g}_\mathrm{b} \times \mathbf{M}_\mathrm{r}}{||\mathbf{g}_\mathrm{b}||^2 \mathrm{m_p}}. \tag{3.72}$$

To sum it up, one needs to find the value of $\mathbf{M}_\mathrm{r}$ and then substitute it in equation (3.72). $\mathbf{M}_\mathrm{r}$ can be determined by calculation via equation (3.67), which in turn is based on the estimated offset vector $\hat{\Theta}$, the vertical component of the angular velocity ω_p, which can be measured by using an IMU, as well as a scalar factor $\mathrm{k_p}$.

3.4.3.3 Vertical Imbalance Estimation

After the transversal imbalance compensation phase, the $\mathbf{r}_\mathrm{x}$ and $\mathbf{r}_\mathrm{y}$ components of the offset vector should be compensated to a satisfactory precision. Naturally it follows that the $\mathbf{r}_\mathrm{z}$ component should be compensated as the next step. It was already previously mentioned that only the transverse components of the offset vector can be compensated as the torque generated by the actuators and MMUs is always perpendicular to the gravity vector. To estimate the $\mathbf{r}_\mathrm{z}$ component, an UKF is utilized as described in section 3.4.2.2.

The state vector of the in [6] proposed filter is made up of the angular velocity of the satellite simulator and the parameter to be estimated $\mathrm{r_z}$,

$$\mathbf{x} = \begin{bmatrix} \omega_\mathrm{x} \\ \omega_\mathrm{y} \\ \omega_\mathrm{z} \\ \mathbf{r}_\mathrm{z} \end{bmatrix}. \tag{3.73}$$

As the MMUs are not used in the vertical imbalance estimation phase, the offset vector does not change and thus $\dot{\mathbf{r}}_\mathrm{z} = 0$ can be assumed. The dynamics of the satellite simulator as described in section 3.4.3.1 do not change except for the torque generated due to the gravitational force. After the transversal phase, it can be assumed that the $\mathbf{r}_\mathrm{x}$ and $\mathbf{r}_\mathrm{y}$ components of the offset vector were nullified and only an offset in the $\mathbf{r}_\mathrm{z}$ component remains. This leads to the updated gravitational torque

$$\mathbf{M}_\mathrm{G} = \mathbf{r} \times \mathrm{m}\mathbf{g}_\mathrm{b} = \begin{bmatrix} -\mathrm{m}g_\mathrm{b}^\mathrm{y} \mathrm{r_z} \\ \mathrm{m}g_\mathrm{b}^\mathrm{x} \mathrm{r_z} \\ 0 \end{bmatrix} \tag{3.74}$$

in which g_b^x is the x-component of the gravity vector in the body frame, and similarly g_b^y is the y-component. The dynamics of the satellite simulator system in state-space from is then given by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{w}, \tag{3.75}$$

in which $\mathbf{w}$ is the process noise and $\mathbf{f}$ is given as

$$\mathbf{f} = \begin{bmatrix} J^{-1} \left(-\omega \times \mathbf{J}\,\omega + \begin{bmatrix} -mg_b^y r_z \\ mg_b^x r_z \\ 0 \end{bmatrix} \right) \\ 0 \end{bmatrix}_{4\times 1}. \tag{3.76}$$

The output equation is given by

$$[\mathbf{y}_k]_{3\times 1} = \mathbf{H}\,\mathbf{x} + \mathbf{v} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{3\times 4} \mathbf{x}_{4\times 1} + \mathbf{v}_{3\times 1} \tag{3.77}$$

in which $\mathbf{v}$ is the measurement noise. To run the UKF, in the time update phase the sigma points $\hat{\mathbf{x}}_k^i$ can be calculated as

$$\hat{\mathbf{x}}_k^{(i)} = \hat{\mathbf{x}}_{k-1}^{(i)} + \mathbf{f} \cdot dT \tag{3.78}$$

in which T is the sampling time. The filter equations are given by

$$\begin{aligned} [\mathbf{K}_k]_{4\times 3} &= [\mathbf{P}_{xy}]_{4\times 3}[\mathbf{P}_y^{-1}]_{3\times 3} \\ [\hat{\mathbf{x}}_k^+]_{4\times 1} &= [\hat{\mathbf{x}}_k^-]_{4\times 1} + [\mathbf{K}_k]_{4\times 3}\left([\mathbf{y}_k]_{3\times 1} - [\hat{\mathbf{y}}_k]_{3\times 1}\right). \\ [\mathbf{P}_k^+]_{4\times 4} &= [\mathbf{P}_k^-]_{4\times 4} - [\mathbf{K}_k]_{4\times 3}[\mathbf{P}_y]_{3\times 3}[\mathbf{K}_k^T]_{3\times 4} \end{aligned} \tag{3.79}$$

Chapter 4

Algorithmic approximation of the Center of Mass Offset

The CoR to CoM offset will be approximated by a basic Kalman Filter and by the adaptive control algorithm presented in chapter 3.4.3. Both were chosen as they are most likely to satisfy the requirements listed in chapter 2. The Kalman Filter is an algorithmic concept that has been used in many different use cases. Its major advantages are that it is able to estimate the state of a system accurately when noise is present in the system measurements, the state estimate is updated recursively by taking new measurements which makes it applicable to online processing, and it is robust in handling noisy measurements. However, as the basic Kalman Filter does not work well with non-linear systems, its usability for the satellite simulator is limited as this system is non-linear. Another drawback is, that the Kalman Filter only approximates the offset vector, but is not able to compensate it. This ability is one of the advantages of the adaptive control algorithm. As it is able to estimate and nullify the offset vector, it is very time efficient.

First, the simulation used for the implementation of the approaches will be presented. Then, the implementations of the Kalman Filter and adaptive control estimation will be shown. The Kalman Filter is run on simulated and data taken from the physical setup. The evaluation of the simulation itself as well as the results from the CoM offset estimation will be discussed in the following chapter 5.

4.1 Simulation of a spherical air bearing satellite simulator

The following chapter presents the simulation used for the implementation and simulation of the different CoR to CoM offset vector approximation algorithms. For this, the simulations are based on the flowchart presented in figure 4.1.

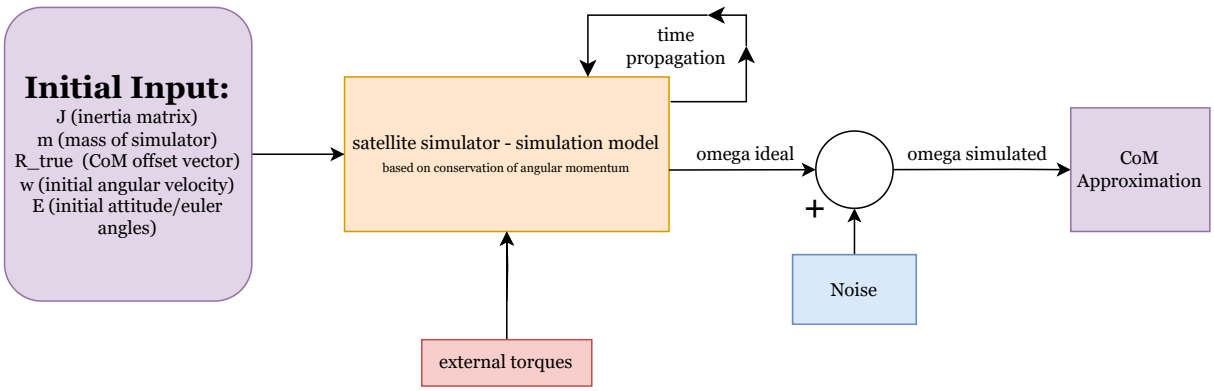


Figure 4.1: Flowchart of the simulation.

The input parameters to the simulation are the inertia matrix and total mass of the physical satellite simulator, the true offset vector which the algorithms will try to approximate, an initial attitude and an optional initial angular velocity. These parameters are then applied to the simulation, which is based on the equations presented in section 3.3. The simulation is run for 100 seconds at 10 Hz, meaning that the attitude and angular velocity is updated every 0.1 seconds.

The system state of the simulation is given as

$$\mathbf{y} = \begin{bmatrix} \varphi & \Theta & \psi & \omega_x & \omega_y & \omega_z \end{bmatrix}^T, \tag{4.1}$$

in which the first three elements are the attitude of the attitude simulator as Euler Angles and the last three elements represent its angular velocity. In the simulation code, a function is defined which calculates the EEoM as presented in (3.22). This function is then executed to calculate the derivative of the system state, which is then integrated over the time-step duration to propagate the system state forwards in time. This is repeated until the targeted simulation time is reached. The system state can be saved at each time-step, enabling the plotting of the system's attitude and angular velocity against time. The presented CoM approximation algorithms all require the input of the system's angular velocity. To create a more realistic simulation, the ideal simulated and noise-free angular velocity values are obscured by adding noise to it before being used by the approximation algorithms. The plots of the attitude and angular velocity of the TOM-CubeSat freely moving in the unbalanced condition are shown in figure 4.2.

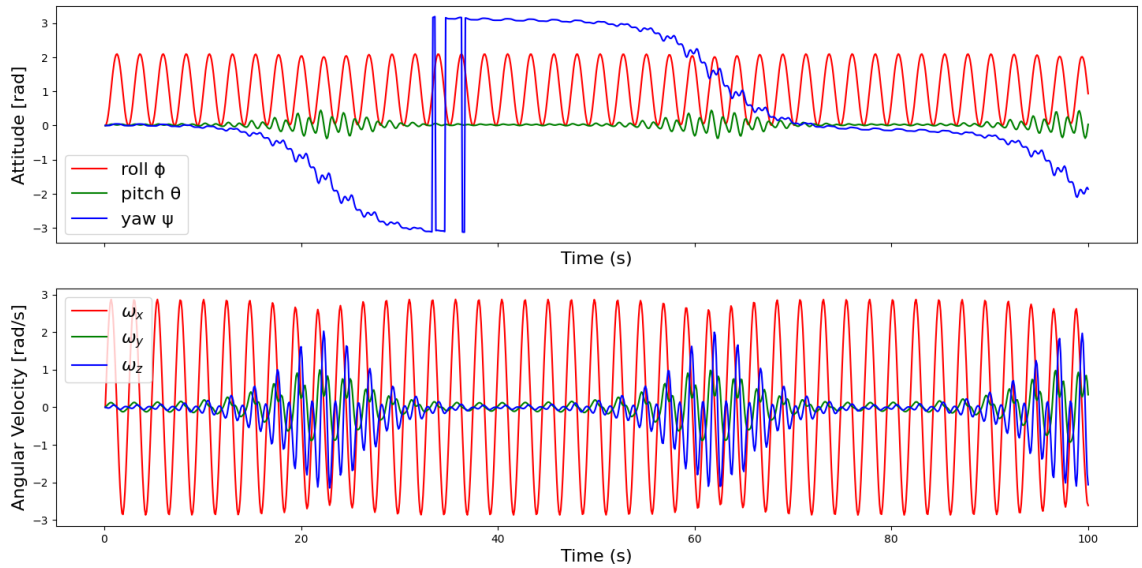


Figure 4.2: Simulated attitude and angular velocity of the TOM CubeSat moving freely on the satellite simulator.

4.2 Approximation using Kalman Filter

In the following section, the setup for taking physical data of an optically tracked object on the satellite simulator is presented. Then, the simulation is run with parameters matching this physical setup and on that simulated data the Kalman Filter based approximation is performed. After that, the same Kalman Filter is provided the recorded data.

4.2.1 Attitude Simulator - Data from Hardware

As mentioned before, the hardware setup at the ZfT was not completed as of the writing of this thesis. In order to still validate the Kalman Filter with actual measurements of the satellite simulator, an optical tracking system already present is used. The camera based solution is using a system by OptiTrack[1]. It comprises a set of four Prime^x 22 infrared cameras, which can optically track an object. Using the infrared camera measurements, the OptiTrack system outputs the orientation of the rigid body in quaternion representations, with a frequency range spanning from 20 to 1000 Hz.

Instead of using the TOM CubeSat's development model, it was chosen to utilize a more simple mass simulator. The choice was made on the basis that the mass simulator is almost completely symmetrical with an even density distribution, as the whole body is made up of the same material. This allows the CAD-model to be more accurate as it is easier to model it correctly. The mass simulator as well as its CAD-model is shown in figure 4.3 and 4.4.

[1] `https://optitrack.com/`

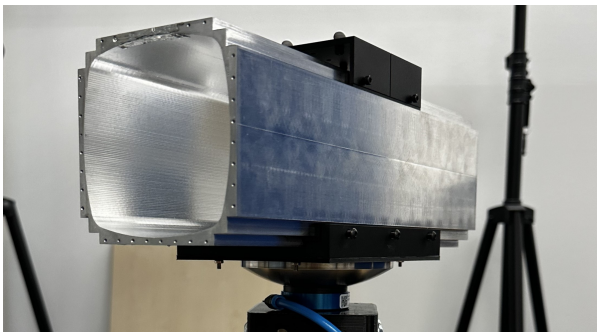


Figure 4.3: The mass simulator used for the Kalman Filter based approximation. Image Credit: ZfT.

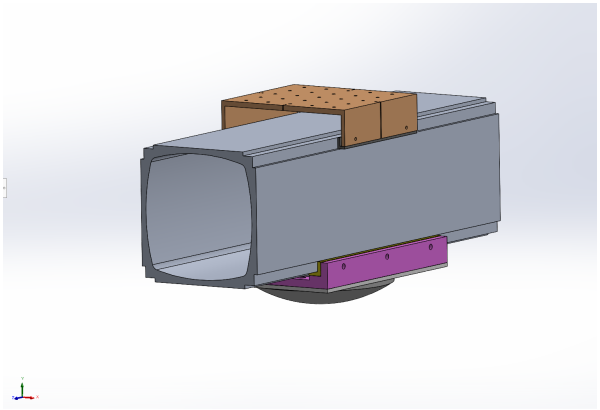


Figure 4.4: The CAD-model of the mass simulator. Image Credit: ZfT.

From the CAD-model, the inertia is calculated as

$$J_{cad} = \begin{bmatrix} 27.19 & -0.00000687 & 0.00 \\ -0.00000687 & 24.93 & 0.0000041 \\ 0.00 & 0.0000041 & 11.23 \end{bmatrix} \cdot 10^{-3} \ \text{kg} \cdot \text{m}^2 \tag{4.2}$$

and the whole system has a mass of 2.7 Kg. Also taken from the CAD-model is the CoM offset vector, as $r_{true} = [0.00, 0.00, -19.88]^T$ mm.

4.2.1.1 Hardware Setup Verification

According to OptiTrack, the 3D tracking accuracy of the four Prime^x 22 cameras is ±0.15mm [23]. To quantitatively verify this precision, a zero rate offset test was done to determine the bias. First, the cameras were calibrated, and a reference frame was set according to the instructions provided by OptiTrack. In this process, the ground plane was also defined. The ground plane was set leveled using a high precision digital level with a resulting offset measured by the digital level of 0.0121 degrees in the X-axis and 0.014 degrees in the Y-axis. Then, a rigid body was defined by a set of five optical markers. The test object was then put inside the point of view of all four infrared cameras on a static plane and data was taken continuously for 180 seconds at 40 Hz. figure A.1 in Appendix A shows the CubeSat model used for the zero rate offset tests with the five optical markers on top. Figure A.2 displays the complete setup used for the tests. To reduce optical interference, the blinds were closed and the light was switched off during the test. For the whole duration of the test, the body was kept still and movement inside the test room was kept to a minimum to not affect the measurement. After the data was taken, a lowpass filter with a windows size of four was used on the raw quaternion data to reduce the noise impact on the test.

Using this data, the error quaternions q_{error} between the first recorded quaternion, q_0 and the following quaternions q_k, $k \in \{1, 2, 3, ..., 300\}$, was calculated as

$$q_{error} = \bar{q}_0 \odot q_k. \tag{4.3}$$

In the context of the bias determination tests, the error quaternion represent the compound rotation of q_0 and q_k. In a perfect system, this compound rotation would be zero as the tracked body is not moving or rotating. However, due to noise and bias present in the tracking system, the error quaternion is not zero. For the bias estimation, a duration of 30 seconds is chosen. From this follows the choice of a maximum k of 300, as data taken at 40 Hz with a lowpass filter with a window size of four leads to converted data at 10 Hz. 10 samples per second over a total duration of 30 seconds leads to a total of 300 samples. The next step in the zero rate offset test is to convert the error quaternions into Euler Angles. Then, the average of each axis is build. This final value represents the measurement bias in each axis. For the system present, the bias is found by this method to being $\mathbf{r}_{\text{bias}} = [-2.94857210, -0.0742939369, -9.19673409]^{\text{T}}\ 10^{-5}$ rad.

