Control of a Ball-Balancing Robot

Cristina Roche, Victor Borja

June 2020

Energy Technology, MCE4-1027

Master Thesis

Simulation software: Matlab R2019b with Simulink 10.0. Text formatting software: Overleaf/Latex. Image formatting software: Inkscape.

Energy Engineering
Aalborg University
http://www.aau.dk

Title:
Control of a Ball-Balancing Robot

Theme:
Control and robotics

Project Period:
Spring Semester 2020

Project Group:
MCE4-1027

Participant(s):
Cristina Roche
Victor Borja

Supervisor(s):
Jan Dimon Bendtsen
Juan de Dios Flores Mendez

Copies: 1

Page Numbers: 184

Date of Completion:
June 25, 2020

Abstract:

A ball balancing-robot is a non-minimum phase, omni-directional mobile robot that balances on a ball with shape accelerated dynamics. It possesses five degrees of freedom and it is driven by three actuators, yielding an under-actuated MIMO system. This thesis covers the analysis of the dynamic model derived through Lagrangian mechanics and using quaternion representation, and it draws conclusions with regard to the implications of the unit quaternion operations.
Based on this model, two Unscented Kalman Filters (UKF) are implemented to deal with the measurement signals and are in charge of orientation and velocity estimation. The Quaternion UKF introduces a novel treatment of the quaternion to assure its unit norm. Velocity and balance control are achieved through a cascade control structure. To deal properly with the MIMO system and quaternion algebra, a modification of the non-linear Feedback Linearization Controller is implemented. An adequate reference tracking, equivalent to a Sliding Mode Controller performance, is achieved.
Simulation results prove the ability of the algorithms to control the system, although further laboratory tests are to be conducted to justify the higher complexity.

Preface

This project has been carried out within the Aalborg University Energy Engineering Master's degree programme in collaboration with Control & Automation Department, by a group formed by two students from the Mechatronic Control Engineering specialization. This report aims to show the results and insights obtained during the development of the project. It has been written in the period comprised between the 20th of February and the 26th of June of the year 2020. The authors would like to thank Jan Dimon and Juan Flores for their support and assistance during the completion of this project, as well as Thomas Kølbæk for his outstanding previous work on the ball-robot which has made possible to reach advanced research topics.

Aalborg University, June 25, 2020

Cristina Roche
<croche18@student.aau.dk>

Victor Borja
<vborja18@student.aau.dk>

Contents

Nomenclature

Table of notation

c	Scalar (non-bold letter)
$\boldsymbol{s}$	Vector (bold lowercase letter)
$\boldsymbol{M}$	Matrix (bold capital letter)
$\mathbf{A}$	System matrix in a state space model
$\mathbf{B}$	Input matrix in a state space model
$\mathbf{u}$	Input vector in a state space model
$\mathbf{y}$	Output vector in a state space model
$\mathbf{x}$	State vector in a state space model
$\dot{\mathbf{x}}$	Derivative of the state vector in a state space model
$\vec{i}, \vec{j}, \vec{k}$	Fundamental Cartesian unit vector
$\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$	Fundamental unit quaternions
${}^{A}_{B}\boldsymbol{R}$	SO(3) Rotation matrix from frame $\{B\}$ to frame $\{A\}$
${}^{A}_{B}\boldsymbol{T}$	SE(3) Transformation from frame $\{B\}$ to frame $\{A\}$
${}^{K}_{B}\boldsymbol{q}$	Quaternion describing the rotation of frame $\{B\}$ with respect to frame $\{K\}$ with a rotation axis defined in frame $\{K\}$
${}^{K}_{B}\dot{\boldsymbol{q}}$	Quaternion derivative describing the rotational velocity of frame $\{B\}$ with respect to frame $\{K\}$ with a rotation axis defined in frame $\{K\}$ derivative
$\boldsymbol{q}*$	Conjugated quaternion

$\vec{\boldsymbol{q}}$	Vector part of the quaternion $\boldsymbol{q}$
$\boldsymbol{q} \circ \boldsymbol{p}$	Quaternion multiplication (product operator)
${}^{K}_{B}\boldsymbol{q}_{e}$	Quaternion error between the reference and the actual value
$\boldsymbol{\Phi}(\boldsymbol{q})$	Matrix operator for Left quaternion multiplication: $\boldsymbol{q}\circ$
$\boldsymbol{\Gamma}(\boldsymbol{q})$	Matrix operator for Right quaternion multiplication: $\circ\boldsymbol{p}$
$\wedge$	Quaternion operator to construct quaternion form a 3-dimensional vector
$\vee$	Quaternion operator to extract 3-dimensional vector from a quaternion
$\hat{\boldsymbol{x}}$	Estimation of $\boldsymbol{x}$
n	Order of the system
r	Relative degree of the output
$G(s)$	Transfer function
s	Laplace transform variable
ϕ	Roll rotation angle
θ	Pitch rotation angle
ψ	Yaw rotation angle
$\sum$	Covariance
μ	Mean value
σ	Standard deviation
σ^2	Variance
$X \sim \mathcal{N}\left(\mu, \sigma^2\right)$	Normal probability distribution with mean μ and variance σ^2
$P(X)$	Probability density functions
ω	Process noise
$\boldsymbol{Q}$	Process noise covariance matrix
v	Measurement noise
$\boldsymbol{R}$	Measurement noise covariance matrix

$L_f h(x)$	Lie derivative of vector field h along vector field f
$L_g h(x)$	Lie derivative of vector field h along vector field g
f_s	Sampling frequency
T_s	Sampling period

Abbreviations

2D	Two Dimensions
3D	Three Dimensions
4D	Four Dimensions
Ball-robot	Ball-balancing robot
CL	Closed-Loop
COM	Center of Mass
DoF	Degrees of Freedom
EKF	Extended Kalman Filter
EMI	Electro Magnetic Interference
ES-EKF	Error State Extended Kalman Filter
FL	Feedback Linearization
FLC	Feedback Linearization Controller
GPS	Global Positioning System
HRI	Human-Robot Interaction
IMU	Inertial Measuring Unit
KF	Kalman Filter
LiDAR	Light Detection and Ranging
LPF	Low pass filter
LQR	Linear Quadratic Regulator

LTI	Linear Time-Invariant
MCU	Micro Controller Unit
MIMO	Multiple Input Multiple Output
MPC	Model Predictive Control
ODE	Ordinary Differential Equation
OL	Open-Loop
PI	Proportional Integral controller
QEKF	Quaternion Extended Kalman Filter
QUKF	Quaternion Unscented Kalman Filter
RHP	Right Half Plane (s-plane)
sgn	Sign function
SISO	Single Input Single Output
SMC	Sliding Mode Controller
UKF	Unscented Kalman Filter
UT	Unscented Transform
VEKF	Velocity Extended Kalman Filter
VUKF	Velocity Unscented Kalman Filter

Part I

Ball-robot system

Chapter 1

Introduction

The world of mobile robotics has been dominated by structures on platforms with one or two pairs of parallel wheels, due to its stable equilibrium with two/four points of contact with the ground and fairly easy and intuitive modeling and control. However, the parallel wheels architecture has the main inconvenience which is its non-holonomic properties in the trajectory. A non-holonomic trajectory implies that to reach certain positions the shortest way might not be available, which requires a roundabout to reach the desired position. Therefore, some of this moves presents difficulties for the parallel wheels architecture. One of the solutions to the non-holonomic limitation is the ball balancing robot.

The ball balancing robot, also called ball-robot, consists of a robot with a ball platform architecture. The path tracking in the ball-robot is achieved by generating the ball movement required. To do so, the ball-robot changes its inclination which generates an acceleration due to the gravity forces, allowing the ball-robot to follow paths in a holonomic way. The challenge for the ball-robot architecture consists of maintaining the desired orientation and therefore the equilibrium while achieving the desired path tracking.

Having holonomic properties allows ball-robots to have the capacity to move sideways without spinning the body, as well as perform more sophisticated maneuvers. Due to this fact, the ball-robot has been proposed as an architectural solution to robots intended for human-robot interaction.

1.1 Problem analysis

The ball-robot architecture was selected in Aalborg University for the 'Robot Digital Signage' [1] project, with the goal of developing a Human-Robot Interaction (HRI) under the use case of guidance and interaction with passengers in an airport. With the use case in mind, the ball-robot V1 was conceived with two main parts, the ball, and the body, whereas the total dimension was designed to reach the eye height, therefore a Human-Machine Interface, such as a tablet, can be displayed at the top of the robot.

The ball-robot should be capable of performing smooth maneuvers in a controlled environment, such as an airport, including path following, obstacles avoidance, standing still, and standing basic interaction with humans i.e. the force arising from a human touching the screen or an unpredictable collision. Thus, the mechanical design as well as the control of the ball-robot is designed to cope with the aforementioned functionality.

The basic operation principles of the ball-robot are based on two processes, standing still and moving.

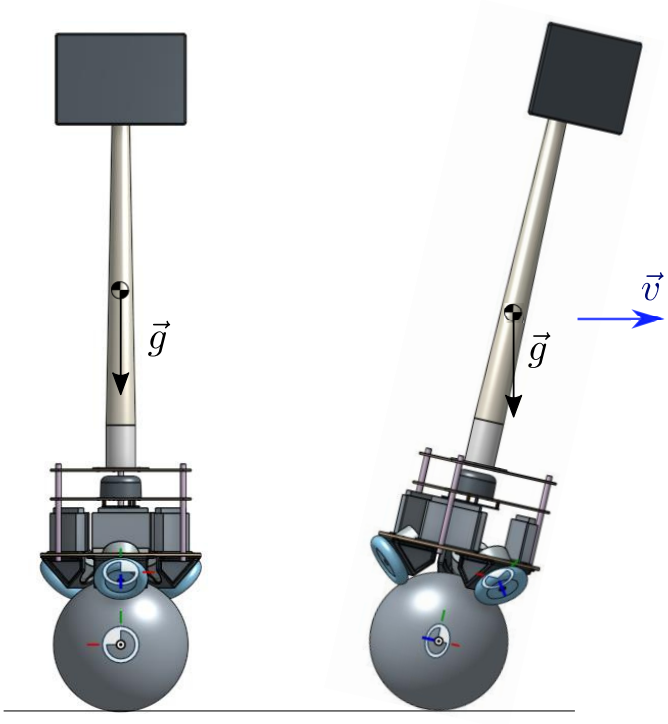


Figure 1.1: Ball-robot in standstill position and tilted for linear displacement [2].

In a standstill process, the ball-robot is ideally held perpendicular to the ground, where the gravitational force act in the center of mass without introducing any acceleration to the system, therefore maintaining the equilibrium.

On the other hand, the moving process is produced when the center of mass is not aligned vertically with the center of the ball, which causes an acceleration producing the movement of the ball-robot.

Both processes have to be controlled by the actuators of the robot through the controllers, assuring that they are performing as expected, either maintaining the equilibrium position and recovering it if lost or by maintaining the inclination required to reach the necessary translational velocity.

1.2 State of the art and previous work at AAU

The ball balancing robot architecture has been researched thoroughly for the last 15 years, capturing the interest of many researchers in different institutions. Among the research consulted, the most remarkable and cited tests are three; The Carnegie Mellon University (CMU), consisting on an inverse mouse-ball with four DC-motors [3], the Tohoku Gakuin University (TGU) using omni-wheels and stepper motors [4] and finally the Eidgenössische Technische Hochschule Zürich (ETH) with omni-wheels and geared brushless DC motors [5]. Besides those, the previous project at Aalborg University [6] makes thorough research and recapitulation on the recent studies around the ball balancing robot state of the art.

In mechanical terms, there are no great differences between the project's set-up characteristics. The typical configuration of the ball balancing robot consists of the aforementioned structure of a ball and a body. The body is designed with a narrow skeleton and allocating the mass as closest as possible to the ball and symmetrically distributed, hence, reducing the inertia and increasing the controllability. The ball-robots are mostly driven by three actuators and different force transmitters, including inverse-mouse drives and omni-wheels, being the last one the most extended. The main characteristic of the omni-wheel is that, besides the common behavior of a wheel, it allows axial translation, therefore transmitting the torque of the motors but not blocking the displacement of the ball caused by the other motors.

In terms of modeling strategies of ball balancing robots, different approaches can be found in the literature; The most simple one consists in modeling the robot in 2-Dimensions which highly decreases the complexity of the model, with the draw-

back of not being so accurate in representing the system. The ETH group used a projection strategy in two planes. Finally, the 3-Dimensions representation with a coupling between the body and the ball dynamics is the more accurate representation used. Inside the 3-Dimensions representations, there is also a distinction between projects in how to represent the rotations, either by Euler-angles, SO(3), or quaternions.

Finally, there is a great variety of control strategies used to stabilize the system and track the references given. The different strategies can be categorized into two groups, linear controllers and non-linear controllers.

The linear controllers vary in its complexity and results, from simple PD controllers, such as [4], to cascade structures to correct disturbances and linearization errors, as shown in [3]. On top of that, gain-schedule strategies for an LQR are developed by the ETH group [7]. The research conducted by ETH university continued the mentioned work by developing a time-varying LQR gain.

For the non-linear control strategies the most common strategy is the use of sliding mode controllers to account for the coupled non-linearities of the model [8], [9], [10], [11] and [6].

The previous ball-robot project at Aalborg University [6] included in its scope the recapitulation and detailed description of the balancing ball-robot state of the art, as well as a comparison between the different studies conducted, therefore, if a deeper insight into the ball-robots states of the art is required, the reader is referred to [6] to find an extended revision.

It is necessary to mention the outstanding work done by [6] as the predecessor of the current Master thesis. His extended job achieved various goals; it contains, among others, a detailed review and comparison of the state of the art in the ball balancing robot world; A quaternion 3D model of the balancing robot is derived and an ODE equation using the Euler-Lagrange equation is computed. Extended Kalman Filters are designed to cope with the noise of the sensors. A cascade control structure is developed using mainly a Sliding mode controller with a velocity controller. Finally, it is tested in the real version of the ball balancing robot at AAU facilities producing satisfactory results.

Therefore, the authors of the current thesis want to specially acknowledge the clear and detailed work presented by Jespersen, and also providing all the documentation and materials open-source.

1.3 Scope of the thesis

The current thesis is developed to face certain challenges and learning goals. First of all, getting insight into how to model and interact with a mechanical device with multiple degrees of freedom, under-actuated, advanced complexity, and couplings between the states. Secondly, learning to develop state of the art observers to deal with sensor noise, non-linear models, and model inaccuracies. Finally, developing advanced control strategies involving Multiple-Input Multiple-Output (MIMO) non-linear systems, where non-linearities and input-output couplings are into consideration.

Due to the characteristics of the ball balancing robot, it offers the challenges sought and satisfies the inquisitiveness and goals set.

Thanks to the previous work developed in [6], there is a solid background to start with the development as well as to compare results and assess the properties achieved by the algorithms developed. Therefore, throughout the current thesis, it is cited several times and its work is taken as a basis. Comparisons with the previous project are provided in order to throw conclusions and make advances in the field.

The "Future work" chapter introduced in [6] as well as his work is taken into consideration to chose the algorithms to be developed in this thesis, in order to explore new techniques that can overcome certain issues found. The variations introduced are summarized in Figure 1.2, where V2019 refers to the system developed by Jespersen and V2020 to the current version. The algorithms that vary from one version to the other are colored in orange to ease the reader.

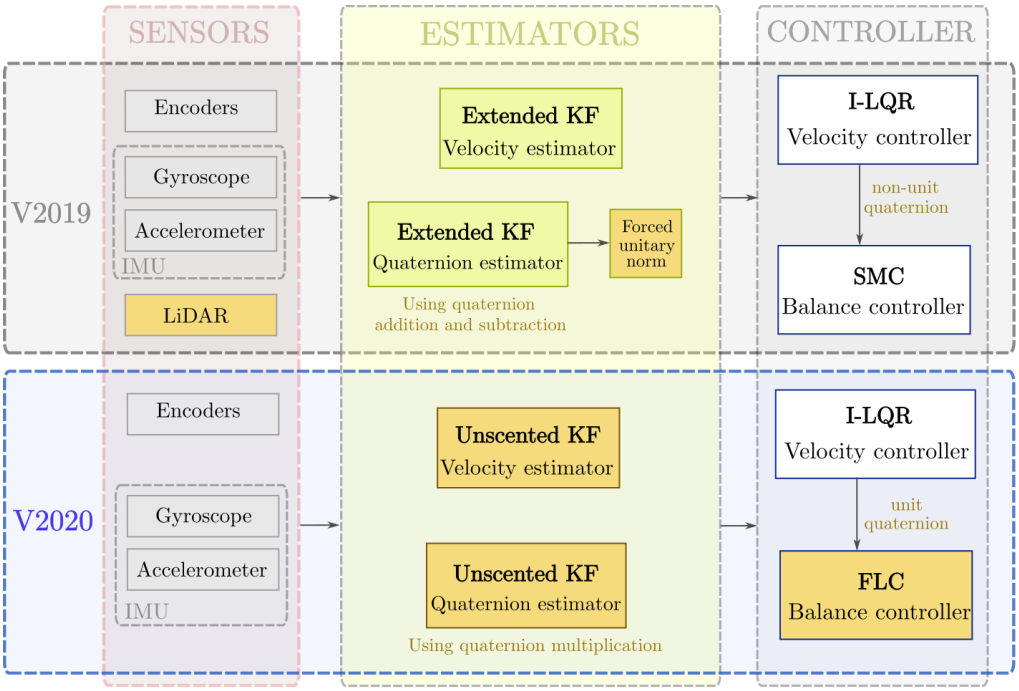


Figure 1.2: Schematic of the changes proposed in the current project compared to the ones implemented in the predecessor [6]. The main changes proposed are highlighted with an orange background or in orange text.

The scope of the current project does not cover interacting with the on-board computer of the robot, therefore the operative system ROS as well as the LiDAR readings are not available for the project development.

For the modeling of the robot, it has been decided to continue with the quaternion representation while introducing a small variation in the way of treating the quaternion and its operations to obtain the error. Previously done through a "brute force normalization" where the quaternions are driven on the unit sphere forcing the unit norm, which can yield deviations in the predictions. The proposed alternative is based on using only quaternion multiplication and no addition and subtraction, therefore the quaternions are kept unitary throughout the algorithm without requiring the forced unit norm.

The observers proposed are an alternative non-linear approach to the previously implemented Extended Kalman Filters. The Extended Kalman Filters linearise the model in every iteration which might lead to linearization errors, whereas the Unscented Kalman Filter is thought to deal with the non-linearities of the model, hence it is presumed to provide preciser results in the whole operation range. Furthermore, as proposed in "Future work" in [6], quaternion additions and subtrac-

tions are substituted also in the estimators' algorithms by multiplications in order not to introduce deviations in the estimates due to the forced norm. It becomes a relevant change to be implemented in the Unscented case, due to the number of additions included in the algorithm that might make the forced norm even more inaccurate. These required changes lead to implement an extension of the classical Unscented Kalman Filter.

Finally, the controller strategy studied in this thesis consists in the Non-linear Feedback Linearization Controller as an alternative to the Sliding Mode Control, seeking better performance in the tracking error and a more suitable control signal. The Sliding Mode Controller shows a chattering in the torque control signal commanded to the motors, that yields wheel slip related problems. On the other hand, when using the continuous switching law the stability can only be proven for the reaching phase. As shown in [6] it creates some oscillations around the desired equilibrium point that are aimed to be reduced by using the Feedback Linearization Controller strategy.

It was originally in the scope of this thesis to implement the results in the physical device. However, due to external circumstances the device and the facilities to test were not available.

1.4 Problem statement

Based on the scope of the thesis and according to the goals aimed to be achieved, the statement of the project can be summarized in three main groups

1. *Review, understand and develop the quaternion model with alternative quaternion operation.*
2. *Development of an advanced observer strategy for non-linear systems.*
3. *Derivation and implementation of an advanced non-linear control strategy.*

The performance of the different algorithms proposed will be evaluated throughout the thesis.

1.5 Project outline

The thesis is disposed in the following chapters

- **Chapter 2** details the main characteristics of the ball-robot as well as the principles of the model derivation.
- **Chapter 3** starts reviewing the observers state-of-the-art applicable to the system to justify the observers' type choice. It shows the derivation of the Unscented algorithm and its principles.
- **Chapter 4** covers the implementation of the orientation estimator, QUKF.
- **Chapter 5** describes the procedure to implement the velocity estimator, VUKF.
- **Chapter 6** presents the results obtained with the estimators.
- **Chapter 7** provides an overview of the system from a control perspective detailing the peculiarities and providing a preview of the control scheme chosen.
- **Chapter 8** describes the principles of the Feedback Linearization Controller and presents the results of the balance controller.
- **Chapter 9** shows the results obtained in simulation tests combining the controller with the estimators.
- **Chapters 10** discusses the insights obtained throughout the project.
- **Chapter 11** suggests possibilities for future work based on the current project achievements.

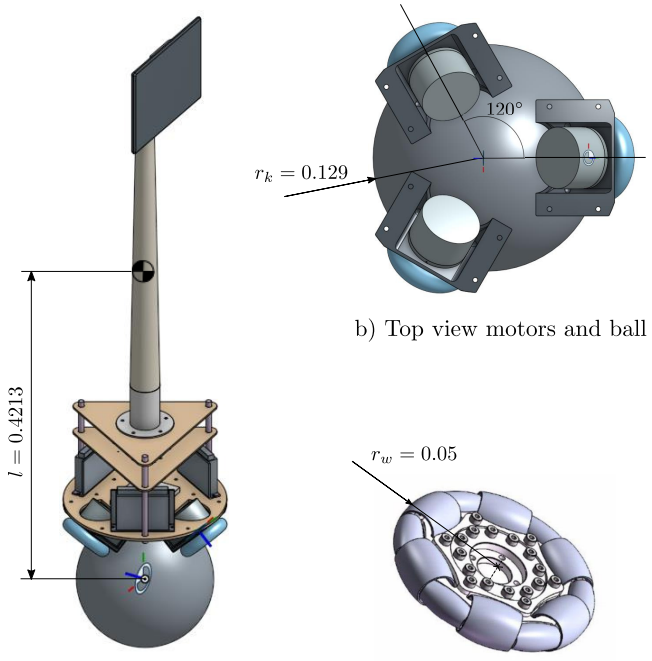
In regard to appendices of the thesis,

- **Appendix A** lists the parameters used in the ball-robot.
- **Appendix B** derives the kinematics of the model while in **Appendix C** the dynamics of the model are obtained.
- **Appendix D** shows a comparison of the different types of 3D rotation representations and justifies the quaternion representation choice.
- Finally, in **Appendix E** quaternion algebra required for the project is explained.

Chapter 2

Description of the system

The ball balancing robot onward referred to as ball-robot is designed seeking to adapt properly to the specifications of the use case. Therefore, it consists of a ball of 129 *mm* of radius with the body on top of it leading to the center of mass ending up at 421.3 *mm* above the center of the ball. For its design, it has been sought to distribute the mass such as the center of mass is the closest possible to the ball and symmetrically distributed. An illustration of the system is seen in Figure 2.1.



a) Front view

b) Top view motors and ball

c) Omniwheel

Figure 2.1: System main geometric parameters. *a*) and *b*) from [2]. *c*) from [12].

Three omni-wheels are equally distributed with a separation of 120° and a zenith inclination of 45°. Each wheel is driven by a different motor and has a radius of 50 *mm*. Finally, a list of the parameters of the ball-robot used for simulation is defined in Appendix A.

2.1 System architecture

The measurement system consists of three sensors sets, i.e., one encoder that measures the rotations of the motor and an Inertial Measurement Unit (IMU), which consists of an accelerometer, a gyroscope, and a magnetometer. However, the magnetometer is not used since the interferences with the environment of the room and the device itself introduce noise in the signal. The sensors readings are fed to the velocity and quaternion estimators and, trough an inverse kinematics computation, to the velocity controller, which is a Linear Quadratic Regulator with integral action (I-LQR).

The measurements of the sensors are normally noisy and not suitable for the controller, therefore two Unscented Kalman Filters estimators are placed in between in order to reduce noises and improve the signal, accounting for possible parameter deviations on the system model. The Quaternion Unscented Kalman Filter (QUKF) estimates the orientation of the body through the quaternion estimate and its derivative. These estimates are fed to the linear velocity Unscented Kalman Filter (VUKF) that estimates the linear velocity of the ball-robot.

A reference generator provides the desired translational velocity as well as the rotational velocity in the heading angle to the controllers' section. The controller scheme is designed as a cascade controller architecture. In the outer loop, a velocity controller (I-LQR) is placed, which computes the reference in the inclination required to obtain the desired reference tracking. The reference signal is expressed in quaternion representation and its derivative. The quaternion reference from the velocity controller together with the states estimated is fed to the inner loop, where a Non-linear Feedback Linearization Controller (FLC) is in charge of computing the signal required by the drivers to reach the desired reference.

An overview of the architecture is presented in Figure 2.2.

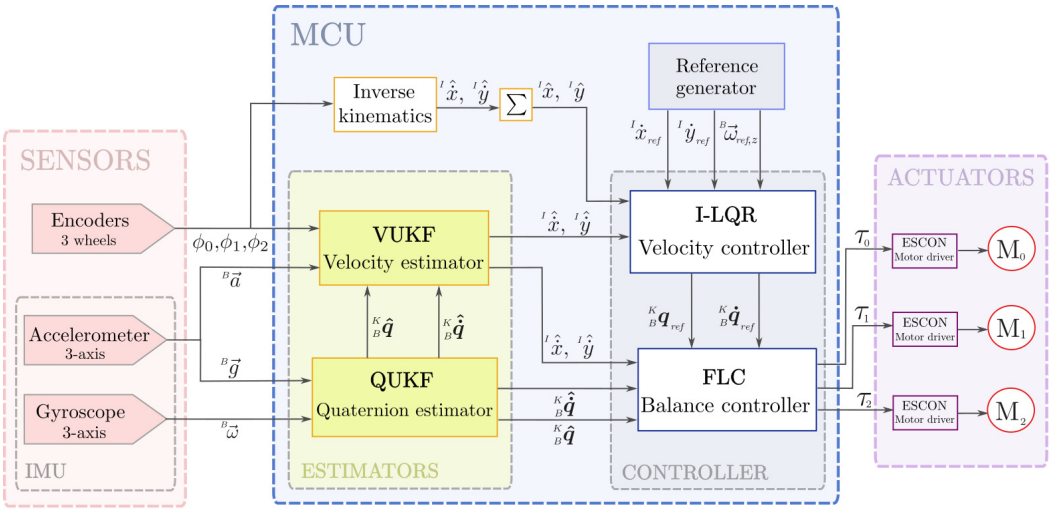


Figure 2.2: System architecture.

2.2 Model assumptions

There is a trade-off between the accuracy of the system and its complexity, and it must be assessed carefully in order to set the goals and characteristics of the model. The assumptions considered in the model and their implications have been defined upon the state-of-the-art of the ball balancing robots [5] and mainly based on the equivalent robot of the one presented in the project [6].

1. The parts of the system are considered rigid bodies. This idealization implies that they do not deform or change shape under the action of external forces, it can only translate or rotate. Three components of translation and three of rotation define a rigid body; hence, a rigid body has a maximum of six degrees of freedom.

2. The ball-robot is defined by two rigid bodies, the ball and the body, including the wheels. The body is simplified as a cylinder. The ball and body connection are considered a spherical joint, avoiding the parts to fall apart and allowing them to conceive the body as an inverted pendulum.

3. Wheels and motors masses are included in the body mass, although the motor and wheel inertia are evaluated within each of the wheels. The reason for it is that, at any given moment, the velocity and direction of the body with respect to the wheels and the wheels with respect to each other might be different.

4. The ball-robot will move in a flat environment, meaning that the ball only translates in the x-y plane, but not on the z-axis, therefore, limiting one degree of freedom in translation.

5. There is no slip between ball and omni-wheels, hence there is a kinetic relationship between the body's movement.

6. There is no slip between ball and ground; this allows us to connect the ball translation movement with its rotational movement.

7. The ball is considered not to rotate around the z-axis. Therefore the rotation of the body in the z-axis is not transmitted to the ball z-axis rotation.

8. The center of mass of the body ${}^{B}\boldsymbol{p}_{COM}$ location is in the geometrical center of the body, which provides an inertia tensor with only diagonals.

9. The actuators' dynamics are considered ideal. They are fast enough and considered without having a delay between the command and the response. The torque provided is the same as the commanded one.

10. The omni-wheels are connected to the ball in a point (${}^{K}\boldsymbol{p}_{c}$) and can only transmit tangential force while the displacement in other directions occurs without friction.

11. The friction forces considered in the model are: Viscous friction between the ball and ground, B_{vk}; viscous friction between the body and the air, B_{vb}; viscous friction arising from the motor angular velocity B_{vw}, which is composed of the internal motor friction and the friction between omni-wheel and the ball.

The ball-robot model is constructed under these assumptions and the parameters of the model are defined according to it in Appendix A.

2.3 System representation

The ball-robot representation shown in Figure 2.3 presents the system characterized by the two rigid bodies simplification, where the upper part of the robot, also referred to as the body, is represented as a cylinder.

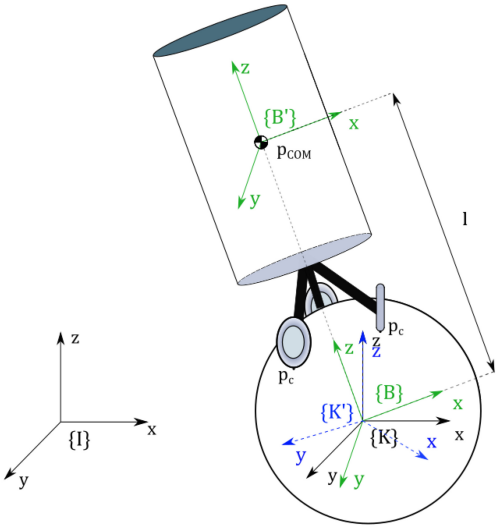


Figure 2.3: Rigid bodies representation of the ball-robot showing the main reference frames.

The parameters are defined in their respective frames. The definition of the frames is required to ease and simplify the representation, have a structured nomenclature, and systematically enable calculations.

2.4 Frames definition

The frames used throughout this project to define the model are the following:

- { I } : The inertial frame is considered the general reference frame. Determined in the initial position of the ball robot. It does not vary with time.
- { K } : The orientation of the axis of the frame K is the same as the inertial frame. It is located in the centre of the ball and translates with respect to the inertial frame in accordance with the horizontal translation of the ball.
- { K' } : It is rigidly attached to the centre of the ball, hence with the rotation of the ball the axis will rotate equally.
- { B } : Rigidly attached to the body, but located at the centre of the ball. The axes will rotate in accordance with the rotation of the body.

- $\{ B' \}$: Same orientation as the frame $\{ B \}$ but translated to be located at the center of mass of the body.

2.5 System characteristics

In this section the main concepts that define the characteristics of the ball-robot are defined and discussed for the system. These concepts are considered by the authors as key concepts for understanding the algorithms developed throughout the project.

The ball-robot is a system with five Degrees of Freedom (DoF). It is able to translate in two directions, i.e., x-axis and y-axis directions, and it can rotate in the three rotational directions, pitch, roll and yaw. The z-axis translation is limited by the assumption that the ball moves on a xy-plane.

The ball-robot is controlled through three motors which transmit torque to three different omni-wheels. An omni-wheel, also considered a poly wheel, is a wheel composed by rotary parts in its circumference. The specific structure of an omni-wheel is designed to transmit the torque tangentially to the circumference and parallel by the plane formed by the wheel, while the rotary parts included in the circumference allow the wheel to "slip" in the other directions [13]. The three motors are the actuators of the system. Hence, the ball-robot is defined by five DoF controlled by only three actuators, which make it an under-actuated system [14].

Depending on the number of actuators versus the DoF, a system can be considered of three types:

1. **Fully-actuated**: A fully actuated system is a system where the number of actuators corresponds to the degrees of freedom, there is a direct relationship between each actuator and the control of one of the coordinates; however, it can exist couplings between them.
2. **Over-actuated**: An over actuated system has more actuators than the required to control the coordinates of the systems. Hence there is some redundancy [15].
3. **Under-actuated**: Systems where the number of degrees of freedom is higher than the number actuators. Hence, by default there are couplings between coordinates. In other words, when controlling one of the coordinates another is collaterally affected.

It is required to use a mathematical representation capable of describing each of the possible positions, also known as configuration. Configurations are defined through a mathematical representation, also known as generalised coordinates of the system. If the generalised coordinates are chosen in a way that represents the different configurations with the minimum amount of independent variables, the generalised coordinates are known as minimal coordinates. However, there are some cases in which it is not possible to use minimal coordinates, or there are some advantages coming from the usage of other nomenclature. For those situations, the generalised coordinates include a set of dependent coordinates. As a consequence, it is required extra definitions, known as constraints, to maintain the problem solvable. The constraints can be of two types, holomonic or non-holomonic;

- **Holomonic:** A holomonic constraint limits the places in space where the system can reach; it is interpreted as a reduction of freedom. It can be integrated into a position state, which means that at any time it is possible to know the position through a geometric relationship.

$$g = (\chi, t) = 0 \tag{2.1}$$

- **Non-holomonic:** A non-holomonic constraint limits the directions of the system. It is a constraint in velocity, and it can not be integrated; it would require to know the information about the previous states, e.g. the order of the rotations applied with Euler angles determines the final position. It is not possible to determine the current position just by knowing the amount of rotation; hence, it requires the previous states, i.e., how the rotations have been produced with respect of time.

$$g = (\chi, \dot{\chi}, t) = 0 \tag{2.2}$$

The ball-robot system has five degrees of freedom; hence, it is required to have at least a set of five coordinates. The coordinates set depends on the system chosen to describe the rotations in the system. Rotations can be described through different systems of representation, while the most common systems are SO(3), Euler-Angles and quaternions. Each system has its properties with advantages and disadvantages. In the previous work in the homologous ball-robot [6] the representation with quaternions is used. Even though it was proven not to be a must, it is desirable to use quaternions due to its extensive use and advantages compared to Euler angles. A comparison of the different systems as well as the reasons of choosing quaternions for this project are explained in detail in Appendix D. The basics in quaternion algebra and their main concepts are developed in Appendix E.

Due to the quaternion representation choice, there are four generalised coordinates to represent the 3 degrees of freedom of the rotation. This results in the following set of generalised coordinates defined relative to the inertial frame.

$$\chi = \begin{bmatrix} x \\ y \\ q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.3}$$

To avoid confusion with the quaternions, the symbol χ is used instead of the commonly used q to refer to as the generalised coordinates. The coordinate z is not included since the displacement in the z-axis is not allowed. Finally, the set of generalised coordinates is composed by six variables, while the system has five degrees of freedom, meaning that it is not a set of minimal coordinates, and it requires the use of constraints arising from the quaternions.

The generalized coordinates define the configuration of the system. The states of the system x that are used to control the system, and therefore to define the state-space, are the generalized coordinates and their derivatives. The choice of these states is because they represent the complete position and orientation of the frame $\{B\}$, i.e. its translational velocity and rotation with respect to the inertial frame, and therefore they define completely the system dynamics.

$$x = \begin{bmatrix} \chi \\ \dot{\chi} \end{bmatrix} = \begin{bmatrix} x \\ y \\ {}^{K}_{B}q \\ \dot{x} \\ \dot{y} \\ {}^{K}_{B}\dot{q} \end{bmatrix} \tag{2.4}$$

The ball-robot is also characterised as being a shape accelerated robot, which means that it is destabilized by gravitational forces and has non-integrable constraints in its dynamics [16]. The body inclination angle with respect to the vertical axis generates a linear acceleration that makes it translate. Being shape-accelerated combined with having the actuators at a lower height than the COM of the body makes the system present non-minimum phase dynamics, which is due to Right Half Plane (RHP) zero behaviour. As a consequence, the ball-robot has inverse response behaviour at the beginning of the response. For example, when com-

manding a reference of linear displacement in the x-axis, the ball-robot initially has to move to the negative x-axis direction before starting moving to its positive steady-state value, yielding a delay in the response.

2.6 Model notation

A standard notation for deriving the model is created for the sake of simplicity and avoid ambiguity to the reader.

The velocity is relative and therefore, it has to be given as a velocity of the subject frame respect to another frame, called reference frame or observer, from where it is observed. Thus, the subscript is composed of the frame from where the velocity is measured, followed by the name of the frame from where the movement is observed. Furthermore, the vector has to be described in a frame coordinates, which is indicated with a superscript. In case of its absence, it is assumed that it is expressed in the reference frame.

According to this convention, the linear velocity $\boldsymbol{v}_{B|A}$ (also referred to as tangential velocity) is the velocity vector at which the origin of the frame $\{B\}$ is moving relative to a reference frame $\{A\}$, i.e. the linear velocity of frame $\{B\}$ seen from an observer placed on frame $\{A\}$.