Figure 4.5 displays the rate of change of the attitude of the CubeSat model during the zero rate offset test. This figure is plotted with data that has the bias removed from the lowpass filtered data. What can be seen is that the rate of change is similar over the whole test's duration. It is centered evenly around 0, which means that the optical tracking of the CubeSat model does not undergo any serious drift or other measurement errors that would affect the attitude measurement. Similarly, the attitude and angular velocities of the CubeSat do not change in any significant way as can be seen in figure 4.6.

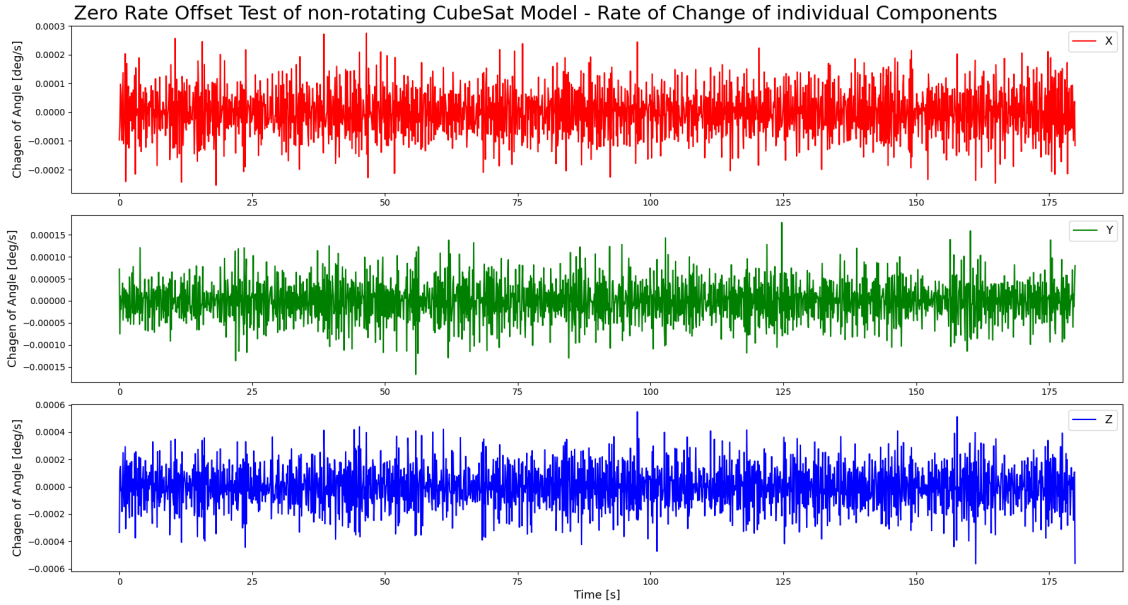


Figure 4.5: The rate of change of the attitude of the CubeSat Model while lying still. The rate of Change is calculated as the derivative of quaternion errors.

Considering the other noise factors not considered here, like the environmental conditions or the frictions of the air bearing cup and air bearing base, the obtained results suggest that the camera based setup is able to take measurements in a sufficient order that matches the expected measurements of the IMU based setup that will be used in the future.

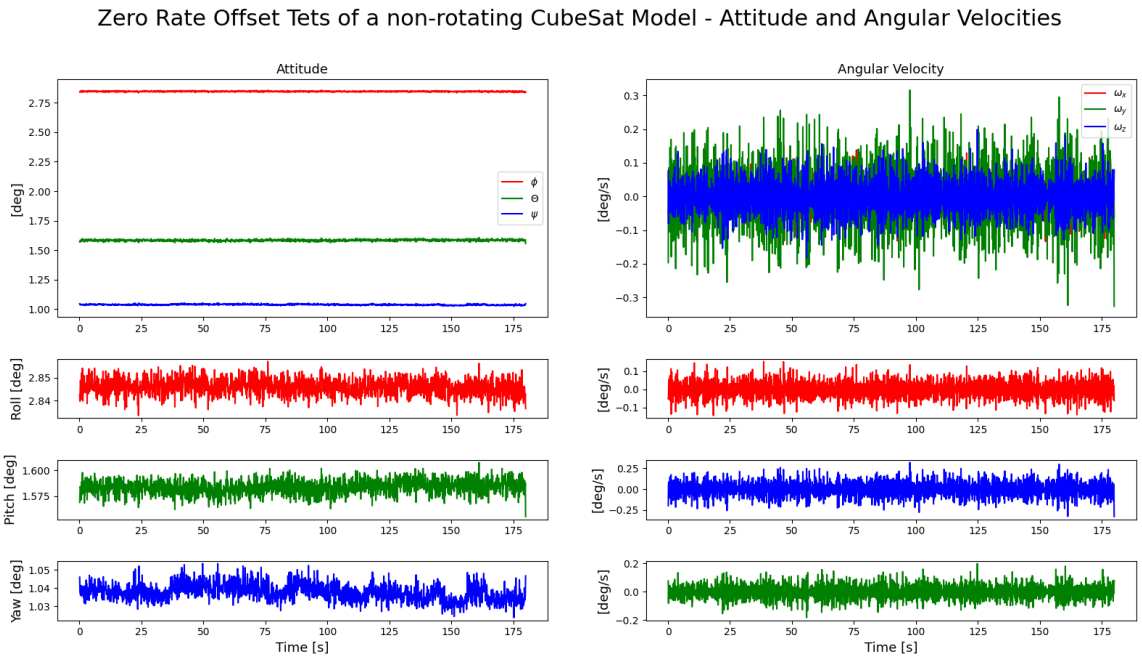


Figure 4.6: The attitude and angular velocity of the CubeSat model undergoing the zero rate offset test. No serious change in the attitude or angular velocities can be seen, which is to be expected as the CubeSat model was stationary and not undergoing any attitude changes.

4.2.1.2 Data Acquisition for Kalman Filtering

To be able to evaluate the Kalman Filter later on, a total of five different data sets are taken. For all the recorded data, the preparations are the same. The data is taken directly after the data for the zero rate measurement described in the preceding section was taken, as to be able to use the derived bias under the same conditions. First, the compressor is turned on, with a working pressure of about $1.9 \cdot 10^6$ Pa. Then, the CubeSat is put onto the air bearing with its air bearing cup put on top of the air bearing base. As the CubeSat is not balanced properly yet, screws on the CubeSat's supporting structure were fine-tuned to put the CubeSat in a balanced position. In Appendix B, figure B.1, B.2 and B.3 show the CubeSat model placed on top of the attitude simulator as used for the data set recording. Table 4.1 lists the properties of the recorded data sets **A** trough **E**. The column *First full rotation* describes how long the duration for the first full rotation about the Z-axis took. This value is presented to give a better understanding of the initial angular velocities.

The main difference in the different data sets is their initial angular velocity. Set **A** does not have any initial angular velocity as the CubeSat is put on the attitude simulator in a stable and stationary position. Data Sets **B** trough **D** have increasing starting initial velocities. The initial angular velocities are generated by applying small impulse onto the CubeSat, which is accomplished by applying a burst of cold gas thrust to its side. For larger initial angular velocities, longer bursts of cold gas thrusts are applied. For data set **E**, the mass simulator is put into a tumbling motion by applying multiple bursts of cold gas to multiple different locations

Data Set	Duration [s]	Frequency [Hz]	First full rotation [s]	Description
A	300	40	-	No initial angular velocity.
B	600	40	15.9	Small initial angular velocity.
C	600	40	3.9	Medium initial angular velocity.
D	600	40	1.3	High initial angular velocity.
E	60	40	1.8	Mass simulator tumbling about all three axes.

Table 4.1: The different data sets taken for the Kalman Filter based offset vector estimation. Initial angular velocity was applied quantitatively to the CubeSat by a cold gas spray. For data set **A** - **D**, only an initial angular velocity about the Z-axis was applied.

of the CubeSat's sides, with a greater velocity created around its Z-axis and smaller velocities created around the X- and Y-axes.

Appendix C shows the attitude and angular velocities of the CubeSat in the different scenarios. Form figure C.1, which displays data set **A**, it can be seen that the CubeSats attitude stays mostly the same over the test duration. This is expected as the CubeSat has no initial angular velocities. Only the rotation about the Z-axis shows a change of about 10 degrees over the duration of 300 seconds. This could be due to multiple factors. Firstly, the CubeSat is placed on the air bearing and its mass balance was adjusted after by fine-tuning the support structure by hand. This by-hand procedure could introduce significant errors. Another factor could be that the mass distribution of the CubeSat model is not completely uniform, and thus the CoM does not align with the CoR which creates torques as explained in preceding Sections. Lastly, environmental factors such as air movement in the room could affect the CubeSats attitude.

In the other data sets **B** - **E** presented in figure C.2 - C.5, it can be seen that the attitude simulator is not entirely frictionless as especially the rotation about the Z-axis slows down. Data set **B** starts with a duration of 15.9 seconds for a full rotation about the Z-axis, however, the last complete rotation recorded has a duration of approximately 150.8 seconds. This slowing down can also be seen in the angular velocity about the Z-axis, it slows down from about 13 $\frac{\text{deg}}{\text{s}}$ to less than 2.5 $\frac{\text{deg}}{\text{s}}$ over the duration of 600 seconds. The reason for this deceleration can be found in the remaining friction between the air bearing cup and base, the aerodynamic friction between the air and the mass simular, air moving against the CubeSat on the attitude simulator as well as other non-modeled influences.

4.2.2 Kalman Filter - Simulation Approximation

The implemented Kalman Filter is based on the Kalman Filter as described in section 3.4.2.1. For the implementation of the Kalman Filter, equation (3.28) of the LSM is changed to

$$\begin{bmatrix} \omega_x^{k+1} \\ \omega_y^{k+1} \\ \omega_z^{k+1} \end{bmatrix} = \begin{bmatrix} \omega_x^{k} \\ \omega_y^{k} \\ \omega_z^{k} \end{bmatrix} + \begin{bmatrix} 0 & \Phi_{12}^{k} & \Phi_{13}^{k} \\ \Phi_{21}^{k} & 0 & \Phi_{23}^{k} \\ \Phi_{31}^{k} & \Phi_{32}^{k} & 0 \end{bmatrix} \begin{bmatrix} r_x^{k} \\ r_y^{k} \\ r_z^{k} \end{bmatrix}. \tag{4.4}$$

The state vector $\mathbf{x}$ is chosen as $\mathbf{x} = \begin{bmatrix} \omega_x^{k} & \omega_y^{k} & \omega_z^{k} & r_x^{k} & r_y^{k} & r_z^{k} \end{bmatrix}^{T}$ which is made up of the angular velocity of the satellite simulator as well as the offset vector being estimated. As the Kalman

Filter approach only estimates the offset vector, but does not actively try to compensate it, the offset vector is static, meaning $\dot{\mathbf{r}} = 0$.

The state system equation (3.40) is set to

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}_k\, \mathbf{x}_k + \mathbf{w}_k \\ \begin{bmatrix} \omega_x^{k+1} \\ \omega_y^{k+1} \\ \omega_z^{k+1} \\ r_x^{k+1} \\ r_y^{k+1} \\ r_z^{k+1} \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \Phi_{12} & \Phi_{13} \\ 0 & 1 & 0 & \Phi_{21} & 0 & \Phi_{23} \\ 0 & 0 & 0 & \Phi_{32} & \Phi_{32} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}_k} \begin{bmatrix} \omega_x^{k} \\ \omega_y^{k} \\ \omega_z^{k} \\ r_x^{k} \\ r_y^{k} \\ r_z^{k} \end{bmatrix} + \mathbf{w}_{6\times 1} \end{aligned} \tag{4.5}$$

The values for $\Phi\_\_$ are the same as presented in (3.29). From the IMU or other available sensors the angular velocity can be measured. Only the offset vector components have to be estimated. The output equation is therefore given as

$$\begin{aligned} \mathbf{y}_k &= \mathbf{H}\, \mathbf{x}_k + \mathbf{v}_k, \\ \mathbf{H} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{4.6}$$

In accordance with the notation of the Kalman Filter presented in section 3.4.2.1, the Kalman Filter equations are implemented as follows. The time update equations are used as

$$\begin{aligned} [\mathbf{P}_k^-]_{6\times 6} &= [\mathbf{F}_{k-1}]_{6\times 6} \left[\mathbf{P}_{k-1}^+\right]_{6\times 6} \left[\mathbf{F}_{k-1}^T\right]_{6\times 6} + [\mathbf{Q}_{k-1}]_{6\times 6} \\ \hat{\mathbf{x}}_k &= \mathbf{F}_{k-1} \hat{\mathbf{x}}_{k-1} \end{aligned}. \tag{4.7}$$

The equations for the measurement update are implemented as

$$\begin{aligned} [\mathbf{K}_k]_{6\times 3} &= [\mathbf{P}_k^-]_{6\times 6} \left[\mathbf{H}_k^T\right]_{6\times 3} \left([\mathbf{H}_k]_{3\times 6} [\mathbf{P}_k]_{6\times 6} \left[\mathbf{H}_k^T\right]_{6\times 3} [\mathbf{R}_k]_{3\times 3} \right)^{-1} \\ \left[\hat{\mathbf{x}}_k^+\right]_{6\times 1} &= [\hat{\mathbf{x}}_k^-]_{6\times 1} + [\mathbf{K}_k]_{6\times 3} \left(\left[\mathbf{y}_k^+\right]_{3\times 1} - [\mathbf{H}_k]_{3\times 6} \left[\hat{\mathbf{x}}_k^+\right]_{6\times 1} \right) \\ \left[\mathbf{P}_k^+\right]_{6\times 6} &= \left([\mathbf{I}]_{6\times 6} - [\mathbf{K}_k]_{6\times 3} [\mathbf{H}_k]_{6\times 3} \right) [\mathbf{P}_k^-]_{6\times 6} \end{aligned}. \tag{4.8}$$

$\mathbf{P}_0$, the initial state covariance is set as $\mathbf{P}_0 = 0_{6\times 6}$ and the noise matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ are set to

the diagonal matrices

$$\mathbf{Q}_\mathrm{k} = \mathrm{diag}\left(\begin{bmatrix} 5\cdot 10^{-4}\frac{\mathrm{rad}^2}{\mathrm{s}^2} \\ 5\cdot 10^{-4}\frac{\mathrm{rad}^2}{\mathrm{s}^2} \\ 5\cdot 10^{-4}\frac{\mathrm{rad}^2}{\mathrm{s}^2} \\ 1\cdot 10^{-8}\mathrm{m}^2 \\ 1\cdot 10^{-8}\mathrm{m}^2 \\ 25\cdot 10^{-8}\mathrm{m}^2 \end{bmatrix}\right) \tag{4.9}$$

and

$$\mathbf{R}_\mathrm{k} = \mathrm{diag}\left(\begin{bmatrix} 0.01^2\frac{\mathrm{rad}^2}{\mathrm{s}^2} \\ 0.01^2\frac{\mathrm{rad}^2}{\mathrm{s}^2} \\ 0.01^2\frac{\mathrm{rad}^2}{\mathrm{s}^2} \end{bmatrix}\right), \tag{4.10}$$

respectively. They are diagonal matrices as the assumption is that the axes of the gyroscope used in the IMU are uncorrelated. The values were chosen based on average noise parameters of IMUs. [2]

The Kalman Filter starts its estimation process with the initial values for the $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ as stated before. The initial system state is chosen to being $\mathbf{x}_0 = \left[\omega_\mathrm{x}^0, \omega_\mathrm{y}^0, \omega_\mathrm{z}^0, \mathrm{r}_\mathrm{x}^0, \mathrm{r}_\mathrm{y}^0, \mathrm{r}_\mathrm{z}^0\right]^\mathrm{T}$ with $\omega_\mathrm{x}^0, \omega_\mathrm{y}^0$ and ω_z^0 being set to 0.1 $\frac{\mathrm{rad}}{\mathrm{s}}$ for the simulation. This value was chosen as it excites the system into movement but keeps the overall dynamics stable.

The true center of mass offset is determined from the mass simulator CAD-model, resulting in $\mathbf{r}_\mathrm{cad} = [0.00, 0.00, -19.88]^\mathrm{T}\mathrm{mm}$. It should be noted, that as this value is also just an approximation, its accuracy is limited by the CAD-Model as well as the center of mass offset calculation of the CAD-Modeling program. The initial offset vector guess is set to $\mathbf{r}_\mathrm{guess} = [10, 10, 10]^\mathrm{T}\mathrm{mm}$. It could be set closer to the offset vector values from the CAD-model, however, as the estimation process will be evaluated for cases where there is no basis to base the initial offset estimate on, more generic values will be used if not otherwise stated.

Using the Kalman Filter with the initial values as described above, the CoM offset vector estimate results are shown in Table 4.2. The estimate is build by averaging the offset vector estimates over the last 30 seconds. In Table 4.2, $\mathbf{r}_\mathrm{kf}$ represents the estimated offset vector and $\mathbf{r}_\mathrm{cad–kf}$ the difference between the CAD-mdel and Kalman Filter estimate. This value represents the estimation error. The chronological progression of the components of the offset estimate vector is shown in figure 4.7.

$\mathbf{r}_\mathrm{guess}$ [mm]	10.0000	10.00000	10.00000
$\mathbf{r}_\mathrm{cad}$ [mm]	0.00000	0.000000	-19.88000
$\mathbf{r}_\mathrm{kf}$ [mm]	-0.01703	0.000201	-18.28388
$\mathbf{r}_\mathrm{cad–kf}$ [mm]	0.01703	0.000201	-1.59612

Table 4.2: The results of the Kalman Filter offset vector estimate.

[2] See www.bosch-sensortec.com/products/motion-sensors/imus/bmi088/, www.bosch-sensortec.com/products/motion-sensors/imus/bmi260/ or www.movella.com/products/sensor-modules/xsens-mti-100-imu#specs as exemplary devices.

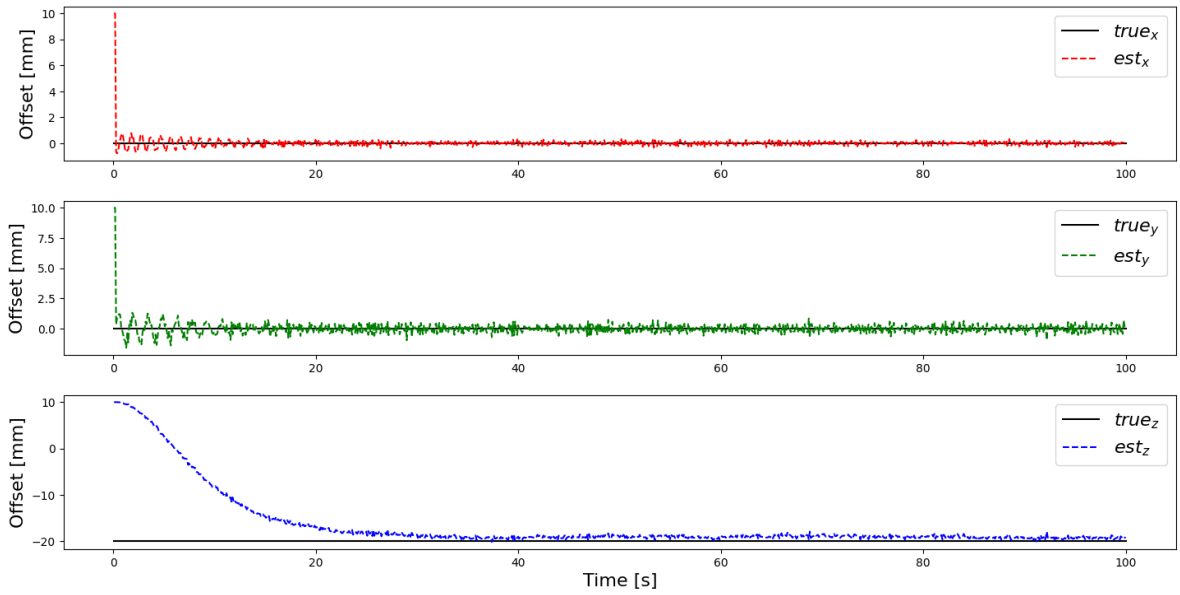


Figure 4.7: The Kalman Filter offset vector estimate with simulated data of the mass simulator on the attitude simulator.

4.2.3 Kalman Filter - Approximation with Hardware Data

The Kalman Filter is used exactly the same as in the preceding chapter. However, instead of simulating the kinematics of the attitude simulator, the optically tracked attitude data is used.

As the data provided by in quaternions, but the Kalman Filter is using Euler Angles, some steps have to be done to be able to use the data. In the first step, the raw quaternion data - which is give as a comma-separated values (CSV) file by OptiTrack - is loaded into the python environment using the pandas data analysis library.[3] On this raw data, a lowpass filter with window size four is applied. As another measure to smooth the resulting quaternion data, the quaternions are tested if the scalar part, w, is negative. As the scalar part w of a quaternion represents the angle of rotation, and a negative rotation about some angle $-w_i$ being the same as w_i, the quaternion can be conjugated to ensure a rotation about a positive angle. This is done to reduce jumps in the quaternion data when the angle moves from close to -1 to 1. The next step in the data preparation is to convert the quaternions into their Euler Angles representation using equations (3.21). From those angles, the previous determined bias is subtracted. From differentiation of the Euler Angles, the angular velocities are derived. The last step is to transform the data from to the original reference frame into the body reference frame of the mass simulator, as this reference frame is also used by the Kalman Filter. The respective reference frames can be seen in figures 4.8 and 4.9. The transformation from the OptiTrack reference frame into the body reference frame can be done by two successive rotations, first a rotation of 90° about the X-axis, $\mathbf{R}_{X,90°}$, and then a rotation of 90° about the new Z-axis, $\mathbf{R}_{Z,90°}$.

After these preoperational steps, the data is passed along to the Kalman Filter, where the

[3] https://pandas.pydata.org/

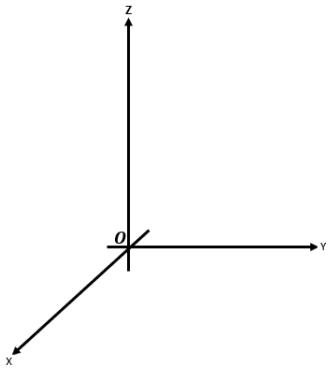


Figure 4.8: The reference frame used by the simulation and the Kalman Filter, which is the body reference frame. If the body is put in its initial attitude, with all its Euler angles being 0°, the X- and Y-axes span a plane that is parallel to the laboratory ground. The Z-axis is perpendicular to this plane. The origin of the reference frame is the CoR.

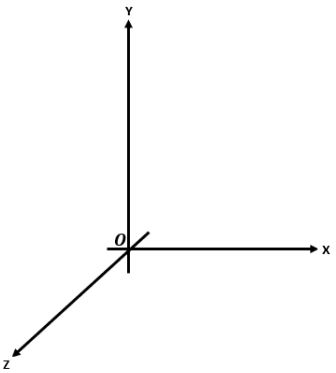


Figure 4.9: The reference frame in which the quaternion data by the OptiTrack camera tracking system is presented in. It is different from the body reference frame by two successive rotations. However, it shares the same origin as the CoR of the attitude simulator.

angular velocities are used as the measurement. In contrast to the simulation, no noise is added. This is because the measured data contains noise already.

In appendix D, graphs for the recorded data sets **A** - **E** are presented. Additionally, for the following evaluation of the Kalman Filter in section 5.2, data set **E** is presented once with an initial r_{guess} of $[10, 10, 10]^T$mm as well as with an initial offset vector guess of $[0.1, 0.1, –20]^T$ mm.

The estimation results for the different data sets are shown in Table 4.3.

4.3 Approximation using adaptive control

The TOM CubeSat will have internal momentum exchange devices such as reaction wheels or gyroscopes for its attitude control in orbit. But, as the development of the CubeSat is

not finished as of the writing of this thesis, the CubeSat-Model used for the development and debugging does not have such devices. Due to this, the adaptive control algorithm as described in section 3.4.3 will only be simulated and not implemented in hardware.

In comparison to the simulation used for the Kalman Filter based CoM approximation, only small simulation properties change. Firstly, the system model is enhanced to the extended system model of section 3.4.3.1. This is necessary, as the adaptive control algorithm utilizes MMUs to approximate and nullify the offset vector. Due to the altering mass properties after the moveable masses have changed their position, the inertia matrix has to be recalculated after each approximation step. Secondly, the adaptive control algorithm is separated in two steps: transversal and vertical compensation. Transversal compensation is the adaptive control law operating. By changing the MMUs position, the derivative of the angular momentum is iteratively nullified. Vertical compensation is the UKF estimating the remaining z-component of the CoM offset. This two-step procedure has to be deployed as the torque generated in the transversal phase can only act perpendicular to the gravity vector. As the z-component is parallel to the gravity vector it can not be nullified directly.

The transversal plane compensation phase is implemented as a python script by evaluating the equations presented in section 3.4.3.2. For that, the scalar parameter k_p is set to 0.2 and the individual masses of the MMUs, m_p, were set to 0.2 kg in accordance to the planned setup at the ZfT. Similarly, the vertical estimation phase utilizes an UKF which is implemented in accordance to section 3.4.3.3.

Below, the results of the adaptive control estimation process are presented. From averaging the last 30 seconds of samples, the estimation of the offset vector was found to being $\mathbf{r}_{\text{adaptive}} = [0.2024687, –7.9243491, –4.6303229]^{\text{T}}$ mm. In Table 4.4, the CoM offset vector by the CAD-Model, $\mathbf{r}_{\text{cad}}$ is compared to the adaptive control based estimate $\mathbf{r}_{\text{adaptive}}$ by building the difference vector, $\mathbf{r}_{\text{cad–adaptive}}$. An in depth evaluation of this approach is conducted in section 5.2.3.

$\mathbf{r}_{\text{cad}}$ [mm]	0.00000	0.00000	-19.88000
$\mathbf{r}_{\text{A}}$ [mm]	0.23260	0.01914	-9.20194
$\mathbf{r}_{\text{B}}$ [mm]	1.23464	0.12935	-6.97470
$\mathbf{r}_{\text{C}}$ [mm]	1.10019	0.19849	-5.48961
$\mathbf{r}_{\text{D}}$ [mm]	0.88430	0.19835	-4.54321
$\mathbf{r}_{\text{E,1}}$ [mm]	0.20727	0.08845	-15.23014
$\mathbf{r}_{\text{E,2}}$ [mm]	0.29674	0.13960	-15.23014

Table 4.3: The Kalman filter based offset vector estimations for the recorded data sets **A** trough **E**. Rows **E_1** and **E_2** are both using data set **E**, but have different initial guesses. **E_1** has a generic guess of $[10, 10, 10]^{\text{T}}$ mm, **E_2** an initial guess close to the CAD based true values as $[0.1, 0.1, –20]^{\text{T}}$mm.

$\mathbf{r}_{\text{guess}}$ [mm]	10.00000	10.00000	10.00000
$\mathbf{r}_{\text{cad}}$ [mm]	0.29000	-8.00000	-4.60000
$\mathbf{r}_{\text{adaptive}}$ [mm]	0.20247	-7.92435	-4.63033
$\mathbf{r}_{\text{adaptive–kf}}$ [mm]	-0.08753	0.07565	0.03032

Table 4.4: The results of the adaptive control law based offset vector estimate.

Chapter 5

Evaluation

In chapter 3.4 different approaches for estimating the CoR to CoM offset vector were shown. Chapter 4 subsequently detailed the implementations of two methods: one based on the Kalman Filter and another utilizing an adaptive control law. This chapter focuses on the evaluation of these implementations.

As the goal of this thesis is to find a time efficient and effective CoM offset vector estimator, the main assessment criteria are the time until the estimation reaches a predetermined threshold and whether the estimation is stable or does show oscillations. Before going into the evaluation of the estimation approaches, it is necessary to discuss the simulation itself first.

5.1 Simulation Assessment

Figure 4.2 in section 4.1 displays the free moving behavior of the TOM CubeSat on the satellite simulator. In the top figure, which shows the attitude in Euler Angles, clear oscillations in the roll, pitch and yaw values can be seen. These oscillations are to be expected when considering the dynamics of the air bearing cup and air bearing base as these dynamics are comparable to that of an inverted pendulum.

Generally, an inverted pendulum is a pendulum whose CoM is above its CoR. In the case of an attitude simulator, the mass of the satellite model is located above the central point of rotation, which is located inside the spherical air bearing. As the mass is seldom evenly distributed in the satellite, the central point of mass is not only offset by a vertical, but also by a horizontal component. This offset and the acting gravitational force create a torque, which acts on the attitude simulator and pushes the side of the satellite simulator that contains the CoM downwards. The opposite side of the satellite naturally gets tilted upwards. At some point, the side of the satellite that does not contain the CoM is elevated to such an extent that the gravitational forces acting upon that side of the satellite are greater than the one acting on the CoM and the satellite tilts back towards the opposite direction. This process is then repeated periodically and creates the seen oscillations.

However, the amplitudes of the oscillations are greater than expected. From the attitude graph in Figure 4.2 it can be seen that roll angle reaches a value of 2 rad in about 2.5 seconds. As the graph represents the simulated attitude of the TOM CubeSat in a free-moving configuration,

which means that the satellite is placed onto the attitude simulator without any initial angular velocity, it would be expected to see a much smaller amplitude in the roll angle. The pitch and yaw angles show more realistic behavior but also show greater than expected amplitudes. The yaw angle shows a constant rotation about the z-axis, which is the axis parallel to the gravity vector. As the external torque is constant over the whole simulation, it is possible that the satellite exhibits a constant drift in the z-component as this component is not limited by the attitude simulator properties - the satellite can rotate freely about its z-axis - and the above described inverted pendulum aspect does not directly affect the z-axis. The pitch angles shows behavior that is expected when considering the inverted pendulum. Its amplitudes show a realistic range with a maximum angle of about $\pm$ 0.4 rad.

There are multiple factors that could lead to the larger than expected amplitudes of the oscillations. Firstly, the friction and adhesion between the air bearing cup and base are not modeled in the idealized simulation model which could introduce dampening effects. With those factors present in the simulation, the system would react slower to the external torque present. As the friction and adhesion are very difficult to calculate or even approximate, they were chosen to not being included in the model. Another reason for the seen behavior is that the simulation is very sensitive as the dynamics are non-linear. Even small initial values can lead to strong responses over a longer simulation which could lead to resonance. Furthermore, the inertia matrix provided by the CAD-Model of the TOM CubeSat has great effects on the simulation. The simulation is based upon the EEoM as presented in Equation (3.22). In this system of equations, the inertia matrix is present multiple times. If the inertia matrix used in those calculations is not very accurate, the whole simulation is based on uncertain values which could also lead to resonance and oscillating behavior. In Appendix E four figures are listed with changed inertia matrix values and magnitudes. From this it can be seen that the inertia matrix has a great influence on the simulated behavior. As previously stated in Section 3.1.1, for this thesis, the inertia matrix can only be determined by the CAD tool as no other possibilities currently exist at the ZfT to measure the inertia matrix.

One possibility to evaluate the simulation is to compare it to other simulations that are based on the same dynamic model. By research one possible source for comparison was found in the work of Rodrigo Da Silva, who researched similarly to this thesis attitude simulator balancing techniques [30]. Da Silva's work implements a satellite simulator simulation, which is based on the work by [7], which is also the theoretical foundation for the simulation in this work. In [30, Appendix K, Source Code K.1] the Matlab code for the attitude simulator, along with the simulation parameters, is provided. Figure 5.1 is generated from executing this Matlab file. These simulation parameters were put into the simulation presented in this work, resulting in figure 5.2. From comparing the Figures, it can be seen that the graphs match each other closely. This alignment suggests that the simulation itself is performing as intended and that any errors may come from the CAD model or other simulation parameters used for the TOM CubeSat, possibly leading to the observed increased oscillations during the simulation of the free-moving TOM CubeSat.

Considering these factors, the simulation appears sufficiently accurate enough to generate angular velocities that can be used for the CoM offset vector estimation techniques, especially as only a preselection of the presented algorithms will be done in this thesis. In works following this thesis, more time has to be spent on improving the dynamic model, the simulation and the

CAD-model to make it more accurate.