A rotation of a vector, applied either by transformation matrices or quaternions, only changes the frame in which the vector is expressed, and it will not affect the reference frame. Equation (2.5) describes the velocity of the reference frame $\{B\}$ relative to the reference frame $\{A\}$, being the velocity expressed in $\{C\}$ frame, which is accomplished through a pure rotation.

$$^{C}\boldsymbol{v}_{B|A} = {}^{C}_{A}\boldsymbol{R}^{A}\boldsymbol{v}_{B|A} \tag{2.5}$$

Same principles and notation apply for angular velocities but describing the speed and direction of a frame rotating relative to the reference frame $\boldsymbol{\omega}_{B|A}$.

2.7 Model of the system

The main goal of the project is to be able to control the ball-robot in a defined environment. The control strategies and estimators studied in the thesis make use of the system model. The model describes mathematically the response of the ball-robot to stimulus coming from the system itself or the environment. Hence,

it is correct to assume that with an accurate model of the system, correct control of the ball-robot is achievable, and simulations can be performed to replicate the real set-up behaviour. The model of the system has been based on previous works, especially in the model from 2019 at AAU [6], seeking for an in-depth understanding while trying to improve the previous work, and providing detailed information about the concepts and methodology needed to create the model. In the current section, a summary of the model derivation is shown, while the main explanation and derivations can be found in Appendix B and C.

To model the system behaviour, firstly, it must be described by the laws of kinematics. The kinematic model is derived and explained in Appendix B. The goal is to relate the system states to the actuator variables, and vice-versa.

The dynamic model makes use of the obtained kinematic relationships to substitute and leave the model in terms of the state variables. The dynamic model describes the behaviour of the system and is used for prediction of the system outputs, i.e. it connects the states and system inputs with the derivative of the system states, which are the next instant values of the system.

The dynamic relationships can be mainly established through the Newton-Euler equation or Lagrangian mechanics. In this project, the dynamic model derivation is developed based on Lagrangian mechanics and can be found in Appendix C. It leads to the standard Euler-Lagrange equation

$$\boldsymbol{M}(\chi)\ddot{\chi} + \boldsymbol{C}(\chi,\dot{\chi})\dot{\chi} + \boldsymbol{G}(\chi) + \boldsymbol{D}(\dot{\chi}) - \boldsymbol{H}_{hol}^{T}(\chi)\boldsymbol{\lambda}_{hol} + \boldsymbol{H}_{non}^{T}(\chi)\boldsymbol{\lambda}_{non} = \boldsymbol{Q}(\chi,\dot{\chi},\boldsymbol{u}) \tag{2.6}$$

Furthermore, the equation is modified for convenience into a more manageable equation including the constraints required for the system while having a suitable form as an Ordinary Differential Equation. This form is better suited for control purposes yielding the final ODE dynamic equation of the system (2.7), which is further used for the estimators and controllers development.

$$\widetilde{\boldsymbol{M}}(\chi)\ddot{\chi} + \widetilde{\boldsymbol{C}}(\chi,\dot{\chi})\dot{\chi} + \widetilde{\boldsymbol{G}}(\chi) + \widetilde{\boldsymbol{D}}(\dot{\chi}) = \widetilde{\boldsymbol{Q}}(\chi,\dot{\chi},\boldsymbol{u}) \tag{2.7}$$

The solution of these equations of motion describes the system states dynamics and therefore the behaviour of the whole system.

Part II

Kalman Filter Estimators

Chapter 3

Estimators introduction

The dynamic model defined in Appendix C is aimed to provide an accurate representation of the behaviour of the system. However, the model states differ from the real states of the system due to a variety of reasons.

One reason comes from the inaccuracy of the mathematical model due to the assumptions and the parameter uncertainties. Besides that, there could be scenarios not considered in the model, including the response of the system to external events, also known as disturbances. Hence, it yields a drift in the values obtained by the model.

Another relevant source of error is caused by the inaccuracy of the sensors used in the system. Most of the systems use mechanism or devices to "read" a specific parameter. Those devices are prompt to drift, either by noise inherent to the device and the system or mechanical limitations, i.e. the precision of an encoder is limited by the number of readings per full rotation.

For some applications, neither the model predictions (sensor-less) or the readings from the sensors are accurate or reliable enough to be fed to the controller. In some cases, the controller is required to control some states that the sensors can not measure and they need to be estimated. A state estimator, also called Luenberger observer, is aimed to account for the inaccuracies of the model and drive the estimated parameters to converge to the real ones in the presence of disturbances or uncertainties [17]. The schematic of an estimator using the linear standard state-space nomenclature is shown in Figure 3.1.

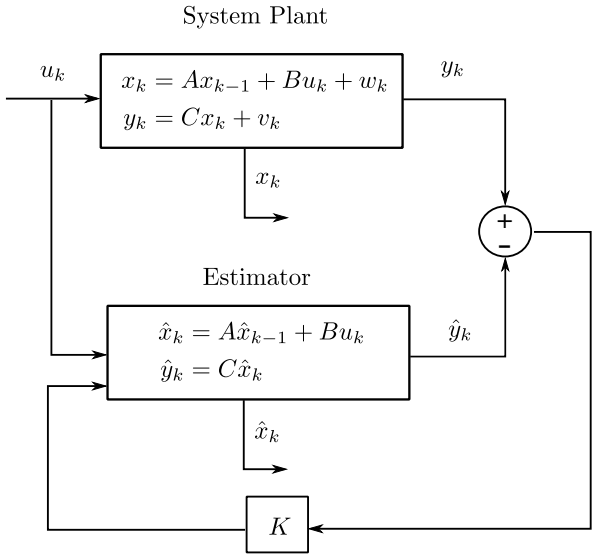


Figure 3.1: Block diagram of a system with an estimator, inspired by [18].

The equations governing the error of a state estimator [19] are in the form of

$$\begin{aligned} e_{est} &= x - \hat{x} \\ \dot{x} &= Ax + Bu \\ \dot{\hat{x}} &= A\hat{x} + Bu + K(y - \hat{y}) \end{aligned} \tag{3.1}$$

while the error dynamics are defined by

$$\begin{aligned} \dot{e}_{est} &= (A - KC)e_{est} \\ e_{est}(t) &= e^{(A-KC)t}e_{est}(0) \end{aligned} \tag{3.2}$$

The error of the mathematical model is determined by subtracting the real states minus the mathematical model.

The Luenberger observer has to be applied to a deterministic system, in which no randomness is involved in the development of future states of the system. However, real systems suffer from random noise, v, in their sensor signals. When dealing with sensors the system is considered stochastic, i.e. a random probability distribution is expected in the output measurements from the sensors [20], and therefore the estimator is required to be able to handle this noise statistically. Hence, despite the fact that the structure of a Luenberger observer and a Kalman filter looks alike, the gain of the Luenberger observer is just an adequately chosen constant for convergence rate. In contrast, in a Kalman filter the gain is optimally

derived taking into account the variance of the measurement and process noise and updated in every iteration. In addition, Kalman Filter estimators allow to combine the measurements of multiple sensors that are reading the same states or indirectly the same states [21] and, applying the combination of the readings obtain a more precise measurement. The use of the technique combining multiple measurements is often referred to as sensor fusion.

In this chapter, the model of the sensors are derived in order to create the noise in the simulation as well as to derive the noise covariance needed for the Kalman filter estimators. At the end, a discussion about the choice of the type of Kalman filter is provided.

In Chapter 4 and Chapter 5, the Unscented Kalman filters for orientation and velocity estimation are derived respectively.

The results of the estimators dealing with the noise created by the sensors in simulation are shown in Chapter 6.

3.1 Sensor modelling

The available sensors' set consists of three-wheel encoders and an Inertial Measurement Unit (IMU), consisting of a three-axis gyroscope, three-axis accelerometer and a three-axis magnetometer.

IMU's are capable of providing orientation changes measurement through the gyroscope, changes in linear velocity through the accelerometer, and heading orientation measurements using the magnetometer. However, experiences in the previous projects on the ball-robot had proven the magnetometer to be under the influence of large disturbances due to the Electro Magnetic Interference (EMI) [6]. Hence, it is not being considered for the measurements. The wheel encoders are electromechanical sensors capable of reading the rotation of the axle, to do so, a full revolution is divided into different reading points or ticks, and every time the angle lands in one of the points a signal is sent through the encoder.

The measurements of the IMU are noisy and integrated to obtain the states of the system. However, integration leads to dead-reckoning positioning systems. Therefore, the IMU measurements are biased, and a bias integrated over the time causes drift in the measurement. For a long-running time, this drift can trigger an utterly wrong state estimation, i.e. wrong position and orientation of the ball-robot. This drift might be negligible in some cases and simplifies the algebra, while it is relevant for most of the cases. It is therefore essential to assess its relevance

for each specific application, e.g. in autonomous cars, a drift effect causes the car not to keep the lane properly and it is a non-admissible effect [21].

Avoiding drift is a matter of fusing this information with absolute position readings, such as the ones from GPS, LiDAR or a vision device that can reset this drift [22]. However, this absolute readings also have noise, and in order to treat them properly, they need to be considered through a Kalman Filter. The accelerometer and gyroscope biases for each axis can be incorporated as states of the Kalman Filter to estimate them and correct them with the sensor sources.

However, it is proven in [6] that the observability of the gyroscope is not possible if LiDAR measurements or any supplementary sensor reading of the states related to the bias are not available. The reasons of the unobservable bias come from the accelerometer model used for quaternion estimation and are explained more in-depth in Section 4.2. As a consequence, only the accelerometer bias can be estimated. The drift effect is expected to be admissible for this application since it is a short running time application and the integration of the bias is not accumulated for a long time. Furthermore, by means of an accurate initial calibration, the biases can be close to zero in the beginning and therefore do not lead to a significant drift.

In the ball-robot set-up, the centre of mass of the body is not accurately known. Furthermore, the IMU installation place might differ from the desired one. For these reasons, it might contain misalignments with the orientation of the body. Hence, a specific frame for the IMU, $\{IMU\}$, is defined. Finally, the models of different parts of the IMU are defined following the work of [23].

3.1.1 Accelerometer

Accelerometer measurements are generated through the use of two components, an internal measurement which is reading the gravity force, and an external case measuring all the accelerations to which the IMU is subjected to. The internal case readings are subtracted to the external case readings to calculate the total acceleration. Hence, in a free fall under the influence of gravity, an accelerometer would measure zero acceleration due to gravity. However, while being on the surface of the earth, it measures a positive gravity vector.

$$ {}^{IMU}\boldsymbol{z}_{acc} = \boldsymbol{K}_{acc}\, {}^{IMU}_{I}\boldsymbol{R} \left({}^{\mathrm{I}}\ddot{\boldsymbol{O}}_{IMU|I} - {}^{I}\boldsymbol{g} \right) + {}^{IMU}\boldsymbol{b}_{acc} + {}^{IMU}\boldsymbol{v}_{acc} \tag{3.3} $$

Equation (3.3) models the general form of an accelerometer, where $\boldsymbol{K}_{acc}$ is a scale factor specific for each IMU. The rotation matrix ${}^{IMU}_{I}\boldsymbol{R}$ transforms the readings

from the IMU frame $\{ IMU \}$ to the inertial frame $\{ I \}$. The bias of the sensor is represented by ${}^{IMU}\boldsymbol{b}_{acc}$ and the noise corresponding to the sensor is denoted as ${}^{IMU}\boldsymbol{v}_{acc}$, which is represented as an additive white noise with a Gaussian distribution; ${}^{IMU}\boldsymbol{v}_{\text{acc}} \sim \mathcal{N}\left(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{acc}}\right)$.

The covariance of the accelerometer $\boldsymbol{\Sigma}_{\text{acc}}$, as well as the rest of the covariances of the sensors defined throughout the project, are defined based on the ones used in the previous project [6]. The accelerometer covariance is extracted from a test performed on a MPU-9250 IMU, and is given by

$$\Sigma_{\text{acc}} = \begin{bmatrix} 0.3945 & 0.0006 & -0.0234 \\ 0.0006 & 0.3922 & -0.0036 \\ -0.0234 & -0.0036 & 0.9941 \end{bmatrix} \cdot 10^{-3} \tag{3.4}$$

The gravity vector in the inertial reference frame is defined as

$${}^{I}\boldsymbol{g} = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} \tag{3.5}$$

The IMU is assumed to be perfectly calibrated and the measurements readings have been transformed to be expressed in the body frame, yielding the following equation for the accelerometer model

$${}^{B}\boldsymbol{z}_{acc} = {}^{K}_{B}\hat{\boldsymbol{q}}^{T} \circ \left({}^{\text{I}}\ddot{\boldsymbol{O}}_{IMU|I} - {}^{I}\boldsymbol{g} \right) \circ {}^{K}_{B}\hat{\boldsymbol{q}} + {}^{B}\boldsymbol{b}_{acc} + {}^{B}\boldsymbol{v}_{acc} \tag{3.6}$$

3.1.2 Gyroscope

The gyroscope measures the changes in orientation of the IMU and depending on the quality of the IMU it is more or less prompt to give a noisy reading and drift due to the time-varying bias.

$${}^{IMU}\boldsymbol{z}_{gyro} = \boldsymbol{K}_{\text{gyro}} {}^{IMU}_{I}\boldsymbol{R}^{IMU}\boldsymbol{\omega}_{IMU|I} + {}^{IMU}\boldsymbol{b}_{gyro} + {}^{IMU}\boldsymbol{v}_{gyro} \tag{3.7}$$

The accelerometer gain $\boldsymbol{K}_{\text{gyro}}$ is also a sensor-specific constant scale factor, as well as the bias of the sensor $\boldsymbol{b}_{gyro}$. The noise of the measurements $\boldsymbol{v}_{gyro}$ is also modeled as a Gaussian white noise, ${}^{IMU}\boldsymbol{v}_{\text{gyro}} \sim \mathcal{N}\left(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{gyro}}\right)$, where $\boldsymbol{\Sigma}_{\text{gyro}}$ is extracted from

the previous project [6] test in a MPU-9250, leading to the following values

$$\Sigma_{\text{gyro}} = \begin{bmatrix} 0.4969 & 0.0201 & 0.0035 \\ 0.0201 & 0.1749 & -0.0260 \\ 0.0035 & -0.0260 & 1.3970 \end{bmatrix} \cdot 10^{-3} \tag{3.8}$$

Due to the nature of the gyroscope, the readings are influenced by the bias, which may drift over time as an stochastic process. The rate of change of the bias is modelled as a white Gaussian noise, ${}^{IMU}\boldsymbol{\omega}_{\text{gyro}} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{bias}})$, with $\boldsymbol{\Sigma}_{\text{bias}}$ being a tunable parameter

$${}^{IMU}\dot{\boldsymbol{b}}_{gyro} = {}^{IMU}\boldsymbol{w}_{bias} \tag{3.9}$$

Assuming the gyroscope to be calibrated, the equation of the gyroscope yields to

$${}^{B}\boldsymbol{z}_{gyro} = {}^{B}\boldsymbol{\omega}_{B|I} + {}^{B}\boldsymbol{b}_{gyro} + {}^{B}\boldsymbol{v}_{gyro} \tag{3.10}$$

3.1.3 Encoder

The encoders measure the variation of the angle position of the motor axle. The variation measured is limited by the resolution of the encoder. Hence, it can only measure the change between one reading and another, and the number of readings per revolution limits the precision. The number of readings for the encoder used in the set-up consists of $n_{ticks} = 70997.33$. The total amount of ticks, $\boldsymbol{i}_{ticks}$, is accumulated and obtained as follows [6]

$$i_{\text{ticks}} = \text{round}\left(\frac{n_{\text{ticks}}}{2\pi}\theta\right) \tag{3.11}$$

where $\boldsymbol{\theta}$ corresponds to the motors position angle vector

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \tag{3.12}$$

The round function forces the angle to the lower or upper tick as follows

$$\text{round}(x) = \text{sgn}(x)\,||x| + 0.5| \tag{3.13}$$

A change in the encoder readings evaluated over a time period, Δt, is used as a numerical approximation to calculate the angular velocity of the motor [6]

$$\Delta i_{\text{ticks}} = \text{round}\left(\frac{n_{\text{ticks}}}{2\pi}\theta(t+\Delta t)\right) - \text{round}\left(\frac{n_{\text{ticks}}}{2\pi}\theta(t)\right) \tag{3.14}$$

However, due to the approximation, it is required to include a small error on the encoder readings, which leads to model the encoder as

$$\boldsymbol{z}_{\text{enc}} = \Delta \boldsymbol{i}_{\text{ticks}} = \frac{n_{\text{ticks}}}{2\pi}\Delta t\dot{\boldsymbol{\theta}} + \boldsymbol{v}_{\text{enc}} \tag{3.15}$$

The noise for the encoder is modelled as Gaussian white noise, $\boldsymbol{v}_{enc} \sim \mathcal{N}\left(\mathbf{0}_{3\times 1}, \boldsymbol{\Sigma}_{\text{enc}}\right)$. The covariance $\boldsymbol{\Sigma}_{\text{enc}}$ is determined as a diagonal matrix due to the lack of influence of the noise of the encoders between one another. The maximum error possible in the encoder is limited by the error in the approximation of the function round (3.13), which corresponds to one tick

$$\left| \left(\text{round}\ \left(\frac{n_{\text{ticks}}}{2\pi}\theta_i(t+\Delta t)\right) - \ \text{round}\ \left(\frac{n_{\text{ticks}}}{2\pi}\theta_i(t)\right)\right) - \frac{n_{\text{ticks}}}{2\pi}\Delta t\dot{\theta}_i \right| \leq 1 \tag{3.16}$$

A Gaussian distribution is built upon the premise that 99.7% of the samples belong to $\pm\ 3\sigma$. This boundary is set to one according to (3.16). Hence, it is possible to define the Gaussian distribution covariance diagonal from the standard deviation as follows

$$3\sigma_{\text{enc}} = 1 \rightarrow \sigma_{\text{enc}} = \frac{1}{3}$$
$$\boldsymbol{\Sigma}_{\text{enc}} = \sigma_{\text{enc}}^2 \boldsymbol{I}_3 = \frac{1}{9}\boldsymbol{I}_3$$

Finally, combining (3.15) with the inverse kinematics from (B.47) yields to

$$\hat{\boldsymbol{z}}_{\text{enc}} = \Delta \boldsymbol{i}_{\text{ticks}} = \frac{n_{\text{ticks}}}{2\pi}\Delta t\widetilde{\boldsymbol{W}}\,\Phi\left({}^{K}_{B}\hat{\boldsymbol{q}}\right)^{T}\left(\Gamma\left({}^{K}_{B}\hat{\boldsymbol{q}}\right)\ \frac{1}{r_k}\begin{bmatrix} 0 \\ -\hat{\dot{y}} \\ \hat{\dot{x}} \\ 0 \end{bmatrix} - 2\,{}^{K}_{B}\hat{\dot{\boldsymbol{q}}}\right) + \boldsymbol{v}_{\text{enc}} \tag{3.17}$$

3.2 Kalman Filter

A static parameter can be estimated, for example, by a linear recursive least squares filter. Similarly, the Luenberger observer aims to drive the state estimate to a real evolving value. However, it has to be applied to a deterministic system, in which no randomness is involved in the development of future states of the system. Rudolf E. Kálmán developed an estimator based on the least-squares method capable of estimating an evolving variable. It is also aimed to handle statistically the noisy signals that come from stochastic systems. For this reason, a Kalman Filter is also called linear–quadratic state estimator [24]. There are several types of Kalman filters, but all of them are settled on the fundamentals of the classic one, i.e. computing the gain optimally over time using measurements which contain noise. The estimates of the real values are computed using the model of the system, which yields deviations due to inherit inaccuracies [25].

The principle of the Kalman filter relies on the combination of the characteristics of the probability distribution of the estimated value, i.e. the mean value estimated and its covariance, combined with the distribution of the measured value, in order to provide a more accurate value with lower covariance. Hence, the more the sources of measurements used to estimate the value, the higher precision is obtained in the estimation process. Therefore, the objective of the Kalman filter is to minimize the mean squared error between the actual states and the estimated ones [26], as shown in Figure 3.2.

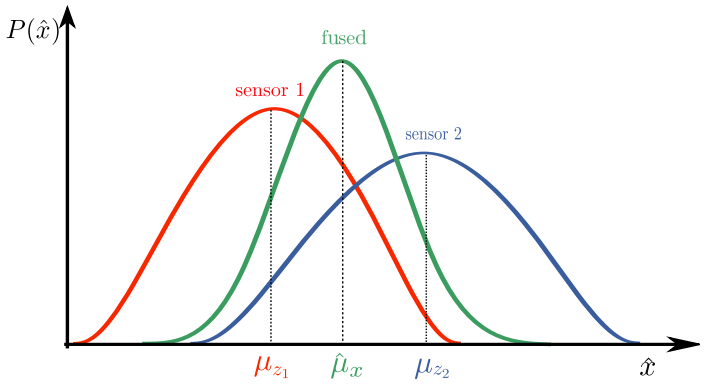


Figure 3.2: Combination of measurements distribution, inspired by [27].

The classic Kalman's estimator structure can be divided in two steps, prediction and update step. A schematic of the steps followed is shown in Figure 3.3.

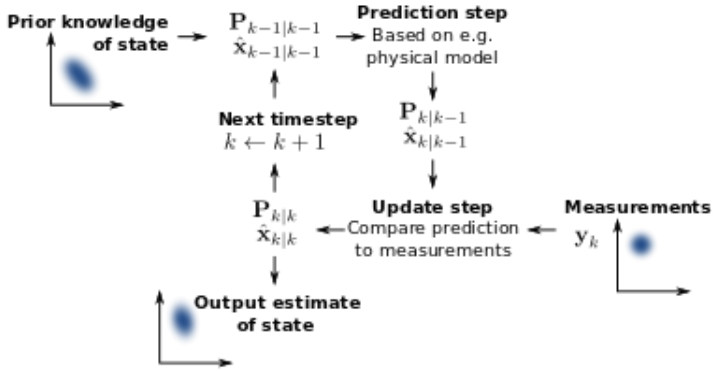


Figure 3.3: General process of a classic Kalman filter estimator [28].

In the first step, also called the prediction step, an estimation of the current states is computed as well as the statistical parameters accounting for the uncertainties. Afterwards, in the update step, the states are transformed into the estimate of the measurement variables. The difference between the values obtained through sensor measurements and the estimated ones are pondered using a dynamic weight and used to correct the state estimate obtained in the prediction step. The result of the update or correction step, known as "a posteriori" estimate, is composed of a mean and a covariance matrix. It is assumed as the correct value and driven to the controller as well as fed-back to the filter algorithm to use it in the next prediction step. Depending on the applications, precision aimed as well as the sensors reading rate, multiple prediction steps may be performed per update step.

The prediction step in a classical KF for a LTI system is defined by the following relationships

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- &= \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{P}_k^- &= \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q} \end{aligned} \tag{3.18}$$

to define the a priori state estimation $\hat{x}_k^-$ and the a priori covariance matrix P_k^-. For the first iteration of the algorithm, the previous state estimation and the previous covariance, $\hat{x}_{k-1}$ and P_{k-1} respectively, are initialized by a fixed value of the initial estimated parameters. The matrix $\boldsymbol{Q}$ corresponds to the noise covariance of the process model.

The second part of the algorithm, or update is given by

$$\begin{aligned} \boldsymbol{K}_k &= \boldsymbol{P}_k^- \boldsymbol{C}^T \left(\boldsymbol{C}\boldsymbol{P}_k^- \boldsymbol{C}^T + \boldsymbol{R}\right)^{-1} \\ \boldsymbol{P}_k &= \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}\right) \boldsymbol{P}_k^- \\ \hat{\boldsymbol{x}}_k &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \left(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k^-\right) \end{aligned} \tag{3.19}$$

where K_k is the optimal weighting factor. It is an m,n-matrix whose components depend on the relative size of the covariances of estimate and measurements. The Kalman gain performs two functions, it weights the update, i.e. it is in charge of deciding how much to trust the measured values and the estimated ones, and at the same time it transforms the residual from measurement space to state vector space [29]. The matrix $\boldsymbol{R}$ corresponds to the noise covariance of the measurements.

3.3 Kalman Filter types comparison and choice discussion

The Kalman Filter algorithm has many variants and are chosen depending on the system and application requirements. For a linear system, a linear Kalman Filter, known merely as Kalman Filter (KF) can be applied. In this case, to propagate the system states from the previous instant to the next time step, a linearised model at the equilibrium point is used.

For some non-linear systems, a linear estimator is not sufficient, in such a case a non-linear estimator is needed, such as an Extended Kalman Filter (EKF). The EKF is different from the ordinary Kalman Filter since it uses the non-linear model instead of the linearised one to predict the mean of the states. The covariance of the states is however propagated linearly, but with a more accurate approach, since the system is linearised in every iteration at the previous value of the state. Therefore, it is usually used instead of the Kalman Filter when the system is considered to be degraded using a global linearisation. However, for a proper performance the system is required to be locally linear, i.e. the system linearised at the previous state value used to calculate the next time state value still needs to fit the non-linear system result. For the ball-robot application, due to the fast sample rate of 200 Hz, the model can be considered locally linear in the vicinity of each sampling instant, since the variable values are not going to change considerably in that short sampling period. An EKF approach is applied in [6], showing accurate results at that sampling rate. However, EKF is not optimal if the system is highly non-linear; besides that, the system is linearized at each iteration, which requires finding the Jacobian matrices of the system. For complex functions finding the Jacobians is a difficult task by itself, i.e. analytically it might prompt to mistakes, and numerically is computationally costly. In case the functions are not differentiable, it is impossible to find the Jacobians [30].

The Error-State Extended Kalman Filter (ES-EKF) is a modification of the EKF to reduce the error in the estimation due to the linearization process. It splits the state value into the nominal part and the error in the estimation. The nominal part is propagated in the prediction step by the non-linear model, leading to having an

error state closer to a linear signal. ES-EKF uses the error in the estimation as the Kalman Filter states. The error in the estimation consists of the accumulation of the modelling errors and the process noise. Therefore, ES-EKF estimates the error state directly and uses it as a correction to the nominal state. This Kalman Filter approach operates close to the origin since the error states are assumed to be small and considered centered around a zero mean, and therefore far from possible parameter singularities and providing a guarantee that the linearisation validity holds at all times. Since the error state is small, the second-order products are negligible, which makes the computation of the Jacobians easy because they are constant or equal to available state magnitudes. Furthermore, the error dynamics are slow because all the large-signal dynamics have been integrated into the nominal-state; this leads to apply corrections at a lower rate than the predictions achieving same results [21].

In case of dealing with highly non-linear systems, there are some extensions of the EKF, such as the Unscented Kalman Filter (UKF) or more complicated versions, such as the Particle Kalman Filter, that handle the non-linearities achieving higher accuracy. However, the more accurate the approximation, the higher the computational cost.

In general terms, both EKF and UKF share the basics of the KF implementation structure; the difference lies in how to compute the mean and noise covariance matrix both in the prediction and correction steps. The EKF transforms the probability distributions through the linearised process and measurement model, which can lead to a covariance that does not accurately capture the uncertainty in the states, triggering the state estimates to vary from the actual value [29]. Conversely, the Unscented Kalman Filter relies on the Unscented Transform (UT) which allows transforming probability distributions through the non-linear functions of the models, maintaining the Gaussian distribution afterwards without distortion. Since it transforms the distributions by using the non-linear process and measurement model, it is a more realistic approach and fits better the real propagation of the noise through the model. The error source, in this case, comes from using the UT, i.e. from approximating a Gaussian distribution to a non-linear function. Nevertheless, this error is lower compared to the error caused by linearising the model, mostly in highly non-linear models. It is founded on the intuition that it is easier to approximate a probability distribution than an arbitrary non-linear function transformation [31]. Moreover, UKF removes the requirement to explicitly calculate Jacobians matrices of the system at every time step.

Once the Kalman Filter algorithm is chosen, there are different approaches on how to apply it to the system. The dynamic model of the system can be used to propagate the states and calculate the output; in such a case, it is said to be a

model-driven filter. It utilizes the actuated input to the system, i.e. the external torques, to predict the next time step state of the system. The other approach is called sensor-driven when it uses the kinematic relationships as the system model. The advantage of a sensor-driven filter is that it does not rely on the model and parameters of the system and therefore makes the estimator applicable to any similar system.

Quaternion estimation is widely researched in Kalman Filters with mainly two different approaches, treating the error in an additive way or multiplicative [32] [33]. Quaternions are not closed under addition and subtraction, therefore correcting the a posteriori estimation with a quaternion addition prevents the result from being a unit quaternion, and forced normalization is required at the end of each estimation iteration to keep the quaternion on the unit-sphere.

An overview of the main characteristics of the different Kalman filters is presented in Figure 3.4.

	EKF	ES-EKF	UKF
Linearization principle	Linearization (Full state)	Linearization (Error only)	Unscented transform
Linearization error in the covariance (P)	High	Medium	Non
Jacobians	Required	Required	Not required
Speed	Slightly faster	Slightly faster	Slightly slower
Implementation difficulties	Requires Jacobians	Requires Jacobians	Normal
Accuracy	Good	Better	Best

Figure 3.4: Comparison of the main characteristics of the Kalman Filters inspired by [21].

Once assessed the differences between the Kalman Filters state-of-the-art, the choice of the estimator approach applied to ball-robot is discussed next. First, the non-linear relationship in the ball-robot between estimated orientation and expected measurement prevent the usage of a classical Kalman Filter. This problem is addressed by choosing a UKF which allows the non-linear measurement and process models and is more accurate and less costly than the common EKF. Furthermore, when dealing with quaternions UKF is preferred compared to an EKF due to the flexibility it offers in dealing with the random variables, v and w. Some extensions

of the original UKF are necessary in order to deal with the inherent properties of unit quaternions, which results in a UKF with similarities to an error-state approach with the multiplicative treatment of the orientation propagation. The UKF approach chosen is sensor-driven due to its simple model and commonly usage for this application. Furthermore, the sensor model approach is proved as a sufficient model for orientation state estimation in [29].

A two connected UKF structure is developed to estimate the ball-robot states. A Quaternion UKF (QUKF) in charge of orientation and angular velocity estimation, and a Velocity UKF (VUKF) dealing with the linear velocity estimation, which requires the quaternion orientation estimation from the QUKF to update its states. This two filter structure approach is chosen because for orientation estimation purposes (QUKF) the accelerometer is usually assumed to measure mainly the gravity vector since the linear accelerations of the ball-robot are considered negligible compared to the gravity one. However, when estimating velocity (VUKF), its rate of change is calculated using the linear acceleration estimation, which is corrected by the linear acceleration read by the accelerometer and hence it can not be disregarded in this case.

The QUKF and VUKF are executed at the same speed rate as the sensors reading speed which coincides also with the sampling time, which is 200 Hz. It is decided to compute the same number of prediction and update steps, since it might provide enough convergence speed. Hence, the filter algorithm has an iteration periodicity of $\Delta t = 1/200 = 5\ ms$.

3.4 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is the one studied in the current project. Hence its characteristics are covered in the current section.

The KF operates on a probability distribution in the n-dimensional state vector space. This distribution is characterized by its first two statistical moments, mean and covariance. The UKF uses a deterministic sampling technique known as the Unscented Transform (UT) to pick a minimal set of sample points from the input distribution. These points are called sigma points since they are placed at a specific rate of standard deviations away from the mean [30]. The process and measurement models transform this distribution by transforming each sigma point, leading to a new set of sigma points belonging to the output distribution, from which a new mean and covariance estimate is then formed to parameterize the transformed distributions. The resulting filter depends on how the transformed statistics of the

UT are calculated and which set of sigma points are used.

The goal of the filter is to compute an estimate of the system state vector x combining the model prediction and the sensor measurements. The distance at which each sigma point is placed away from the mean of each state is related to the square root of the noise covariance of that state in each of the dimension directions. Since the noise covariance matrices are by definition symmetric positive definite matrices, the Cholesky Decomposition is used to compute their square root. The most basic example is in the case of a one-dimensional input probability distribution, in which the Cholesky Decomposition is the square root of the variance, which is the standard deviation [29].

$$\mathbf{L}\mathbf{L}^T = \hat{\mathbf{P}}_{k-1} \tag{3.20}$$

where $\boldsymbol{L}$ is a lower triangular matrix. At the beginning of every UKF recursion the previous estimate of the state vector $\boldsymbol{\mu}_{x_{k-1}}$, or the mean at time step $k-1$, and its covariance $\hat{\mathbf{P}}_{k-1}$, are known. The n,n-matrix $\hat{\mathbf{P}}_{k-1}$ is transformed into the set of sigma points $\mathcal{X}_{k-1}$ which has the same mean and covariance $\hat{\mathbf{P}}_{k-1}$.

$$\mathcal{X}_{k-1}^{(i)} \quad = \boldsymbol{\mu}_{x_{k-1}} \pm \propto \mathrm{col}_i \mathbf{L}_{k-1} \tag{3.21}$$

where $i \in \{0, D\}$ corresponds to the index of the sigma point and D to the total number of sigma points. The corresponding column of the L matrix previously calculated is refers to as col_i $\mathbf{L}_{k-1}$. The proportional parameter will be defined according to the transformed statistics of the UT and the number of sigma points chosen.

The sigma points are propagated ahead to the next iteration k through the nonlinear process model of the system. Note that no process noise vector w is being considered in the process model function since it will be added through its covariance matrix $\boldsymbol{Q}$ to the sigma points when computing the covariance matrix $\check{\mathbf{P}}_k$. The covariance matrix $\boldsymbol{Q}$ of the random variable w is a measure of the rate at which the uncertainty of the system state estimates increase with time.

$$\mathcal{X}_k^{(i)} = f\left(\mathcal{X}_{k-1}^{(i)}, \mathbf{u}_{k-1}, \mathbf{0}\right) \tag{3.22}$$

The prediction step concludes computing the "a priori" estimation of the mean and covariance of the state vector at time k, weighted by a proportional parameter that is specific for each UT characteristics chosen and that approximates a Gaussian distribution using the properties of the sigma points deviations due to the model propagation.

$$\boldsymbol{\mu}_{x_k} = \check{\mathbf{x}}_k \propto \sum \mathcal{X}_{k-1}^{(i)} \tag{3.23}$$

$$\check{\mathbf{P}}_k \propto \sum \left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)\left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)^T + \mathbf{Q} \tag{3.24}$$

The update step requires to obtain a mean and covariance of the measurement estimation vector. The previous sigma points are transformed from the n-dimensional state space to the m-dimensional measurement space through the measurement model function. The measurement noise vector v is neglected in the model function since its covariance $\boldsymbol{R}$ is added afterwards when computing the covariance matrix of the estimated measurement of $\hat{\mathbf{P}}_{zz}$.

$$\mathcal{Z}_k^{(i)} = \mathbf{h}_k\left(\mathcal{X}_k^{(i)}, \mathbf{0}\right) \tag{3.25}$$

The measurement mean and covariance of the measurement estimates are computed as previously defined

$$\hat{\mathbf{z}}_k = \boldsymbol{\mu}_z \propto \sum \mathcal{Z}_k^{(i)} \tag{3.26}$$

$$\hat{\mathbf{P}}_{zz} = \sum_{i=0}^{2N} \alpha^{(i)} \left(\mathcal{Z}_k^{(i)} - \boldsymbol{\mu}_z\right)\left(\mathcal{Z}_k^{(i)} - \boldsymbol{\mu}_z\right)^T + \mathbf{R} \tag{3.27}$$

Finally the Kalman gain that optimally weights the update is computed. To do so, the cross-covariance of the predicted state and the predicted measurements needs to be computed. This cross-covariance relates how the measurement space is correlated with the state space

$$\mathbf{P}_{xz} \propto \sum \left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)\left(\mathcal{Z}_k^{(i)} - \boldsymbol{\mu}_z\right)^T \tag{3.28}$$

leading to a Kalman gain that drives the residual between the measurement and the predicted measurement to an update of the "a priori" states value and transform from the m-dimensional measurement space to the n-dimensional state space. Therefore K is a n,m-matrix computed as follows

$$\mathbf{K}_k = \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1} \tag{3.29}$$

The "a posteriori" state estimate corresponds to the "a priori" mean updated with the residual. The update of the "a posteriori" covariance matrix is computed following the same principle.

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right) \tag{3.30}$$

$$\hat{\mathbf{P}}_{zz_k} = \check{\mathbf{P}}_{zz_k} - \mathbf{K}_k \mathbf{P}_{zz} \mathbf{K}_k^T \tag{3.31}$$

Figure 3.5 shows the transformation of the sigma points through the non-linear model and the UT approximation to a Gaussian distribution. In a classical KF, the measurement estimate is simply the states mean transformed through the model ($h(\mu_x)$), while UKF uses the mean value of the projected distribution instead (μ_z), which generally does not correspond as shown in the figure. The covariance is also calculated more realistically in the UKF using the UT instead of using the linear model, which constitutes a relevant improvement of the UKF compared to the EKF if the model functions are highly non-linear.