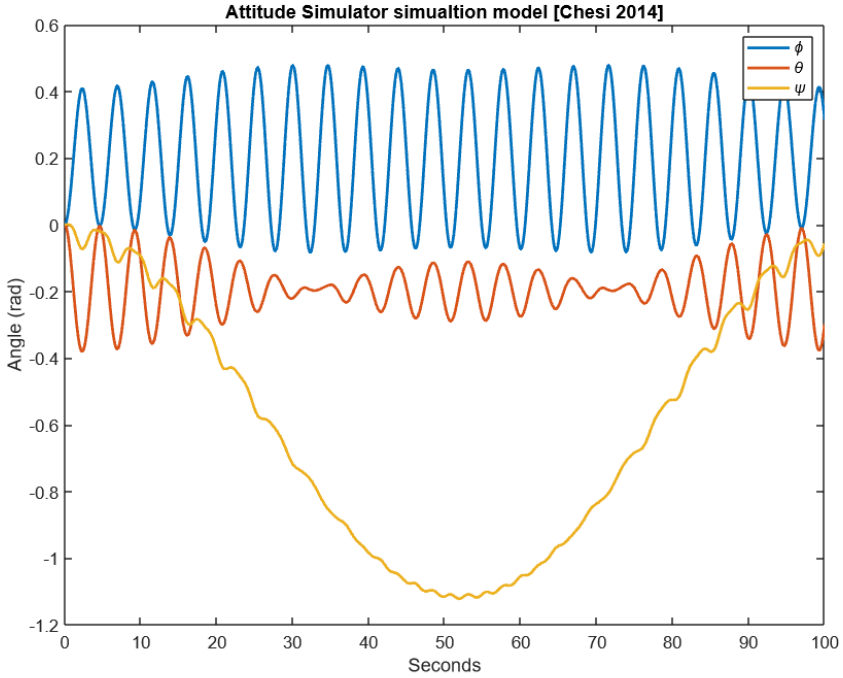


Figure 5.1: Simulated attitude of a satellite simulator described in [30] using the simulation from [30] as well.

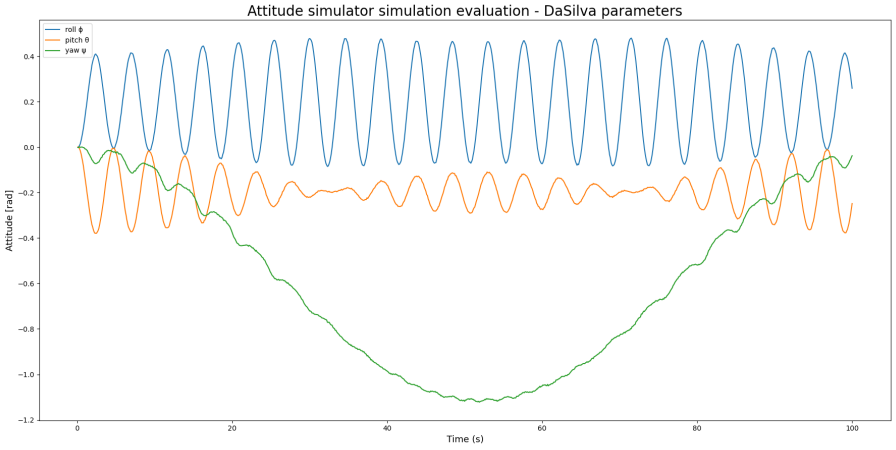


Figure 5.2: Simulated attitude of a satellite simulator described in [30] using the simulator from this thesis.

5.2 Center of Mass Approximation Assessment

For the CoM vector offset estimation a Kalman Filter and an adaptive control law were chosen. In section 4 these approximation techniques were implemented into the simulation and first results were shown. Now, an in depth evaluation of the implementations is conducted.

5.2.1 Kalman Filter Results - Simulation

Figure 4.7 presents the estimation of the offset vector components in the simulation environment. Additionally, Figure 5.3 presents the individual components of the Kalman estimation which also displays a zoomed in region of the estimation's first 15 seconds, except for the Z-component as this value converges slower compared to the X- and Y-components. Also, in each sub-figure, a shaded area depicts the region around the true value, extending 2 millimeters in width and centered on the true value. When the data points remain within this area for at least 10 seconds, it indicates that the value has converged. The duration of 10 seconds and width of 2 mm are chosen in consideration of the system's fast dynamics and the desired accuracy as well as the requirement of keeping the overall test time relatively short. In these graphs it can be seen that the Kalman Filter is able to estimate most the offset vector components. Especially the X- and Y-components reach a converged state after just about 5 seconds and remain in the convergence area after. However, the Z-component never enters the convergence area significantly. It converges to a value that is off to the true value determined by the CAD-model of the mass simulator by about 1.6 mm.

This could be due to multiple reasons, one of which is the CAD-model. Minor differences between the CAD-model and the true physical model can impact the estimation in such a way that it converges to a value offset to the true value. The simulations dynamic model can also introduce errors that lead to the offset. If the simulation is not accurate, or minor errors get introduced in one or multiple of the calculations, the propagation of the system state could diverge from the true behavior. Lastly, the Kalman filter estimation could be more accurate than the true value taken from the CAD model. In that case, the estimation seen in the graph would show the true value, and the CAD model or the internal calculations of the CAD modeling tool leads to a false positive true CoM offset vector. More in depth tests are necessary to eliminate some of these possibilities presented. However, as the scope of this thesis is to evaluate which of the presented CoM to CoR estimation techniques could be used in the future as an improvement to existing offset vector estimation tools used at the ZfT, more in depth tests are too much time-consuming to include and thus out of scope of this thesis.

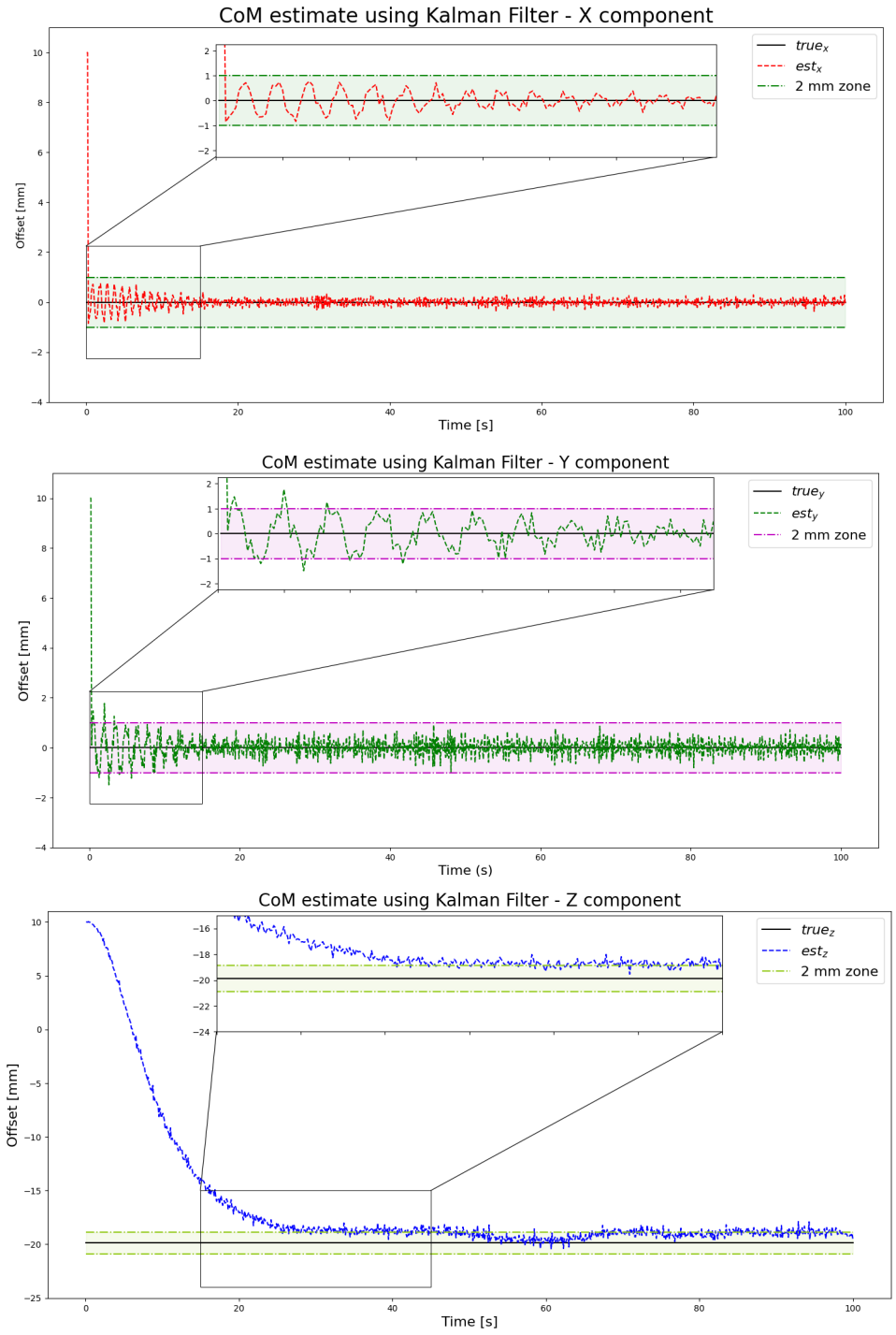


Figure 5.3: Kalman Filter approximation of the CoM of TOM CubeSat on the simulated attitude simulator

5.2.2 Kalman Filter Results - Hardware

The in the Appendix D presented results of the Kalman filter estimation of the offset vector show an at first glance unexpected behavior. None of the data sets **A** trough **D** lead to a converged Z-component. Also, the X- and Y- components converge immediately to a value close to the true value, but diverge over time from those values, which can be seen especially prominent in Figures D.2 and D.3. Table 4.3 presents the Kalman filter based offset vector estimate. In this table, it is apparent that the first and second column, presenting the X- and Y- components of the estimates are relatively close to the true value. Only for data set **B** and **B**, the estimate are offset by more than 1 mm from the CAD value. For the Z-component, only data set **E** presents estimates that are nearing the true value, but are still offset by about 4.65 mm.

From the data presented relating the Kalman filter based CoM offset estimate, it would seem that the Kalman filter does not work on the actual data and just in the simulated environment as it is not able to accurately determine the Z-component. However, the fault in this behavior does not lie in the Kalman filter, but in the data taken.

Setting an initial attitude of $\mathbf{E}_0 = [0, 0, 0]^{\mathrm{T}}$ rad and an initial angular velocity of $\omega_0 = [0.1, 0.1, 0.1]^{\mathrm{T}} \frac{\mathrm{rad}}{\mathrm{s}}$ for the simulation, the resulting Kalman filter estimation is presented in figure 4.7. A different behavior is visible when reducing the initial angular velocity, as an example figure 5.4 presents the Kalman estimation for an initial angular velocity of $\omega_0 = [0.01, 0.01, 0.01]^{\mathrm{T}} \frac{\mathrm{rad}}{\mathrm{s}}$. Here, the Z-component is also not converging properly. Figure 5.5 further supports this observation, as it shows the Z-component estimate against increasing initial angular velocities. It can be clearly seen, that a minimal initial angular velocity of about 0.15 $\frac{\mathrm{rad}}{\mathrm{s}}$ or about 9 $\frac{\mathrm{deg}}{\mathrm{s}}$ is necessary for the Kalman filter to estimate the Z-component correctly. As an example, when running the simulation with an initial angular velocity of $\omega_0 = [0.175, 0.175, 0.175]^{\mathrm{T}} \frac{\mathrm{rad}}{\mathrm{s}}$, the Z-components estimate is -19.2093 mm, resulting in an estimation error of 0.6707 mm.

This can further be confirmed when looking at the Kalman estimate of data set **D** (Figure D.4) while considering its angular velocity (Figure C.4). While the angular velocities of the X- and Y-components are above the mentioned 9 $\frac{\mathrm{deg}}{\mathrm{s}}$ - which is true until about 300 seconds -, the Z-component's estimate converges towards the true Z-components value. However, after the angular velocities of the X- and Y-components drop below 9 $\frac{\mathrm{deg}}{\mathrm{s}}$, the Z-component's estimate begins to diverge.

Data set **E** further supports this. For the recording of data set **E**, the satellite was put into a tumbling motion. It's initial angular velocities are approximately $\omega_0 = [210, 48, 35]^{\mathrm{T}} \frac{\mathrm{deg}}{\mathrm{s}}$, exceed the needed minimal initial angular velocity of 9 $\frac{\mathrm{deg}}{\mathrm{s}}$ in all components. Figure C.5 displays the attitude and angular velocity over the test duration. Presented in figure D.5, is the Kalman filter approximation of the offset vector. It is clearly visible, that the Kalman filter shows better convergence, while still not converging properly, as it is still offset by the true value by about 4.6 mm. To evaluate if an improved initial guess for the offset vector is able to improve the convergence, figure D.6 displays the filter's behavior when starting with an initial offset vector estimate close to the CAD model. In the scenario of an actual test involving the attitude simulator, this improved guess would be available most of the time as it can be extracted from the CAD model. Using the closer initial estimate, the filter is not able to improve the estimation. The fault for this could be that the initial velocities were too high, or that the Kalman filter is run using a too small frequency, or it could be based on both the initial velocity and too few

attitude updates. More tests have to be done with different initial angular velocities and different frequencies of recording the data. For this thesis, it was not possible to record additional data sets.

From this follows the hypothesis, that the Kalman filter as implemented is working as intended, and for an accurate z-component estimation using physical data, the angular velocities must be kept over at least 9 $\frac{\text{deg}}{\text{s}}$. In possible future works, this hypothesis must be confirmed with multiple data sets by exciting the mass simulator to different, but sufficient fast angular velocities and running the Kalman filter on that data.

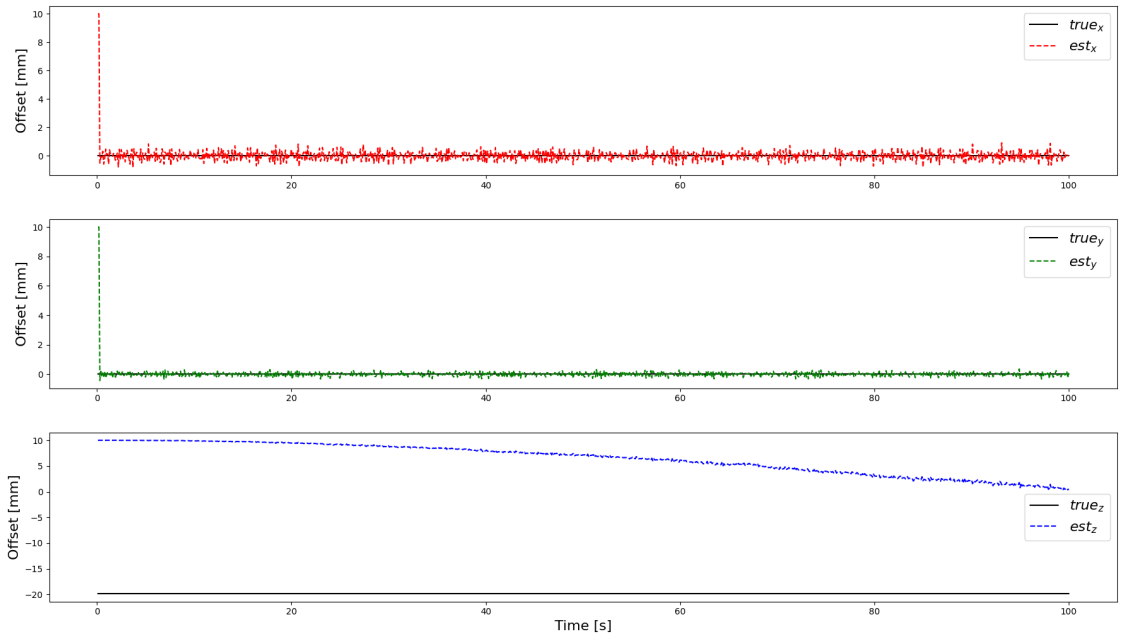


Figure 5.4: The Kalman filter estimation for the simulated mass simulator on the attitude simulator with small initial velocities.

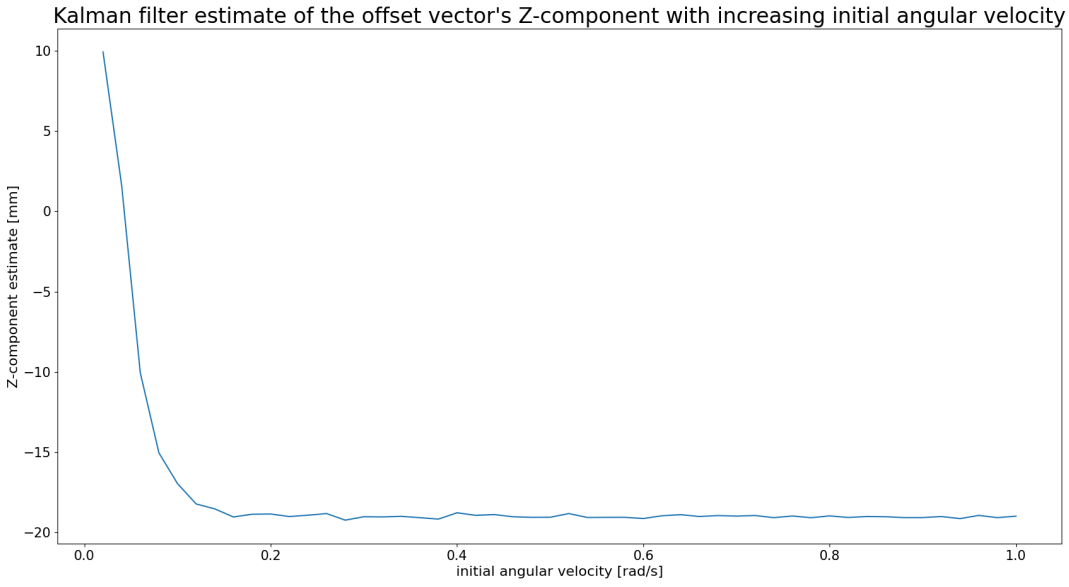


Figure 5.5: Kalman filter estimated Z-component compared to increasing initial angular velocities.

5.2.3 Adaptive Control Law Results

Similarly to the evaluation of the Kalman Filter in section 5.2.1 the adaptive controller as presented in section 3.4.3 will be assessed using the simulated attitude simulator. As stated previously, the current testing setup at the ZfT does not allow the implementation of the adaptive control approach.

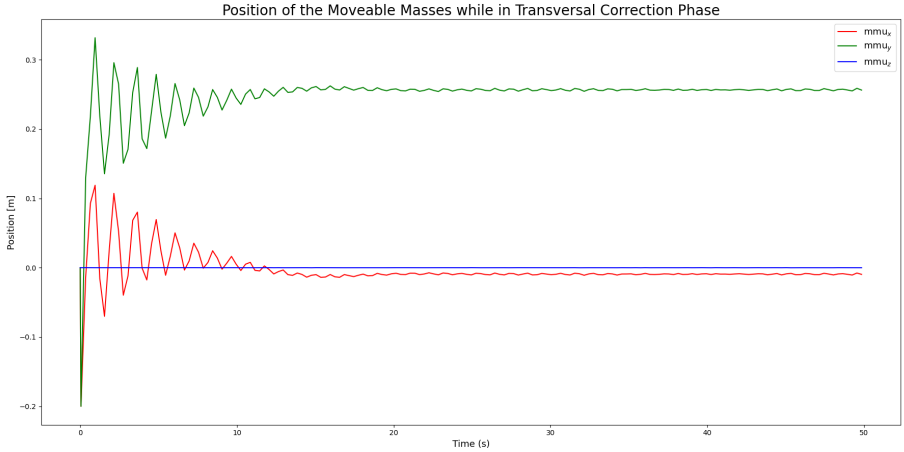
For the adaptive control law it is necessary to have three masses that can be moved by actuators in the three axes of the body coordinate system. Only by utilizing the MMUs, the mass properties of the satellite simulator and device under test on the attitude simulator can be changed and the goal of nullifying the external torque and in turn reduction of the offset vector can be accomplished. As the development of the more sophisticated attitude simulator is still in progress at the ZfT, the adaptive control based approach will be evaluated using the simulation only.

It should also be noted again that the movement of the MMUs is not modeled itself as stated previously in section 3.4.3. In the simulation, the masses move instantly between iteration and the introduced angular momentum change is not considered. This simplification of the dynamic system was made as the velocity of the MMUs while in motion is relatively low. Additionally, the individual masses m_p of the MMUs are small in relation to the total mass of the attitude simulator and the CubeSat model. For the TOM CubeSat, the total mass will be about 4.5 Kg. In contrast, the planned masses of the MMUs are at most 0.2 Kg. The combined effects of the low velocity and small mass of the MMUs contribute to the observation that their movement has a negligible impact on the total angular momentum.

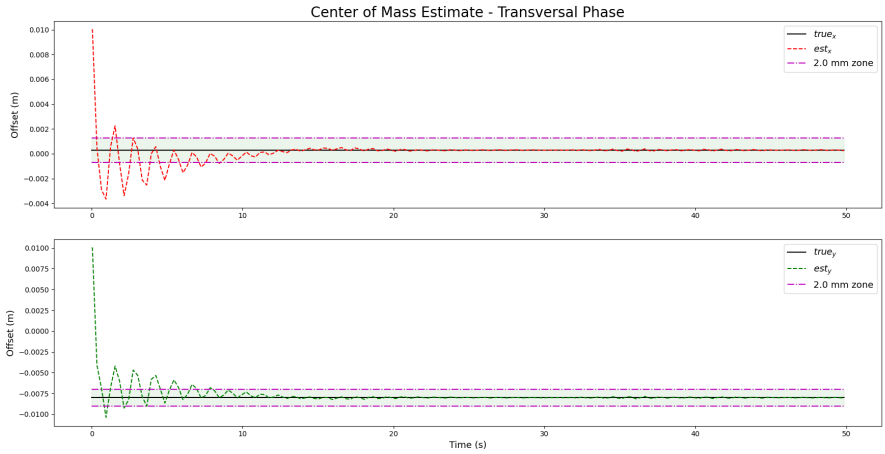
Figure 5.6 shows the simulated behavior of the adaptive control CoM offset vector estimation process. Although the simulation lacks precision in modeling the time duration, the plots still provide a time reference derived from the frequency of iterations per second, albeit more as an approximation than a precise measurement. Even just as a rough estimate, it provides a useful frame of reference for understanding the temporal aspects of the simulation results.

Subfigure (a) presents the positions of the MMUs. It can be seen that the position of the MMUs in the X- and Y-Axis converge to a final position in about 20 seconds. After those 20 seconds, the positions of the MMUs seem to oscillate slightly around the converged value, which is due to the noise present. Based on direct data from the simulation, a converged result is found after approximately 100 iterations. The following subfigure (b) presents the CoM offset vector components estimates. Similarly to the evaluation of the Kalman Filter in the preceding section, an area with a width of 2 mm was drawn around the true value taken from the CAD-Model. It can be seen that the adaptive control technique is able to find a good estimate after an initial oscillating phase. As the estimation process is coupled with the movement of the MMUs, the estimate do not change significant once the MMUs start to oscillate around the converged MMUs position. To overcome this indefinite oscillation, a minimal value for the external torque acting on the system could be defined. In the introduction to section 3.4 the gravitational torque acting on a CubeSat was found to range form 10^{-4} Nm to 10^{-6} Nm. Thus, a lower limit for the estimation should be set to a value of about 10^{-6} Nm as to replicate the environment found in low earth orbit. Subfigure (c) finally presents the estimated value of the Z-Component of the offset vector. The X- and Y-Components, and Z-Component is displayed in different graphs as the X- and Y-Components are estimated together in the transversal phase by the adaptive controller, and the

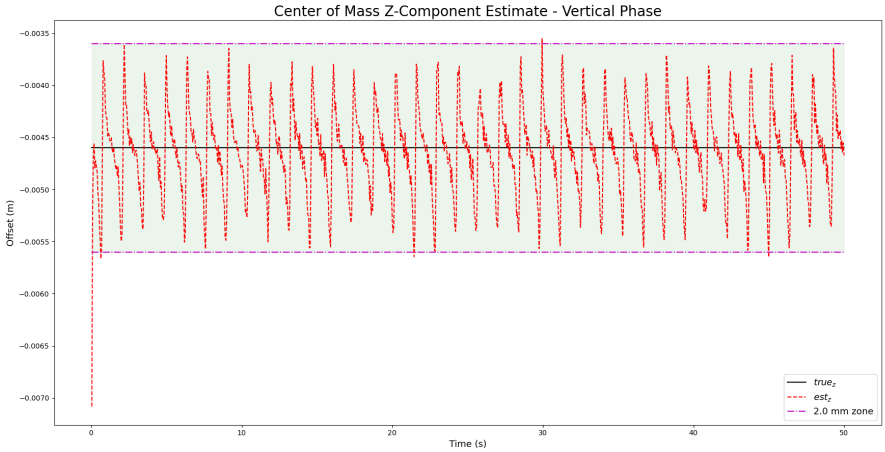
Z-Component is estimated by an UKF in the vertical phase. The estimate of the Z-Component in subfigure (c) shows a very fast convergence into the 2 mm wide convergence area. In contrast to the X- and Y-Component estimation, the UKF based approach never converges on a stable value but continues to oscillate around the true value. This is due to the filter based design of the Kalman filter, it does not converge perfectly on a stable value due to inherent uncertainties in the system dynamics and measurement noise of new data, leading to continual oscillations around the true value. All estimated values stay in the convergence area once they entered it, which shows that the adaptive control law based approach is able to produce stable estimated values for the offset vector. Additionally, as the MMUs converge around a final position in about 20 seconds, the adaptive control technique is able to present this stable value in a timely manner. Finally, the estimated offset vector by the adaptive control law was presented in Table 4.4 as $\mathbf{r}_{\text{adaptive}} = [0.0002024687, –0.0079243491, –0.0046303229]^{\text{T}}$ m, leading to an error between the estimate and the true value of $\mathbf{r}_{\text{cad–adaptive}} = [0.0875312557, –0.0756509321, 0.0303229323]^{\text{T}}$ mm, satisfying the requirement of an accuracy of ± 0.1 mm.



(a)



(b)



(c)

Figure 5.6: Adaptive control approximation of the CoM in the simulated environment.

Chapter 6

Conclusion

This thesis has presented different approaches to estimate the CoM to CoR offset vector of a CubeSat mounted on an air bearing attitude simulator with the goal of improving the ADCS's development tools at the ZfT. Three main techniques were discussed: batch estimation, filtering and adaptive control techniques. The adaptive control law uses three sliding masses to estimate and nullify the offset vector, while the other techniques are only able to estimate the offset vector.

From the presented estimators, a Kalman filter and an adaptive control law based approach were selected to be further investigated. For this investigation, a simulation of the attitude simulator was created. Using the angular velocities created by the simulation, the Kalman filter and adaptive control law were evaluated. Both techniques present promising results, however the adaptive control estimator is able to achieve better convergence in all offset vector components in the simulation. It is able to approximate the offset components by a precision of up to ± 0.1 mm, which is a requirement that the estimator must meet. The Kalman filter is able to match this precision in the X- and Y-components, but is just accurate to ± 1 mm in the Z-component.

Using data from the attitude simulator present at the ZfT, the Kalman filter was also evaluated using actual data. The results of the physical based data estimation match the magnitudes reached in the estimation using the simulated data. By evaluating and comparing the different data sets recorded, it became apparent that the Z-components accuracy is closely linked to the initial angular velocity of the CubeSat on the attitude simulator. This becomes especially noticeable when the Z-component's estimate is being compared to the initial angular velocity of the simulation. If the initial angular velocity exceeds approximately 9 $\frac{\text{deg}}{\text{s}}$, the estimation of the Z-component is able to reach the required accuracy of ± 0.1 mm.

Both estimation techniques present comparable results and accuracies, making a definitive selection difficult. The adaptive control estimation's evaluation is limited by the unavailability of physical data due to the incomplete setup at the ZfT. Although estimations from simulated data favor the adaptive control law for accuracy, validation is lacking, unlike the Kalman filter, which met accuracy requirements in estimating X- and Y-components using physical data. The evaluation suggests that with angular velocities of at least 9 $\frac{\text{deg}}{\text{s}}$, the Z-component accuracy may meet requirements for a satellite on the attitude simulator. Further tests are necessary to validate this claim.

Chapter 7

Future Work

As no definitive selection of the presented estimation tools could be made on the basis that not enough data was available, more data has to be recorded in further subsequent work. It should especially be recorded more data with higher initial angular velocities in all axes as to validate the claim that higher initial velocities improve the Kalman filter estimation of the offset vector's Z-component.

Additional work must also be spent on the dynamic modeling and simulation of the attitude simulator. Currently, present oscillations introduce potential error sources that could carry into the adaptive control and Kalman filter estimation.

To be able to evaluate the adaptive controller, the setup at the ZfT must be also be improved by adding MMUs as well as actuators and other supporting structures. Only then, both approaches, the Kalman Filter and the adaptive controller, can be directly compared, and a selection can be made.

Bibliography

[1] AR Abdulghany. Generalization of parallel axis theorem for rotational inertia. *American Journal of Physics*, 85(10):791–795, 2017.

[2] Paolo Baerlocher and Ronan Boulic. Parametrization and range of motion of the ball-and-socket joint. In *Deformable Avatars: IFIP TC5/WG5. 10 DEFORM'2000 Workshop November 29–30, 2000 Geneva, Switzerland and AVATARS'2000 Workshop November 30–December 1, 2000 Lausanne, Switzerland*, pages 180–190. Springer, 2001.

[3] Anton Bahu and Dario Modenini. Automatic mass balancing system for a dynamic cubesat attitude simulator: development and experimental validation. *CEAS Space Journal*, 12:597–611, 2020.

[4] Moti Ben-Ari. A tutorial on euler angles and quaternions. *Weizmann Institute of Science, Israel*, 524, 2014.

[5] Evandro Bernardes and Stéphane Viollet. Quaternion to euler angles conversion: A direct, general and computationally efficient method. *Plos one*, 17(11):e0276302, 2022.

[6] Simone Chesi. *Attitude control of nanosatellites using shifting masses*. University of California, Santa Cruz, 2015.

[7] Simone Chesi, Qi Gong, Veronica Pellegrini, Roberto Cristi, and Marcello Romano. Automatic mass balancing of a spacecraft three-axis simulator: Analysis and experimentation. *Journal of Guidance, Control, and Dynamics*, 37(1):197–206, 2014.

[8] Rodrigo Cardoso da Silva, Renato Alves Borges, Simone Battistini, and Chantal Cappelletti. A review of balancing methods for satellite simulators. *Acta astronautica*, 187:537–545, 2021.

[9] James Diebel et al. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[10] Roberta Veloso Garcia, Paula Cristiane Pinto Mesquita Pardal, HK Kuga, and MC Zanardi. Nonlinear filtering for sequential spacecraft attitude estimation with real data: Cubature kalman filter, unscented kalman filter and extended kalman filter. *Advances in Space Research*, 63(2):1038–1050, 2019.

[11] Irina Gavrilovich, Sébastien Krut, Marc Gouttefarde, François Pierrot, and Laurent Dusseau. Test bench for nanosatellite attitude determination and control system ground tests. In *4S: Small Satellites Systems and Services Symposium*, 2014.

[12] Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton university press, 2008.

[13] Noel H Hughes. Quaternion to/from euler angle of arbitrary rotation sequence & direction cosine matrix conversion using geometric methods. 2017.

[14] Yan-Bin Jia. Quaternions and rotations. *Com S*, 477(577):15, 2008.

[15] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[16] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[17] Jae Jun Kim and Brij N Agrawal. Automatic mass balancing of air-bearing-based three-axis rotational spacecraft simulator. *Journal of Guidance, Control, and Dynamics*, 32(3):1005–1017, 2009.

[18] Dorothea Knopf. G-extractor, 2015. Physikalisch-Technische Bundesanstalt, www.ptb.de/cms/ptb/fachabteilungen/abt1/fb-11/fb-11-sis/g-extractor.html, Accessed: 02.01.2024.

[19] Jack B Kuipers. *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

[20] Breitband und Vermessung Landesamt für Digitalisierung. Geodätische referenzpunkte würzburg, 2019. www.ldbv.bayern.de/vermessung/satellitenpositionierung/referenzpunkte/wuerzburg, Accessed: 02.01.2024.

[21] F Landis Markley and John L Crassidis. *Fundamentals of spacecraft attitude determination and control*, volume 1286. Springer, 2014.

[22] Ramakrishnan Mukundan and Ramakrishnan Mukundan. Quaternions. *Advanced Methods in Computer Graphics: With examples in OpenGL*, pages 77–112, 2012.

[23] OptiTrack. Optitrack primex 22 - specs. https://optitrack.com/cameras/primex-22/specs.html, Accessed: 23.01.2024.