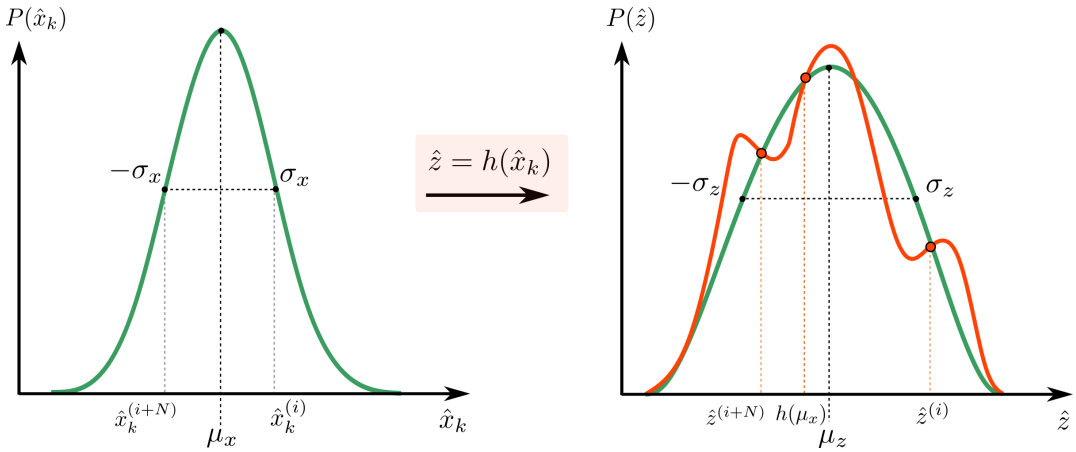


Figure 3.5: The sigma points are chosen equally separated from the mean according to the Gaussian properties of the variable. They are transformed through the non-linear model (orange dots) leading to a non-linear distribution (orange line). The sigma points are used to calculate through the UT the mean and covariance to properly approximate a Gaussian distribution to the non-linear one. Inspired by [21].

Chapter 4

Quaternion Unscented Kalman Filter (QUKF)

A modified Unscented Kalman Filter specifically designed to work with quaternions is implemented for orientation estimation. An schematic of the workflow is shown in Figure 4.1 in order to clarify the steps followed.

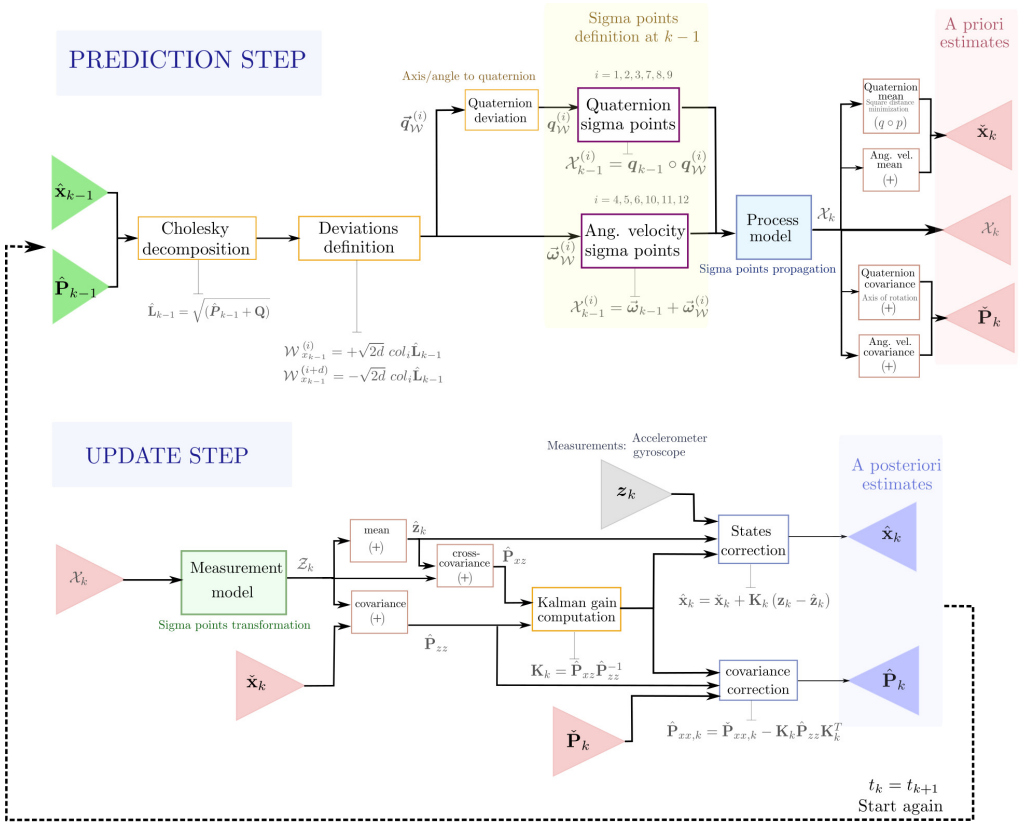


Figure 4.1: Schematic of the modified quaternion unscented Kalman filter estimator.

The modifications were needed since the UKF algorithm contains a large number of additions and subtractions operations. Those operations can not be implemented freely with quaternions due to the accumulated error caused by exiting the surface of the unit-sphere. The proposed forced unit norm after the addition operations that worked properly for an EKF in [6], did not yield to good results in the UKF algorithm due to the aforementioned reasons.

The QUKF estimates the orientation of the ball-robot and the rotational velocity, i.e. the $\{B\}$ reference frame orientation with respect to the $\{K\}$ reference frame represented by a unit quaternion and its angular velocity associated. As explained in Section 3.1, the gyroscope bias can not be estimated since it is unobservable without an additional sensor available. The filter states to be estimated form a 7-dimension vector as follows

$$\hat{\boldsymbol{x}} = \begin{bmatrix} {}^{K}_{B}\hat{\boldsymbol{q}} \\ {}^{B}\hat{\boldsymbol{\omega}}_{B|K} \end{bmatrix} \tag{4.1}$$

for the algorithm explanation, the super and sub-scripts indexes of the states will be removed for clarity purposes. As explained in Section C.4, the unit quaternion constraint (C.45) on the norm deprives the quaternion of one degree of freedom. Therefore, the first four components of the filter space vector are dependent, and any spatial rotation could be represented by their three degrees of freedom (in fact, there are two quaternions that represent each rotation E.1.2). The dependence between filter states causes a conflict with the concept underlying a KF and the way how noise is treated as additive [29]. Furthermore, unit quaternions are not closed under addition and subtraction, and therefore many UKF operations need to be adapted to unit quaternions arithmetic. For these reasons, a quaternion orientation estimation filter requires special handling of the quaternion associated states.

The available measurements for the update step to correct the states estimate comes from the IMU, accelerometer and gyroscope.

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_{acc} \\ \boldsymbol{z}_{gyro} \end{bmatrix} = \begin{bmatrix} {}^{B}\hat{\vec{g}}_{B|K} \\ {}^{B}\hat{\boldsymbol{\omega}}_{B|K} \end{bmatrix} \tag{4.2}$$

The previous iteration state vector and covariance are known for each iteration of the algorithm. For the first iteration, they are initialized with a realistic initial position of the ball-robot, which consists in starting in a pre-calibrated vertical position and standstill. This position corresponds to a unit quaternion orientation with zero angular velocity.

$$\hat{\boldsymbol{x}}^{0} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]^{T} \tag{4.3}$$

Since the state vector has six degrees of freedom ($d = 6$), the covariance of the process noise vector w uses for the orientation components an axis/angle representation instead of a quaternion form, and therefore it is a 6,6-matrix.

$$\boldsymbol{P}^0 = \text{diag}\left(1 \cdot 10^{-7} \cdot \mathbf{I}_2, \quad 1 \cdot 10^{-9}, \quad 1 \cdot 10^{-7} \cdot \mathbf{I}_2, \quad 1 \cdot 10^{-9}\right) \tag{4.4}$$

4.1 Prediction step

Sigma points definition

In the beginning of every QUKF recursion the previous d,d-matrix $\hat{\mathbf{P}}_{k-1}$ is known and transformed into the set of sigma points through the Cholesky decomposition. In this UKF it is desired to apply the process noise before the process model in order to include it in the sigma points creation at $k-1$. The process noise has zero mean and a d,d-matrix covariance $\mathbf{Q}$ that makes it more suitable to be implemented when defining the sigma points deviations from the mean [29]. Each sigma point is a seven-dimensional vector, while the process noise is defined as a six dimensional vector, therefore a simple process noise addition would not be possible because of size mismatch. This yields to the following d,d-matrix $\boldsymbol{L}$

$$\hat{\mathbf{L}}_{k-1} = \sqrt{(\hat{\mathbf{P}}_{k-1} + \mathbf{Q})} \tag{4.5}$$

where $\mathbf{Q}$ is defined in (4.37). The matrix $\boldsymbol{L}$ is transformed to create the $2d = 12$ sigma points deviations $\mathcal{W}^{\,i}_{x_{k-1}}$ from the mean. This deviations have a covariance $\hat{\mathbf{P}}_{k-1} + \mathbf{Q}$ scaled by a factor following [29]. These UT characteristics are optimal for a quaternion orientation UKF.

$$\begin{aligned} \mathcal{W}^{\,(i)}_{x_{k-1}} &= +\sqrt{2d}\ \text{col}_i \hat{\mathbf{L}}_{k-1} \quad i = 1 \ldots d \\ \mathcal{W}^{\,(i+d)}_{x_{k-1}} &= -\sqrt{2d}\ \text{col}_i \hat{\mathbf{L}}_{k-1} \quad i = 1 \ldots d \end{aligned} \tag{4.6}$$

The deviations form a 6,12-matrix in which each column correspond to the deviation of a specific sigma point and the rows to the variable corresponding to a degree of freedom, therefore

$$\begin{bmatrix} \vec{\boldsymbol{q}}^{\,(i)}_{\mathcal{W}} \\ \vec{\boldsymbol{\omega}}^{\,(i)}_{\mathcal{W}} \end{bmatrix} = \mathcal{W}^{\,(i)}_{x_{k-1}} \tag{4.7}$$

the first three components correspond to the orientation. Since the sigma points need to be seven-dimension vectors to match with the state vector, the orientation components are then transformed to a quaternion form applying the axis/angle properties,

$$\alpha^{\,(i)}_{q_{\mathcal{W}}} = |\vec{\boldsymbol{q}}^{\,(i)}_{\mathcal{W}}| \tag{4.8}$$

$$\vec{e}\,_{q_{\mathcal{W}}}^{(i)} = \frac{\vec{q}\,_{\mathcal{W}}^{(i)}}{|\vec{q}\,_{\mathcal{W}}^{(i)}|} \tag{4.9}$$

where the angle is the magnitude of rotation needed for the deviation of the sigma point i and it is expressed in radians. The deviation axis of rotation is the previous vector in its unitary form. The quaternion orientation of the deviation of the sigma point i can be composed by applying (E.23)

$$q_{\mathcal{W}}^{(i)} = \left[\cos\left(\frac{\alpha_{q_{\mathcal{W}}}^{(i)}}{2}\right), \vec{e}\,_{q_{\mathcal{W}}}^{(i)} \sin\left(\frac{\alpha_{q_{\mathcal{W}}}^{(i)}}{2}\right)\right]^T \tag{4.10}$$

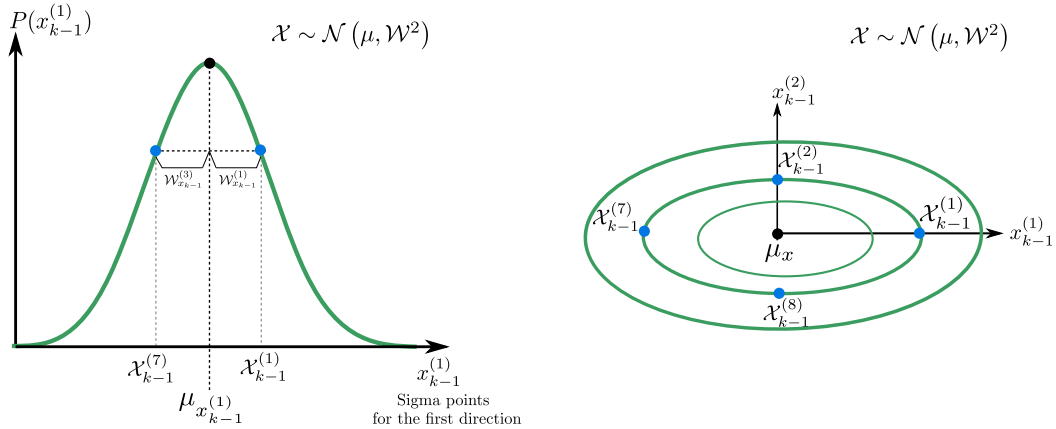
The set of sigma points $\mathcal{X}_{k-1}$ is a 7,12-matrix created shifting the deviations by the mean value. The deviations will be symmetrically distributed to both sides of each previous state value, which corresponds to the state mean given by

$$\hat{\mathbf{x}}_{k-1} = \boldsymbol{\mu}_{x_{k-1}} = \begin{bmatrix} q_{k-1} \\ \vec{\omega}_{k-1} \end{bmatrix} \tag{4.11}$$

Since unit quaternions are not closed under addition, the mean is rotated by the deviation quaternion $q_{\mathcal{W}}^{(i)}$ to create each sigma point i, and this has to be done by quaternion multiplication (E.9) to keep it in the unit-sphere. For the angular velocity each deviation vector $\vec{\omega}_{\mathcal{W}}^{(i)}$ is added to the mean.

$$\mathcal{X}_{k-1}^{(i)} = \hat{\mathbf{x}}_{k-1} + \mathcal{W}_{x_{k-1}}^{(i)} \quad \rightarrow \quad \mathcal{X}_{k-1}^{(i)} = \begin{bmatrix} q_{k-1} \circ q_{\mathcal{W}}^{(i)} \\ \vec{\omega}_{k-1} + \vec{\omega}_{\mathcal{W}}^{(i)} \end{bmatrix} \tag{4.12}$$

The sigma points are seven dimensional vectors, however, to clarify a simplified schematic is depicted in Figure 4.2. It shows the sigma points creation process using the deviations from the mean, for the first and second state space directions, hence creating a 1D and 2D planes respectively.



a) Sigma points probability distribution plane, first direction $\left(x_{k-1}^{(1)}\right)$

b) Sigma points probability distribution plane, first and second direction $\left(x_{k-1}^{(1)}, x_{k-1}^{(2)}\right)$

Figure 4.2: 2D and 3D probability distribution planes for one and two state space directions respectively, represented in a 1D and 2D drawings to show the distribution of the sigma points from the mean according to their deviation.

Process model

These sigma points are propagated ahead in time through the non-linear process model to get a new set of sigma points for the predicted state at time k. The process noise vector w is not considered in the model since the process noise has already been applied at the sigma points definition.

$$\mathcal{X}_k^{(i)} = f\left(\mathcal{X}_{k-1}^{(i)}, \mathbf{0}\right) \quad i = 1 \dots 2d \tag{4.13}$$

The process model of the angular velocity is defined as the simplest one which considers it constant in time. It is proven that this model is enough for these types of applications [29], despite a more complex model like the dynamic one can also be used.

$$\omega_k = \omega_{k-1} \tag{4.14}$$

Furthermore, the process noise attached to this angular velocity model is defined in (4.39) and considered as a tunable parameter for the process model. The bigger the noise covariance of the angular velocity model, the less trust in stating that the angular velocity is constant, leading to trust more the measurement from the sensors than the process model. If the covariance is low, it means that a constant angular velocity is a good estimation to our application. As a consequence, this noise parameter has a filtering effect in the angular velocity sensor readings, which is higher if more trust is put in the model.

The process model function for the orientation is derived from the angular velocity

one. The angle rotated is computed as

$$\alpha_\Delta = |\vec{\omega}_{k-1}| \cdot \Delta t \tag{4.15}$$

and the unitary rotation axis is

$$\vec{e}_\Delta = \frac{\vec{\omega}_{k-1}}{|\vec{\omega}_{k-1}|} \tag{4.16}$$

The corresponding process model for the orientation is a quaternion given by

$$q_\Delta = \left[\cos\left(\frac{\alpha_\Delta}{2}\right), \vec{e}_\Delta \sin\left(\frac{\alpha_\Delta}{2}\right)\right]^T \tag{4.17}$$

which results in the following process model from (4.13) that is used to propagate the set of sigma points

$$\mathcal{X}_k^{(i)} = \begin{bmatrix} q_{k-1}^{(i)} \circ q_\Delta \\ \vec{\omega}_{k-1}^{(i)} \end{bmatrix} \tag{4.18}$$

Computation of the "a priori" mean

The predicted mean $\check{\mathbf{x}}_k$ and covariance $\check{\mathbf{P}}_k$ for the state at time k, are the "a priori" state and covariance estimation. The mean is computed applying the barycentric mean.

$$\check{\mathbf{x}}_k = \mu_{x_k} = \frac{1}{2d} \sum_{i=1}^{2d} \mathcal{X}_k^{(i)} \tag{4.19}$$

However, orientation is periodic. Therefore (4.19) can yield to invalid results, e.g the mean of $\{-176, 180\}$ gives $2°$ when it is $-178°$ [29]. Furthermore, the quaternion mean computation is not trivial since unit quaternions are not closed under addition. Therefore, for the first four components of the sigma points that corresponds to the quaternion, the mean is computed using quaternion multiplication instead of a sum, this is applied through a square distance minimization algorithm. It is establish on the idea that the mean is the least total distance to the addends considered. The quaternion term of each sigma point is first transformed into its rotation matrix [34]. The mean quaternion is the result of minimizing the squared Frobenius norm of the difference between a sweep quaternion and the quaternions of the sigma points. The sweep matrix consists of quaternions that represent rotations from 0 to 180 degrees about each axis and is also transformed into a rotation matrix.

$$\bar{q} = \arg\left(\min_{q \in S^3} \sum_{i=1}^{2d} \left\| R(q_{sweep}) - R\left(\mathcal{X}_q^{(i)}\right) \right\|_F^2\right) \tag{4.20}$$

In an iterative process the quaternion of the sweep matrix that minimizes the distance between itself and each sigma quaternion is stored. Once all the sweep

quaternions have been tried, the one that minimizes the sum of the distances to each sigma point quaternion is the mean quaternion of the quaternions sigma points [34]. The algorithm is limited to compute the mean of quaternions that are close to each other [35]. The Frobenius norm consists in the matrix norm of a matrix, i.e. the square root of the sum of the absolute squares of its elements.

Computation of the "a priori" covariance

The covariance $\check{\mathbf{P}}_{xx_k}$ is usually calculated by the square of the subtraction to the mean of each sigma point value and then divided by the total number of sigma points. This subtraction term is similar to the deviations $\mathcal{W}_{x_{k-1}}$ computed in (4.12) to define the sigma points but in this case for time step k.

$$\check{\mathbf{P}}_{xx_k} = \frac{1}{2d}\sum_{i=1}^{2d}\left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)\left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)^T = \frac{1}{2d}\mathcal{W}_{x_k}\,\mathcal{W}_{x_k}^T \tag{4.21}$$

This is true for the angular velocity terms $\mathcal{W}_{\omega,x_k}$, but in order to avoid unit quaternions subtraction the orientation terms, i.e. the first 4,12-matrix $\mathcal{W}_{q,x_k}$, of the 7,12-matrix $\mathcal{W}_{x_k}$, are computed applying instead quaternion multiplication. The conjugated quaternion (E.15) of the mean is required to represent a orientation subtraction in the quaternion multiplication.

$$\mathcal{W}_{q,x_k}^{(i)} = \mathcal{X}_{q,k}^{(i)} \circ \boldsymbol{\mu}_{q,x_k}^T \tag{4.22}$$

Furthermore, in order to match the dimensions of the covariance matrix, which is a 6,6-matrix due to the noise terms definition, the quaternion deviation must be converted to its axis/angle representation to be three component vector. Following (E.23) the angle is extracted from the scalar part of the quaternion $\mathcal{W}_{q,x_k}^{(1,i)}$

$$\boldsymbol{\alpha}_{\mathcal{W}_{q_k}}^{(i)} = 2\cdot\arccos(\mathcal{W}_{q,x_k}^{(1,i)}) \tag{4.23}$$

and the unitary rotation axis from the complex terms is given by

$$\vec{\boldsymbol{e}}_{\mathcal{W}_{q_k}}^{(i)} = \frac{\mathcal{W}_{q,x_k}^{(2:4,i)}}{\sin\left(\boldsymbol{\alpha}_{\mathcal{W}_{q_k}}^{(i)}/2\right)} \tag{4.24}$$

by multiplying both terms the rotation vector is defined

$$\mathcal{W}_{q,x_k}^{(i)} = \vec{\boldsymbol{e}}_{\mathcal{W}_{q_k}}^{(i)} \cdot \boldsymbol{\alpha}_{\mathcal{W}_{q_k}}^{(i)} \tag{4.25}$$

This way the covariance 6,6-matrix can be computed applying (4.21) with the modifications introduced for the orientation terms

$$\check{\mathbf{P}}_{xx_k} = \frac{1}{2d}\cdot\begin{bmatrix}\mathcal{W}_{q,x_k}\\ \mathcal{W}_{\omega,x_k}\end{bmatrix}\cdot\begin{bmatrix}\mathcal{W}_{q,x_k}\\ \mathcal{W}_{\omega,x_k}\end{bmatrix}^T \tag{4.26}$$

4.2 Correction step

The predicted state sigma points $\mathcal{X}_k^{(i)}$ which are seven-dimensional and were obtained in (5.11) are transformed through the non-linear measurement model to get a new set of sigma points of the predicted measurement, which are six-dimensional.

$$\mathcal{Z}^{(i)} = h\left(\mathcal{X}_k^{(i)}, \mathbf{0}\right) \quad i = 0 \dots 2d \tag{4.27}$$

Two measurement models are required to compute the predicted measurements corresponding to the sensors reading, one for the accelerometer and another for the variable read by the gyroscope. The measurement vector is defined as

$$\hat{\boldsymbol{z}} = \begin{bmatrix} \hat{\boldsymbol{z}}_{acc} \\ \hat{\boldsymbol{z}}_{gyro} \end{bmatrix} = \begin{bmatrix} {}^{B}\hat{\vec{\boldsymbol{g}}}_{B|K} \\ {}^{B}\hat{\boldsymbol{\omega}}_{B|K} \end{bmatrix} \tag{4.28}$$

Measurement model

The accelerometer is assumed in the QUKF to measure the gravity vector in the $\{B\}$ reference frame. Therefore its measurement model transforms the known gravity vector in the inertial reference frame to the $\{B\}$ reference frame by rotating it using the quaternion estimation obtained through the process model.

$$\hat{\boldsymbol{z}}_{\text{acc}} = {}^{B}\hat{\vec{\boldsymbol{g}}}_{B|K} = -\vee\left({}^{K}_{B}\hat{\boldsymbol{q}}^{T} \circ \left(\wedge \frac{{}^{\text{I}}\vec{\boldsymbol{g}}}{\|{}^{\text{I}}\vec{\boldsymbol{g}}\|}\right) \circ {}^{K}_{B}\hat{\boldsymbol{q}}\right) = \left(\vee \boldsymbol{\Phi}\left({}^{K}_{B}\hat{\boldsymbol{q}}\right)^{T} \boldsymbol{\Gamma}\left({}^{K}_{B}\hat{\boldsymbol{q}}\right) \wedge\right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.29}$$

Since the gyroscope measures the angular velocity in the $\{B\}$ reference frame and this variable was already component of the state vector its measurement model is simple

$$\hat{\boldsymbol{z}}_{\text{gyro}} = {}^{B}\hat{\boldsymbol{\omega}}_{B|K} \tag{4.30}$$

The accelerometer estimation has a limitation inherit from its model. A yaw rotation reference in the ball-robot does not vary the projection of the gravity vector in the $\{B\}$ reference frame, since the gravity vector has only z-component when the ball-robot is vertical, and the $\{B\}$ reference frame z-axis coincides with the inertial reference frame z-axis. Therefore, when there is a yaw rotation in the reference it produces a drift in the heading estimation that increases with time. A magnetometer or Li-DAR sensor reading can correct the estimate of the heading and assure that the drift stays into the acceptable threshold for the application. In that case, the bias would be observable and could be estimated by the QUKF. It would be added in (4.30) to improve the angular velocity estimate compensating for the drift.

Computation of the measurement mean and covariance of the estimate

In this case the measurement vector (5.2) is a six-dimensional and does not have quaternions, therefore the mean of the sigma points can be easily computed as

$$\hat{\mathbf{z}}_k = \boldsymbol{\mu}_z = \frac{1}{2d}\sum_{i=1}^{2d} \mathcal{Z}^{(i)} \tag{4.31}$$

Same applies for the covariance computation that can be computed with the subtraction formula without modifications. The measurement noise covariance $\mathbf{R}$ is defined in (4.40) and added to the sigma points covariance to compute the final covariance of the predicted measurements, providing the noise is additive.

$$\hat{\mathbf{P}}_{zz} = \frac{1}{2d}\sum_{i=1}^{2d} \left(\mathcal{Z}_k^{(i)} - \boldsymbol{\mu}_z\right)\left(\mathcal{Z}_k^{(i)} - \boldsymbol{\mu}_z\right)^T + \mathbf{R} \tag{4.32}$$

Computation of cross-covariance and Kalman gain

The cross-covariance relates how the measurements are correlated with the states and transform from the m-dimensional measurement space to the n-dimensional process space, therefore it is a 7,6-matrix,

$$\hat{\mathbf{P}}_{xz} = \frac{1}{2d}\sum_{i=1}^{2d} \left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)\left(\hat{\mathcal{Z}}^{(i)} - \boldsymbol{\mu}_z\right)^T \tag{4.33}$$

The Kalman gain is a 7,6-matrix and is given by

$$\mathbf{K}_k = \hat{\mathbf{P}}_{xz}\hat{\mathbf{P}}_{zz}^{-1} \tag{4.34}$$

Computation of the "a posteriori" mean and covariance

The Kalman gain is used to optimally correct the mean and covariance of the predicted "a priori" state estimate with the residual between the sensors measurement and the measurement estimate from the model, leading to the "a posteriori" state estimate and covariance.

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right) \tag{4.35}$$

$$\hat{\mathbf{P}}_{xx,k} = \check{\mathbf{P}}_{xx,k} - \mathbf{K}_k\hat{\mathbf{P}}_{zz}\mathbf{K}_k^T \tag{4.36}$$

These two variables are the initial data for the next iteration of the QUKF algorithm. Moreover, the part of the mean and covariance related to the quaternion estimation is driven to the VUKF since it needs the estimation of the quaternion to compute

the linear velocity estimate. The estimated quaternion and angular velocity of the QUKF transformed to quaternion derivative are also required to estimate the position of the ball-robot through integration and forward kinematics. Therefore, an accurate and fast convergence is required to the QUKF in order not to propagate its error to the other state estimates.

4.3 QUKF covariance matrices

The process noise covariance matrix Q is a 6,6-matrix and is defined for the QUKF following the explanation of the noise sources in Section 3.1 [6]

$$Q = \begin{bmatrix} \mathbf{\Sigma}_q & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{\Sigma}_\omega \end{bmatrix} \tag{4.37}$$

For the process model of the QUKF the numerical integration error that comes from the discretization is normally that small that can be simply set to be zero. However, to be able to apply the Cholesky decomposition in (4.5) to $\hat{\mathbf{P}}_{k-1} + \mathbf{Q}$ the sum of these matrices must be a positive definite matrix. Hence, since $\hat{\mathbf{P}}_{k-1}$ is computed automatically and can not be modified, $\mathbf{\Sigma}_q$ is set to a small covariance but different from zero to prevent having a 3,3-matrix of zeros that would make impossible to compute the Cholesky decomposition.

$$\mathbf{\Sigma}_q = \sigma_q \cdot \mathbf{I}_3 = 1 \cdot 10^{-20} \cdot \mathbf{I}_3 \tag{4.38}$$

The angular velocity process covariance is defined as a diagonal matrix with a smoothing factor that is tunable for acquiring the desirable filtering [6]. The angular velocity process model was defined as to propagate the angular velocity as a constant. Therefore, the noise with covariance $\mathbf{\Sigma}_\omega$ which is tunable, indicates how reliable is to state that the angular velocity is constant. The higher the noise, the less trust in the model of constant angular velocity, and therefore less filtering effect in the angular velocity estimation from the sensors reading.

$$\mathbf{\Sigma}_\omega = \sigma_\omega \cdot \mathbf{I}_3 = 1 \cdot 10^{-5} \cdot \mathbf{I}_3 \tag{4.39}$$

The measurement noise covariance is a 6,6-matrix that is defined as follows

$$R = \begin{bmatrix} \alpha_{\text{trust}}\, \mathbf{\Sigma}_{\text{acc}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{\Sigma}_{\text{gyro}} \end{bmatrix} \tag{4.40}$$

where $\mathbf{\Sigma}_{\text{acc}}$ and $\mathbf{\Sigma}_{\text{gyro}}$ where already extracted from experimental data collected by the MPU-9250 IMU with internal LPF as shown in (3.4). and (3.8) The ratio α_{trust} is set to 2 and it is a tunable parameter, the higher value the more trust is put into the gyroscope measurements by increasing the accelerometer covariance.

Chapter 5

Velocity Unscented Kalman Filter (VUKF)

The VUKF aims to estimate the linear velocity of the ball-robot in the $\{K\}$ reference frame with respect to the inertial reference frame. However, the system states to be estimated form the following 7-dimensional vector

$$ {}^{I}\hat{\boldsymbol{x}} = \begin{bmatrix} {}^{I}\hat{x}_{K|I} \\ {}^{I}\hat{y}_{K|I} \\ {}^{I}\hat{\dot{x}}_{K|I} \\ {}^{I}\hat{\dot{y}}_{K|I} \\ {}^{I}\hat{b}_{gyro} \end{bmatrix} \tag{5.1} $$

since also the linear acceleration and the gyro bias are needed to perform the linear velocity estimation process model.

The available measurements for the update step to correct the states estimate comes from the encoder placed in each motor and from the accelerometer in the IMU. The state vector and measurement vector share the linear acceleration terms, but the measurements ones are defined in the $\{B\}$ reference frame.

$$ \boldsymbol{z} = \begin{bmatrix} z_{enc} \\ z_{acc} \end{bmatrix} = \begin{bmatrix} {}^{M}\phi_0 \\ {}^{M}\phi_1 \\ {}^{M}\phi_2 \\ {}^{B}\ddot{x}_{B|K} \\ {}^{B}\ddot{y}_{B|K} \\ {}^{B}\ddot{z}_{B|K} \end{bmatrix} \tag{5.2} $$

The previous iteration state vector and covariance are known for each iteration of the algorithm. For the first iteration, the state vector is initialized with a standstill

position and pre-calibrated with zero bias value,

$$\hat{\boldsymbol{x}}^{\,0} = \left[\begin{array}{cc|cc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]^T \tag{5.3}$$

The covariance matrix is a 7,7-matrix initialized at

$$\boldsymbol{P}^{\,0} = \text{diag}\left(1 \cdot 10^{-1} \cdot \mathbf{I}_2, \quad 1 \cdot 10^{-2} \cdot \mathbf{I}_2, \quad 1 \cdot 10^{-9} \cdot \mathbf{I}_3\right) \tag{5.4}$$

5.1 Prediction step

Sigma points definition

In the beginning of every QUKF recursion the previous N,N-matrix $\hat{\mathbf{P}}_{k-1}$ is known and transformed into the set of sigma points through the Cholesky decomposition.

$$\hat{\mathbf{L}}_{k-1}\,\hat{\mathbf{L}}_{k-1}^{T} = \hat{\mathbf{P}}_{k-1} \tag{5.5}$$

In this case, the number of sigma points required is chosen to be $2N+1$ for an N-dimensional probability distribution, since there are N states to estimate, where $N = 7$ since it is the state vector dimension.

The matrix $\boldsymbol{L}$ is transformed to create sigma points deviations $\mathcal{W}^{\,i}_{x_{k-1}}$ from the mean. The parameter κ is a tuning parameter and is chosen to be $\kappa = 3 - N$, since it is shown in [21] as a good choice in case of a Gaussian probability distribution. The higher κ the faster the convergence. the These UT characteristics are commonly assumed as optimal for a general state estimation UKF.

$$\begin{array}{lll} \mathcal{W}^{\,(i)}_{x_{k-1}} & = +\sqrt{N+\kappa}\ \text{col}_i\hat{\mathbf{L}}_{k-1} & = +\sqrt{3}\ \text{col}_i\hat{\mathbf{L}}_{k-1} \quad i = 1\ldots N \\ \mathcal{W}^{\,(i+N)}_{x_{k-1}} & = -\sqrt{N+\kappa}\ \text{col}_i\hat{\mathbf{L}}_{k-1} & = -\sqrt{3}\ \text{col}_i\hat{\mathbf{L}}_{k-1} \quad i = 1\ldots N \end{array} \tag{5.6}$$

The deviations form a 7,14-matrix in which each column correspond to the deviation of a specific sigma point and the rows each of the state vector components. The previous state estimation is taken as the mean and also constitutes the first sigma point. The other 14 left sigma points are concatenated to this one forming a 7,15-matrix.

$$\begin{array}{ll} \mathcal{X}^{(0)}_{k-1} & = \boldsymbol{\mu}_{x_{k-1}} \\ \mathcal{X}^{(i)}_{k-1} & = \boldsymbol{\mu}_{x_{k-1}} + \mathcal{W}^{\,(i)}_{x_{k-1}} \quad i = 1\ldots 2N \end{array} \tag{5.7}$$

Process model

These sigma points are propagated ahead in time through the non-linear process model to get a new set of sigma points for the predicted state at time k.

$$\mathcal{X}^{(i)}_{k} = f\left(\mathcal{X}^{(i)}_{k-1}, \mathbf{0}\right) \quad i = 0\ldots 2N \tag{5.8}$$

The velocity of the VUKF is calculated through the first order relationship between the velocity and the acceleration, hence the new velocity would be updated through the Forward Euler method

$$\begin{aligned} {}^{I}\dot{x}_k &= {}^{I}\dot{x}_{k-1} + \Delta t\, {}^{I}\ddot{x}_{k-1} \\ {}^{I}\dot{y}_k &= {}^{I}\dot{y}_{k-1} + \Delta t\, {}^{I}\ddot{y}_{k-1} \end{aligned} \tag{5.9}$$

where Δt corresponds to the sampling time.