[24] Mason A Peck, Les Miller, Andrew R Cavender, Mario Gonzalez, and Tim Hintz. An airbearing-based testbed for momentum control systems and spacecraft line of sight. *Advances in the Astronautical Sciences*, 114:427–446, 2003.

[25] Marcello Romano and Brij N Agrawal. Acquisition, tracking and pointing control of the bifocal relay mirror spacecraft. *Acta Astronautica*, 53(4-10):509–519, 2003.

[26] Jana L Schwartz, Mason A Peck, and Christopher D Hall. Historical review of air-bearing spacecraft simulators. *Journal of Guidance, Control, and Dynamics*, 26(4):513–522, 2003.

[27] Ghasem Sharifi, Mehran Mirshams, and Hamed Shahmohamadi Ousaloo. 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*, 233(3):896–907, 2019.

[28] Malcolm D Shuster et al. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993.

[29] Marcel J Sidi. *Spacecraft dynamics and control: a practical engineering approach*, volume 7. Cambridge university press, 1997.

[30] Rodrigo Cardoso da Silva. Filtering and adaptive control for balancing a nanosatellite testbed. 2018.

[31] Rodrigo Cardoso da Silva, Fernando Cardoso Guimarães, João Victor Lopes de Loiola, Renato Alves Borges, Simone Battistini, and Chantal Cappelletti. Tabletop testbed for attitude determination and control of nanosatellites. *Journal of Aerospace Engineering*, 32(1):04018122, 2019.

[32] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[33] Mathieu St-Pierre and Denis Gingras. Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 831–835. IEEE, 2004.

[34] John Robert Taylor and John R Taylor. *Classical mechanics*, volume 1. Springer, 2005.

[35] Gabriel A Terejanu. Unscented kalman filter tutorial. *University at Buffalo, Buffalo*, 2011.

[36] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[37] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[38] Ramazan Yeşilay, A Ankaralı, and Mustafa Atakan Afşar. A review paper: The dynamics, kinematics, design and control of satellite simulators with spherical air bearing. In *Global Conference on Engineering Research*, 2021.

[39] Jeff S Young. Balancing of a small satellite attitude control simlator on an air bearing. 1998.

[40] Jefferson Stephen Young. *Development of an automatic balancing system for a small satellite attitude control simulator*. Utah State University, 1998.

Appendices

Appendix A

Zero Rate Offset Test

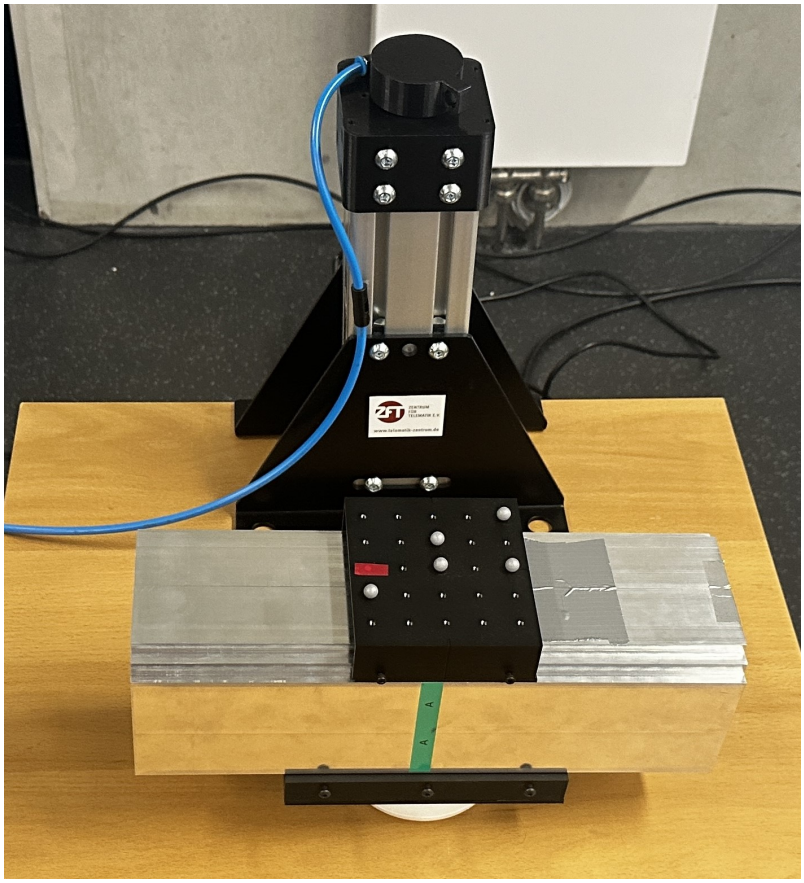


Figure A.1: A closeup picture of the CubeSat model with its OptiTrack markers fixed on top. Lying on a styrofoam bed, the CubeSat model was kept still while conducting the zero rate offset test. The green sticker denotes the geometric center of the CubeSat. Image Credit: ZfT.

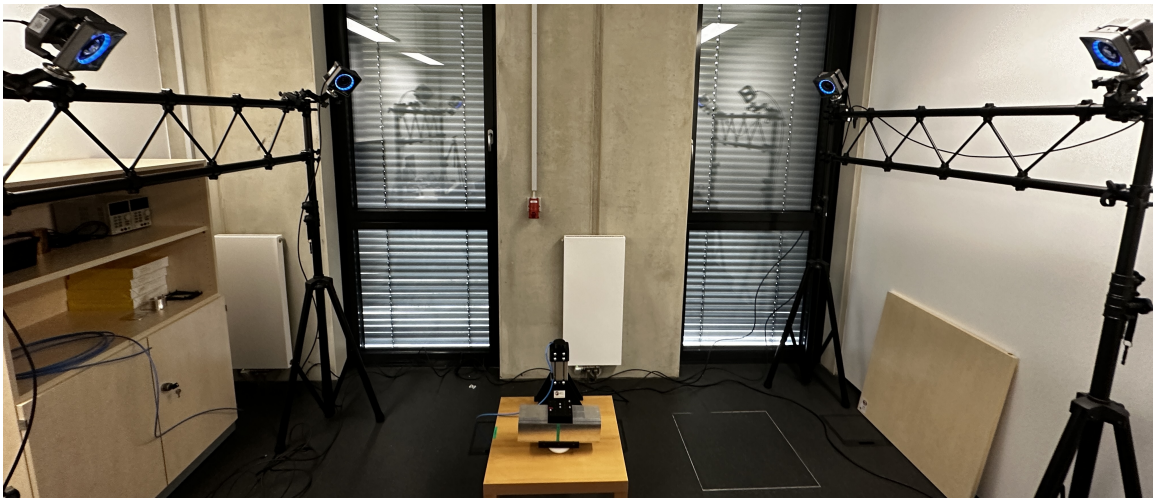


Figure A.2: The setup used for the zero rate measurement test. The CubeSat model with its OptiTrack markers on top is visible in the lower center of the image. The four infrared cameras are visible in the top section of the image. Image Credit: ZfT.

Appendix B

Attitude Simulator - Data Taking Setup

Figure B.1: Full view of the setup used for the data acquisition. Image Credit: ZfT.

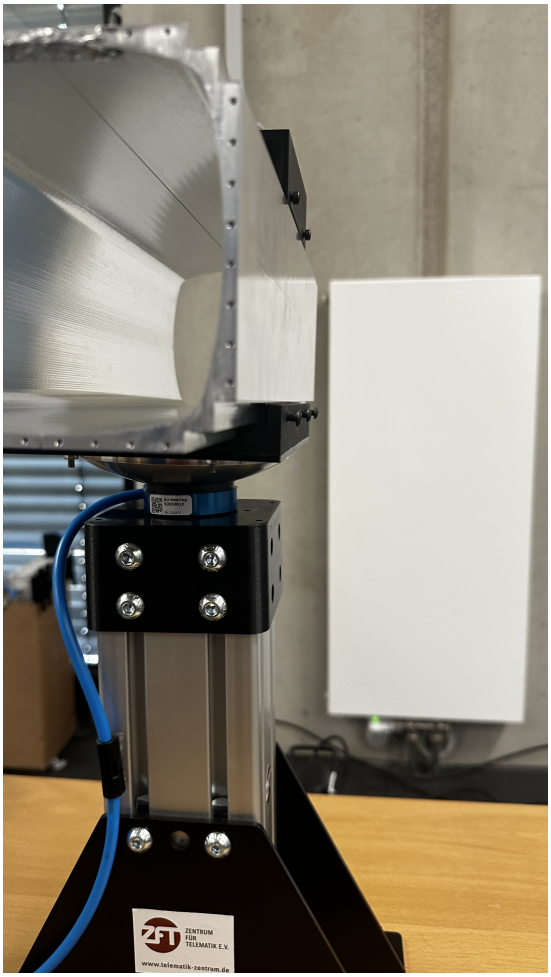


Figure B.2: CubeSat Model on the air bearing closeup with air bearing cup and base visible. Image Credit: ZfT.

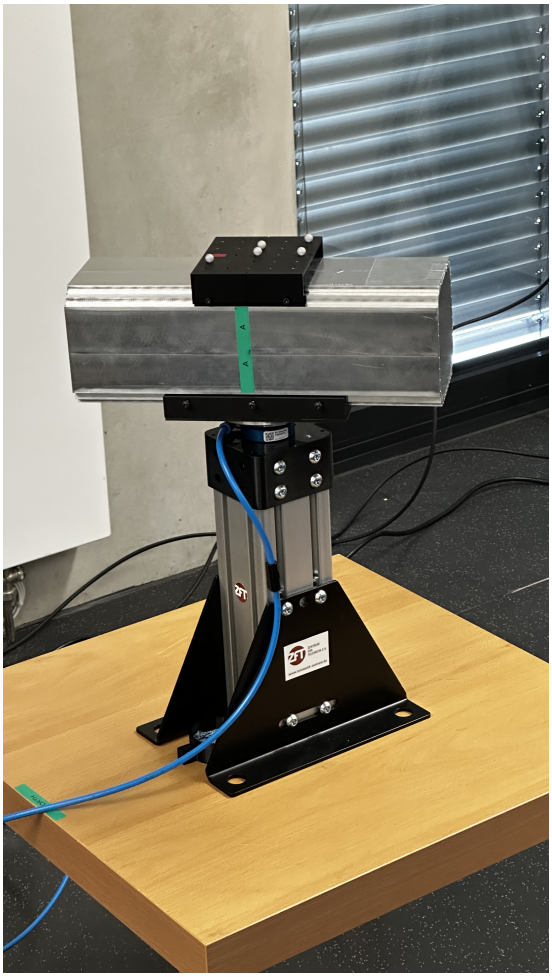


Figure B.3: Full view of the CubeSat on the air bearing satellite simulator. Image Credit: ZfT.

Appendix C

Kalman Filter - Data Sets

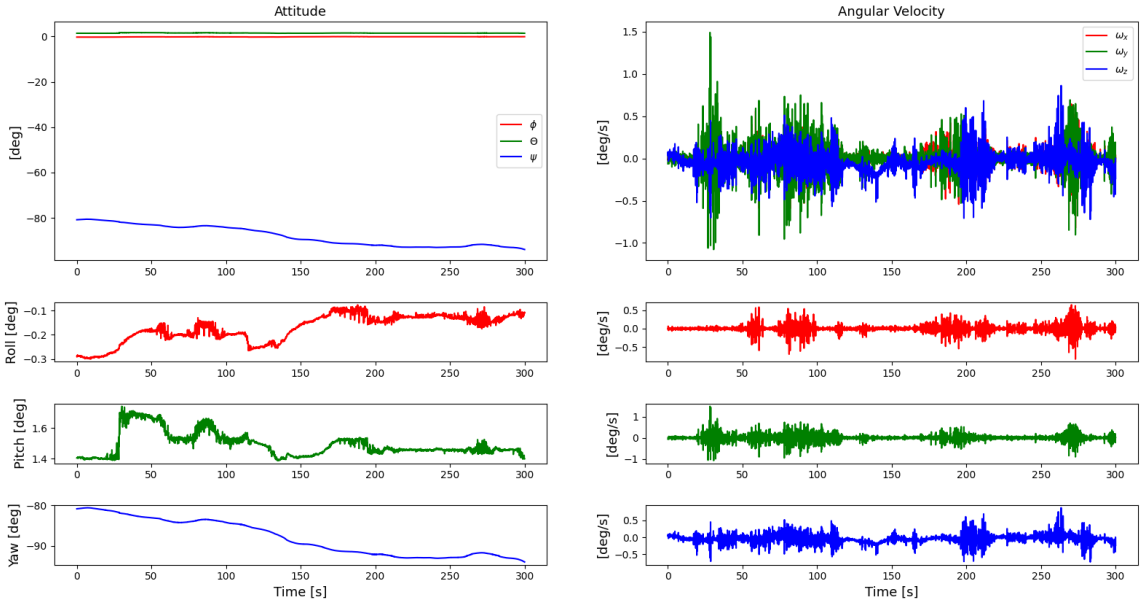


Figure C.1: Data Set A - no initial angular velocity. Attitude as Euler Angles (left), Angular Velocities (right).

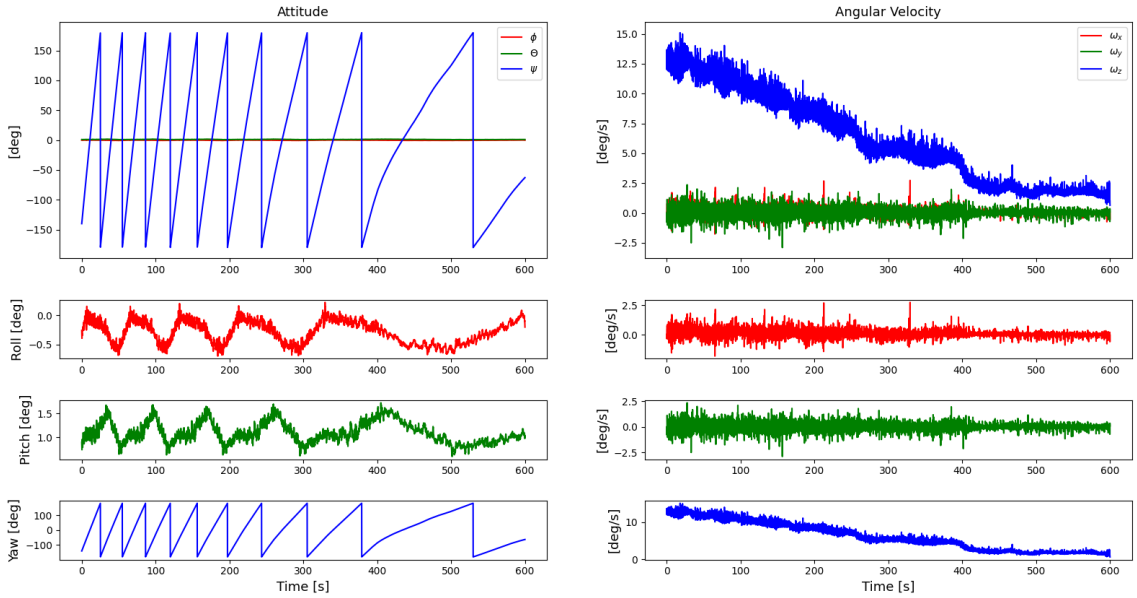


Figure C.2: Data Set B - small initial angular velocity. Attitude as Euler Angles (left), Angular Velocities (right).