The changes in acceleration can be described by the dynamic model but it is computationally heavy, hence, the acceleration is defined kinetically as a constant process model where the changes are defined as random noise and included in the noise covariance matrix $\boldsymbol{Q}$.

$$\begin{aligned} {}^{I}\ddot{x}_k &= {}^{I}\ \ddot{x}_{k-1} \\ {}^{I}\ddot{y}_k &= {}^{I}\ \ddot{y}_{k-1} \end{aligned}$$

The accelerometer bias is modelled as a constant in which the changes correspondent to a tunable white noise variable whose covariance is applied in the noise covariance matrix $\boldsymbol{Q}$

$$\boldsymbol{b}_{acc,k} = \boldsymbol{b}_{acc,k-1} \tag{5.10}$$

Hence, the complete process model used for propagating the sigma points following (5.8) is defined as

$$\mathcal{X}_k^{(i)} = \begin{bmatrix} {}^{I}\hat{\boldsymbol{x}}_{k-1}^{(i)} + \Delta t\, {}^{I}\hat{\ddot{\boldsymbol{x}}}_{k-1}^{(i)} \\ {}^{I}\hat{\boldsymbol{y}}_{k-1}^{(i)} + \Delta t\, {}^{I}\hat{\ddot{\boldsymbol{y}}}_{k-1}^{(i)} \\ {}^{I}\hat{\dot{\boldsymbol{x}}}^{(i)k-1} \\ {}^{I}\hat{\dot{\boldsymbol{y}}}_{k-1}^{(i)} \\ \hat{\boldsymbol{b}}_{\text{acc},k-1}^{(i)} \end{bmatrix} \tag{5.11}$$

Computation of the "a priori" mean and covariance

The a priori mean and covariance of the sigma points are computed weighted by a factor α_i.

$$\check{\mathbf{x}}_k = \boldsymbol{\mu}_{x_k} = \sum_{i=0}^{2N} \alpha^{(i)} \mathcal{X}_k^{(i)} \tag{5.12}$$

This weight depends on the parameter kappa and the dimension N of the probability distribution.

$$\alpha^{(i)} = \begin{cases} \frac{\kappa}{N+\kappa} & i = 0 \\ \\ \frac{1}{2}\frac{1}{N+\kappa} & i \neq 0 \end{cases} \tag{5.13}$$

The predicted covariance is the result of adding the covariance of the propagated sigma points (3.22) to the noise covariance matrix $\mathbf{Q}$, defined in (5.26). Therefore, it results in a 7,7-matrix covariance matrix $\check{\mathbf{P}}_{xx_k}$ which is updated with the integration of the process noise from the process model, and grows more and more every iteration.

$$\check{\mathbf{P}}_{xx_k} = \sum_{i=0}^{2N} \alpha^{(i)} \left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right) \left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)^T + \mathbf{Q} \tag{5.14}$$

5.2 Correction step

The predicted state sigma points $\mathcal{X}_k^{(i)}$ which are seven-dimensional are transformed through the non-linear measurement

$$\mathcal{Z}^{(i)} = h\left(\mathcal{X}_k^{(i)}, \mathbf{0}\right) \quad i = 0 \dots 2N \tag{5.15}$$

In this case the measurement models correspond to transform the states to the measurement variables read by the encoder and accelerometer

$$\hat{\boldsymbol{z}} = \begin{bmatrix} \hat{\boldsymbol{z}}_{enc} \\ \hat{\boldsymbol{z}}_{acc} \end{bmatrix} = \begin{bmatrix} \hat{\phi}_0 \\ \hat{\phi}_1 \\ \hat{\phi}_2 \\ {}^B\hat{\hat{x}}_{B|K} \\ {}^B\hat{\hat{y}}_{B|K} \\ {}^B\hat{\hat{z}}_{B|K} \end{bmatrix} \tag{5.16}$$

Measurement model

The model h is defined by the sensors providing the measurements. Using the definition of the encoder from (3.15) discretizing without the noise yields to

$$\hat{\boldsymbol{z}}_{\text{enc},k} = \frac{n_{\text{ticks}}}{2\pi} \Delta t \widetilde{\boldsymbol{W}} \, \boldsymbol{\Phi}\left({}^K_B\hat{\boldsymbol{q}}_k\right)^T \left(\boldsymbol{\Gamma}\left({}^K_B\hat{\boldsymbol{q}}_k\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -\hat{y}_k \\ \hat{x}_k \\ 0 \end{bmatrix} - 2\,{}^K_B\hat{\boldsymbol{q}}_k \right) \tag{5.17}$$

In the case of the VUKF, the accelerometer measures are not simplified as it was done with the QUKF since it is required to obtain the influence of the different

accelerations in the system. The measurement model of the accelerometer is discretized from (3.3) yielding to

$$ {}^{IMU}\hat{\boldsymbol{z}}_{\text{acc},k} = \left(\vee\boldsymbol{\Phi}\left({}^{K}_{B}\hat{\boldsymbol{q}}_k\right)^T\boldsymbol{\Gamma}\left({}^{K}_{B}\hat{\boldsymbol{q}}_k\right)\wedge\right)\begin{bmatrix} {}^{I}\hat{\ddot{x}}_k \\ {}^{I}\hat{\ddot{y}}_k \\ g \end{bmatrix} + {}^{IMU}\hat{\boldsymbol{b}}_{\text{acc}} \tag{5.18} $$

Finally, the measurement sigma points are computed transforming them using both measurements

$$ \mathcal{Z}_k{}^{(i)} = \begin{bmatrix} \hat{\boldsymbol{z}}^{(i)}_{\text{enc,k}} \\ \hat{\boldsymbol{z}}^{(i)}_{\text{acc,k}} \end{bmatrix} \tag{5.19} $$

Computation of the measurement mean and covariance of the estimate

The mean and covariance of the predicted measurements are estimated using the weighted α_i parameter from Equation (5.13)

$$ \hat{\boldsymbol{z}}_k = \boldsymbol{\mu}_z = \sum_{i=0}^{2N}\alpha^{(i)}\mathcal{Z}^{(i)} \tag{5.20} $$

$$ \hat{\mathbf{P}}_{zz} = \sum_{i=0}^{2N}\alpha^{(i)}\left(\mathcal{Z}^{(i)} - \boldsymbol{\mu}_z\right)\left(\mathcal{Z}^{(i)} - \boldsymbol{\mu}_z\right)^T + \mathbf{R} \tag{5.21} $$

The measurement noise covariance is defined in (5.28).

Computation of cross-covariance and Kalman gain

The cross-covariance and Kalman gain are 7,6-matrices given by

$$ \hat{\mathbf{P}}_{xz} = \sum_{i=0}^{2N}\left(\mathcal{X}_k^{(i)} - \boldsymbol{\mu}_{x_k}\right)\left(\hat{\mathcal{Z}}^{(i)} - \boldsymbol{\mu}_z\right)^T \tag{5.22} $$

$$ \mathbf{K}_k = \hat{\mathbf{P}}_{xz}\hat{\mathbf{P}}_{zz}^{-1} \tag{5.23} $$

Computation of the "a posteriori" mean and covariance

The Kalman gain is used to optimally correct the mean and covariance of the predicted "a priori" state estimate with the residual between the sensors measurement

and the measurement estimate from the model, leading to the "a posteriori" state estimate and covariance.

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k \right) \tag{5.24}$$

$$\hat{\mathbf{P}}_{xx_k} = \check{\mathbf{P}}_{xx_k} - \mathbf{K}_k \hat{\mathbf{P}}_{zz} \mathbf{K}_k^T \tag{5.25}$$

Since the VUKF makes use of the quaternion estimation obtained from the QUKF, its convergence and error are influenced by the QUKF one.

5.3 VUKF covariance matrices

The process noise covariance matrix $\boldsymbol{Q}$ is build collecting all the covariances from the process model, giving a 6,6-matrix [6]

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_v \mathbf{I}_2 & \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 2} & \sigma_a \mathbf{I}_2 & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} & \boldsymbol{\Sigma}_{\text{bias}} \end{bmatrix} \tag{5.26}$$

where σ_v is the covariance of the distribution of errors occurring due to the discretization of the system, hence they are tunneable. As for σ_a is the covariance corresponding to the expected change in acceleration, thus, if it is expected to have large changes in acceleration the σ_a can be increased to give less trust to the predictions and relay more in the readings. Finally Σ_{bias} is tuned depending on how fast the accelerometer bias changes in the process.

The measurement noise covariance is a 6,6-matrix built upon the error covariances of the encoder and the accelerometer in addition to the covariance of the accelerometer coming from the QUKF (4.36)

$$\hat{\mathbf{P}}_{xx,k} = \begin{bmatrix} \hat{\mathbf{P}}_{xx,k,acc} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \hat{\mathbf{P}}_{xx,k,gyro} \end{bmatrix} \tag{5.27}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\Sigma}_{\text{enc}} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{\Sigma}_{\text{acc}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \hat{\mathbf{P}}_{xx,k,acc} \end{bmatrix} \tag{5.28}$$

where the covariance of the encoder $\boldsymbol{\Sigma}_{\text{enc}}$ is defined in (3.1.3) as $\frac{1}{9}\boldsymbol{I}_3$ and the covariance of the accelerometer $\boldsymbol{\Sigma}_{\text{acc}}$ defined in (3.4) as a 3,3-matrix.

Chapter 6

Estimators results

The estimator is implemented and tested in simulation, where the output of the system model, χ, is fed to a model of the sensors which adds noise to the states. The noise of the sensors is shown in Figure 6.1, Figure 6.2 and Figure 6.3, corresponding to noise of the encoder, gyroscope and accelerometer respectively.

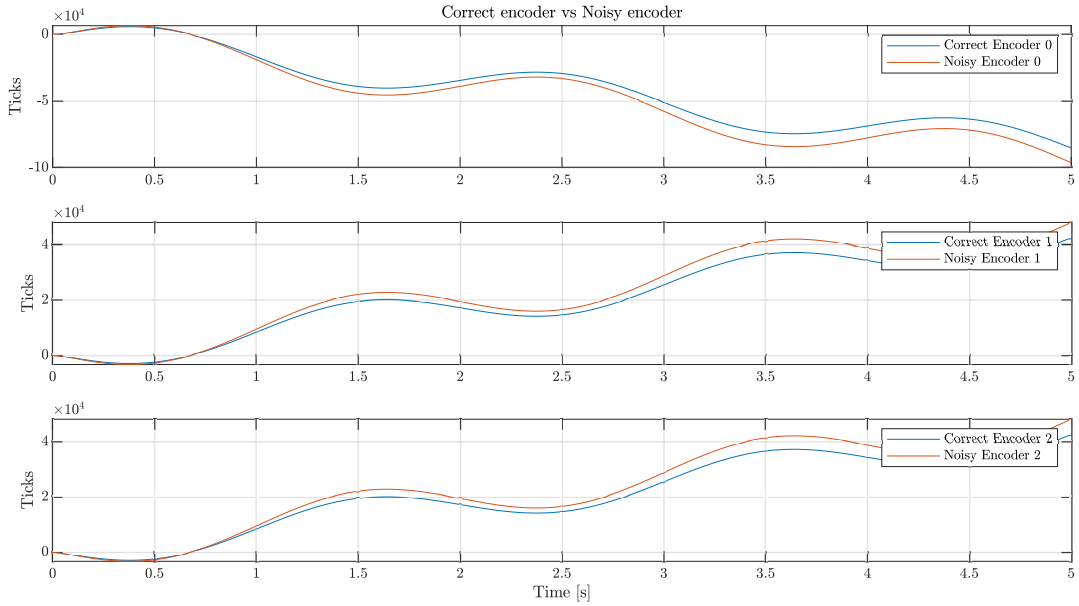


Figure 6.1: Ideal encoder signal compared to the deviated encoder signal.

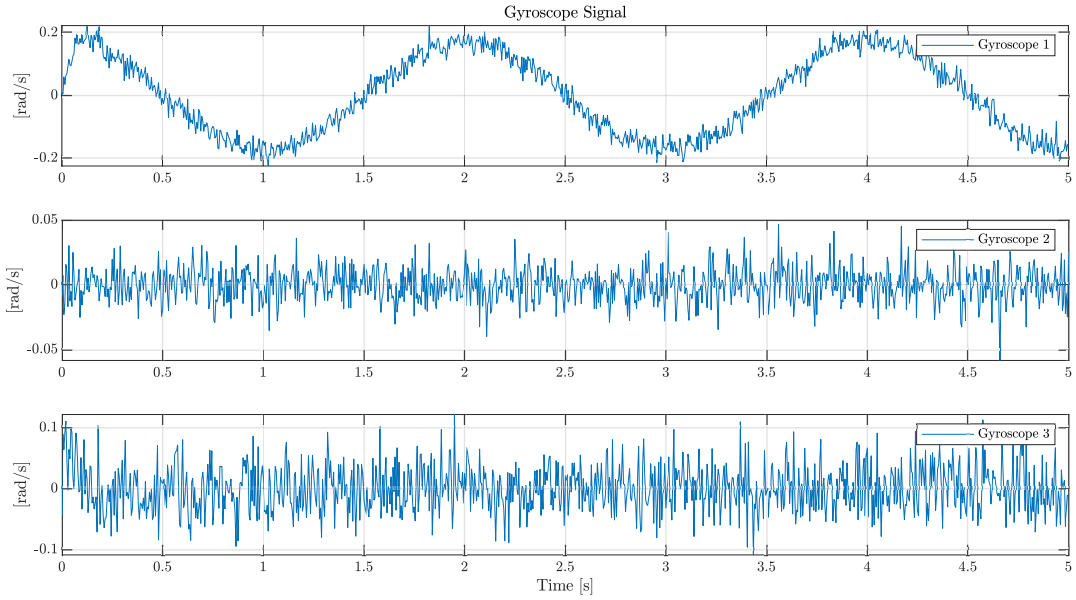


Figure 6.2: Signal given by the gyroscope with the measurement noise included.

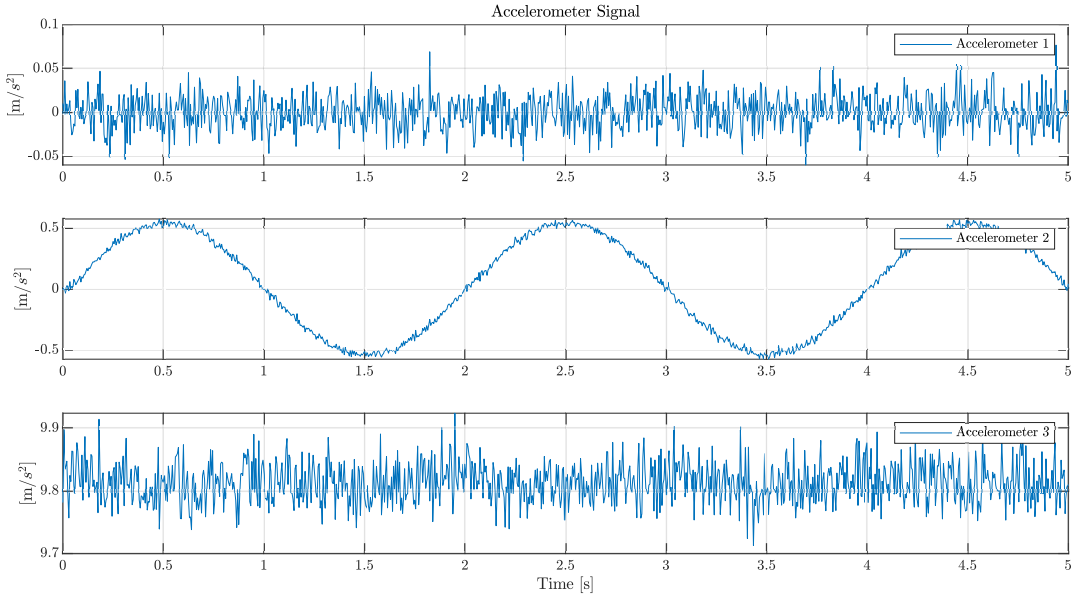


Figure 6.3: Signal given by the accelerometer with the measurement noise included

The measurements obtained by the model of the sensors are used as input to the estimator. Furthermore, it is desired to compare the performance of the proposed UKF against the EKF to assess if the use of a more advanced estimator is justified with remarkable improvements. The comparison is possible thanks to the use

of the previous results obtained by the V2019 ball-robot in the project [6], which makes uses of the mentioned EKF to estimate the states.

The time lapse chosen to show the following results covers the main characteristics of the response, further time lapses only add repetitive patterns similar to the results presented.

6.1 QUKF results

The first estimator implemented is the QUKF, since its output is required to be fed to the VUKF. The QUKF provides an accurate tracking of the quaternion reference as it is seen in Figure 6.4. The real part of the quaternion as well as the q_1 component, follows the reference correctly. The q_2 and q_3 components are slightly noisy with an error of order ± 0.001 and ± 0.002 respectively.

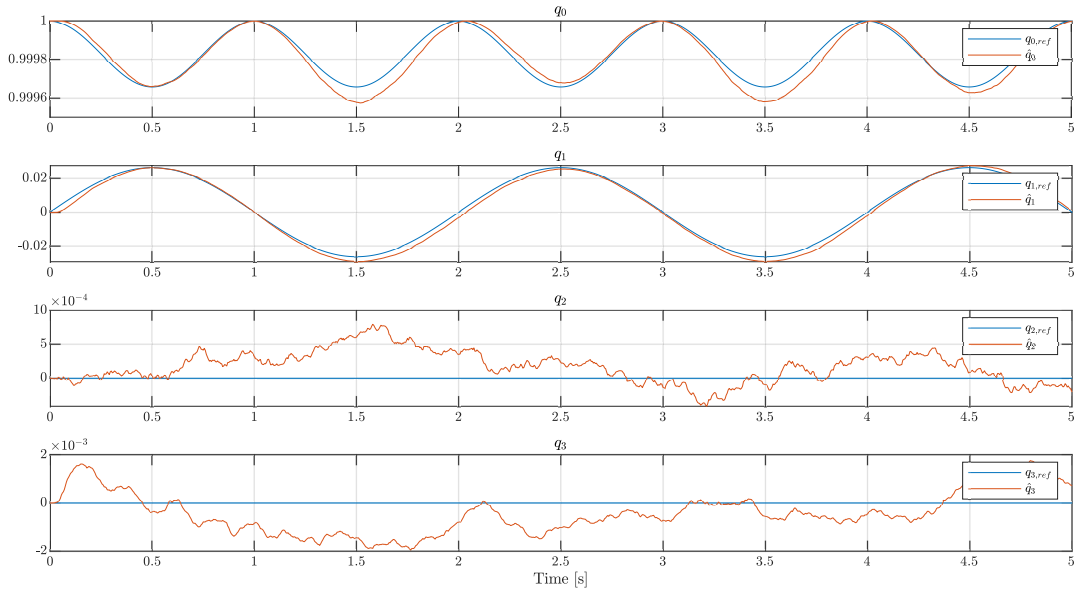


Figure 6.4: Comparison of the quaternion obtained by the QUKF against the referenced quaternion signal.

More intuitive visualization of the results expressed in Euler Angles is shown in Figure 6.5, where it can bee seen that the movement is tracked achieving low error. The error seen in the yaw angle might be corrected using another sensor, such as a LiDAR, which would allow to correct the heading estimate.

Part of the error seen in the quaternions might be due to the yaw error distributed

between the components of the quaternion.

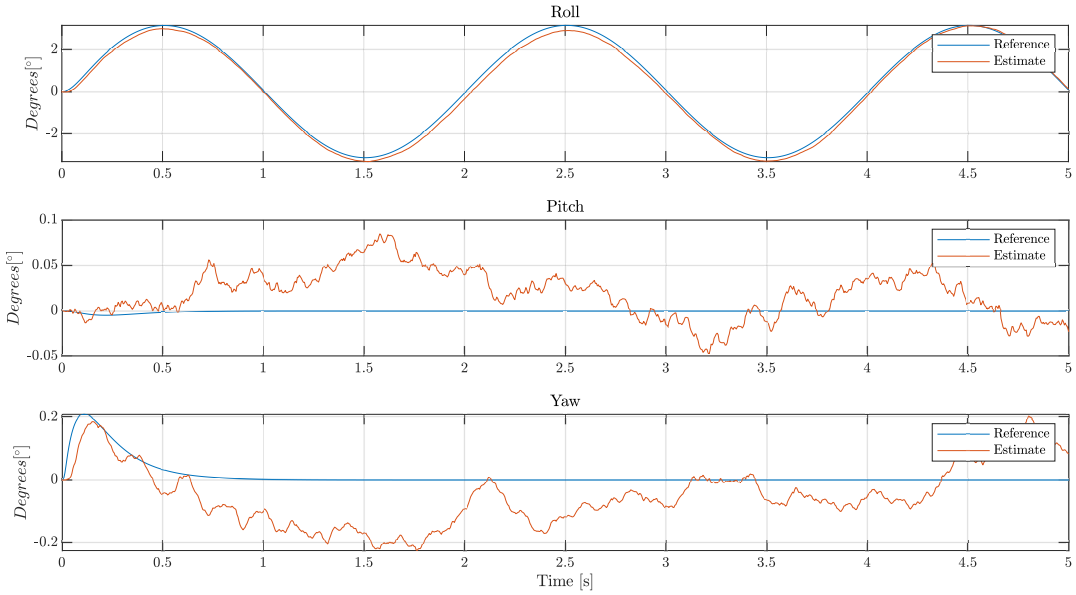


Figure 6.5: Comparison of the estimated Euler angle against the reference orientation.

To assess the quality of the QUKF, the results are compared with the ones obtained in the V2019 ball-robot [6]. Figure 6.6 shows the error in tracking for both of them in Euler angles representation. It is observed that there is no mayor difference in the performance of the observers. Both of the estimators yield an error considered admissible for the application requirements.

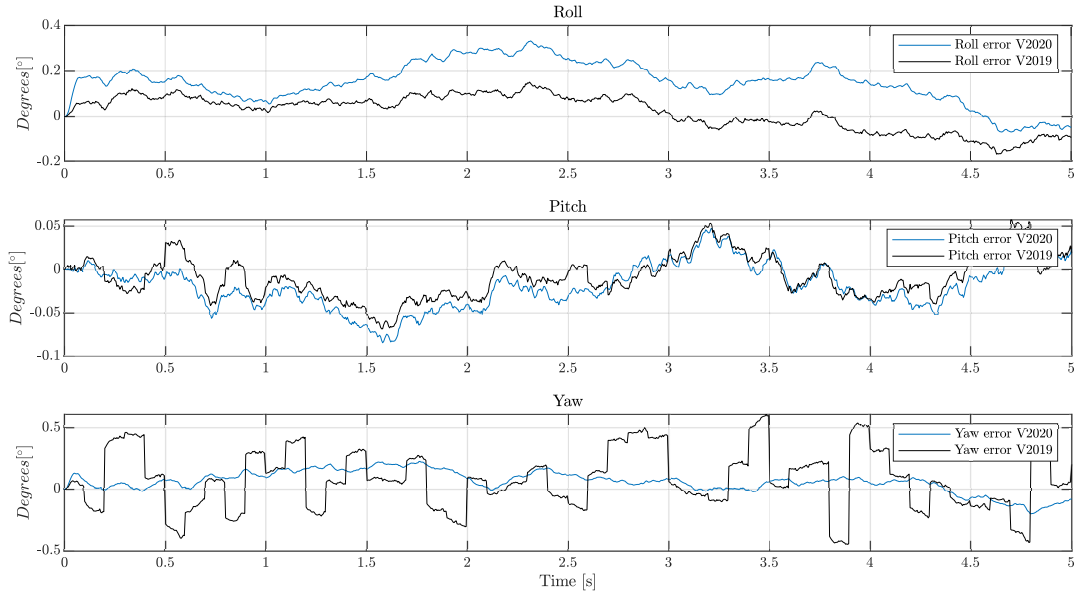


Figure 6.6: Comparison in error between the QUKF from V2020 and QEKF from V2019 expressed in Euler Angles.

For the quaternion derivative tracking shown in Figure 6.7, it is quite remarkable the precision obtained by the estimator, since the estimation for every component follows adequately the references, being the signals centered around the actual value. As it was expected, the estimation in the quaternion derivative have more noise than the quaternion estimation.

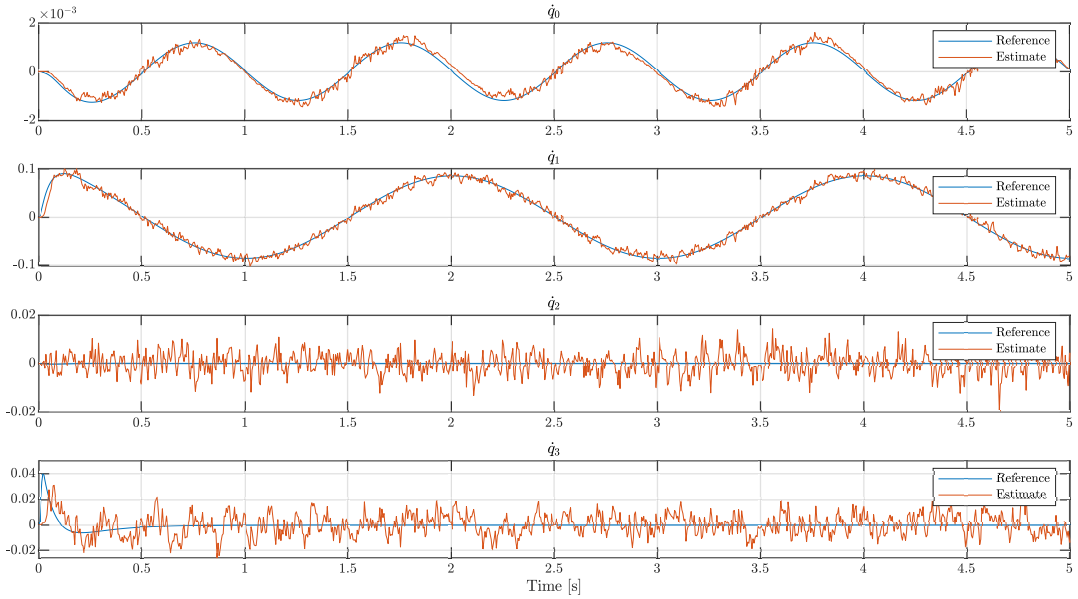


Figure 6.7: Quaternion derivative tracking obtained by the quaternion estimator.

A comparison between the errors obtained by the QUKF (V2020) and the QEKF (V2019) for the quaternion derivative can be observed in Figure 6.8. It is seen that there are no noticeable differences in the performance.

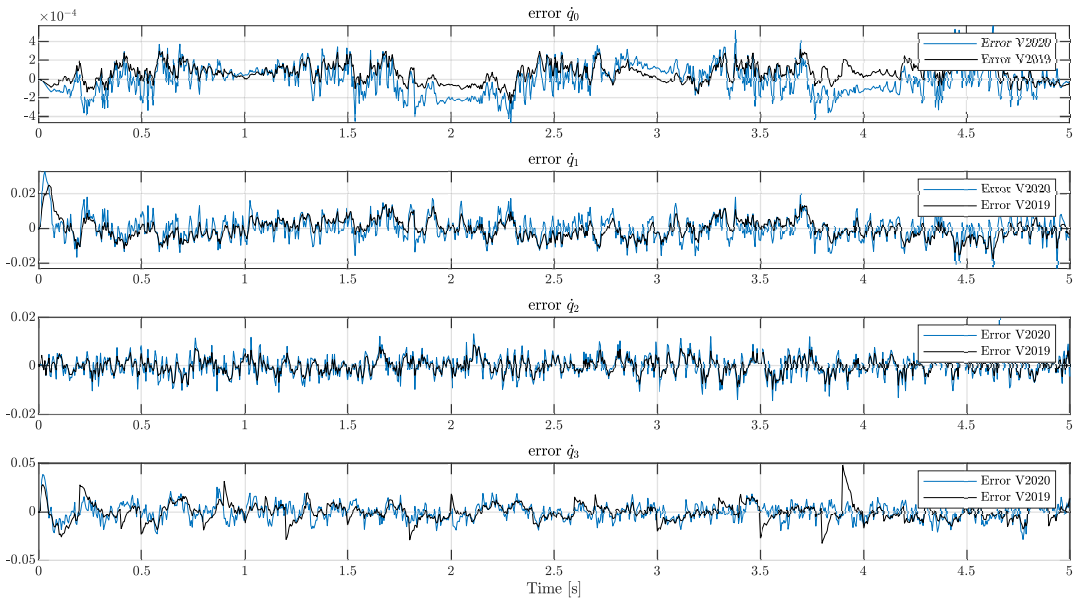


Figure 6.8: Comparison of the error of the quaternion derivative between the QUKF from V2020 and QEKF from V2019.

6.2 VUKF results

The quaternion and its derivative estimation signals are used to calculate the position and the linear velocity estimates through the Velocity Unscented Kalman Filter. The results are compared with the reference in Figure 6.9.

The estimations obtained by the VUKF are considered precise enough for the ball-robot application, with a slight constant drift of $-0.0002 m/s$ for the x-axis position, and a small chattering in the velocity x-axis.

It is ventured that it can be caused by the quaternion error coming from the QUKF, given that an error in the original orientation from the body can produce a small deviation in the translational velocity. Despite that, e.g. the error in the x-axis position after 100 seconds would accumulate to 2 cm, which might be considered acceptable for the current scope of the project. An additional sensor correcting the heading in the QUKF might mitigate the deviation in the position and velocity estimate.

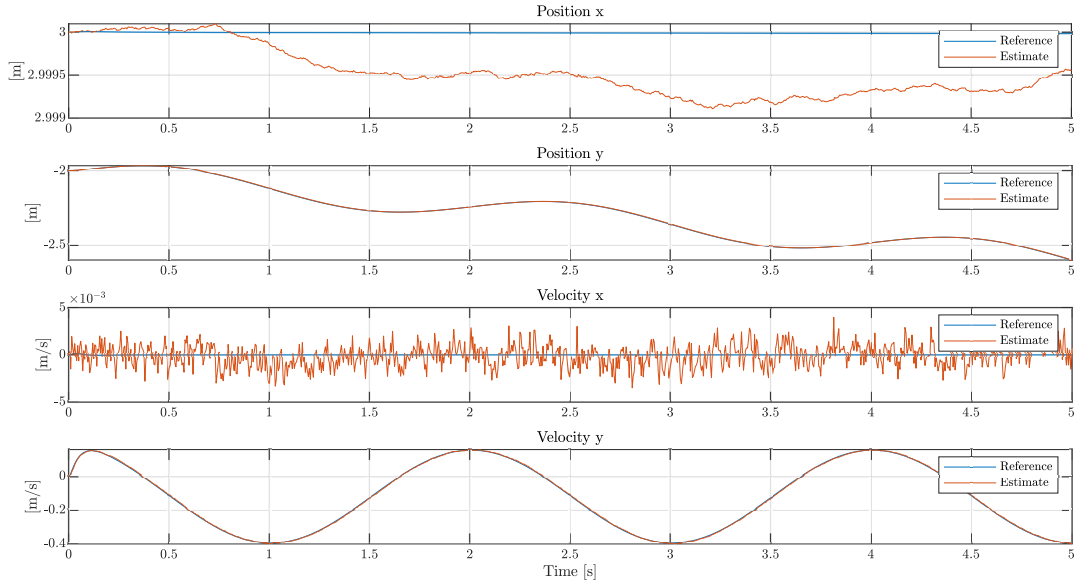


Figure 6.9: Tracking performance in the position and linear velocity.

Comparing the errors of the VUKF and the VEKF in Figure 6.10, it is observed that the results are similarly accurate, with a slightly better performance in the y-axis velocity estimated by the VEKF. The difference might be caused by the non-use of the heading in the QUKF versus a heading correction in the QEKF.

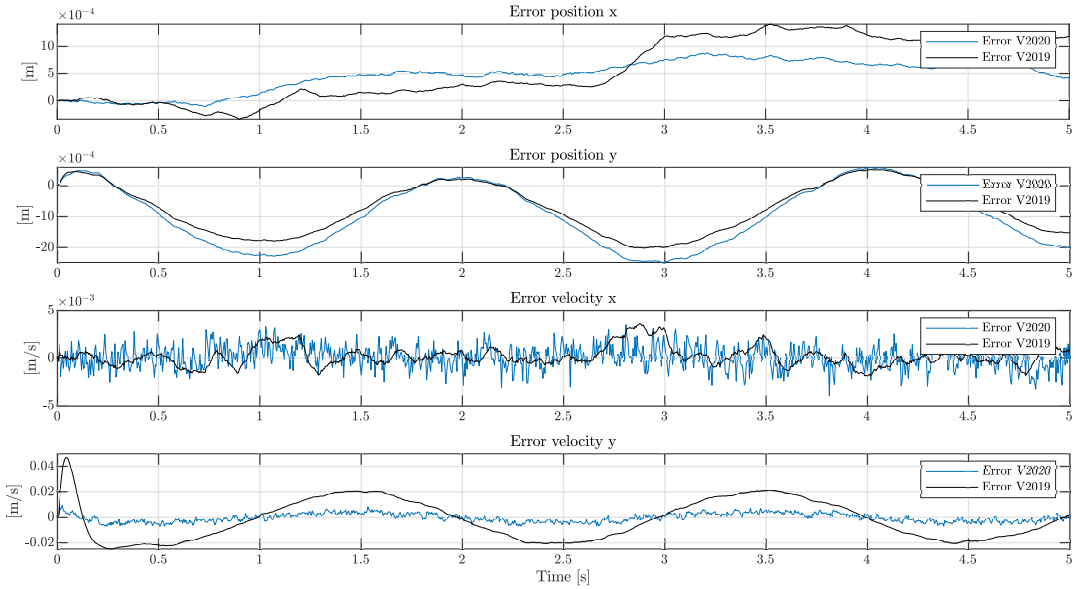


Figure 6.10: Comparison of error for the UKF in V2020 and the EKF in V2019 to estimate the position and linear velocity.

6.3 Estimators conclusion

From the previous results, some conclusions regarding the Kalman filter strategy chosen are drawn. Firstly, it was assumed that a model-driven UKF was going to lead to better results but with the trade-off of being computationally more costly than the implemented sensor-driven UKF. However, the accuracy achieved by the sensor-driven approach validates the use of the kinematic relationships, since it is sufficient for the ball-robot application with a less computational cost.

From the comparison with the previous work done in V2019, it can be synthesized that the Unscented Kalman Filter produces similar results as the Extended Kalman Filter in simulation. Theoretically, the UKF is supposed to lead to more accurate results because it avoids linearizing the system. However, the similarity in the results between the UKF and EKF leads to conclude that the model of the system is not highly non-linear around the references generated. The combination of this premise with a high-speed computation in the algorithm relative to the dynamics of the system allows assuming the linearization error of the EKF to be negligible.

Furthermore, due to the sensor-driven approach, the functions for the process and measurement models are relatively easy Forward-Euler integration and kinematic relationships. These equations can be easily differentiated to form analytically the

Jacobians matrices of the system required by the EKF. As a consequence, it is simple to implement an EKF which on top of that requires less computational cost.

Regarding the quaternion algebra implemented in the QUKF, it is essential to point out that there is a significant difference desired to be tested between the QUKF and the QEKF implementation. The QEKF makes use of unit quaternion additions and subtractions during the steps of the algorithm, which is then corrected by a forced unitary norm of the resulting quaternion. This is not desirable in quaternion algebra since it introduces error in every iteration to force the quaternion to go back to the unit-sphere. The QUKF is implemented with modifications from the classical UKF in order to substitute all the quaternion additions and subtractions adequately by quaternion multiplications and iterative algorithms. Since the results provided by the QUKF are almost similar to the ones obtained with the QEKF quaternion strategy, it could be concluded that the error coming from the forced normalization is not noticeable, which is mentioned as a possibility in [36].

Also, it is remarkable how the UKF with no heading available produces as accurate estimations as to the EKF that has heading readings available for correction. Hence, it might be possible to produce better results with the UKF if the heading would be available.

In conclusion, an Unscented Kalman Filter approach was tested, where the main differences against the previous Extended Kalman Filter lie in transforming the covariances with the non-linear model and a different approach to manage the quaternion algebra and its error. The variations had led to similar results which might induce that is more practical to develop an EKF instead of a UKF for the ball-robot case. However, it would be required to evaluate the results in the real device, as well as increase the range of operation evaluated to provide conclusive statements.

Part III

Controllers

Chapter 7

Controllers introduction

In the following chapters, the control scheme applied to the balancing robot is presented, simulated, and the results are discussed.

The current chapter describes the system characteristics, and the controller scope is provided in order to set the requirements of the controller. Furthermore, the velocity controller and how to cope with quaternion errors are presented.

In Chapter 8, the controller providing the balance of the ball-robot is explained in detail, and the results obtained presented.

In Chapter 9 the results of the balance controller combined with the velocity controller, the sensor models and the estimators are presented and analyzed.

7.1 System characteristics and design specification

The ball-robot is conceived to track a defined path, avoiding obstacles while maintaining the balance of the body for expected operating conditions as well as under disturbances, to a certain extend. In the previous project [6], a use case is suggested where the ball-balancing robot acts as a human assistant and guidance in environments like airports, providing information or showing the desired path. For the current project, the trajectory planning, as well as obstacle avoidance, is not contemplated. Nonetheless, it is decided to define the specifications based in the predecessor project [6]

1. Maximum translational velocity of $0.5 m/s$.

2. Linear acceleration limit of $1 m/s^2$.
3. Maximum heading velocity ($\dot{\psi}$) of $1 rad/s \approx 60°/s$.
4. Angular acceleration limit of $10\ rad/s^2$.
5. Inclination limit $\pm 5°$.
6. Angular velocity on inclination less than $20°/s$ to avoid aggressive jumps and to reduce the risk of wheel-slip.

On top of the desired specifications, it is necessary to account for the hardware and design characteristics. The ball-robot consists of an idealized cylindrical body above a ball. The assumption of an idealized cylindrical body is upfront imprecise. Hence it would be required to define the controller accounting for a possible misalignment in the Center Of Masses (COM). The ball-balancing robot is commanded by three motors in charge of providing torque to the ball to maintain the balance and accomplish the translational movement. The system has five Degrees Of Freedom (DoF). However, it is commanded by three motors providing three torques. Hence, the system is under-actuated; there are more degrees of freedom than actuators. Due to its under-actuated properties, it means that there exists a coupling between the degrees of freedom; i.e. to provide a rotational movement in one of the axes, there might be an influence in other axis or the translational velocity. The existing couplings between the states of the system are critical when developing the controllers.

The physical properties of the ball-robot besides being under-actuated, classifies it as a shape accelerated robot. An accelerated shape system is destabilized by gravitational forces and has non-integrable constraints in their dynamics [16]. Furthermore, the shape configuration provides a mapping between the accelerations of the system and the configuration. The functionality of the ball-robot is built on its shape accelerated characteristics. Hence the body inclination angle generates a linear acceleration that makes it reach the desired translational velocity, while at the same time seeking to maintain the balance.

An accelerated shape system is significantly influenced by its physical properties, such as the location of the COM and inertias. On top of that, a possible misalignment in the IMU position provides an accumulative error for the readings and the calculations. Therefore it is decided to use a cascade controller structure, with an outer loop in charge of controlling the velocity and an inner loop controlling the balance. The velocity controller, placed in the outer loop, ensures that the reference orientation and reference rotational speed are correct for the desired velocity. It corrects the COM misalignment and deviations in the physical properties. The

inner loop receives the desired angle and rotational speed from the velocity controller and generates the torque control commands to track those references. A schematic of the cascade control scheme used for the simulation implementation is shown in Figure 7.1.

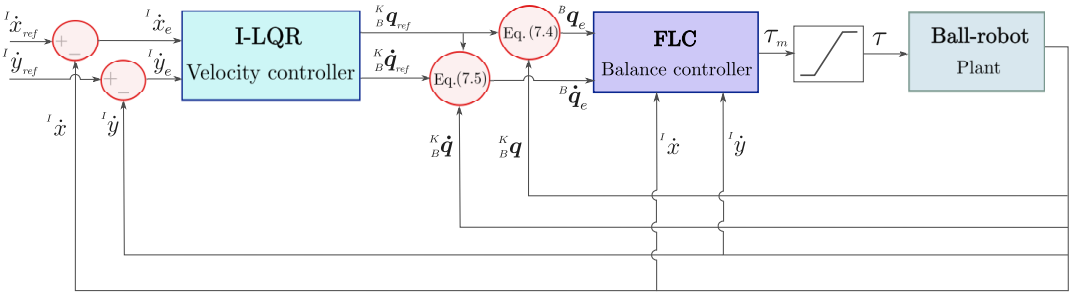


Figure 7.1: Schematic of the cascade control scheme.

In a cascade control structure, it is required that the inner control loop behaves with bandwidth at least three times faster than the outer loop [37]; consequently, the inner loop dynamics can be neglected for the outer dynamics. Besides, it is commonly recommended to have a sampling rate ten times faster than the bandwidth [38]. The sampling rates of the sensor should be chosen to avoid aliasing, sensor noise and other problems derived from a fast sampling rate. Nevertheless, it is proved in [6] that is possible to have enough fast sampling rate compared to the dynamics of the controller. Therefore, it is possible to design the controller algorithm in the continuous space, or in other words, it is not required to use discretized controllers. It is also required to take into consideration the bandwidth of the estimators, which should be two to six times faster than the controllers in order to omit its dynamics in the controller design. The sample rate of the controller and the quaternion and velocity estimator is set to 200 Hz.

The ball-robot presents a non-minimum phase dynamics due to Right Half Plane (RHP) zero behaviour. As well as in a SISO system, in a MIMO system, the RHP-zeros can impose fundamental limitations in closed-loop performance. However, they can be useful in some applications, e.g. RHP-zeros is an excellent method to model the system delay in power electronics converters and analyze its influence in closed-loop stability. The zeros of a MIMO system are where the transfer function $G(s)$ loses rank. Therefore the direction of zero can be found by looking at the direction where the matrix $G(s)$ has zero gain. However, this method has limitations since sometimes the zeros are cancelled by close poles [39].

The RHP zeros effect can also be observed experimentally since the ball-robot has inverse response behaviour at the beginning of the response. When commanding

a reference of linear displacement in the x-axis, the output initially has to move to the negative x-axis direction before starting moving to its positive steady-state value. Same applies to the y-axis. This inverse response is needed in the ball-robot since it is shape-accelerated and tilts the body to the desired orientation to start moving due to the gravity. This effect yields a delay in the response due to a sudden negative phase added by the RHP zero. The controller is designed accordingly. Otherwise, the reference is created taking this effect into account to compensate for it.

This ball-bot characteristic is due to its shaped accelerated characteristic combined with actuators at a lower height than the COM of the body.

7.2 Performance specification

The project scope is limited to the simulation environment, and as a consequence, it is decided to use the previous project [6] as the benchmark in performance:

1. The inclination, based on roll and pitch, should be more aggressive than heading.
2. Inclination tracking error in the steady-state reference should be held between $\pm 1°$.
3. Heading error should stay between $\pm 5°$.
4. The tracking lag should be less than $0.1s$ when tracking a step, ramp or sine wave.
5. Less than $1s$ lag when tracking a quaternion even tough no angular velocity reference is provided.

7.3 Velocity controller

As part of the cascade design, the outer loop is a velocity controller in charge of generating the quaternion and quaternion derivative references that are fed to the balance controller. The reference to the velocity controller is the linear velocities and the rotational velocities. The current velocity controller is developed to ensure that the linear velocity is tracked correctly, as well as to provide robustness against some disturbances, for example, COM miss-alignment. Different solutions

are applicable to control the velocity, such as a Model Predictive Control. However, it was proven in [6] that a Linear Quadratic Regulator performs properly for the solution requirements. Due to the good results of the previous velocity controller, it is decided to use the solution presented by [6] adapted to the new balance dynamics and focus the efforts in the balance controller. Nevertheless, an overview of the velocity controller used is presented next in order to take into account its influence in the balance controller performance.

As mentioned previously, the shape accelerated properties of the ball-robot makes it susceptible to misalignments in the COM, parameter mismatch in the system model and noise and drift in the IMU readings. These deviations may cause an error between the real orientation and the measured one by the sensors, despite having zero error in the orientation tracking, i.e. between the reference orientation and the measured one. Therefore, the acceleration of the body caused by gravity, also known as a run-away, would vary from the desired one, and as a consequence, the ball-robot linear velocity would deviate from the referenced one. A velocity controller is included to compensate for the deviation. It tracks the velocity error and produces an orientation reference to the balance controller according to the shape accelerated variation. The compensation is done defining two different states, the moving process and the standstill process.

A standstill process is considered when the velocity reference is zero and the ball-robot is intended to be maintained perpendicular to the ground. Hence there should be no translational velocity. However, due to the run-away effect, the controller has to account for a change in acceleration and command an orientation reference compensation. Therefore, an integral term corresponding to the position error is included in the velocity controller. Otherwise, the ball-robot is tracking a velocity. Hence it will behave like a typical error tracking controller, where the error is considered the actual states minus the reference signal

$$\begin{aligned} {}^{\mathrm{I}}\dot{x}_e &= {}^{\mathrm{I}}\dot{x} - {}^{\mathrm{I}}\dot{x}_{\mathrm{ref}} \\ {}^{\mathrm{I}}\dot{y}_e &= {}^{\mathrm{I}}\dot{y} - {}^{\mathrm{I}}\dot{y}_{\mathrm{ref}} \end{aligned} \tag{7.1}$$

Finally, a Linear Quadratic Regulator was defined to track the velocity. The performance of the velocity controller is shown in Figure 7.2.

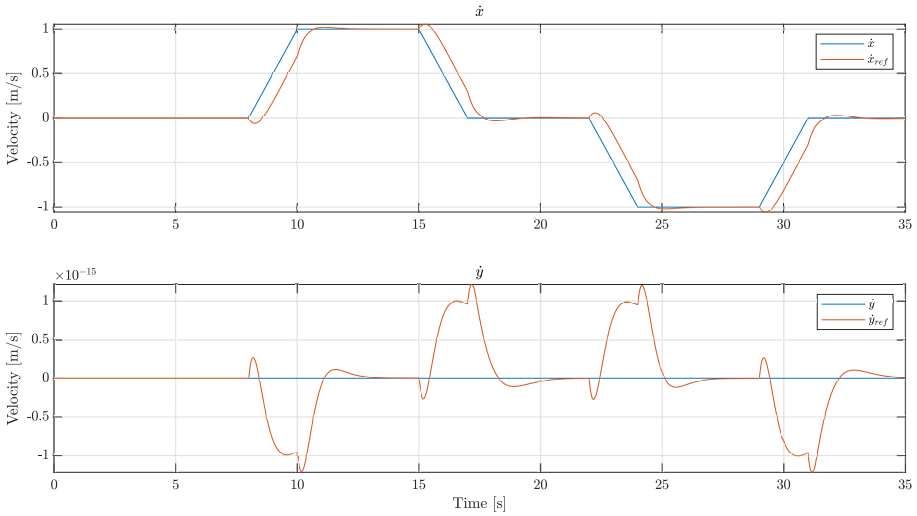


Figure 7.2: Comparison of velocity reference vs velocity tracking by the velocity controller.

It can be seen how the velocity controller converges the states to the reference ones in steady-state. However, a ramp input is tracked with a constant delay due to the non-minimum phase dynamics of the ball-robot. The RHP zeros introduce an inverse response which generates the delay. The performance of the velocity controller is considered satisfactory under the performance requirements.

7.4 Quaternion tracking error

Controllers are commonly designed to track a reference; hence, the controller input signal is defined as the error or deviation between the reference and the actual states, which in most of the cases is computed as a subtraction

$$x_e = x - x_{ref} \tag{7.2}$$

The balance controller addressed in this project requires to track a quaternion and its derivative. The error fed to the controller is. Therefore, the quaternion deviation and its derivative. Due to the use of quaternions, it is not possible to use the algebraic subtraction in (7.2), since unit quaternions are not closed under subtraction. The orientation error must be computed by quaternions multiplication which is given by

$$q_c = q_a \circ q_b^* \tag{7.3}$$

The quaternion multiplication is a non-commutative operation. Therefore the order of the quaternions matter. Depending on the multipliers order, the quaternion error

might be expressed in the ball $\{K\}$ or in the body $\{B\}$ reference frame. An example and an extended comparison between these two approaches can be found in [6]. It remarks the benefits of using the quaternion error defined in the body frame. Thus, it is decided to choose the body frame quaternion error, which is computed by the reference quaternion left-multiplied.

$$^{B}\boldsymbol{q}_e = {}^{K}_{B}\boldsymbol{q}^{*}_{ref} \circ {}^{K}_{B}\boldsymbol{q} \tag{7.4}$$

The quaternion error can induce a non-optimal tracking path, where the rotation introduced by the controller would be longer than the intuitive one, due to the "double-cover" effect explained in Appendix E. The long path rotation can be avoided by evaluating the scalar value of the quaternion. The scalar would be positive for $\theta_e < 180$ and negative for $\theta_e > 180$. Thus, when the computation of the error gives a long rotation such that $\theta_e > 180$, it is possible to correct to the shortest part by inverting the sign of each of the elements of the quaternion.

The derivative of the quaternion error can be computed by taking the derivative of the quaternion error following Equation E.35

$$^{B}\dot{\boldsymbol{q}}_e = {}^{K}_{B}\dot{\boldsymbol{q}}^{*}_{ref} \circ {}^{K}_{B}\boldsymbol{q} + {}^{K}_{B}\boldsymbol{q}^{*}_{ref} \circ {}^{K}_{B}\dot{\boldsymbol{q}} \tag{7.5}$$

Chapter 8

Balance controller

The goal of the balance controller consists of following an orientation reference required to accelerate the robot and drive it at a specific linear velocity.

The balance equilibrium of the body varies depending on the current orientation, and the translation acceleration required due to its shape-accelerated dynamics. The cascade control structure proposed allows the balance controller in the inner loop to focus on reaching the reference orientation and rotational velocities. The velocity controller commands the reference, and it is in the form of the body orientation quaternion and its derivative. The non-linear controller proposed and studied for the balance purposes is the Non-linear Feedback Linearization Controller (FLC).

However, in the current chapter the FLC is analyzed and simulated without the influence of the velocity controller, therefore, as seen in Figure 8.1 the input to the FLC is fed by a reference generator.

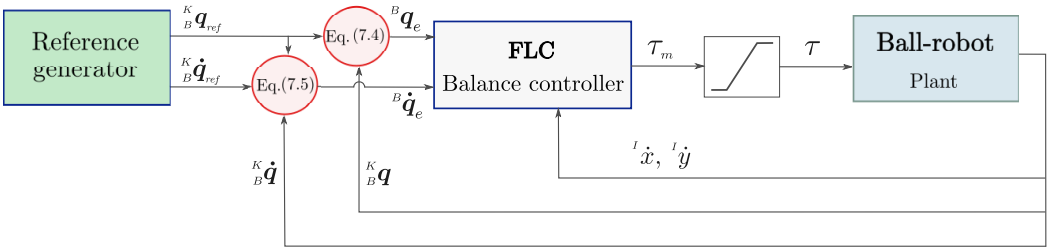


Figure 8.1: Schematic of the simulation of the balance controller using Equation 7.4 and 7.5 to calculate the error fed to the controller.

8.1 Non-linear feedback linearization controller (FLC)

Non-linear Feedback Linearisation Control (FLC), often referred to as Computed-torque control, aims to algebraically transform a non-linear system dynamics into a (fully or partially) linear one, so that linear control techniques can be applied. Feedback Linearisation (FL) is achieved by exact state transformations and feedback, as opposed to the Jacobian linearisation/Taylor expansion techniques, which linearly approximates the dynamics. The idea of simplifying the form of a system's dynamics by choosing a different state representation is widely used in other fields, such as mechanics when a system model reduces its complexity considerably by choosing correctly the reference frames or coordinate system [40].

For a SISO non-linear system, FL aims to cancel the non-linearities so that the closed-loop dynamics are in a linear form and impose a desired linear dynamics; in a MIMO case, it is also in charge of decoupling the system. The most common procedures in the literature for the Feedback Linearisation are input-output linearisation and state-space linearisation.

Input-output linearisation is characterised by linearising the relation between the transformed input v and the actual outputs y. It is the Feedback linearization strategy chosen to implement the balance controller and its structure is depicted in Figure 8.2.

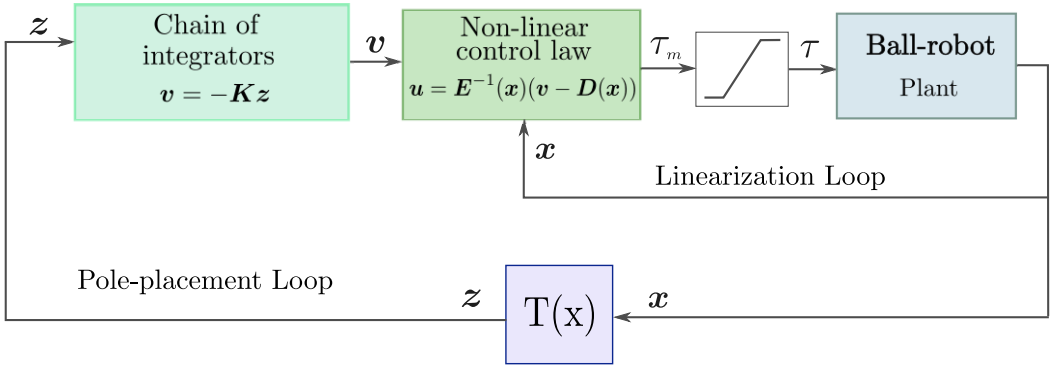


Figure 8.2: Schematic of the zero reference tracking Input-State Feedback Linearization Controller.

On the other hand, state-space linearisation makes a linearisation of the transformed input and a vector of transformed states. The state-space incurs on having more outputs than the real ones. Hence, the extra ones are considered artificial outputs w [41].

In the current project, the focus is put in Input-output linearisation, in which once

the change of coordinates is done, and the input-output linearisation is obtained, linear control techniques are applicable.

8.2 Implementation

The implementation of the FLC scheme in the ball-robot system implies a series of assumptions and extensions from the general form of the algorithm and are going to be presented sequentially throughout the definition of the algorithm. The input-output linearisation approach to multiple-input multiple-output (MIMO) is often called input-output decoupling because the input-output response is both linearised and decoupled. More precisely, it consists in finding a diffeomorphism and a state feedback control law such that, the map between the transformed inputs v and controlled outputs y is linear, and the i-th output y_i is decoupled from all inputs v_j for $i \neq j$ [41].

The non-linear differential equations of the ball-robot dynamics in the so-called controllability canonical form are given by

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{y} &= h(\boldsymbol{x}) \end{aligned} \tag{8.1}$$

where the states are defined by the generalized coordinates and their derivatives

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\chi} \\ \dot{\boldsymbol{\chi}} \end{bmatrix} = \begin{bmatrix} x \\ y \\ {}^{K}_{B}\boldsymbol{q} \\ \dot{x} \\ \dot{y} \\ {}^{K}_{B}\dot{\boldsymbol{q}} \end{bmatrix} \tag{8.2}$$

To control the states a cascade control scheme is chosen, which has a balance controller in the inner loop and a velocity controller in the outer loop. Since it is a shape-acclerated system the orientation of the body is what induces the movement, and therefore what defines the torque control inputs of the motors. The FLC is applied to the balance controller and is in charge of controlling the orientation, which implies controlling the quaternion and quaternion derivative states. A quaternion-based FLC design implies transforming the state-space from eight variables to six since the system possesses six differential equations (system dimension n is equal to 6); this is due to the fact that the two removed equations corresponds to the q_0 and $\dot{q}_0$ components that are algebraic equations, and therefore can be obtained at any given time by an algebraic operation of the other quaternions.

The system for the balance controller in the controllability canonical form is thus reduced to

$$\begin{aligned} \dot{\boldsymbol{x}}_q &= \tilde{\boldsymbol{f}}(\boldsymbol{x}_q) + \tilde{\boldsymbol{g}}(\boldsymbol{x}_q)\boldsymbol{u} \\ \boldsymbol{y}_q &= \tilde{\boldsymbol{h}}(\boldsymbol{x}_q) \end{aligned} \tag{8.3}$$

where the state vector is now a six-dimensional vector

$$\boldsymbol{x}_q = \begin{bmatrix} \chi_{\vec{q}} \\ \dot{\chi}_{\vec{q}} \end{bmatrix} = \begin{bmatrix} {}^{K}_{B}\vec{\boldsymbol{q}} \\ {}^{K}_{B}\dot{\vec{\boldsymbol{q}}} \end{bmatrix} \rightarrow \dot{\boldsymbol{x}}_q = \begin{bmatrix} \dot{\chi}_{\vec{q}} \\ \ddot{\chi}_{\vec{q}} \end{bmatrix} = \begin{bmatrix} {}^{K}_{B}\dot{\vec{\boldsymbol{q}}} \\ {}^{K}_{B}\ddot{\vec{\boldsymbol{q}}} \end{bmatrix} \tag{8.4}$$

and so is the f vector

$$\tilde{\boldsymbol{f}}(\boldsymbol{x}_q) = \begin{bmatrix} {}^{K}_{B}\dot{\boldsymbol{q}} \\ \boldsymbol{f}_q({}^{K}_{B}\boldsymbol{q}, {}^{K}_{B}\dot{\boldsymbol{q}}) \end{bmatrix} \tag{8.5}$$

The function $\boldsymbol{g}$ becomes a 6,3-matrix given by,

$$\tilde{\boldsymbol{g}}(\boldsymbol{x}_q) = \begin{bmatrix} \boldsymbol{0}_{4\times 3} \\ \boldsymbol{g}_q({}^{K}_{B}\boldsymbol{q}) \end{bmatrix} \tag{8.6}$$

An orientation FLC design involves transforming the output state-space from six variables to three since it controls the three degrees of freedom of the 3D orientation. This output space reduction leads to having three outputs and three control inputs, which converts the system into a fully actuated MIMO system. It enables to inverse the dynamics so as to cancel the couplings and apply SISO control. The output vector of the system is formed of three outputs ($m = 3$), which are chosen to be the components of the vector part of the quaternion

$$\boldsymbol{y}_q = \boldsymbol{h}(\boldsymbol{x}_q) = \begin{bmatrix} h_1(\boldsymbol{x}_q) \\ h_2(\boldsymbol{x}_q) \\ h_3(\boldsymbol{x}_q) \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{8.7}$$

Since the unit quaternion satisfies the unit norm constraint, the variable q_0 can be solved by applying Equation E.18.

8.2.1 Feedback linearization

In the FL technique, the Lie derivative and the relative degree of the non-linear system play an essential role.

In an LTI system, the relative degree is the difference between the number of poles (i.e., degree of the transfer function's denominator polynomial) and the number of zeros (i.e., degree of its numerator polynomial). It can also be derived as the number of times the output has to be differentiated before the input appears in its expression [42]. For this system, each output needs to be derived twice to achieve quaternion acceleration and therefore, the control input to appear. For a MIMO system, the relative degree is called vector relative degree and is equivalent to the sum of each output relative degree

$$r = \sum_{i=1}^{m} r_i = r_1 + r_2 + r_3 = 2 + 2 + 2 = 6 \tag{8.8}$$

The Lie derivatives can be obtained by differentiating y from the controllability canonical form from (8.1) with respect to time

$$\dot{y} = \frac{\partial h}{\partial x} f(x) + \frac{\partial h}{\partial x} g(x) u = L_f h(x) + L_g h(x) u \tag{8.9}$$

where $L_f h(x)$ and $L_g h(x)$ are defined as the Lie derivatives of h with respect to f and g, respectively. Defining x_0 as the equilibrium point where $f(x)$ becomes null, then the Lie derivative of h with respect to g is bounded away from zero for all x [42]. Therefore, the state feedback law

$$u = \frac{1}{L_g L_f^{r-1} h(x)} \left[v - L_f^r h(x) \right] \tag{8.10}$$

yielding a linear first order system from the supplementary input v to the initial output of the system, y. Thus, there exists a state feedback law, similar to (8.10), that makes the nonlinear system in (8.9) linear [42].

The control law yields to $n - r$ states of the non-linear system that are unobservable through the state feedback. In the systems where $n > r$, only the input-output map can be linearised while the state equation is partially linearised, leading to having zero dynamics. It is called partial feedback linearisation or input-output linearisation. On the other hand, if $n = r$ the state equation is completely linearised through the input transformation u and the algorithm is categorised as fully feedback linearisation or input-output linearisation [41] [43]. Since the dimension of the current system evaluated in this project is six and equal to the vector relative degree, the non-linear system can be fully feedback linearised, and there are no zero dynamics.

Expressing u in terms of the Lie derivatives is very convenient to see how the input-output map through (8.9) and (8.10) is reduced to

$$y^{(r)} = v \tag{8.11}$$

which is linear and can be written as a chain of integrators. It shows that the output function $h(x)$ must be derived r times until it obtains an expression relating the input u with the output y.

$$\begin{bmatrix} y^{(r_1)} \\ y^{(r_2)} \\ y^{(r_3)} \end{bmatrix} = \boldsymbol{D}(\boldsymbol{x}) + \boldsymbol{E}(\boldsymbol{x})\boldsymbol{u} \tag{8.12}$$

where $\boldsymbol{E}(x)$ is called the decoupling matrix. FL must satisfy $L_g L_f^{r-1} h(x_o) \neq 0$, i.e. the control input does only appears at the Lie derivative with respect to f of r relative degree. The equivalent of the condition for a MIMO system implies non-singularity of the decoupling matrix $\boldsymbol{E}(x)$, since it is required for $\boldsymbol{E}(x)$ to be invertible in order to apply inverse dynamics for decoupling.

$$\boldsymbol{E}(\boldsymbol{x}) = \begin{bmatrix} L_{g_1} L_f^{r_1-1} h_1(\boldsymbol{x}) & \cdots L_{g_3} L_f^{r_1-1} h_1(\boldsymbol{x}) \\ \vdots & \ddots \\ L_{g_1} L_f^{r_3-1} h_3(\boldsymbol{x}) & \cdots L_{g_3} L_f^{r_3-1} h_3(\boldsymbol{x}) \end{bmatrix} \tag{8.13}$$

where $\boldsymbol{E}(x)$ and $\boldsymbol{D}(x)$ are computed through the Lie derivatives.

$$\boldsymbol{D}(\boldsymbol{x}) = \begin{bmatrix} L_f^{r_1} h_1(\boldsymbol{x}) \\ L_f^{r_2} h_2(\boldsymbol{x}) \\ L_f^{r_3} h_3(\boldsymbol{x}) \end{bmatrix} \tag{8.14}$$

The derivative of h along the trajectories of the system $\dot{x} = f(x)$ [43] is given by

$$L_f h_i(\boldsymbol{x}) = \sum_{j=1}^{6} \left(\frac{\partial h_i}{\partial x_j} f_j(\boldsymbol{x}) \right) \tag{8.15}$$

and leads to the following results

$$L_f h_1(\boldsymbol{x}) = \left(\frac{\partial h_1}{\partial q_1} \frac{\partial h_1}{\partial q_2} \ldots \frac{\partial h_1}{\partial \dot{q}_3} \right) \begin{pmatrix} f_1 \\ f_2 \\ \ldots \\ f_6 \end{pmatrix} = (1\,0\,0\,0\,0\,0) \begin{pmatrix} f_1 \\ f_2 \\ \ldots \\ f_6 \end{pmatrix} = f_1 = \dot{q}_1 \tag{8.16}$$

$$L_f h_2(\boldsymbol{x}) = f_2 = \dot{q}_2 \tag{8.17}$$

$$L_f h_3(\boldsymbol{x}) = f_3 = \dot{q}_3 \tag{8.18}$$

The matrix $\boldsymbol{D}(x)$ in Equation 8.14 is computed through the direction of the vector field from the previous results along f

$$L_f^{r_i} h_i(\boldsymbol{x}) = L_f \left(L_f^{r_i - 1} h_i(\boldsymbol{x}) \right) = L_f^2 h_i(\boldsymbol{x}) = \sum_{j=1}^{6} \left(\frac{\partial L_f^{r_i - 1} h_i}{\partial x_j} f_j(\boldsymbol{x}) \right) \tag{8.19}$$

which yields to

$$L_f^2 h_1(\boldsymbol{x}) = \left(\frac{\partial \dot{q}_1}{\partial q_1} \frac{\partial \dot{q}_1}{\partial q_2} ... \frac{\partial \dot{q}_1}{\partial \dot{q}_3} \right) \begin{pmatrix} f_1 \\ f_2 \\ ... \\ f_6 \end{pmatrix} = (0\,0\,0\,1\,0\,0) \begin{pmatrix} f_1 \\ f_2 \\ ... \\ f_6 \end{pmatrix} = f_4 \tag{8.20}$$

$$L_f^2 h_2(\boldsymbol{x}) = f_5 \tag{8.21}$$

$$L_f^2 h_3(\boldsymbol{x}) = f_6 \tag{8.22}$$

The matrix $\boldsymbol{E}(x)$ is a 3,3-matrix composed by the Lie derivative of the vector field obtained in (8.18) along g, which is defined as

$$L_{g_k} L_f^{r_i - 1} h_i(\boldsymbol{x}) = \sum_{j=1}^{6} \left(\frac{\partial L_f^{r_i - 1} h_i}{\partial x_j} g_{j,k}(\boldsymbol{x}) \right) \tag{8.23}$$

where k indicates each of the three columns of g. Substituting the previous values from (8.18) and applying the gradients yield to

$$L_{g_1} L_f h_1(\boldsymbol{x}) = \left(\frac{\partial \dot{q}_1}{\partial q_1} \frac{\partial \dot{q}_1}{\partial q_2} ... \frac{\partial \dot{q}_1}{\partial \dot{q}_3} \right) \begin{pmatrix} g_{1,1} \\ g_{2,1} \\ ... \\ g_{6,1} \end{pmatrix} = (0\,0\,0\,1\,0\,0) \begin{pmatrix} 0 \\ 0 \\ 0 \\ g_{4,1} \\ g_{5,1} \\ g_{6,1} \end{pmatrix} = g_{4,1} \tag{8.24}$$

which forms the following $\boldsymbol{E}(x)$ matrix defined in (8.13).

$$\boldsymbol{E}(\boldsymbol{x}) = \begin{bmatrix} \tilde{g}_{6,1} & \tilde{g}_{6,2} & \tilde{g}_{6,3} \\ \tilde{g}_{7,1} & \tilde{g}_{7,2} & \tilde{g}_{7,3} \\ \tilde{g}_{8,1} & \tilde{g}_{8,2} & \tilde{g}_{8,3} \end{bmatrix} \tag{8.25}$$

There is no value of $\boldsymbol{E}(x)$ making the determinant zero. Hence, the non-singularity requirement is satisfied, and the output linearisation and decoupling problem is solvable for the non-linear system by isolating from (8.12) the non-linear control input.

The control law u in matrix form is given by

$$\boldsymbol{u} = \boldsymbol{E}^{-1}(\boldsymbol{x})(\boldsymbol{v} - \boldsymbol{D}(\boldsymbol{x})) \tag{8.26}$$

Applying this control law, the non-linearities are cancelled since the input-output is decoupled and linearised.

8.2.2 Control of the decoupled linearised system

The design of the linear control law v, as well as the transformation from the x-state space to z-state space, are to be defined in order to compute u in (8.26).

It is desired to design a new set of n independent coordinates named $\boldsymbol{z}$, so the mapping from x-state space to z-space transforms the non-linear system defined in (8.3) into a system which is input-output linearized and controllable. The first r coordinates are denoted by $\boldsymbol{\xi}$ and the last $n - r$ coordinates that create the zero dynamics and are named η. The ball-robot has no zero dynamics since $n = r = 6$, and therefore the new set of coordinates z are collected in a 6-dimensional vector denoted by $\boldsymbol{\xi}$.

The mapping from z-space to x-space must be invertible in order to be unique. The coordinates are chosen following the SISO description in [44] but extended for a MIMO system. Therefore, for each output i with relative degree r_i the coordinates are defined as

$$\boldsymbol{\xi} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \\ \dots \\ z_{r_i}^{(i)} \end{bmatrix} = \begin{bmatrix} h_i(x) \\ L_f h_i(x) \\ \dots \\ L_f^{r_i - 1} h_i(x) \end{bmatrix} \tag{8.27}$$

The derivative of the states are derived applying the Lie derivative. The last derivative of the state $z_{r_i+1}^{(i)}$ corresponds to the linear control law v of the corresponding

output i.

$$\dot{\boldsymbol{\xi}}_i = \begin{bmatrix} \dot{z}_1^{(i)} \\ \dot{z}_2^{(i)} \\ ... \\ \dot{z}_{r_i}^{(i)} \end{bmatrix} = \begin{bmatrix} z_2^{(i)} \\ z_3^{(i)} \\ ... \\ z_{r_i+1}^{(i)} \end{bmatrix} = \begin{bmatrix} L_f h_i(x) \\ L_f^2 h_i(x) \\ ... \\ v_i \end{bmatrix} \tag{8.28}$$

The new coordinates for each output is computed and yields to

$$\begin{aligned} \boldsymbol{\xi}_1 = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \end{bmatrix} \quad &\rightarrow \quad \dot{\boldsymbol{\xi}}_1 = \begin{bmatrix} \dot{z}_1^{(1)} \\ \dot{z}_2^{(1)} \end{bmatrix} = \begin{bmatrix} z_2^{(1)} \\ v_1 \end{bmatrix} \\ \boldsymbol{\xi}_2 = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix} \quad &\rightarrow \quad \dot{\boldsymbol{\xi}}_2 = \begin{bmatrix} \dot{z}_1^{(2)} \\ \dot{z}_2^{(2)} \end{bmatrix} = \begin{bmatrix} z_2^{(2)} \\ v_2 \end{bmatrix} \\ \boldsymbol{\xi}_3 = \begin{bmatrix} z_1^{(3)} \\ z_2^{(3)} \end{bmatrix} \quad &\rightarrow \quad \dot{\boldsymbol{\xi}}_3 = \begin{bmatrix} \dot{z}_1^{(3)} \\ \dot{z}_2^{(3)} \end{bmatrix} = \begin{bmatrix} z_2^{(3)} \\ v_3 \end{bmatrix} \end{aligned} \tag{8.29}$$

Since the system is decoupled and linearized after applying LF, the new set of coordinates for each output create a linear system on the form of

$$\begin{aligned} \dot{\boldsymbol{\xi}}_i &= \boldsymbol{A}\xi_i + \boldsymbol{B}\boldsymbol{v}_i \\ \boldsymbol{y}_i &= \boldsymbol{C}\xi_i \end{aligned} \tag{8.30}$$

therefore yielding three linear systems with one control output each to be defined. The control signal is equal to the last state of each output $\dot{z}_{r_i}^{(i)}$ and then transforms the nonlinear system into a linear system. Thus, it stabilizes the system by standard linear techniques and is chosen to be

$$v_i = -K\boldsymbol{\xi}_i \tag{8.31}$$

substituting (8.31) in (8.30) gives

$$\dot{\boldsymbol{\xi}}_i = (\boldsymbol{A} - \boldsymbol{B}K_i)\,\xi_i \tag{8.32}$$

The solution of this equation is given by

$$\boldsymbol{\xi}_i = e^{(\boldsymbol{A}-\boldsymbol{B}K_i)t}\xi_i(0) \tag{8.33}$$

where $\xi_i(0)$ is the initial state caused by external disturbances. The stability and transient response characteristics are determined by the eigenvalues of the matrix

$\boldsymbol{A} - \boldsymbol{B}K_i$. Therefore K_i must be properly chosen to make it an asymptotically stable matrix and be a Hurwitz matrix. These regular poles or eigenvalues need to be placed in the left half of the s plane, to make $\xi_i(t)$ approach zero as t approaches infinity [45].

The three linear systems can be rearranged from (8.29) to be presented in an single state-space system

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{A}\xi + \boldsymbol{B}\boldsymbol{v} \\ \boldsymbol{y} &= \boldsymbol{C}\xi \end{aligned} \tag{8.34}$$

where the states are rearranged as

$$\boldsymbol{\xi} = \begin{bmatrix} z_1^{(1)} \\ z_1^{(2)} \\ z_1^{(3)} \\ z_2^{(1)} \\ z_2^{(2)} \\ z_2^{(3)} \end{bmatrix} = \begin{bmatrix} h_1(\boldsymbol{x}) \\ h_2(\boldsymbol{x}) \\ h_3(\boldsymbol{x}) \\ L_f h_1(\boldsymbol{x}) \\ L_f h_2(\boldsymbol{x}) \\ L_f h_3(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \rightarrow \dot{\boldsymbol{\xi}} = \begin{bmatrix} \dot{z}_1^{(1)} \\ \dot{z}_1^{(2)} \\ \dot{z}_1^{(3)} \\ \dot{z}_2^{(1)} \\ \dot{z}_2^{(2)} \\ \dot{z}_2^{(3)} \end{bmatrix} = \begin{bmatrix} L_f h_1(\boldsymbol{x}) \\ L_f h_2(\boldsymbol{x}) \\ L_f h_3(\boldsymbol{x}) \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{8.35}$$

and the state-space matrices are trivial and given by

$$A = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{8.36}$$

$$\boldsymbol{B} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{I}_{3\times3} \end{bmatrix} \tag{8.37}$$

$$C = [\mathbf{I}_{3\times3}\, \mathbf{0}_{3\times3}] \tag{8.38}$$

The non-linear system response created by u is designed by the control law of the linear system v. For a SISO system it is defined as

$$v = -K_0\, z_1 - K_1\, z_2 - \ ... \ - K_r\, z_r = -k_0\, y - K_1\, \dot{y} - \ ... \ - K_{r-1}\, y^{r-1} \tag{8.39}$$

the second part of the equality shows that the open-loop dynamics is a chain of integrators. For a MIMO system this expression is extended for each output i. As seen in (8.39), each v_i is indeed a chain of r_i integrators in open-loop and describes the relationship between the output and the new input v [43]. For the ball-robot it

yields to

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -K_1 z_1^{(1)} - K_{d1} z_2^{(1)} \\ -K_2 z_1^{(2)} - K_{d2} z_2^{(2)} \\ -K_3 z_1^{(3)} - K_{d3} z_2^{(3)} \end{bmatrix} = \begin{bmatrix} -K_1 q_1 - K_{d1} \dot{q}_1 \\ -K_2 q_2 - K_{d2} \dot{q}_2 \\ -K_3 q_3 - K_{d3} \dot{q}_3 \end{bmatrix} \tag{8.40}$$

The control law drive the state variables to zero, and follows the response characteristics determined by the closed-loop poles set by v. For a SISO system it leads to the closed-loop output dynamics [43]

$$z_{r+1} + K_r z_r + \ldots + K_0 z_1 = 0 \tag{8.41}$$

with K_i chosen so that the polynomial in the left side of (8.41) has all its roots strictly in the left-half complex plane, it leads to the exponential stable dynamics (8.41). This implies that $x(t) \to 0$. Isolating from (8.40) yields to the ball-robot closed-loop dynamics

$$\begin{aligned} K_1 q_1 + K_{d1} \dot{q}_1 + \ddot{q}_1 &= 0 \\ K_2 q_2 + K_{d2} \dot{q}_2 + \ddot{q}_2 &= 0 \\ K_3 q_3 + K_{d3} \dot{q}_3 + \ddot{q}_3 &= 0 \end{aligned} \tag{8.42}$$

It can be seen that no approximation arising from the linearisation has been performed, the only linearisation by first feedback has been applied in order to make the system linear, but this linearisation did not include any approximation [46].

8.2.3 Zero-reference with non-zero initial condition results

The FLC with zero-reference configuration is implemented and simulated to test the convergence of the states. The initial orientations of the ball-robot are given by $\phi = 5^\circ$, $\theta = 5^\circ$ and $\psi = 0^\circ$, while the rest of the states are initialized to zero.

The convergence of the FLC to the zero-value is tested in Figure 8.3 and provides zero steady-state error. With a more aggressive gain in the controller, it would be possible to achieve faster reaching time if needed.