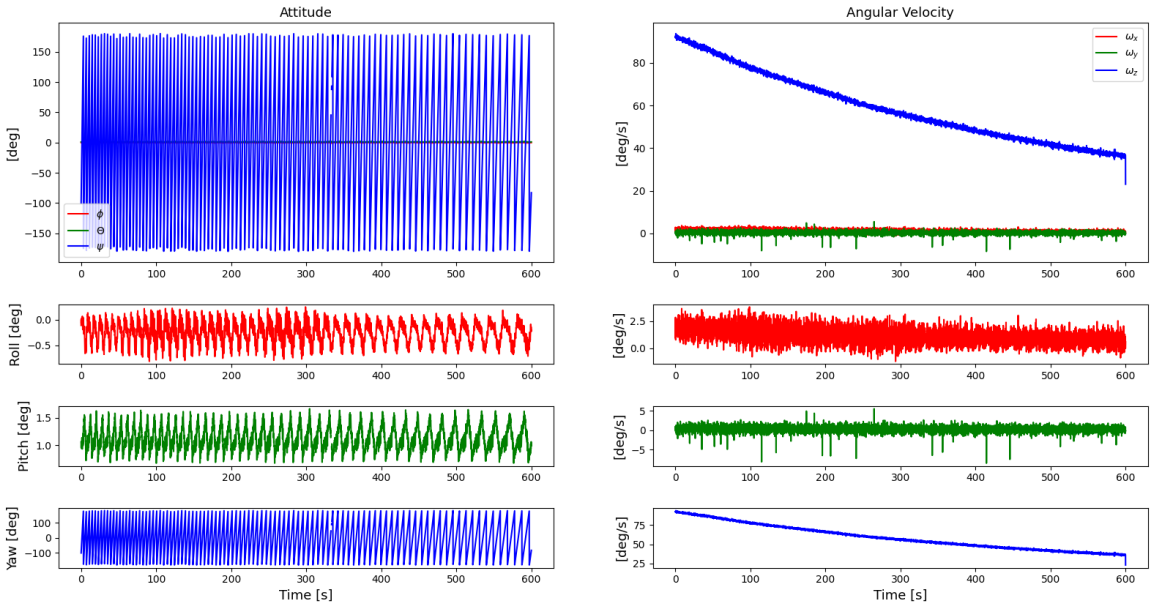


Figure C.3: Data Set C - medium initial angular velocity. Attitude as Euler Angles (left), Angular Velocities (right).

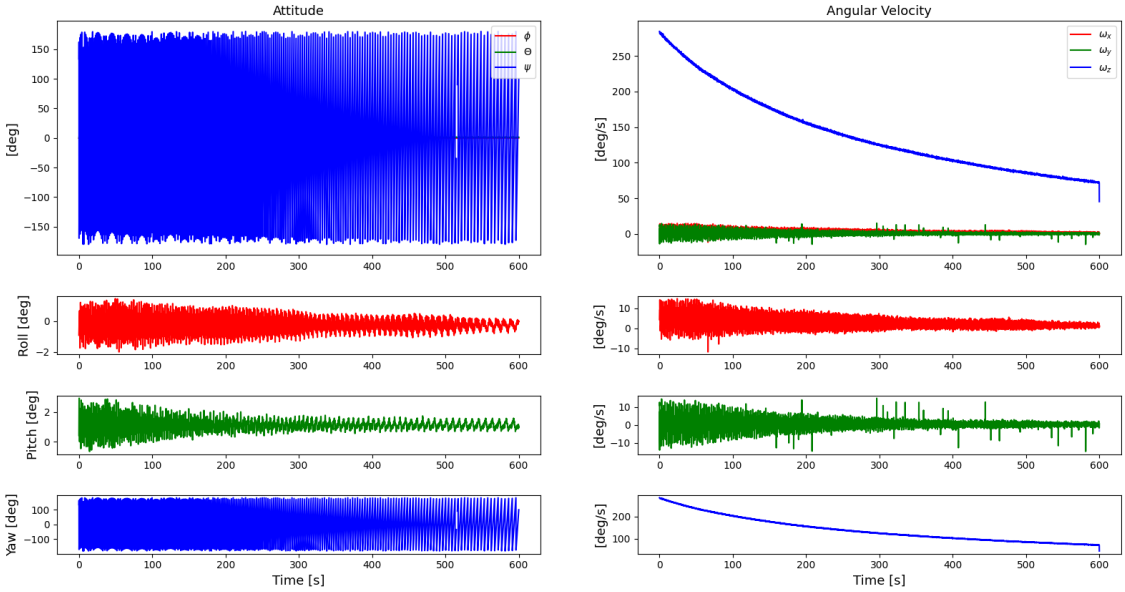


Figure C.4: Data Set D - high initial angular velocity. Attitude as Euler Angles (left), Angular Velocities (right).

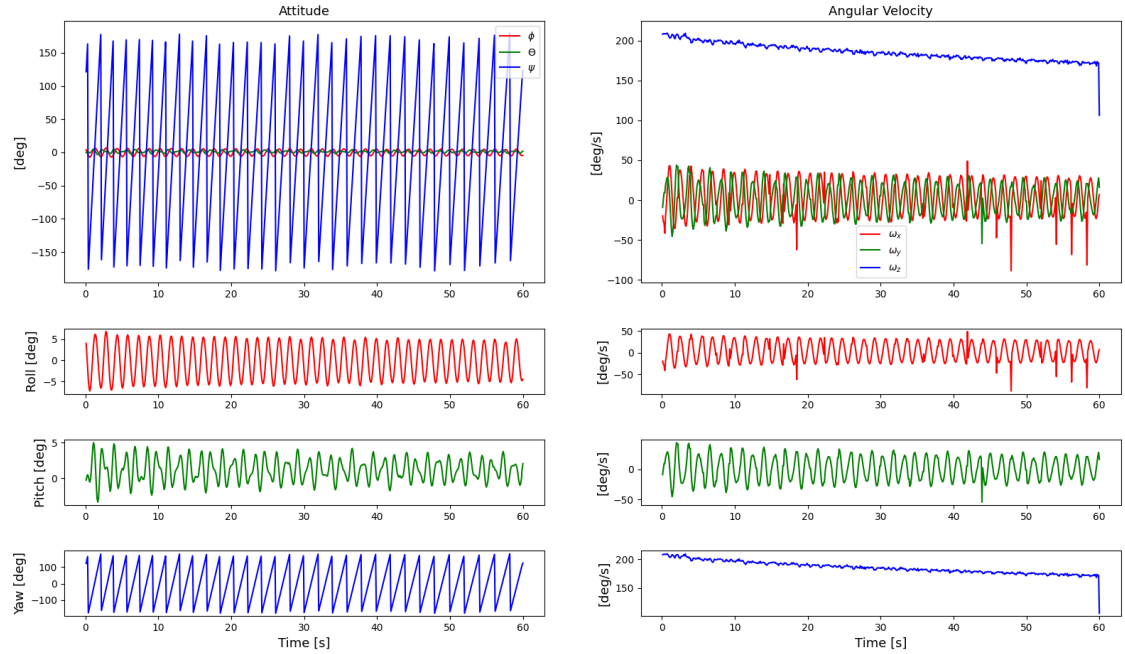


Figure C.5: Data Set E - tumbling motion. Attitude as Euler Angles (left), Angular Velocities (right).

Appendix D

Kalman Filter Approximation using recorded Data Sets

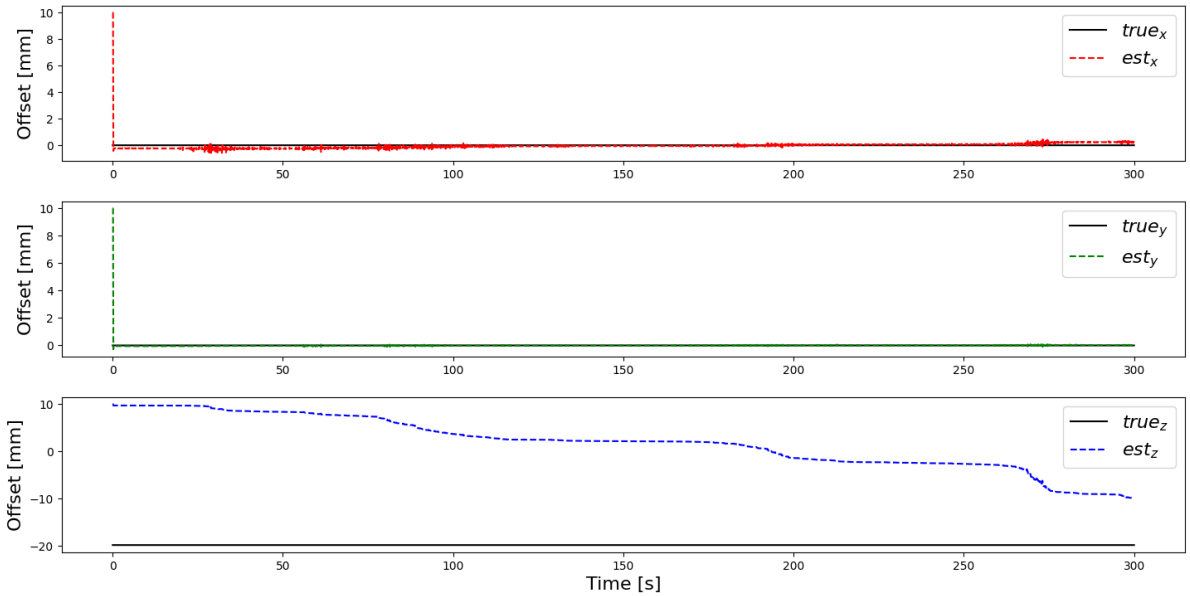


Figure D.1: Kalman Filter estimation with Data Set A - no initial angular velocity.

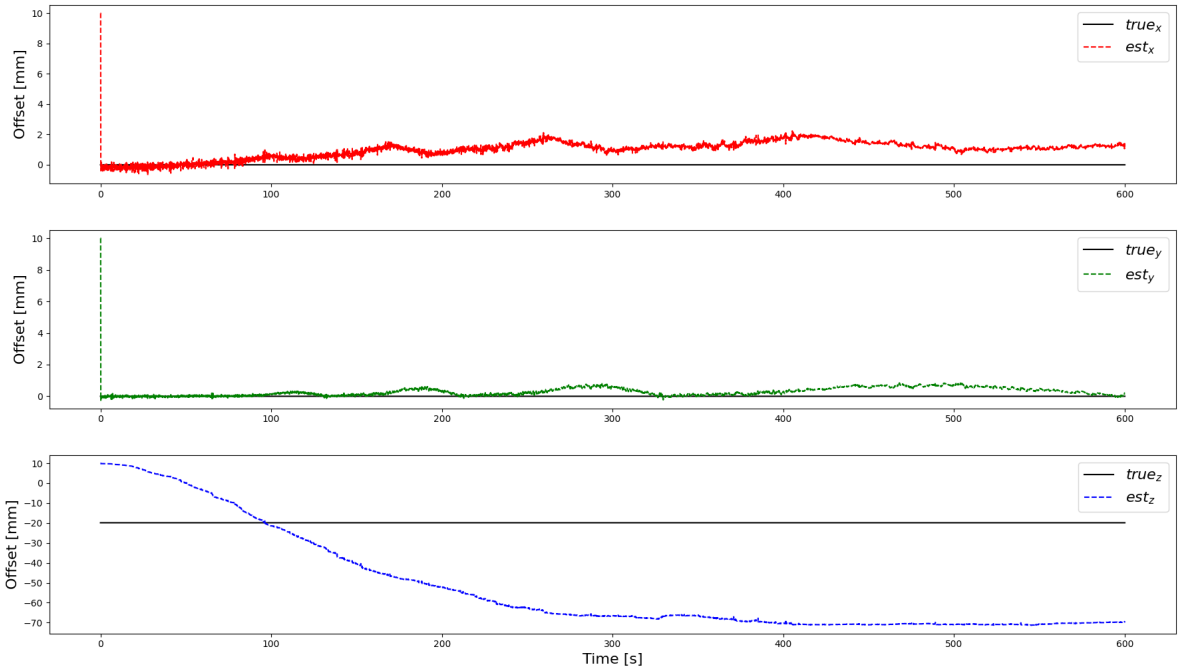


Figure D.2: Kalman Filter estimation with Data Set B - small initial angular velocity.

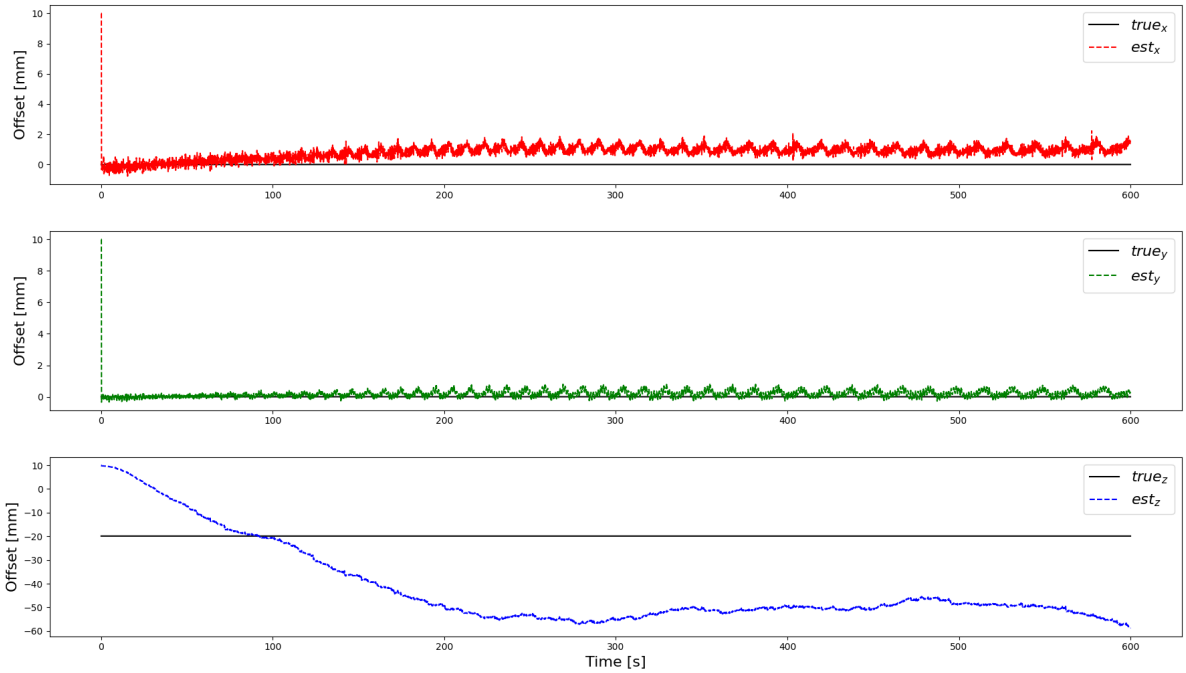


Figure D.3: Kalman Filter estimation with Data Set C - medium initial angular velocity.

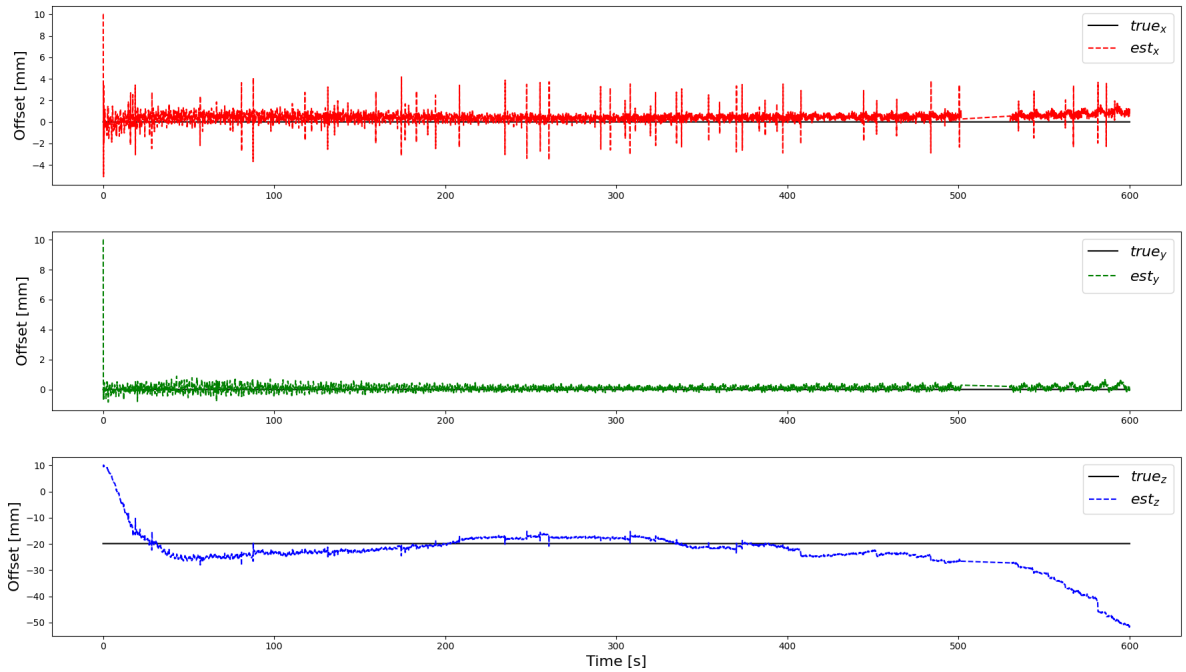


Figure D.4: Kalman Filter estimation with Data Set D - high initial angular velocity. From around 505 seconds until 525 seconds the data was corrupted during recording, thus no Kalman estimate can be calculated.

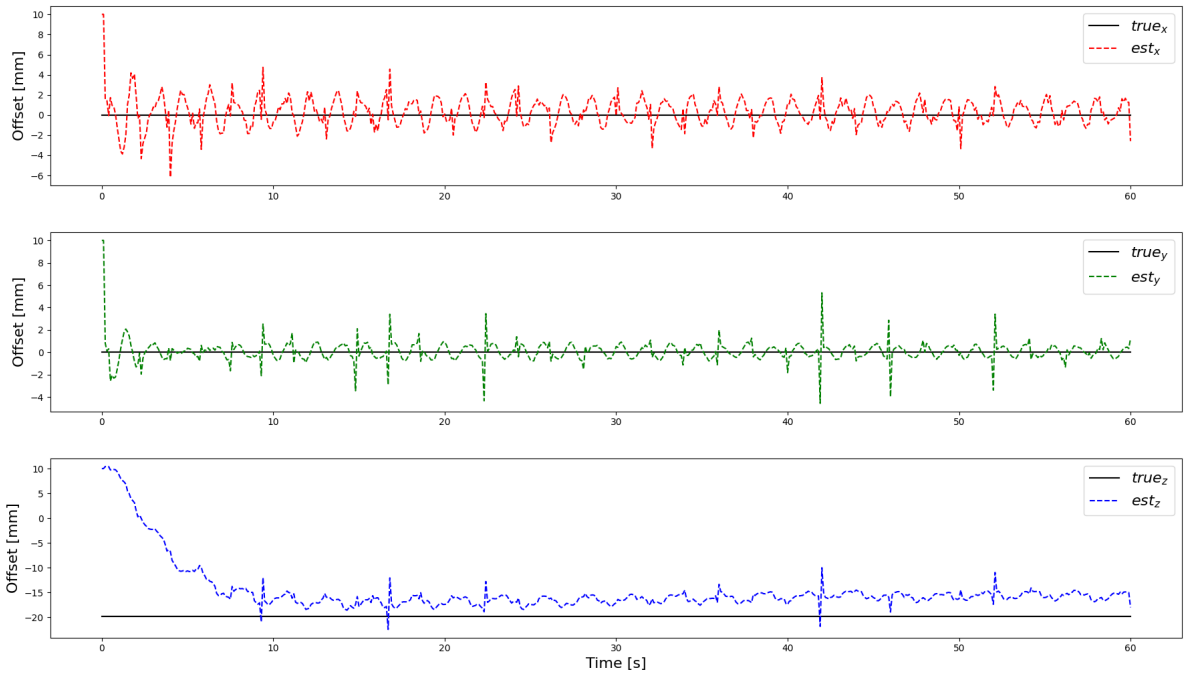


Figure D.5: Kalman Filter estimation with Data Set E - tumbling motion. Initial $r_{guess} = [10, 10, 10]^T$ mm

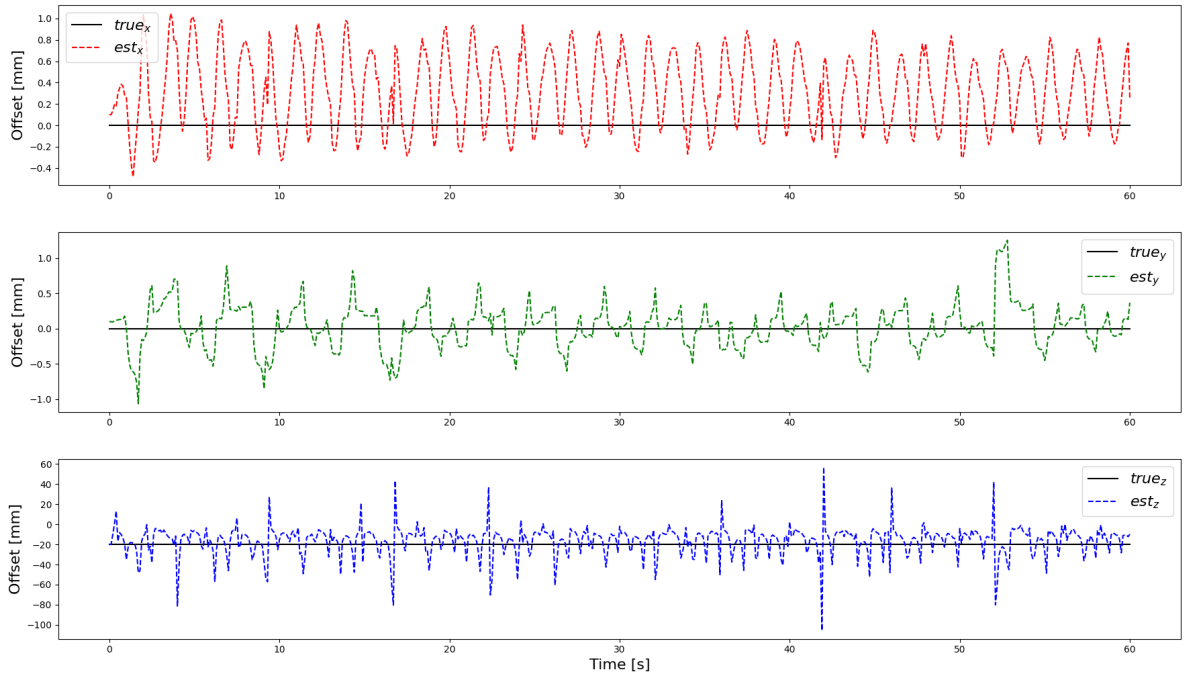


Figure D.6: Kalman Filter estimation with Data Set E - tumbling motion. Initial $r_{guess} = [0.1, 0.1, -20]^T$ mm

Appendix E

Impacts of changes in inertia matrix on simulation

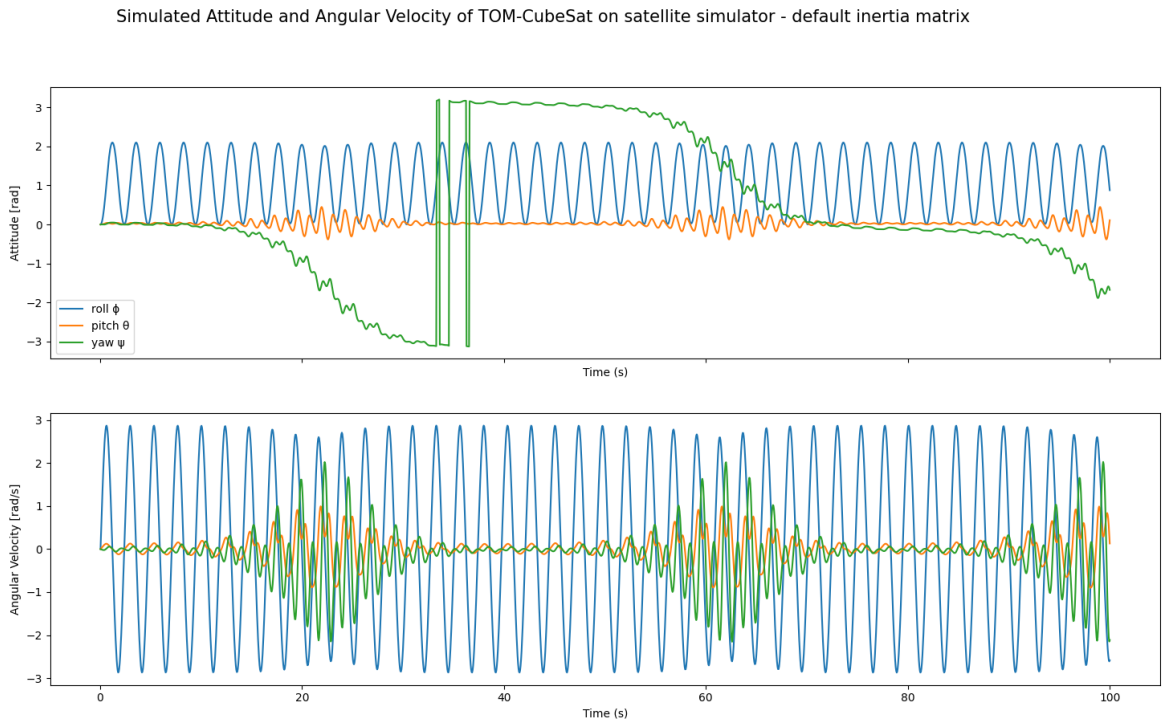


Figure E.1: Inertia matrix from TOM CAD-Model

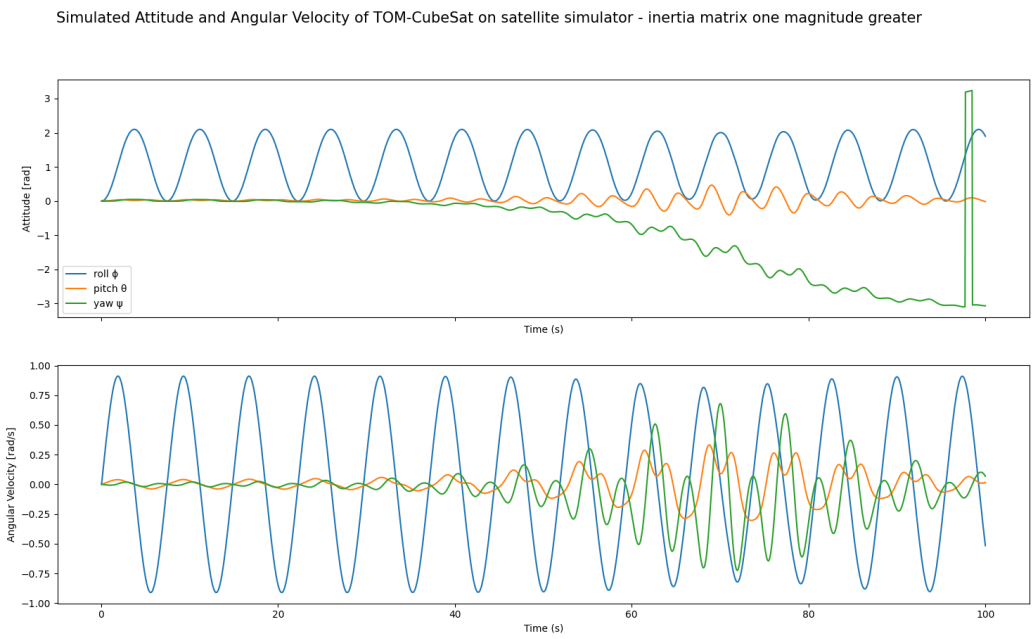


Figure E.2: Inertia matrix scaled by a factor of 10^1

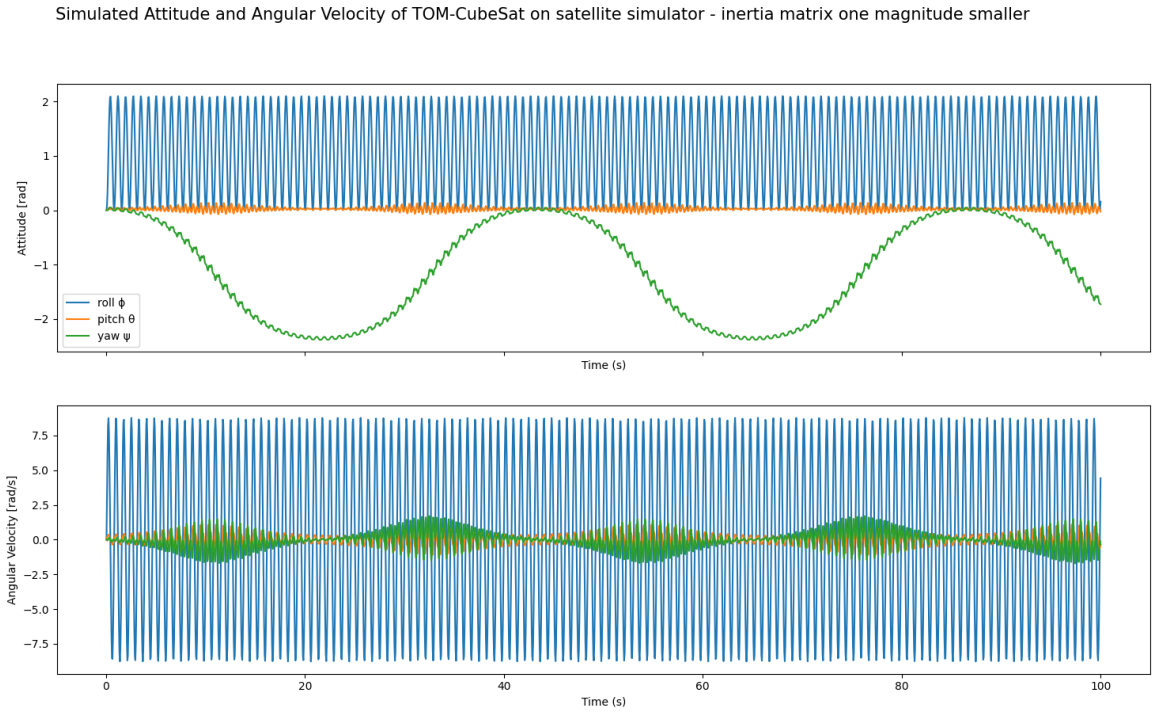


Figure E.3: Inertia matrix scaled by a factor of 10^{-1}

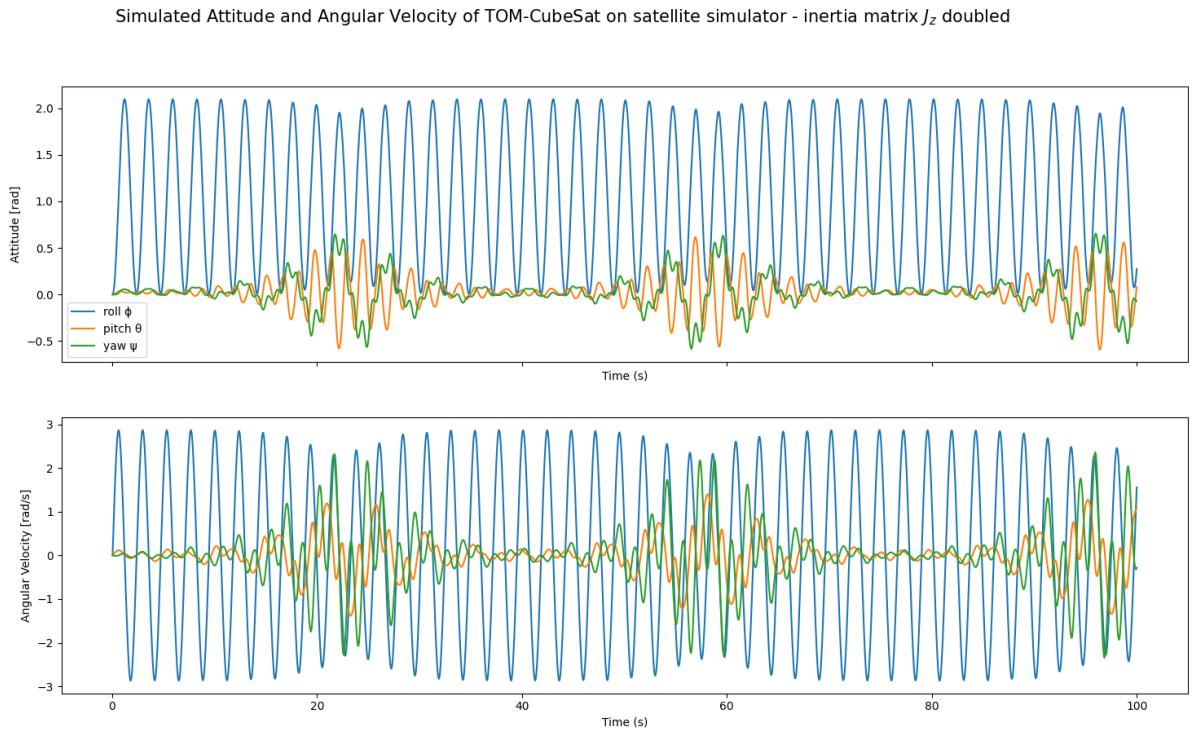


Figure E.4: Inertia matrix with J_z scaled by a factor of 2.