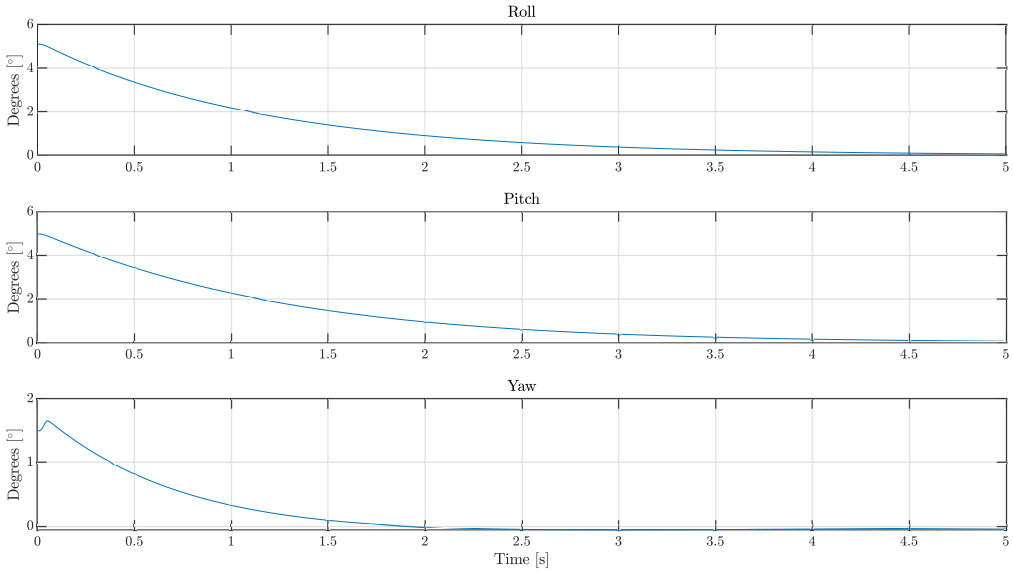


Figure 8.3: Zero reference convergence test of the FLC controller.

To make the ball-robot follow an orientation reference, the tracking error needs to be included in the control scheme.

8.2.4 Reference tracking

The idea of applying FLC to under-actuated mechanical systems is not new, but most of the work has been devoted to the stabilisation problem. There is a lack of literature dealing with how to address the reference tracking in the FLC. Mainly some authors are defining the algorithm for a zero reference and a SISO system. Other sources do not reach an agreement on whether to apply the z-transformation before computing the error or afterwards. Moreover, since the balance controller belongs to a cascade control, it is indeed relevant in this case to consider how to deal with the velocity controller dynamics in the balance controller computation [40] [43].

The goal is to drive the error of the system states to converge towards zero at a finite time. The reference tracking is implemented, as shown in Figure 8.4.

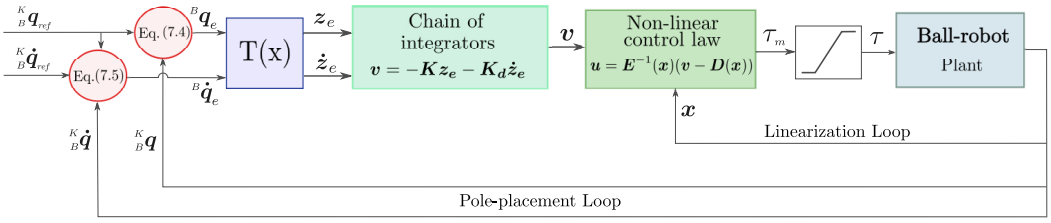


Figure 8.4: Schematic of the reference tracking Input-State Feedback Linearization Controller.

The computation of the matrices $\boldsymbol{E}(x)$ and $\boldsymbol{D}(x)$ is done using the values of the states, while the z-transformation and the linear controller takes the error of the states. For this purpose, the control law from (8.39) needs to be redefined in terms of the error of the states.

$$\begin{aligned} v &= -k_0\,(z_{ref,0} - z_0) - K_1\,(z_{ref,1} - z_1) - \ ... \ - K_{r-1}\,(z_{ref,r-1} - z_{r_1}) = \\ &= z_{ref,r+1} - K_0\,e - K_1\,\dot{e} - \ ... \ - K_{r-1}\,e^{r-1} \end{aligned} \tag{8.43}$$

In the ball-robot case, the states error corresponds to the quaternion error and its derivative, and due to quaternions algebra these errors need to be computed as explained in Section 7.4 and defined in Equation 7.4.

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -K_1\,e_1^{(1)} - K_{d1}\,\dot{e}_2^{(1)} \\ -K_2\,e_1^{(2)} - K_{d2}\,\dot{e}_2^{(2)} \\ -K_3\,e_1^{(3)} - K_{d3}\,\dot{e}_2^{(3)} \end{bmatrix} = \begin{bmatrix} \ddot{q}_{ref,1} - K_1\,{}^{B}q_{1_e} - K_{d1}\,{}^{B}\dot{q}_{1_e} \\ \ddot{q}_{ref,2} - K_2\,{}^{B}q_{2_e} - K_{d2}\,{}^{B}\dot{q}_{2_e} \\ \ddot{q}_{ref,3} - K_3\,{}^{B}q_{3_e} - K_{d3}\,{}^{B}\dot{q}_{3_e} \end{bmatrix} \tag{8.44}$$

The reference acceleration is assumed to be zero in the implementation of the control law v in (8.44), since it is a common assumption for control due to its small value. This chain of integrators yields to the following closed-loop dynamics

$$\begin{aligned} K_1\,{}^{B}q_{1_e} + K_{d1}\,{}^{B}\dot{q}_{1_e} + \ddot{q}_{1_e} &= 0 \\ K_2\,{}^{B}q_{2_e} + K_{d2}\,{}^{B}\dot{q}_{2_e} + \ddot{q}_{2_e} &= 0 \\ K_3\,{}^{B}q_{3_e} + K_{d3}\,{}^{B}\dot{q}_{3_e} + \ddot{q}_{3_e} &= 0 \end{aligned} \tag{8.45}$$

which provides an exponentially stable error dynamics if K_i are chosen to be positive. Therefore, if initially ${}^{B}q_{i_e}(0) = {}^{B}\dot{q}_{i_e}(0) = 0$, then ${}^{B}q_{i_e}(t) = 0$ for $t \geq 0$, i.e., perfect tracking is achieved; otherwise, ${}^{B}q_{i_e}(t)$ converges to zero exponentially.

8.2.5 Feedback linearisation reference tracking results

The FLC is simulated to track the reference signal and tested using a sine wave reference. The sine wave reference consists of a roll movement of $0.5Hz$ with 3° of

amplitude, which is transformed into a quaternion reference to fed it to the FLC. The periodical roll movement is tracked in a precise way by the controller as seen in Figure 8.5, whereas there is a visible oscillatory error in the third and fourth quaternion components, q_3 and q_4 respectively; furthermore, q_3 has a small offset with respect to the reference. The errors could be explained by a combination of the tracking dynamics as well as the condition to satisfy the unitary quaternion norm.

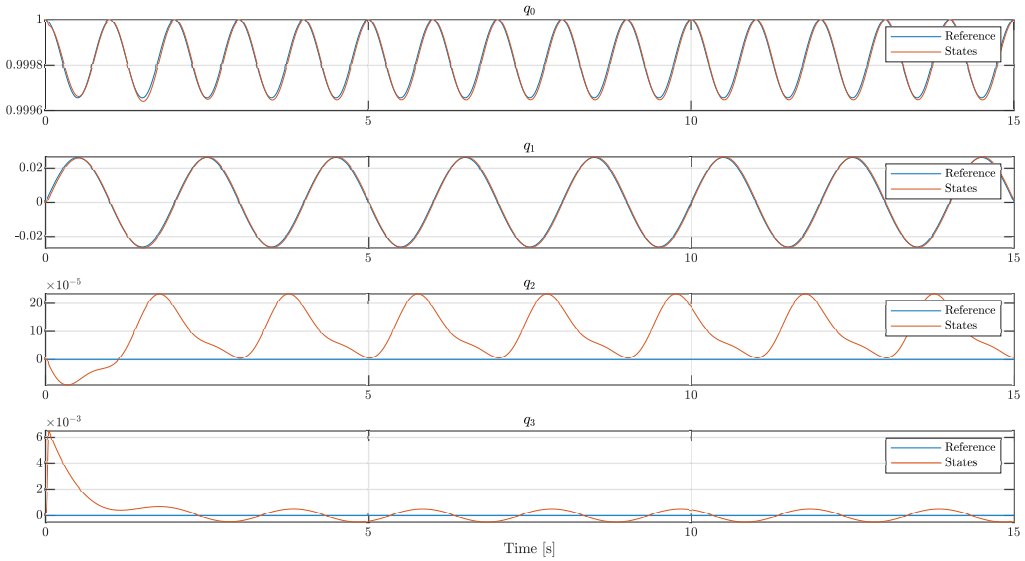


Figure 8.5: Quaternion results with a reference signal of a sine wave in the roll angle.

Despite the deviations in tracking, the magnitude of the error is not significant, with a maximum of 0.25° of error for the pitch direction and less than 0.1° for the yaw, fulfilling the desired performance specifications.

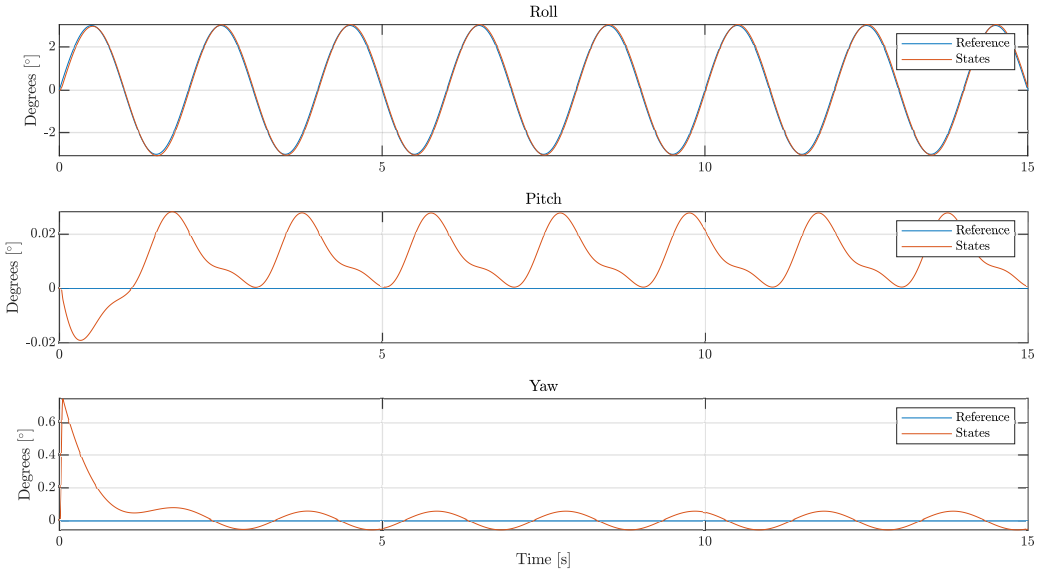


Figure 8.6: Euler-angles results with a reference signal of a sine wave in the roll angle.

In Figure 8.7 is shown the torque signals commanded to the motors to track the sine wave reference. It is seen that the control input behaves smoothly and does not present chattering.

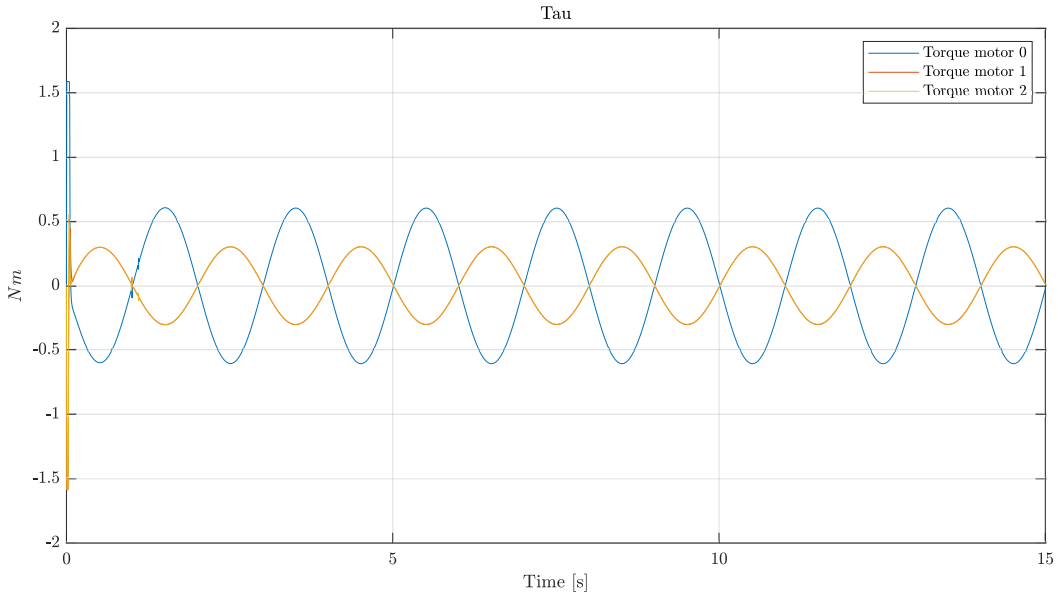


Figure 8.7: Motor torques of the system with velocity controller, balance controller under a sine wave signal.

8.3 Performance comparison with previous version

The system developed by [6] which implements a Sliding mode controller for the balance stabilization is compared with the actual system using Feedback Linearization control.

In Figure 8.8, the results of the tracking error of both systems is presented in Euler-angles. It is seen that the error magnitude of the roll angle is quite similar in both systems, whereas in the pitch the V2019 provides an error with one order magnitude lower than the V2020 proposed in the current thesis. The yaw performance in the V2019 converges adequately to zero while in the V2020 has small oscillations.

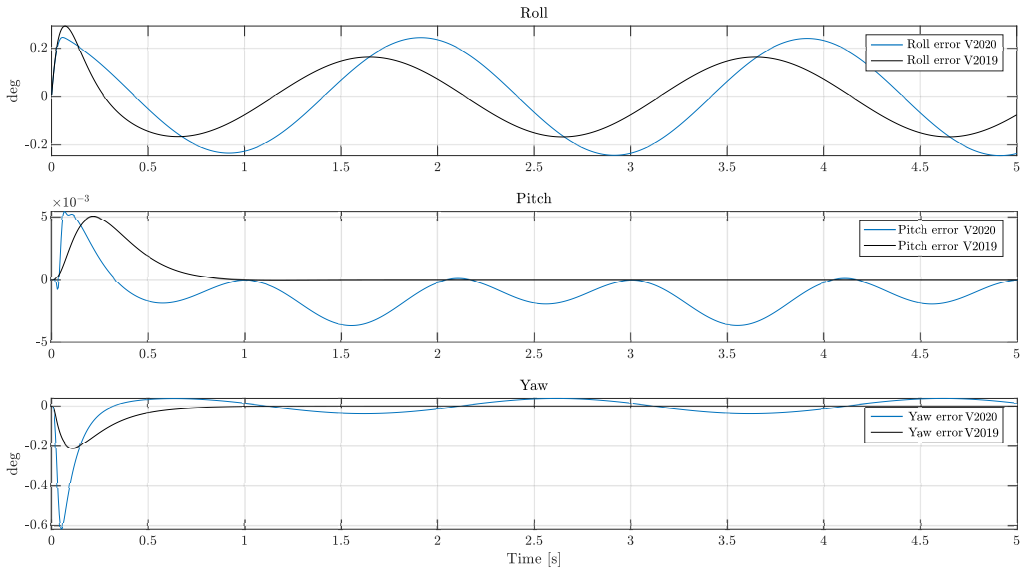


Figure 8.8: Euler angle error comparison feedback linearization vs sliding mode.

Finally, comparing the torque signals commanded to the motors it can be asserted that the behaviour of the controllers is quite similar when facing a sine wave reference.

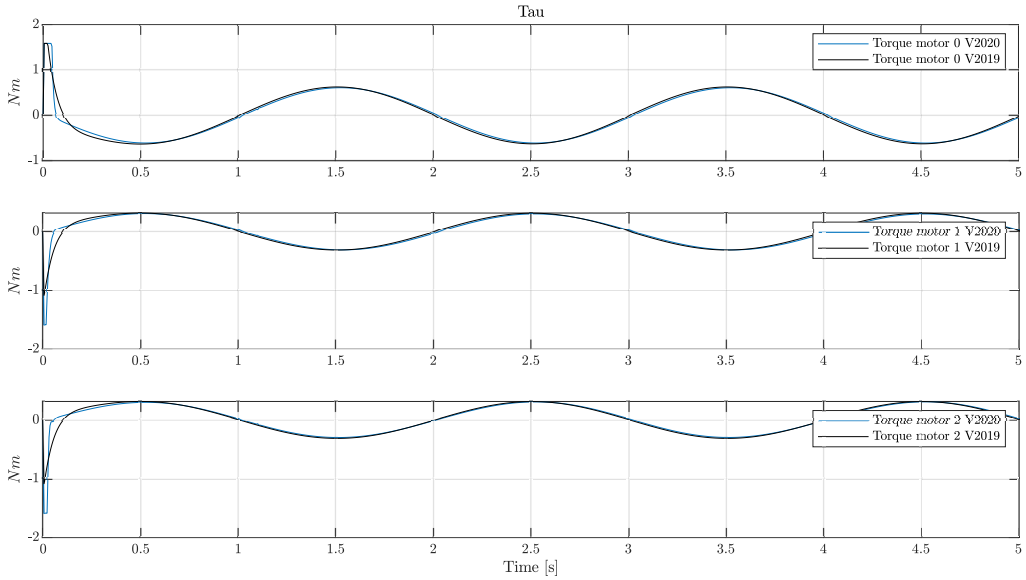


Figure 8.9: Torques comparison between the feedback linearization and the sliding mode.

8.4 FLC robustness test

The previous results obtained by the FLC validate the performance under the expected specifications for the nominal system and without disturbances. However, it is common to find deviations between the model and the real device, hence, it is necessary to evaluate the performance of the controller under disturbances and parametric deviations. To evaluate it, three experiments are performed: Mass and inertia disturbance, COM miss-alignment and force disturbance.

The FLC does not have implicitly implemented any strategy to reject the disturbances, such as a dynamic control law or an extension of the control law including the disturbance rejection terms. However, the controller is tested against the disturbances expected to be found in the set-up to assess its robustness.

8.4.1 Mass and inertia disturbance

The first experiment is designed to evaluate how a change in mass in the device would influence in the performance of the controller to track the desired signal. The mass is increased by a 10% rate in the model and the inertias are increased by a factor of 3.2. It can be observed in Figure 8.10 how the quaternion components track properly the reference despite the mass and inertia differences. It is also

seen how the change in inertias has a bigger influence in the forth quaternion component, q_3, which is correlated with the influence of the inertia corresponding to the yaw, as seen in Figure 8.11

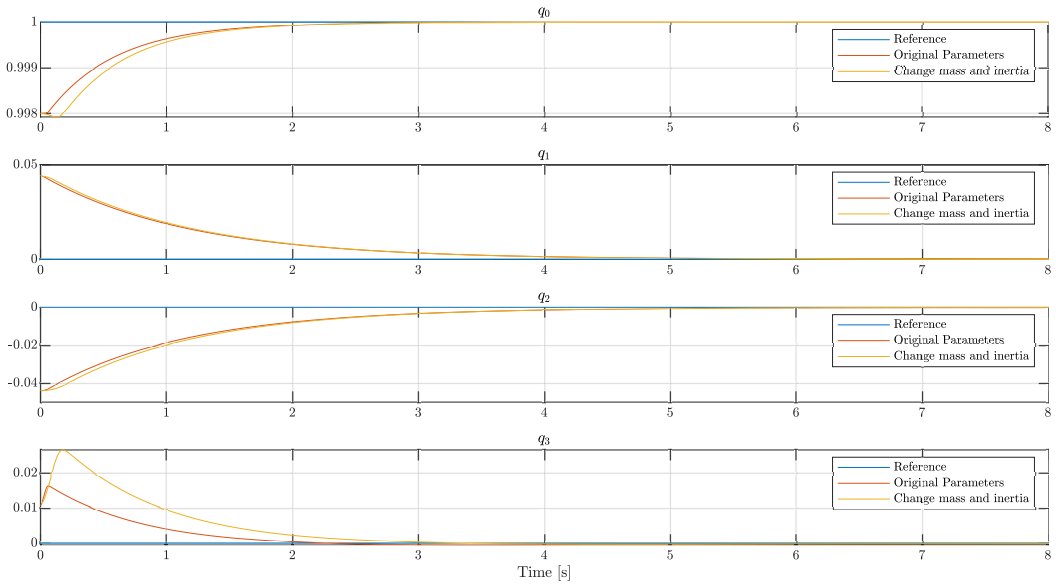


Figure 8.10: Comparison in performance to reach upright position from a $\phi = 5°$ and $\theta = -5°$ inclination between the original model and a model with $+10\%$ change in the mass and 3.2 times the inertia, in quaternion representation.

It is noted the different impacts in the yaw rotation compared with the other two rotations under the effect of mass and inertia changes.

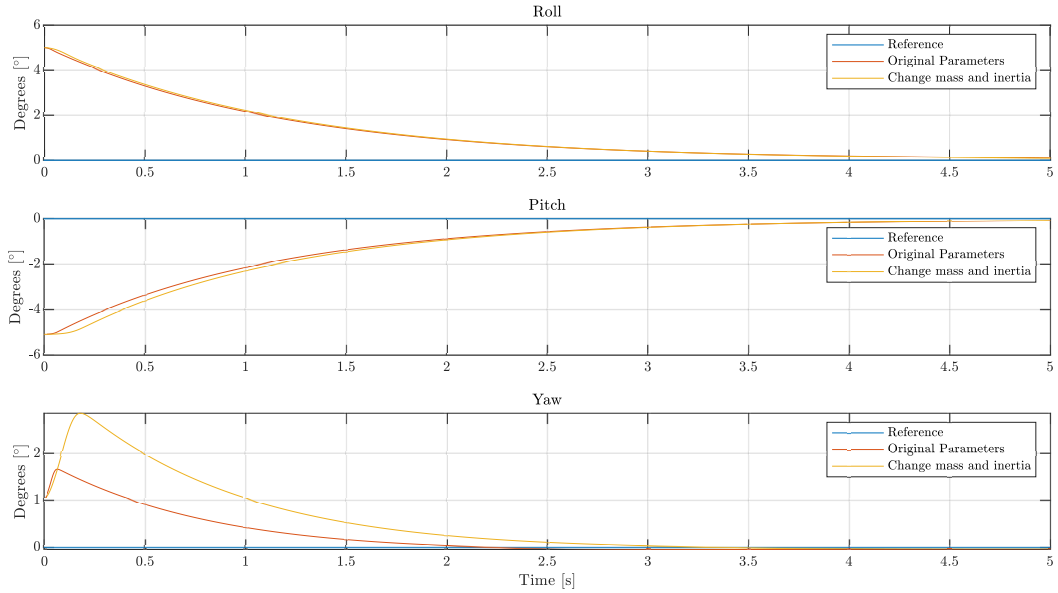


Figure 8.11: Comparison in performance to reach upright position from a $\phi = 5°$ and $\theta = -5°$ inclination between the original model and a model with $+10\%$ change in the mass and 3.2 times the inertia, in Euler-angle representation.

8.4.2 COM miss-alignment

It was discussed previously the effects of the shape-accelerated properties of the ball-robot where a miss-alignment of the COM can cause the runaway effect, mainly, due to miss-calculations over the gravity effect due the inclination. Therefore it is tested the influence and impact in the balance controller with a COM deviated by $[-2\ -3\ +3]\ cm$ in Cartesian reference from the original position express in the inertial frame. This deviation is translated into the ball-robot inertia unbalanced to one side and the COM placed higher than expected. Thus, in Figure 8.12 it is seen how the FLC is not capable of tracking correctly the reference angles.

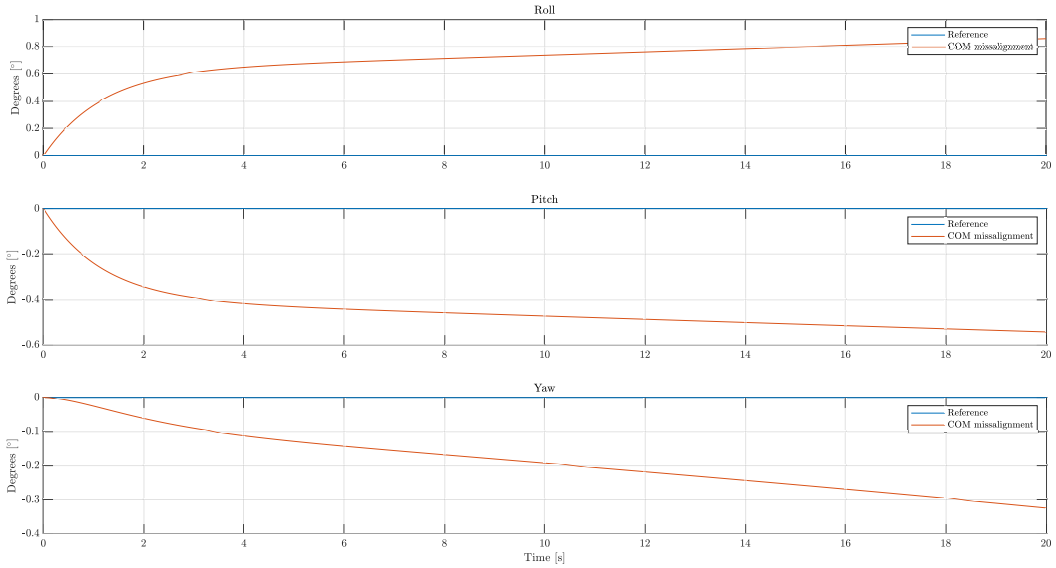


Figure 8.12: Effects of the miss-alignment of the COM causing the run-away effect in the orientation of the ball-robot.

The COM miss-alignment is also reflected in the linear velocity, extension of the wrong orientation of the ball-robot, causing the run-away effect.

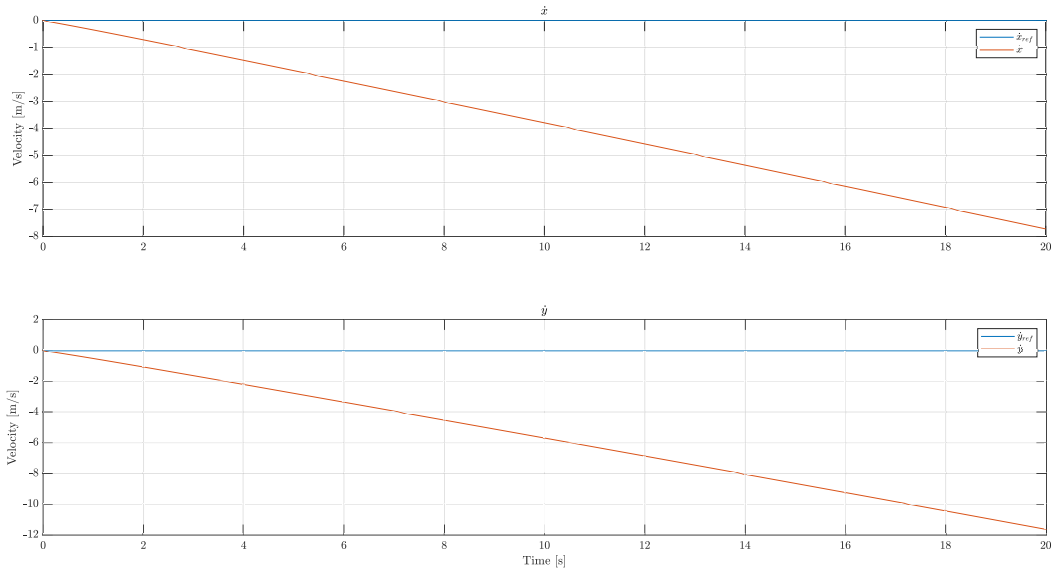


Figure 8.13: Effects of the miss-alignment of the COM causing the run-away effect in the linear velocity of the ball-robot.

The above results confirm the necessity of a velocity controller, capable of correct-

ing the velocities derived of the miss-alignment by compensating it with orientation.

8.4.3 Force disturbance

Finally, it is aimed to assess how susceptible is the balance controller to sudden disturbances caused for example by a collision or a human push. To emulate it, a sudden change of four degrees in the roll angle ϕ is introduced as a disturbance w at the plant output.

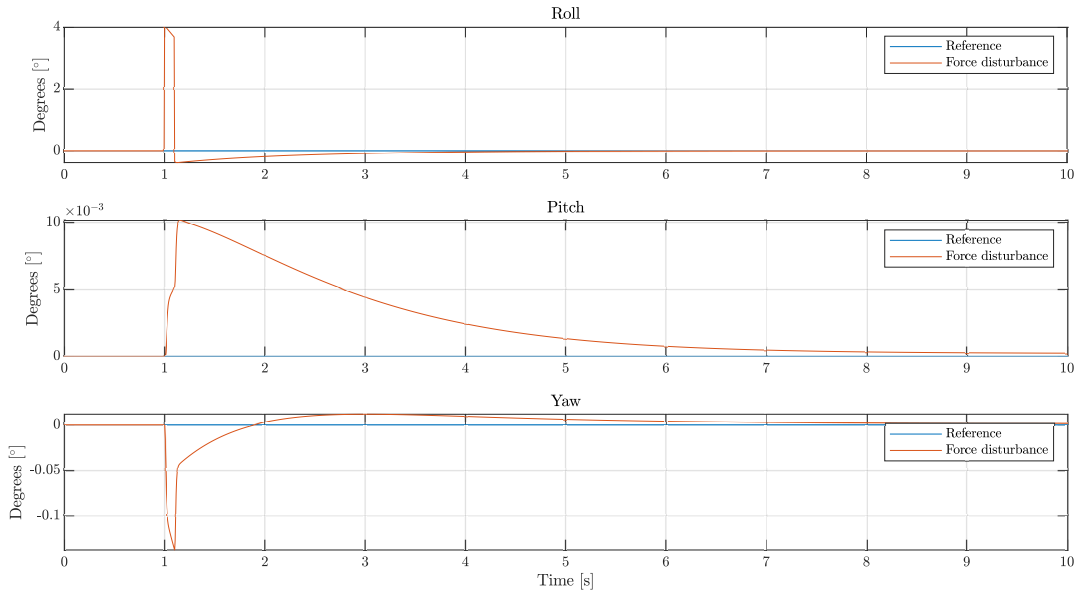


Figure 8.14: Response to a force disturbance expressed in Euler-angles.

It is seen in the Euler-angle representation how the angle is rapidly corrected and converges to the upright position satisfactorily, showing that the algorithm would be robust against this disturbance. This would need to be tested in the real set-up in order to draw further conclusions, since the simulation test is a simplification of a push disturbance and could cause the results to vary.

Chapter 9

Implementation and results

The controller is merged with the rest of the model to simulate the whole ball-robot system. It is evaluated the bandwidth of the system, as well as the performance of the balance controller along with the rest of the set-up, i.e. sensor models, estimators and velocity controller.

9.1 Bandwidth of the system

First of all, it is necessary to evaluate the maximum frequency at which the system is able to perform adequately, or in other words, the bandwidth of the system. To do so, different techniques are at hand, for example, a bode diagram analysis. However, it is decided to perform a chirp test, which consist of exciting the system with a frequency varying signal which keeps increasing in frequency over time. For the test, the roll reference signal is a chirp signal with a frequency starting 0.05 Hz and increases by 0.05 each second, to a limit of 5 Hz at second 100. The rest of the states are initialized at values corresponding to the upright position.

Under the chirp analysis, the roll and pitch response is presented in Figure 9.1. The roll angle tracking is lost at around 25 seconds, while the pitch reflects how the tracking starts to degrade at around the 23 second, meaning that it starts losing control at 1.25 Hz. Therefore, the bandwidth of the system is approximately at 1.25 Hz.

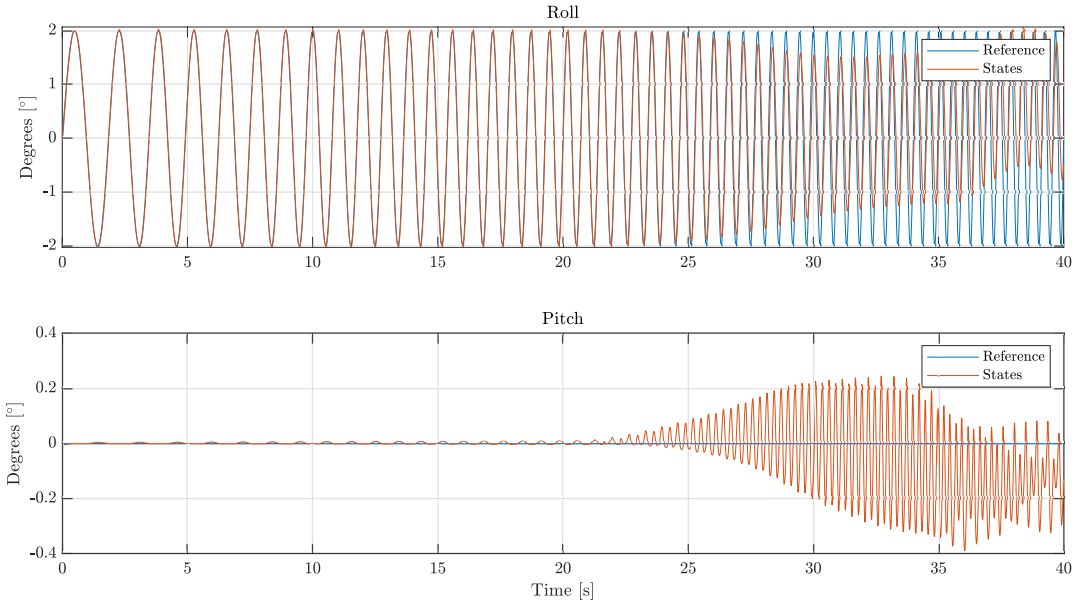


Figure 9.1: Response of the angles of the system under a frequency varying chirp signal exciting the system, represented Euler-angles.

Furthermore, it is desired to analyze the point where the motors reach their saturation point and are not capable of providing a sufficient torque to track the signal. As seen in Figure 9.2, the torque of the motor 0 saturates at a similar time as the angles start losing track, confirming the bandwidth limitation.

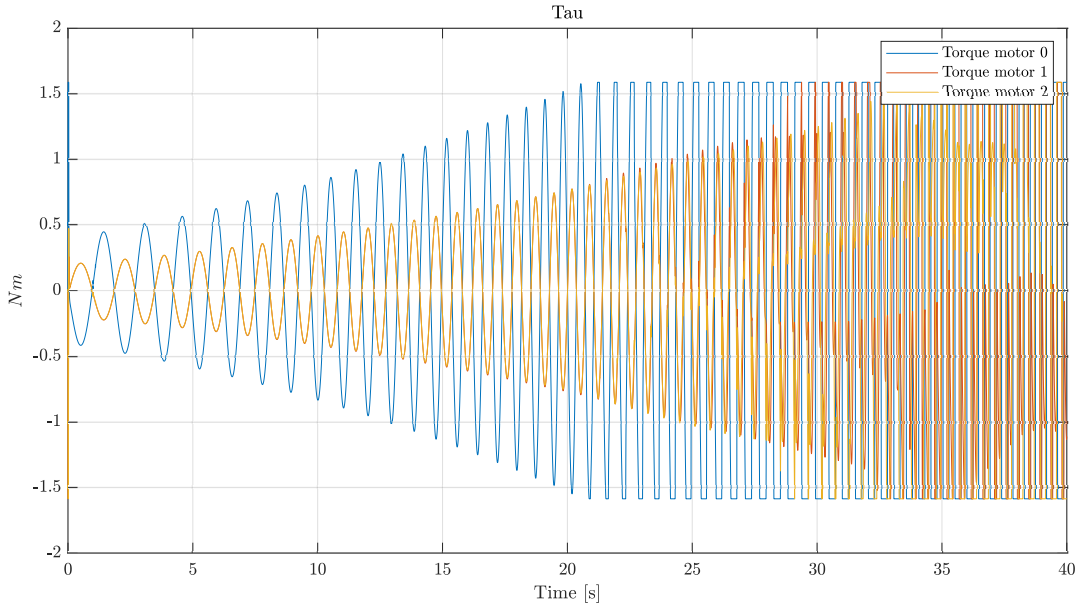


Figure 9.2: Torque response of the motors under a frequency varying chirp signal exciting the system. The torque saturation can be observed.

The synergy of the balance controller with the rest of the system is addressed next, starting with testing the cascade controller, i.e. the balance controller combined with the velocity controller.

9.2 Cascade controller performance

The following tests are conducted to evaluate the merged performance of the velocity controller and the balance controller. Hence, they are an extension of the tests performed in Chapter 7 and Chapter 8. The difference lies in the balance controller reference signal being created by the velocity controller.

The reference linear velocity signal fed to the velocity controller can be seen in Figure 9.3

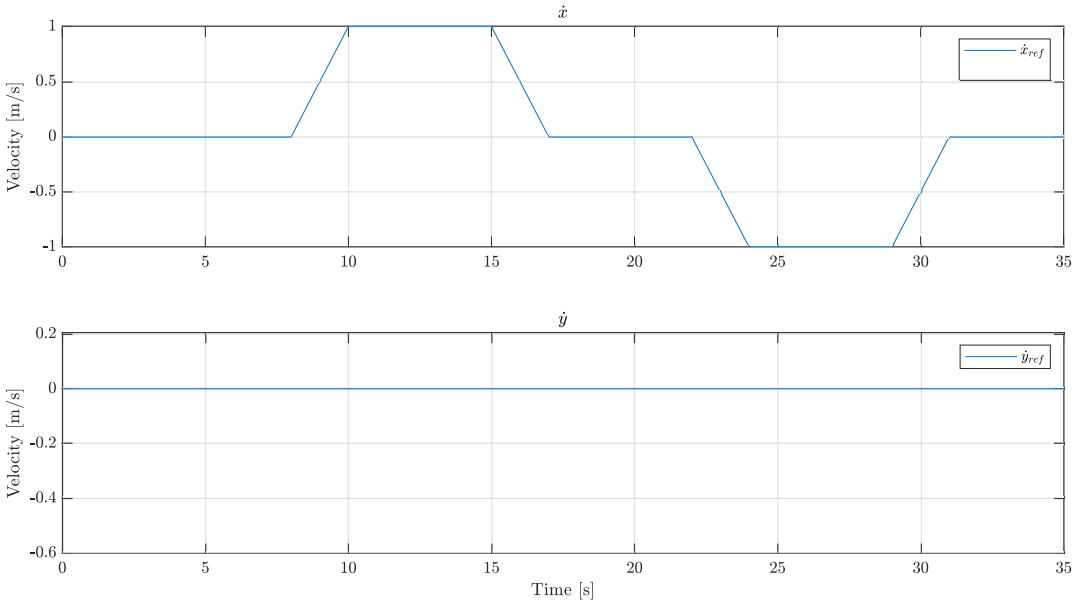


Figure 9.3: Reference signal of the velocity controller.

Figure 9.4 shows the performance of the controllers when tracking the time-varying linear velocity signal shown in the previous figure. It is seen that the system is stable and the controllers are capable of tracking correctly the reference, with no significant error. Nonetheless, it is noted that the reference signal for the fourth quaternion component, q_3, is continuously increasing.

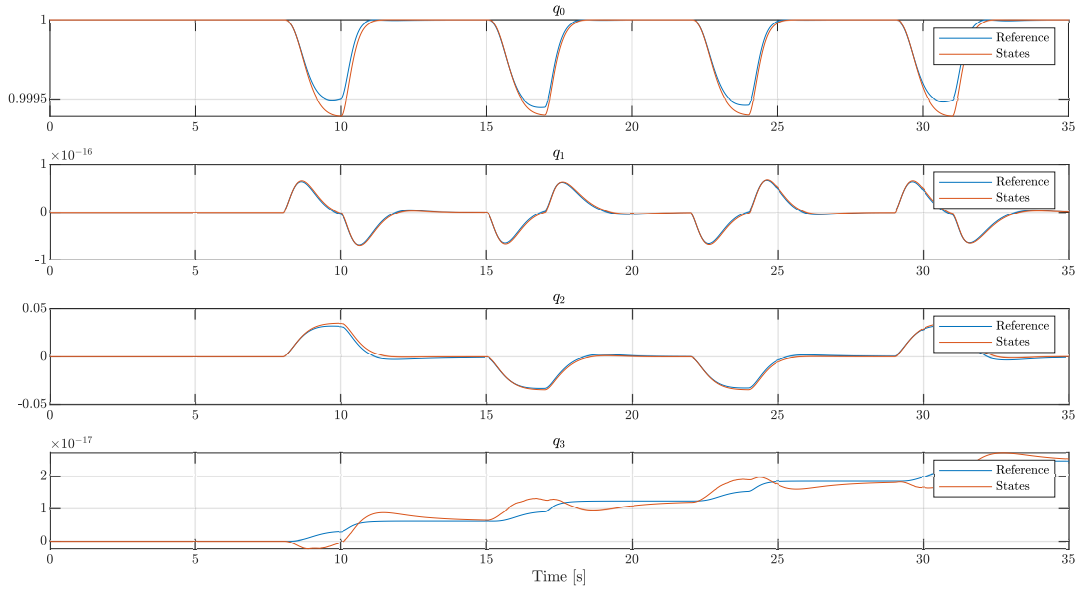


Figure 9.4: Effect of a velocity time-varying reference signal in the inclination of the system, represented in quaternions. The test is carried out using the velocity controller (LQR) and the balance controller (FLC).

It is important to point out the characteristic shape of the orientation reference fed to the balance controller that is created by the velocity controller. In order to follow the ramps imposed as references in linear velocity, the body is required to tilt showing the shape-accelerated characteristic of the ball-robot.

These findings can be confirmed more intuitively by looking at the Euler-angles representation. The tracking is performed properly, and the error in the yaw is indeed negligible.

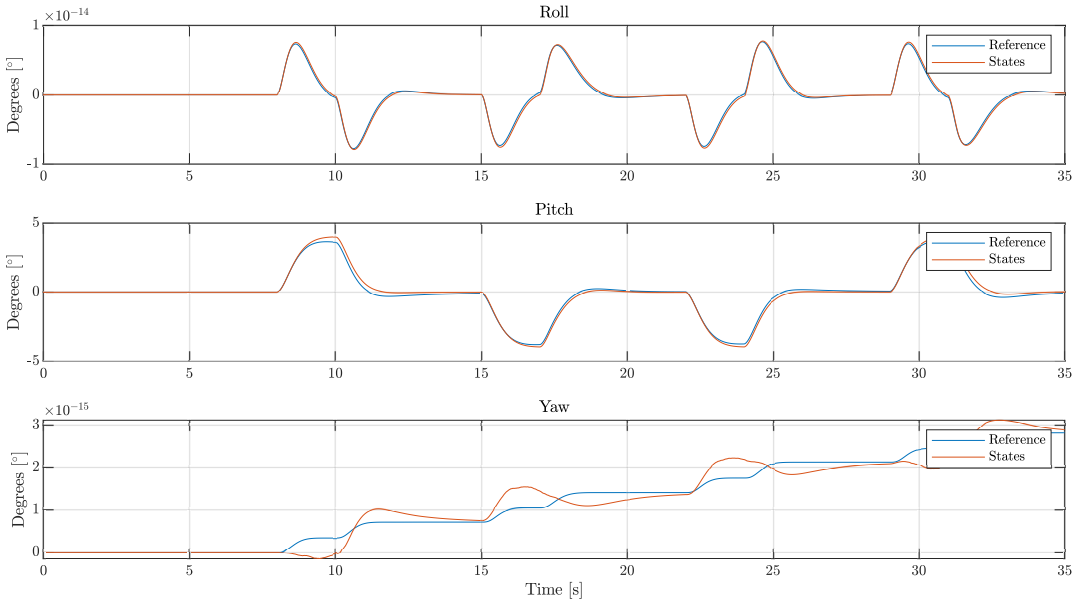


Figure 9.5: Effect of a velocity time-varying reference signal in the inclination of the system, represented in Euler-angles. The test is carried out using the velocity controller (LQR) and the balance controller (FLC).

COM miss-alignment with the cascade controller

Once the velocity controller is tested with the balance controller, it follows a test to assess if a possible COM miss-alignment would be, as intended, corrected by the velocity controller. It is seen in Figure 9.6 how the velocity controller corrects the miss-alignment which in Figure 8.12 was not being corrected. The correction is very accurate in the roll and pitch directions, while the error in the yaw is not properly corrected.

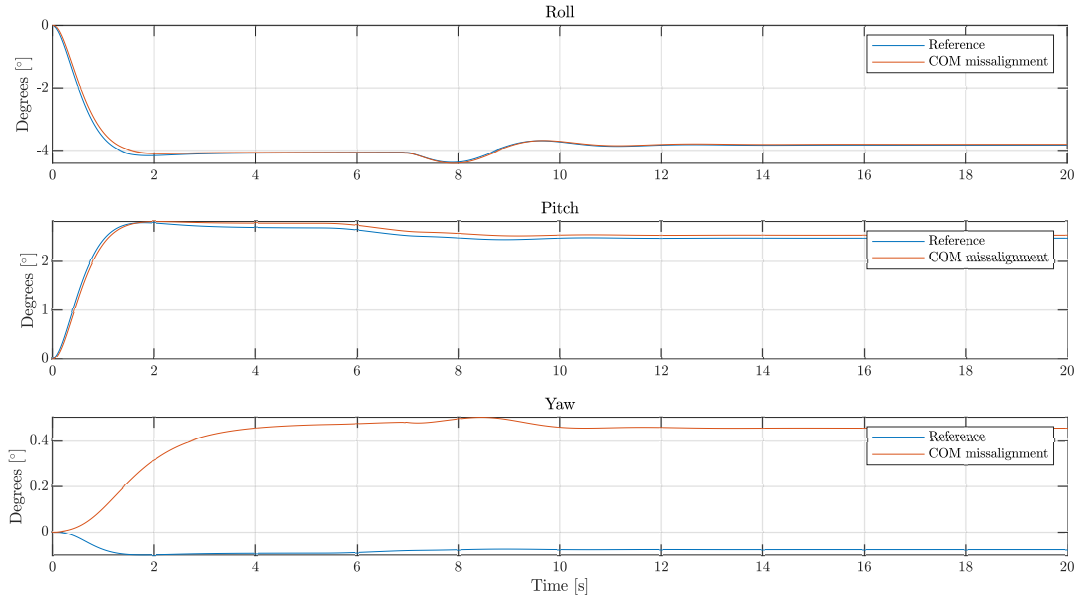


Figure 9.6: Response of the velocity and balance controller against a COM miss-alignment represented in Euler-angles.

The COM was deviated by $[-2\ -3\ +3]\ cm$ in Cartesian coordinates from the original position, expressed in the inertial frame. It is curious to see its impact in the new equilibrium position found by the velocity controller, which now is set at $\phi = -4°$ and $\theta = 2.5°$.

The compensation of the COM miss-alignment can also be observed by looking at the velocity tracking, where the run-away effect has disappeared.

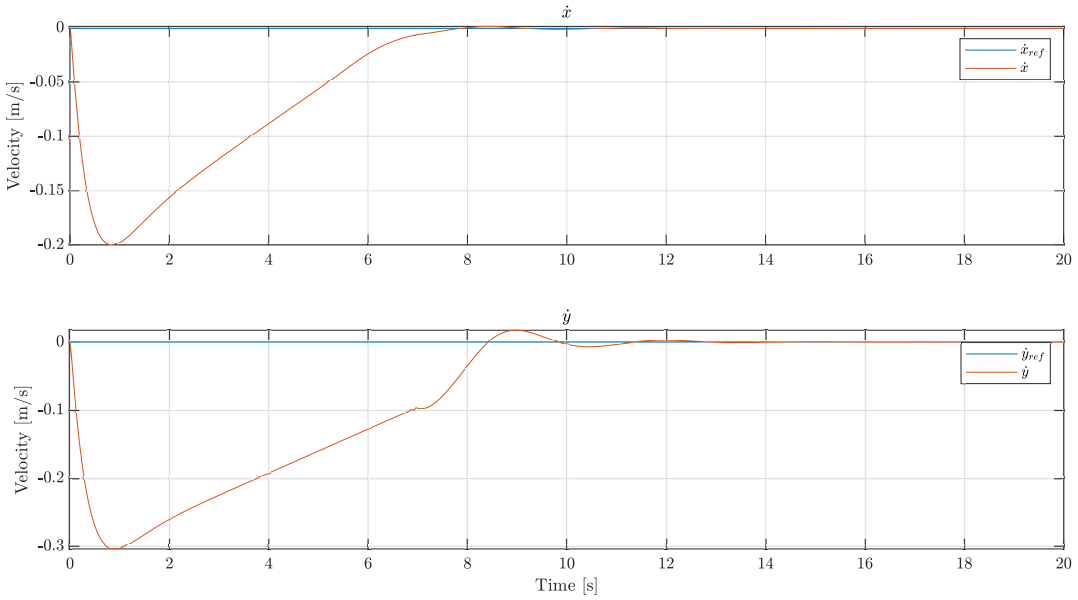


Figure 9.7: Response of the velocity controller against a COM miss-alignment.

9.3 Estimator influence

Finally, the controllers are tested under the influence of noise in the sensors, and thus, the velocity and quaternion estimators are introduced to evaluate its effect.

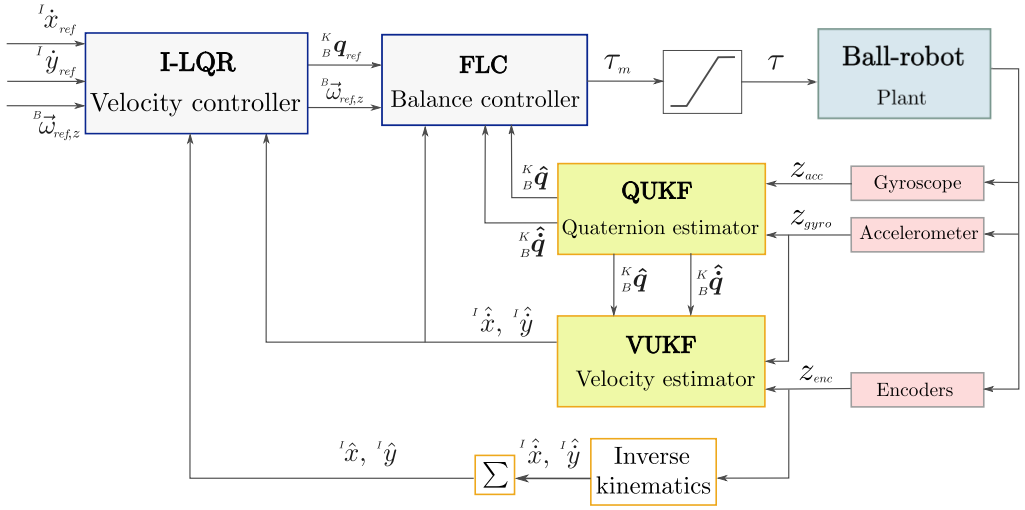


Figure 9.8: Schematic of the simulation for the complete system.

9.3.1 Complete system under a sine wave input test

The complete system is first tested against a sine wave reference signal. The influence of the estimators degrades slightly the tracking performance and generates noisy signals in the third and fourth quaternion components, q_2 and q_3 respectively.

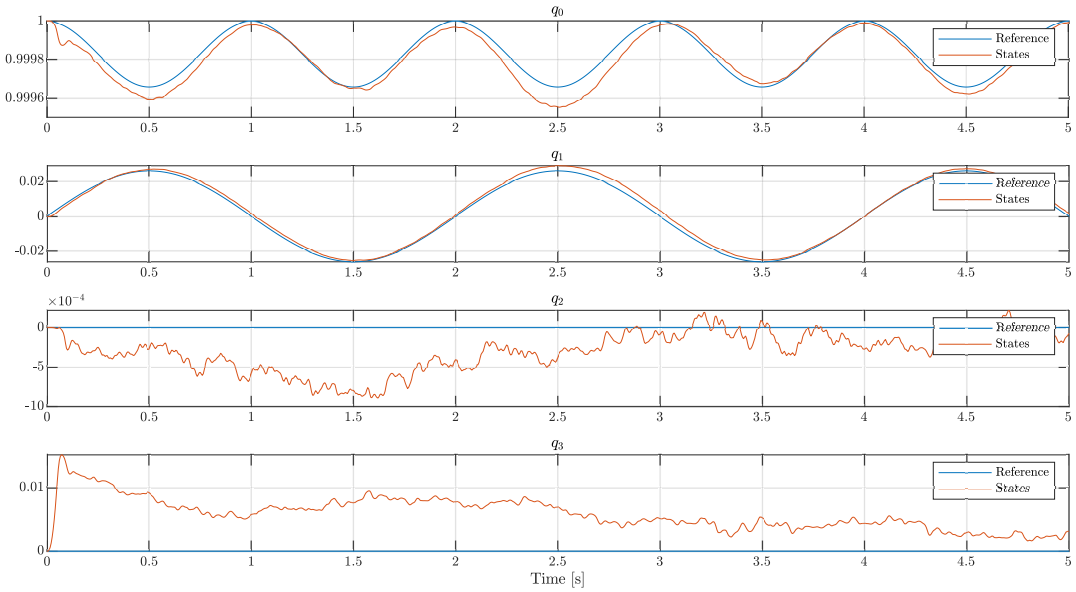


Figure 9.9: Response of the system with velocity controller, balance controller, sensor noise and state estimators under the effect of a sine wave, represented in quaternions.

The quaternion deviations expressed in Euler-angles representation yields satisfactory results that fulfill the performance specifications. The pitch signal converges to an error close to $\pm 0.05°$, whereas the yaw error converges slowly to zero ending up with an offset of $\pm 0.3°$.

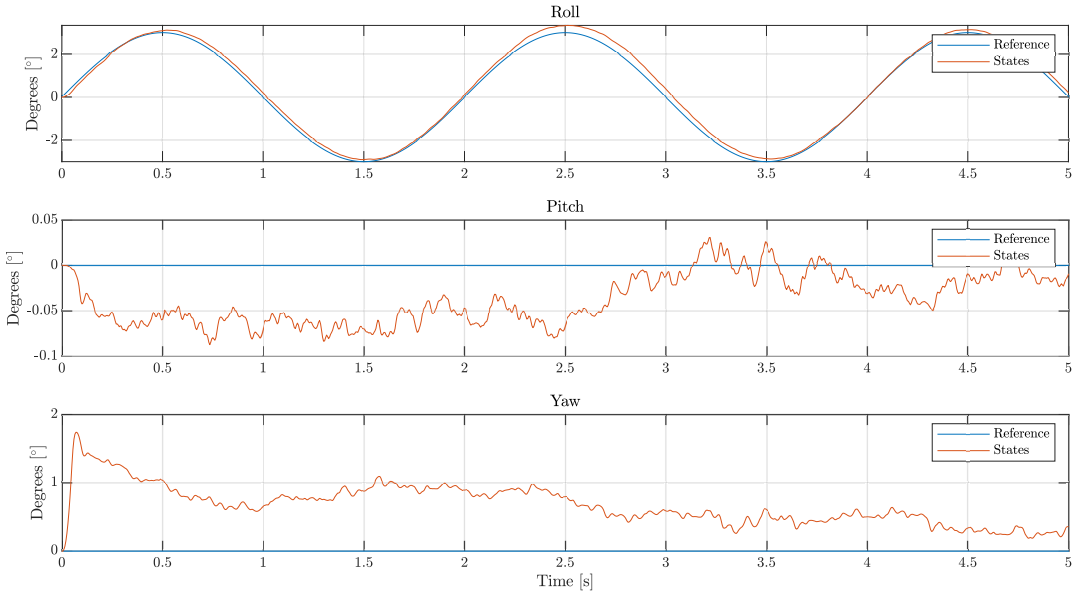


Figure 9.10: Response of the system with velocity controller, balance controller, sensor noise and state estimators under the effect of a sine wave, represented in Euler-angles.

Finally, the torque signals commanded to the motors are shown in Figure 9.11. They reflect that under the influence of a sine wave with frequency lower than the bandwidth of the system, the torque signals do not saturate but produce noisy signals. Since the balance controller itself did not produce chattering in the control input signals, the noise in the signals is attributed to the estimators performance. Therefore, it might be important to check this behaviour in the real set-up, where the noise in the sensors could be lower than the one expected in the simulations and the estimators would perform adequately. Otherwise, the noise covariance matrices of the estimators might need to be adjusted to deal with this undesired effect.

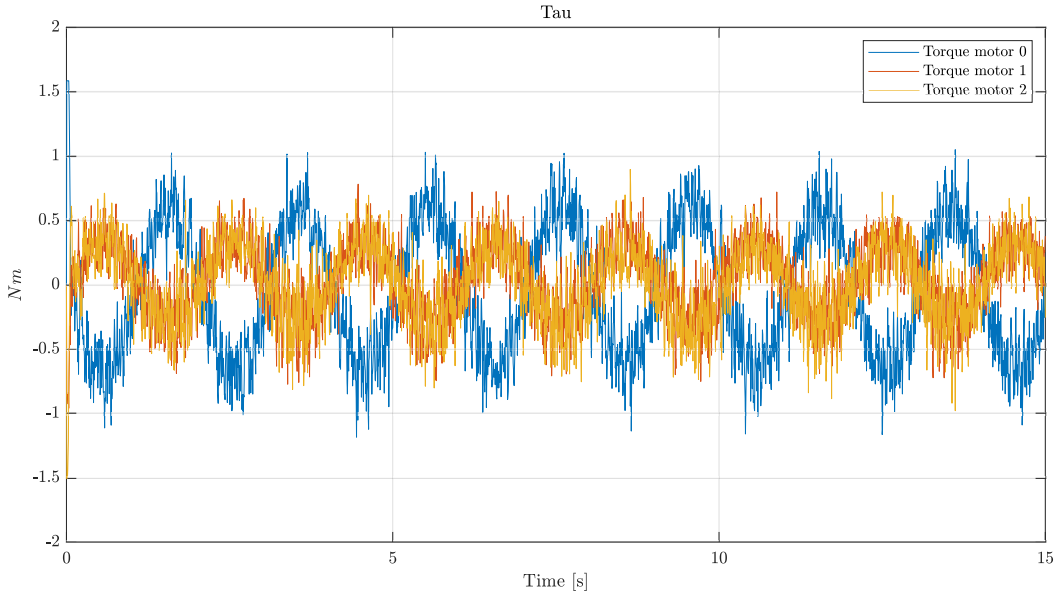


Figure 9.11: Motor torques of the system with velocity controller, balance controller, sensor noise and estimators under a sine wave signal.

9.3.2 Complete system under the influence of a varying velocity signal

The effect of the time-varying velocity signal showed in Figure 9.3 is simulated in this case for the complete system. Figure 9.12 confirms that the system is capable of effectively tracking a time-varying velocity reference. However, small fluctuations are observed mainly in the second and fourth quaternion components, q_1 and q_3.

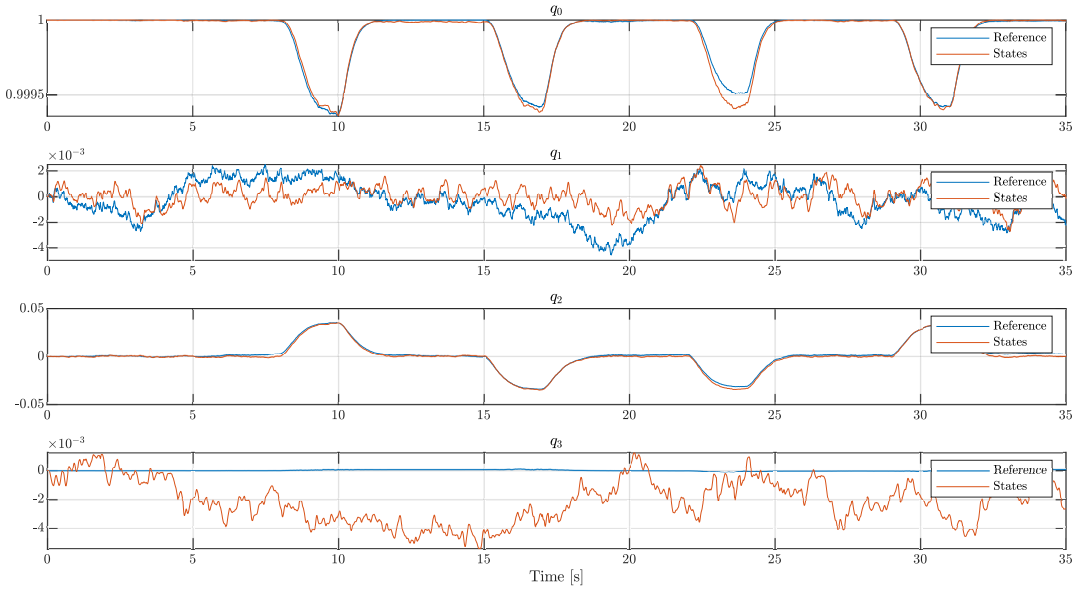


Figure 9.12: Results of the quaternion tracking under a velocity varying signal when the estimator and the velocity and balance controller are active.

However, it can be seen how in the Euler-angle representation the yaw is miss-tracked by approximately $-0.2°$ error on average. Roll and yaw tracking provide some noisy but acceptable results.

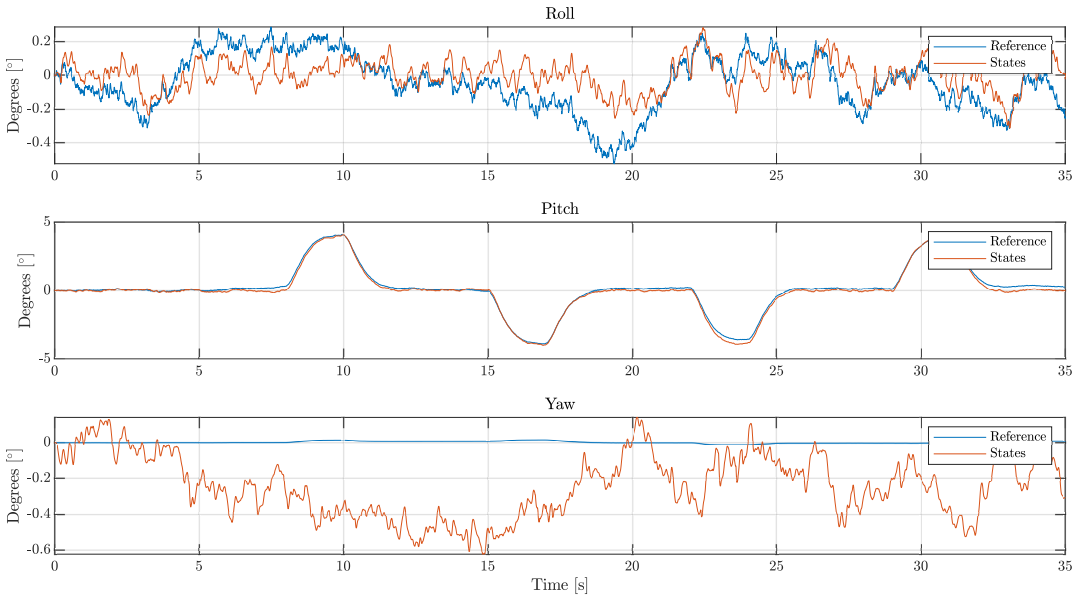


Figure 9.13: Results of the tracking under a velocity varying signal when the estimator and the velocity and balance controller are active shown in Euler-angles.

9.4 Performance comparison with previous version

The system developed by [6] that includes a Sliding mode controller and Extended Kalman filters (V2019) is compared with the actual system that uses Feedback Linearization control and Unscented Kalman Filters (V2020).

In Figure 8.8 the results of the tracking error for both systems is presented in Euler-angles. It is observed that the error magnitude obtained is quite similar in both systems.

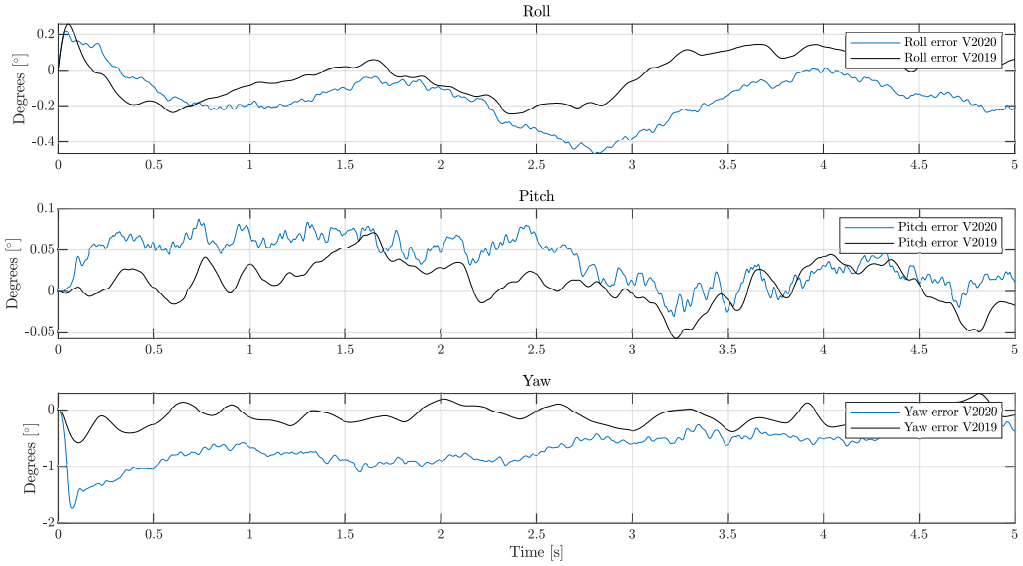


Figure 9.14: Euler angle error comparison between the implementation in the V2019 project and the V2020 including estimators, controllers and noise signals.

Finally, comparing the torque provided by the motors it can be asserted that the behaviour of the controllers is almost the same when facing a sine wave reference.

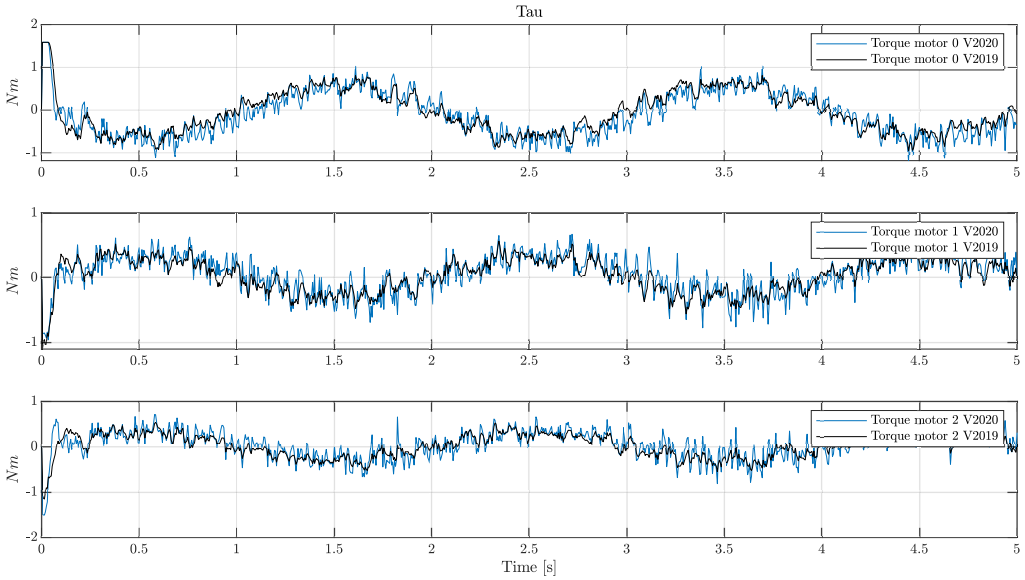


Figure 9.15: Torque signals comparison between the implementation in the V2019 project and the V2020 including estimators, controllers and noise signals.

Chapter 10

Conclusion

This chapter aims to summarize the main issues encountered in the development of this project and to discuss the reasoning behind the decisions taken to overcome them. Furthermore, it intends to provide commentary on the obtained results and findings.

The main challenges sought while choosing the topic of the current thesis consisted in studying a complex mechanical system with some minimum requirements such as having multiple degrees of freedom, the possibility of including estimators, and a multiple-input multiple-output system. These research topics were found to suit well in the ball-robot.

As stated in the problem statement in Chapter 1, the main goals of the project were defined as:

1. *Review, understand and develop the quaternion model with alternative quaternion operation.*
2. *Development of an advanced observer strategy for non-linear systems.*
3. *Derivation and implementation of an advanced non-linear control strategy.*

The goals have been properly achieved and draw a series of conclusions that are presented next together with a summary of the work done.

The ball-robot was modeled using a two body representation which consists in a robot balancing on a ball. The orientation of the body was represented using quaternions. Hence, the five degrees of freedom of the ball-robot were described using six generalized coordinates. The quaternion representation avoids singularities compared to Euler-angles system. However, for the ball-robot operation range,

it is proved unnecessary since it does not reach the condition for the gimbal lock to occur. Other representations, such as Euler-angles, might reduce the complexity of the model.

The kinematic model was derived defining the forward and inverse kinematic relations. Upon the kinematic model, the energy equations were derived and introduced in the Lagrangian formulation to obtain the Euler-Lagrange equations of the system. The input torques were introduced as a relation between the motors and the system trough the kinematic relationship. Furthermore, it was required to introduce the constraints arising from the quaternion representation. Thus, it was necessary to include a diffeomorphism to represent the dynamics of the system in an ODE closed form that is more suitable for control theory. The mentioned transformations yielded a control-affine system of equations, relating the six generalized coordinates and their derivatives with the inputs of the system.

Developed algorithms are intended to be applied in a real device, therefore they have to be able to cope with the possible challenges arising in the real set-up. The states of the system are obtained through measurements from sensors. Thus, aspects such as noise, bandwidth and sampling time have been contemplated. In addition, possible imprecision in the derived dynamic model is considered. Two estimators were developed as a solution to the aforementioned problems; a Quaternion Unscented Kalman Filter (QUKF) and a Velocity Unscented Kalman Filter (VUKF). The Unscented Kalman Filters were aimed to deal with the non-linearities of the model, in contrast to the Extended Kalman Filter which is based on Jacobian linearization, therefore losing precision.

The QUKF was firstly implemented using the forced unit norm quaternion, similarly to how the Extended Kalman Filter was treating the quaternions in [6]. However, the resultant quaternion estimate was deeply deviated from the correct solution, showing that the forced norm was imprecise for the UKF due to the number of additions and subtractions of quaternions required. A variation of the classical Unscented Kalman Filter was introduced to overcome this issue, in which the quaternions comply with the unit norm throughout the algorithm.

The VUKF and the QUKF proved to work properly and fulfill the performance specifications expected in the project. The Unscented solutions were compared with the previous version of the ball-robot. Despite the Unscented Kalman Filter being a more complex estimator than the Extended Kalman Filter, the results closely match. Therefore it can be concluded that the Extended Kalman Filter is a better option from an analytical point of view, since the performance are closely the same but the complexity of the Extended Kalman Filter is lower. On top of that, it confirms that forcing the quaternion unit norm provides proper results for

some algorithms as mentioned in [36].

Finally, the controller scheme of the system was defined and implemented. It consists of a cascade structure, where the I-LQR velocity controller in the outer loop, previously defined in [6], is used to generate the signals to a balance controller, which is placed in the inner loop. The balance controller assures that the required orientation, either to achieve the translational velocity or to maintain the equilibrium, is obtained. The balance controller was implemented through a non-linear Feedback Linearization Controller (FLC) modified to deal with quaternions and MIMO systems.

The FLC proved to track properly a sine reference with almost non noticeable error in the roll rotation. The error magnitude in pitch angle is around $\pm 0.2°$, and for the yaw angle reaches $\pm 0.08°$.

The robustness of the algorithm was evaluated both trough a change of mass and inertia, and in a force disturbance rejection. Those tests showed an adequate convergence time and stability. Furthermore, a misalignment in the center of mass was introduced. The balance controller was not capable of dealing with it by itself, leading to a "run-away" effect. By introducing the velocity controller the new equilibrium point was calculated and the balance controller managed to reach stability and satisfactory performance. Overall, the controller complies with the performance specified in the thesis and it can be concluded to be suitable for this application.

As a final remark, it was shown that the FLC strategy for the balance controller, compared with the Sliding Mode Controller (SMC), does not present a significant improvement. It is concluded that both non-linear controllers are suitable for the ball-robot stabilization problem. However, the stability in the SMC with a continuous law is only assured for the reaching phase, leading to oscillations around the equilibrium point. However, it is necessary to remark that both solutions shown a degraded performance when using the studied estimators. Nonetheless, a proper test in the real set-up should be performed to assert the results.

Chapter 11

Future work

The current thesis addresses advanced algorithms both for parameter estimation and control. However, the ball balancing robot is a source of deep and extensive research, therefore, in the current chapter, some possible modifications and improvements that may be implemented in future research are cataloged.

The Quaternion Unscented Kalman Filter uses a model of the accelerometer to estimate the orientation of the ball-robot that makes the heading bias unobservable. The simplification incurs in a lack of tracking of the rotation around the z-axis, therefore miss-calculating the yaw. Usually, the deviation can be corrected in the update step by the use of other sensors sources, such as a LiDAR or GPS for large displacements. Therefore, a LiDAR is suggested when implementing it in a real environment. However, in case the LiDAR reading was not available, a more accurate model of the accelerometer measuring the acceleration instead of the gravity vector is proposed. In addition, a model-driven estimator scheme should be evaluated due to the observed degradation of the controller performance generated by the estimators.

It was stated that one of the advantages of the shape-accelerated properties of the ball-robot is the observability of the states, e.g. a change in inclination is easily reflected in the states. Hence, the model of the system could be updated dynamically to cope with changes in the properties or inaccuracies in the a priori model. One of the properties that are susceptible to deviations and at the same time have a great impact on the tracking is the COM [6]. As a consequence, it is suggested developing an observer for the COM to increase the accuracy of the controllers.

It has been proved that the Non-linear Feedback Linearization Control (FLC) is capable of rejecting the expected disturbances that are thought as feasible to occur in

the ball-robot. It would be possible to extend the FLC to cope with more demanding disturbances. To do so, the controller could be extended to include a dynamic law or extra terms related to the disturbances in the control law, as it is explained in [41]. Therefore, it is suggested exploring these options to increase the range of operation of the ball-robot.

In relation to the conclusions drawn from the comparison between the controller implemented in this project and in the previous one [6], the FLC and Sliding Mode Controller provide similar results in terms of stability and performance. Therefore, an improvement in the Sliding manifold might out-stand achieved results. It is proposed to investigate the effects of implementing a second-order Sliding Mode which would overcome the chattering problem without oscillating around the equilibrium point behavior created by a continuous law. For the relative degree of the ball-robot, the proper second-order sliding mode algorithm is often referred to as the "super twisting algorithm". The main difference lies in the switching control term, which rejects the disturbances. Instead of being defined as a sign signal that depends on the sliding variable, it takes the integral of it reducing the control activity and assuring convergence. The Super Twisting algorithm was already tested in the current project, but without achieving convergence, which might be due to a non-exhaustive tuning of the parameters.

Finally, the results obtained in the current thesis are subjected to the limitations of simulation scope. Therefore, in order to make a final and valuable conclusion on the results achieved, it would be necessary to implement the algorithms in the real set-up and assess the expected properties of the estimators and controllers. Furthermore, it would be interesting to observe the performance of the system under more challenging environments and operating scenarios.

Bibliography

[1] *Robot Digital Signage*. Last accessed: 21 June 2020. URL: https://vbn.aau.dk/da/projects/robot-digital-signage.

[2] Karl Hansen. *Onshape model of the V0 ball-robot*. Last accessed: 10 June 2020. URL: https://cad.onshape.com/documents/24e69a83be385396bc5b5e2c/w/87f4e9ceff52e4f7a92a446f/e/ef0bdc419cd42c33f12e4459.html.

[3] Tom B Lauwers, George A Kantor, and Ralph L Hollis. "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 2884–2889.

[4] Masaaki Kumagai and Takaya Ochiai. "Development of a robot balancing on a ball". In: *2008 International Conference on Control, Automation and Systems*. IEEE. 2008, pp. 433–438.

[5] Peter Fankhauser and Corsin Gwerder. "Modeling and control of a ballbot". B.S. thesis. Eidgenössische Technische Hochschule Zürich, 2010.

[6] Thomas Kølbæk Jespersen. "Modelling and Control of a Ball-balancing Robot". In: (2019).

[7] Peter Fankhauser and Corsin Gwerder. "Modeling and control of a ballbot". B.S. thesis. Eidgenössische Technische Hochschule Zürich, 2010.

[8] Ching-Wen Liao, Ching-Chih Tsai, Yi Yu Li, and Cheng-Kai Chan. "Dynamic modeling and sliding-mode control of a ball robot with inverse mouse-ball drive". In: *2008 SICE Annual Conference*. IEEE. 2008, pp. 2951–2955.

[9] Lopes G.A.D and Verdier CF. "Geometric control of an under-actuated balancing robot, Master thesis". In: (2015).

[10] Amin Lotfiani, Mehdi Keshmiri, and Mohamad Danesh. "Dynamic analysis and control synthesis of a spherical wheeled robot (Ballbot)". In: *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE. 2013, pp. 481–486.

[11] Dinh Ba Pham and Soon-Geul Lee. "Aggregated hierarchical sliding mode control for a spatial ridable ballbot". In: *International Journal of Precision Engineering and Manufacturing* 19.9 (2018), pp. 1291–1302.

[12] *100mm stainless steel rollers omni wheel for ball balance ballbot*. Last accessed: 10 June 2020. URL: https://es.aliexpress.com/item/32818724751.html.

[13] *Omni wheel*. Last accessed: 21 April 2020. URL: https://en.wikipedia.org/wiki/Omni_wheel.

[14] *Fully Actuated vs. Underactuated Systems*. Last accessed: 21 April 2020. URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-832-underactuated-robotics-spring-2009/readings/MIT6_832s09_read_ch01.pdf.

[15] Ahmed Khelassi, Didier Theilliol, and Philippe Weber. "Control design for over-actuated systems based on reliability indicators". In: (2010).

[16] Umashankar Nagarajan and Ralph Hollis. "Shape space planner for shape-accelerated balancing mobile robots". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1323–1341.

[17] *TState Estimation*. Last accessed: 31 May 2020. URL: https://www.cds.caltech.edu/~murray/wiki/images/b/b3/Stateestim.pdf.

[18] *Understanding Kalman Filters*. Last accessed: 31 April 2020. URL: https://www.youtube.com/watch?v=4OerJmPpkRg&feature=youtu.be&t=367.

[19] *Understanding Kalman Filters*. Last accessed: 31 April 2020. URL: https://uk.mathworks.com/help/control/ug/kalman-filtering.html.

[20] *Deterministic system*. Last accessed: 25 May 2020. URL: https://en.wikipedia.org/wiki/Deterministic_system.

[21] Kelly Jonathan. *Online course: State Estimation and Localization for Self-Driving Cars*. Last accessed: 22 May 2020. URL: https://www.coursera.org/learn/state-estimation-localization-self-driving-cars.

[22] Joan Sola. "Quaternion kinematics for the error-state Kalman filter". In: *arXiv preprint arXiv:1711.02508* (2017).

[23] Angelo Maria Sabatini. "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation". In: *Sensors* 11.10 (2011), pp. 9182–9206.

[24] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: (1960).

[25] Paul Zarchan and Howard Musoff. *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, Inc., 2013.

[26] Tony Lacey. *Tutorial: The Kalman Filter*. Last accessed: 31 May 2020. URL: http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf.

[27] Sian Cousins, Diego Kaski, Nicholas Cutfield, Barry Seemungal, John F Golding, Michael Gresty, Stefan Glasauer, and Adolfo M Bronstein. "Vestibular perception following acute unilateral vestibular lesions". In: *PLoS One* 8.5 (2013).

[28] *Kalman filter*. Last accessed: 31 May 2020. URL: https://en.wikipedia.org/wiki/Kalman_filter.

[29] Edgar Kraft. "A quaternion-based unscented Kalman filter for orientation tracking". In: *Proceedings of the Sixth International Conference of Information Fusion*. Vol. 1. 2003, pp. 47–54.

[30] Fredrik Gustafsson and Gustaf Hendeby. "Some relations between extended and unscented Kalman filters". In: *IEEE Transactions on Signal Processing* 60.2 (2011), pp. 545–555.

[31] Simon J Julier and Jeffrey K Uhlmann. "Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations". In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*. Vol. 2. IEEE. 2002, pp. 887–892.

[32] F Landis Markley. "Multiplicative vs. additive filtering for spacecraft attitude determination". In: *Dynamics and Control of Systems and Structures in Space* 467-474 (2004).

[33] Landis Markley and John L Crassidis. *Fundamentals of spacecraft attitude determination and control*. Vol. 33. Springer, 2014.

[34] Landis Markley, Yang Cheng, John Crassidis, and Yaakov Oshman. "Averaging Quaternions". In: *Journal of Guidance, Control, and Dynamics* 30 (July 2007), pp. 1193–1196. DOI: 10.2514/1.28949.

[35] MathWorks. *Quaternion mean computation "meanrot"*. Last accessed: 1 June 2020. URL: https://se.mathworks.com/help/fusion/ref/quaternion.meanrot.html#d120e5306.

[36] Itzhack Bar-Itzhack, J Deutschmann, and F Markley. "Quaternion normalization in additive EKF for spacecraft attitude determination". In: *Navigation and Control Conference*. 1991, p. 2706.

[37] Geoff Barton. *Process control: Designing processes and control systems for dynamic performance: by Thomas E. Marlin (McGraw-Hill, New York, 1995, ISBN 0-07-040491-7, pp 954+ xxii)*. 1996.

[38] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*. Vol. 3. Addison-Wesley Reading, MA, 1994.

[39] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*. Vol. 2. Wiley New York, 2007.

[40] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 199. 1. Prentice hall Englewood Cliffs, NJ, 1991.

[41] Michael A Henson and Dale E Seborg. "Feedback linearizing control". In: *Nonlinear process control*. Vol. 4. Prentice-Hall. 1997, pp. 149–231.

[42] Cristina Ioana Pop, Eva Henrietta Dulf, and A Mueller. "Robust feedback linearization control for reference tracking and disturbance rejection in nonlinear systems". In: *Recent Advances in Robust Control-Novel Approaches and Design Methods* (2011), pp. 273–290.

[43] Javier Moreno-Valenzuela and Carlos Aguilar-Avelar. *Motion control of underactuated mechanical systems*. Springer, 2018.

[44] Yangbo Long, Sean Lyttle, Nicholas Pagano, and David J Cappelleri. "Design and quaternion-based attitude control of the omnicopter MAV using feedback linearization". In: *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection. 2012, pp. 1413–1421.

[45] William S Levine. *The Control Handbook (three volume set)*. CRC press, 2018.

[46] Luc Jaulin. *Mobile robotics*. John Wiley & Sons, 2019.

[47] InvenSense. *MPU-9250 Product Specification datasheet*. 2018.

[48] Maxongroup. *Maxon EC 60 flat datasheet*. 2018.

[49] Maxongroup. *ESCON Module 50/5 datasheet*. 2018.

[50] Jens Wittenburg. *Dynamics of systems of rigid bodies*. Vol. 33. Springer-Verlag, 2013.

[51] Matthew J Benacquista and Joseph D Romano. *Classical mechanics*. Springer, 2018.

[52] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[53] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, 2020.

[54] Alessandro De Luca. *Dynamic model of robots: Newton-Euler approach*. Last accessed: 5 December 2019. URL: `http://www.diag.uniroma1.it/~deluca/rob2_en/04_LagrangianDynamics_2.pdf`.

[55] Martin Bailyn. *A survey of thermodynamics*. AIP, 1994.

[56] Analytical Mechanics, LN Hand, and JD Finch. "Cambridge University Press". In: (2008).

[57] *The Calculus of Variations and the Euler-Lagrange Equation*. Last accessed: 21 April 2020. URL: https://www.youtube.com/watch?v=V0wx0JBEgZc&feature=youtu.be.

[58] *Calculus of variations*. Last accessed: 21 April 2020. URL: https://en.wikipedia.org/wiki/Calculus_of_variations.

[59] *The Euler-Lagrange equation*. Last accessed: 21 April 2020. URL: http://mathsci.kaist.ac.kr/~nipl/am621/lecturenotes/Euler-Lagrange_equation.pdf.

[60] Sunil Golwala. *Lecture Notes on Classical Mecahnics for Physics 106ab*.

[61] Patrick Hamill. *A student's guide to Lagrangians and Hamiltonians*. Cambridge University Press, 2014.

[62] Benjamin C Kuo and Farid Golnaraghi. "Automatic control systems". In: *Englewood Cliffs, NJ* (2003).

[63] Andrew Witkin. *Physically Based Modeling: Principles and Practice Constrained Dynamics*. Last accessed: 21 April 2020. URL: http://www.cs.cmu.edu/~baraff/sigcourse/notesf.pdf.

[64] Joachim Baumgarte. "Stabilization of constraints and integrals of motion in dynamical systems". In: *Computer methods in applied mechanics and engineering* 1.1 (1972), pp. 1–16.

[65] Shaoping Bai. *Advanced Mechanics Modeling of Mechanical Systems*.

[66] CH Robotics. *Understanding Quaternions*. URL: http://www.chrobotics.com/library/understanding-quaternions.

[67] Erik B. Dam, Martin Koch, and Martin Lillholm. *Quaternions, Interpolation and Animation*. Tech. rep. Last accessed: 21 April 2020. University of Copenhagen, 1998. URL: https://web.mit.edu/2.998/www/QuaternionReport1.pdf.

[68] Wikipedia. *Rotation formalisms in three dimensions*. Last accessed: 21 April 2020. URL: https://en.wikipedia.org/wiki/Rotation{_}formalisms{_}in{_}three{_}dimensions.

[69] Neil Bickford. *Why do the unit quaternions double-cover the space of rotations?* URL: https://www.gathering4gardner.org/g4g13gift/math/BickfordNeil-GiftExchange-WhyDoTheUnitQuaternionsDoubleCoverTheSpaceOfRotations-G4G13.pdf.

[70] *Euler's rotation theorem*. Last accessed: 21 April 2020. URL: https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem.

[71] Grant Sanderson. *Visualizing quaternionsquaternions,An explorable video series*. Last accessed: 21 April 2020. URL: https://eater.net/quaternions.

[72] *Quaternions*. Last accessed: 21 April 2020. URL: https://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/.

[73] Redouane Bouhennache. "Truly Hypercomplex Numbers: Unification of Numbers and Vectors". In: *arXiv preprint arXiv:1409.2757* (2014).

[74] Eric W. Weisstein. *Hypercomplex Number*. Last accessed: 21 April 2020. URL: https://mathworld.wolfram.com/HypercomplexNumber.html.

[75] Pipho Evan. *3D Models*. 2017, pp. 20–20. ISBN: 1592000339. DOI: 10.1007/978-3-319-17885-1_100007. URL: http://index-of.co.uk/Game-Development/Designing/FocusOn3DModels.pdf.

[76] Wikipedia. *Quaternions and spatial rotation*. URL: https://en.wikipedia.org/wiki/Quaternions{_}and{_}spatial{_}rotation.

[77] Grant Sanderson and 3blue1brown. *Double cover quaternions*. Last accessed: 21 April 2020. URL: https://eater.net/quaternions/video/doublecover.

[78] José Luis Blanco Claraco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Tech. rep. Last accessed: 21 April 2020. Universidad de Malaga, 2019. URL: https://ingmec.ual.es/~jlblanco/papers/jlblanco2010geometry3D_techrep.pdf.

[79] *Rotation Matrix To Euler Angles*. Last accessed: 21 April 2020. URL: https://www.learnopencv.com/rotation-matrix-to-euler-angles/.

[80] Erik B Dam, Martin Koch, and Martin Lillholm. "Quaternions , Interpolation and Animation". In: (1998).

Part IV

Appendix

Appendices

Appendix A

System parameters

List of parameters of the ball-robot components extracted from datasheets [47], [48], [49] and from [6].

Group	Symbol	Parameter	Value
Body	M_b	Ball-robot mass included all components	16.154 kg
	l	Distance from center of ball to center of mass	0.4213 m
	J_{bx}	Robot body inertia around x-axis	4.173 $kg\ m^2$
	J_{by}	Robot body inertia around y-axis	4.161 $kg\ m^2$
	J_{bz}	Robot body inertia around z-axis	0.1004 $kg\ m^2$
	α	Wheel zenith angle	45°
	γ	Wheel separation	120°
Motor	J_m	Motor inertia	$1.21 \cdot 10^{-4}\ kg\ m^2$
	J_g	Gear inertia	$1.2 \cdot 10^{-6}\ kg\ m^2$
	n_{gear}	Gear ratio (4.3:1)	13/3
	τ_{max}	Max output torque	1.9825 Nm
Wheel	M_w	Mass of omniwheel	0.27 kg
	r_w	Radius of omniwheel	0.05 m
	J_{ow}	Inertia of omniwheel	$9 \cdot 10^{-4}\ kg\ m^2$
	J_w	Total inertia of wheel, gear and motor	$3.19 \cdot 10^{-3}\ kg\ m^2$
	n_{ticks}	Encoder ticks per wheel revolution	70997.33
Friction	B_{vk}	Viscous friction of ball to ground	$0\ N\ /\ (m/s)$
	B_{vm}	Viscous friction of motor and wheel to ball	$0\ N\ /\ (rad/s)$
	B_{vb}	Viscous (air) friction of body	$0\ N\ /\ (rad/s)$
Ball	r_k	Radius of ball	0.129 m
	M_k	Mass of ball	1.478 kg
	J_k	Inertia of ball	$15.4 \cdot 10^{-3}\ kg\ m^2$

Appendix B

Kinematics

The kinematic 3D model general goal is to relate kinematic variables, such as positions and velocities, to the inertial frame. In other words, they express certain variables in terms of the system state-space variables, which are the generalized coordinates and their derivatives. This way, these variables can be substituted by their kinematic relation to the generalized coordinates throughout the dynamic model derivation to have the dynamic model just in terms of the state space variables. Classic mechanics of motion theory will be applied following [50] [51].

It can be seen that each of the sections, in which this Appendix is divided, follows the same main goal defined above, just applied for defining different intermediate variables in the inertial frame. First, in Section B.2 called 'Intermediate kinematic relationships', relevant variables will be defined, through a series of kinematic transformations, in terms of the system state variables. Afterwards, in inverse and forward kinematics Sections, B.8 and B.9 respectively, the complete relationship between actuator variables and state variables will be described making use of the intermediate relations previously calculated where inverse kinematics maps from state variables to actuator variables and forward do the opposite.

B.1 Ball-robot reference frames relationships

The ball-robot frames have a series of relationships that are useful for making equivalences in algorithms derivations and that will be used to simplify and establish the kinematics. The equivalences are explained in this section and are used throughout the project.

As seen in Figure B.1, the linear velocity of the moving frame $\{K'\}$ seen from $\{K\}$ is zero, because it only has a relative rotation over the frame $\{K\}$, so it does not translate. The observed linear velocity of a frame is not affected by a rotational movement of neither the observed frame nor the reference frame.

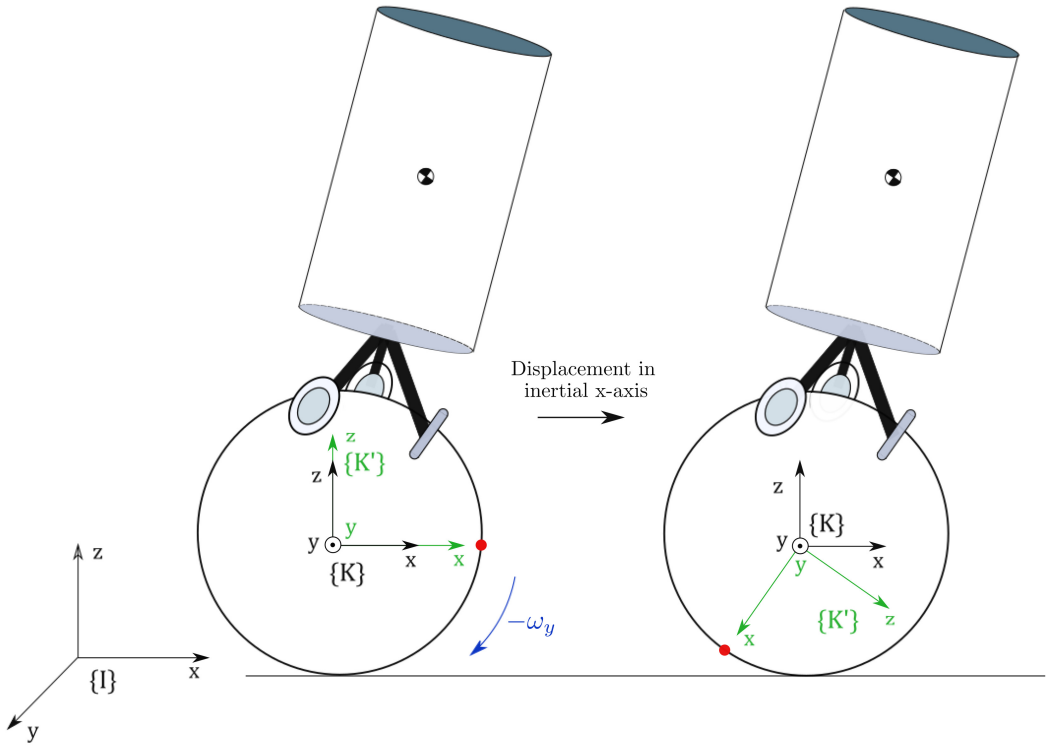


Figure B.1: Linear displacement in the x-axis shows how the moving frame $\{K'\}$ has a relative rotation over the frame $\{K\}$, but zero relative translation.

Thus, the linear velocity of frame $\{K'\}$ and $\{K\}$ are the same seen from $\{I\}$, because $\{K'\}$ rotation is defined over $\{K\}$ and $\{K\}$ translates over $\{I\}$.

$$\boldsymbol{v}_{K'|K} = 0 \neq \boldsymbol{v}_{K'|I} = \boldsymbol{v}_{K|I} \tag{B.1}$$

This relationship applies for relationships between other frames in the ball-robot when they are related by pure rotation.

For the same reason, the angular velocity of $\{K'\}$ is the same on frame $\{K\}$ and on $\{I\}$ because $\{K\}$ only translates over $\{I\}$.

$${}^{K}\omega_{K'|K} = {}^{K}\omega_{K'|I} = {}^{I}\omega_{K'|I} \tag{B.2}$$

This relationship applies for relationships between other frames in the ball-robot when they are related by pure translation.

In summary, it has been shown one example that explains the reason why in terms of linear velocity, $\{K'\}$ is equivalent to $\{K\}$, because of pure rotation. The example with the angular velocity shows that $\{K\}$ is equivalent to $\{I\}$, due to pure translation.

The logic applied to these particular frames is extended to other frames with the same characteristics in the ball-robot. Therefore, equivalences that do not appear explicitly in this section but are analog to the previous explained will be obtained throughout the chapter.

B.2 Intermediate kinematics variables to state variables

The ball-robot kinematic consists in deriving an expression for the individual omniwheel angular velocities based on the system states and vice versa. Therefore, intermediate steps relating intermediate variables to state variables must be done.

B.3 Ball linear velocity to state variables

The ball linear velocity describes the linear velocity of its center of mass and is already defined in the inertial frame.

$$ {}^{I}\boldsymbol{v}_{K|I} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} \tag{B.3} $$

For convenience, from this point forward, the components of the ball linear velocity vector will also be used to derive other relationships as well as the vector itself.

B.4 Ball angular velocity to state variables

The goal is to relate the ball angular velocity with the state variables, i.e. to define it in the inertial frame. This is needed because either the inverse kinematics or the dynamic model for the ball rotational kinetic energy equations make use of it. There is no z component in the angular velocity of the ball considered as mentioned

in Section 2.2, this yields to (B.4) for the ball angular velocity.

$$ {}^{K}\boldsymbol{\omega}_{K'|K} = \begin{bmatrix} \omega_{K',x} \\ \omega_{K',y} \\ 0 \end{bmatrix} \tag{B.4} $$

The linear or tangential velocity of the ball at the point of contact can be calculated through the cross product between the ball angular velocity and the position vector that connects the centre of the ball with the point where the linear velocity is to be calculated. In this case, the instantaneous point in contact with the ground.

$$ {}^{K}\boldsymbol{v}_{G|K'} = {}^{K}\boldsymbol{\omega}_{K'|K} \times {}^{K}\boldsymbol{p}_{G} = \begin{bmatrix} \omega_{K',x} \\ \omega_{K',y} \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -r_k \end{bmatrix} = \begin{bmatrix} -\omega_{K',y}\ r_k \\ \omega_{K',x}\ r_k \\ 0 \end{bmatrix} \tag{B.5} $$

This relationship is explained through a graphical example in Figure B.2

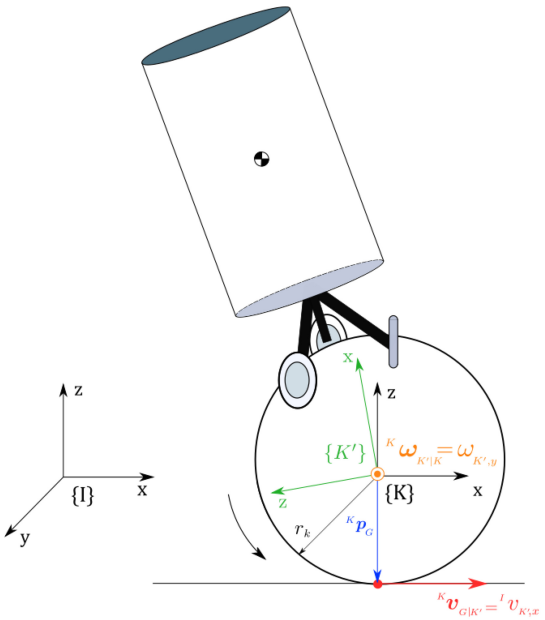


Figure B.2: Graphical explanation of the Equation B.5 through an example showing that an angular velocity in the y-axis creates a linear velocity in the inertial x-axis direction, which can be calculated through the cross-product of the angular velocity and the vector relating the center of the ball with the point of contact.

Symmetry is applied in case of swapping the observed frame and the reference frame. This way, the velocity of ball seen from the ground has the same speed but opposite direction as the velocity of the ground seen from the ball, as seen in (B.6).

$$ {}^{K}\boldsymbol{v}_{K'|G} = -{}^{K}\boldsymbol{v}_{G|K'} \tag{B.6} $$

The state variables x, y define the position of the center of the ball in the inertial frame and thus, its derivative corresponds to the linear velocity. To remove from the equations the intermediate variables (${}^{K}\boldsymbol{v}_{G|K'}$) and leave them in terms of angular velocity and state variables, it is needed a relationship that connects the ball linear velocity in the inertial frame ($\dot{x}, \dot{y}$) to the linear velocity of the contact point previously calculated in (B.5).

The constraint that links them is the no-slip condition, which implies that the ball is assumed to roll without slipping with the floor and yields to having zero instantaneous linear velocity at the point of contact between the ball and the ground G. The rolling movement is entirely converted into ball translation motion.

Consequently, the linear ball velocity in the inertial frame from (B.3) and the tangential velocity of the ball seen from the contact point will be equal.

$$ {}^{I}\boldsymbol{v}_{K'|I} = {}^{K}\boldsymbol{v}_{K'|G} = -{}^{K}\boldsymbol{v}_{G|K'} \tag{B.7} $$

Making them equal and substituting the result obtained from (B.5) yields to

$$ {}^{I}\boldsymbol{v}_{K'|I} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} = -{}^{K}\boldsymbol{v}_{G|K'} = \begin{bmatrix} \omega_{K',y}\ r_k \\ -\omega_{K',x}\ r_k \\ 0 \end{bmatrix} \tag{B.8} $$

Isolating the angular velocity terms from (B.8) and placing them in order as (B.4) shows, results in the angular velocity expression in terms of the translational state variables.

$$ \boldsymbol{\omega}_{K'|I} = \frac{1}{r_k} \begin{bmatrix} -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} \qquad \tilde{\boldsymbol{\omega}}_{K'|I} = \frac{1}{r_k} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} \tag{B.9} $$

B.5 Body linear velocity to state variables

The body linear velocity needs to be defined in the inertial frame, i.e. in terms of the state variables, because the dynamic model makes use of it inside the kinetic energy equations of the body.

The linear velocity of the body refers to its centre of mass velocity and is going to be derived using the quaternion derivative. Thus, the position of the centre of mass of the body needs to be defined first. It coincides with the origin of the $\{B'\}$ frame. In case the ball-robot is entirely vertical, and the sensors are correctly calibrated, the origin of the $\{B'\}$ frame with respect to the $\{B\}$ frame would be a vector with a value different from zero only in the z direction. That value would be the distance between the centre of mass of the body and centre of the ball, l.

$$ {}^{B}\tilde{\boldsymbol{p}}_{B'} = {}^{B'}\tilde{\boldsymbol{p}}_{COM} = {}^{B}\boldsymbol{O}_{B'} \tag{B.10} $$

It needs to be defined in the inertial frame, to do so, it first needs to be rotated to the $\{K\}$ frame by quaternion rotation.

$$ {}^{K}\tilde{\boldsymbol{p}}_{B'} = {}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{p}}_{B'} \circ {}^{K}_{B}\boldsymbol{q}^{*} = \Phi({}^{K}_{B}\boldsymbol{q})\,\Gamma({}^{K}_{B}\boldsymbol{q})^{T}\,{}^{B}\tilde{\boldsymbol{p}}_{B'} \tag{B.11} $$

The velocity of the body with respect to the ball can be computed with the derivative in time of its orientation quaternion, as it is defined in (E.36). The velocity vector is recovered from its quaternion form by using (E.7).

$$ {}^{K}\boldsymbol{\nu}_{B'} = {}^{K}\dot{\boldsymbol{p}}_{B'} = \vee\,{}^{K}\dot{\tilde{\boldsymbol{p}}}_{B'} = \vee\,\left({}^{K}_{B}\dot{\boldsymbol{q}} \circ {}^{B}\tilde{\boldsymbol{p}}_{B'} \circ {}^{K}_{B}\boldsymbol{q}^{*} + {}^{K}_{B}\boldsymbol{q} \circ {}^{B}\dot{\tilde{\boldsymbol{p}}}_{B'} \circ {}^{K}_{B}\dot{\boldsymbol{q}}^{*}\right) \tag{B.12} $$

It can be simplified by using the operator $\boldsymbol{\Pi}_{p}$, as it is shown in (E.39)

$$ {}^{K}\boldsymbol{\nu}_{B'} = 2\Pi_{p}({}^{K}_{B}\dot{\boldsymbol{q}})\,{}^{K}_{B}\boldsymbol{q} \tag{B.13} $$

In the inertial frame, the $\{K\}$ frame has a relative translation. Hence, to define the position of the body (B.11) in the inertial frame, it needs to be translated by the distance that $\{K\}$ has separated from it. As a remark, the potential energy of the body will make use of this vector to extract the vertical distance in the inertial

frame of the body centre of mass.

$$ {}^{I}\boldsymbol{p}_{B'} = {}^{I}\boldsymbol{p}_{K} + {}^{K}\boldsymbol{p}_{B'} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + {}^{K}\boldsymbol{p}_{B'} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + \vee \left(\Phi({}^{K}_{B}\boldsymbol{q})\, \Gamma({}^{K}_{B}\boldsymbol{q})^{T}\, {}^{B}\tilde{\boldsymbol{p}}_{B'} \right) \tag{B.14} $$

Since the $\{K\}$ frame do not rotate relative to the inertial frame, the velocity of the centre of mass in the inertial frame will be the sum of the velocity of the $\{B'\}$ frame with respect to frame $\{B\}$ and the velocity of the $\{B\}$ frame with respect to the inertial frame. Finally, the linear velocity of the body with respect to the inertial frame is computed as shown next by substituting (B.13).

$$ {}^{I}\boldsymbol{\nu}_{B'} = {}^{I}\dot{\boldsymbol{p}}_{B'} = {}^{I}\dot{\boldsymbol{p}}_{K} + {}^{K}\dot{\boldsymbol{p}}_{B'} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + {}^{K}\boldsymbol{\nu}_{B'} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + 2\Pi_{p}({}^{K}_{B}\dot{\boldsymbol{q}})^{K}_{B}\boldsymbol{q} \tag{B.15} $$

B.6 Body angular velocity to state variables

It is aimed to define the body angular velocity in terms of the state variables. The body rotational kinetic energy equations make use of it in the dynamic model; it is, therefore, needed to be defined in the body frame.

The angular velocity is related to the quaternion orientation and its velocity through (E.34).

$$ {}^{B}\tilde{\boldsymbol{\omega}}_{B|I} = {}^{B}\tilde{\boldsymbol{\omega}}_{B|K} = 2\,{}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}_{B}\dot{\boldsymbol{q}} = 2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\, {}^{K}_{B}\dot{\boldsymbol{q}} \tag{B.16} $$

B.7 Wheels angular velocity to state variables

The no-slip condition is a constraint that enables to connect kinetics from the ball with kinetics from the body at the point of contact. This way, the wheels angular velocity can be defined in the inertial frame, i.e. the state variables. This step is needed either for the inverse and forward kinematics or the wheel rotational kinetic energy calculation in the dynamic model. For that purpose, the first linear velocity at the contact point from both the ball and from the body is derived and then connected with the no-slip constraint.

B.7.1 Tangential velocity of the ball at the omni-wheels contact point

The tangential velocity of the ball at the contact point with the omni-wheels are calculated through the cross product between the angular velocity of the ball seen from $\{B\}$, i.e. $\boldsymbol{\omega}_{K'|B}$, and the position vector to the contact point for each wheel. Therefore, firstly the angular velocity of the ball has to be derived.

The omni-wheel index i indicates each of the omni-wheels of the ball-robot and is defined in Figure B.3. The subscript i appears in different derivations throughout this section and takes the values: 0, 1 and 2, in order to define the relations for each one of the omni-wheels.

Since in $\boldsymbol{\omega}_{K'|B}$ the reference frame $\{B\}$ is rotating, then, the resulting observed angular velocity is the sum of angular velocities. This addition can be done if every term is defined in the same frame.

Equation (B.17) states that the velocity of the body with respect to the inertia frame is the velocity of the body seen from the ball added to the velocity of the ball seen from the inertia frame. It is easier to picture this velocities relation and, from it, isolate the velocity of interest.

$$ {}^{B}\boldsymbol{\omega}_{B|I} = {}^{B}\boldsymbol{\omega}_{B|K'} + {}^{B}\boldsymbol{\omega}_{K'|I'} \tag{B.17} $$

Applying the symmetry property from (B.6)

$$ {}^{B}\boldsymbol{\omega}_{K'|B} = -{}^{B}\boldsymbol{\omega}_{B|K'} \tag{B.18} $$

Then, the angular velocity of frame $\{K'\}$ with respect to $\{B\}$ frame can be isolated:

$$ {}^{B}\boldsymbol{\omega}_{K'|B} = {}^{B}\boldsymbol{\omega}_{K'|I} - {}^{B}\boldsymbol{\omega}_{B|I} \tag{B.19} $$

Hence, both components need to be calculated and defined in the $\{B\}$ reference frame. The first term of the sum is already derived in (B.9), but it needs to be rotated to be defined in $\{B\}$ frame instead of in the inertia frame as it was. This is accomplished by rotation quaternion multiplication as explained in (E.24). Changes from the index $\{K\}$ to $\{I\}$ are done during the derivation as explained in (B.2). Through the use of the matrix operators, the quaternion multiplication is transformed into matrix multiplication (E.25).

$$ {}^{B}\tilde{\boldsymbol{\omega}}_{K'|I} = {}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}\tilde{\boldsymbol{\omega}}_{K'|K} \circ {}^{K}_{B}\boldsymbol{q} = \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\Gamma\left({}^{K}_{B}\boldsymbol{q}\right){}^{K}\tilde{\boldsymbol{\omega}}_{K'|K} \tag{B.20} $$

The second term of (B.19) is the angular velocity of $\{B\}$ frame seen from the inertia frame and must be defined in the $\{B\}$ frame. It is already derived when defining the body angular velocity in (B.16). Once the two terms are derived, Equation (B.19) is applied to calculate the angular velocity of the ball.

$$ {}^{B}\boldsymbol{\omega}_{K'|B} = \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\Gamma\left({}^{K}_{B}\boldsymbol{q}\right){}^{K}\tilde{\boldsymbol{\omega}}_{K'|K} - 2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}{}^{K}_{B}\dot{\boldsymbol{q}} \tag{B.21} $$

The linear velocity of the ball at the point of contact is thus calculated for each omni-wheel i through the cross product.

$$ {}^{B}\boldsymbol{v}_{C_i|K} = {}^{B}\boldsymbol{\omega}_{K'|B} \times {}^{B}\boldsymbol{p}_{C,i|B} \tag{B.22} $$

The cross product of these two vectors can be replaced by its equivalent in quaternions, the quaternion multiplication, as was explained in (E.9). Then, it is simplified by using the operator (E.12).

$$ {}^{B}\tilde{\boldsymbol{v}}_{C_i|K} = {}^{B}\tilde{\boldsymbol{\omega}}_{K'|B} \circ {}^{B}\tilde{\boldsymbol{p}}_{C_i|B} = \Gamma\left({}^{B}\tilde{\boldsymbol{p}}_{C_i|B}\right){}^{B}\tilde{\boldsymbol{\omega}}_{K'|B} \tag{B.23} $$

Inserting (B.21) in (B.23) and taking the common factor yields to

$$ {}^{B}\tilde{\boldsymbol{v}}_{C_i|K} = \Gamma\left({}^{B}\tilde{\boldsymbol{p}}_{C_i|B}\right)\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\left(\Gamma\left({}^{K}_{B}\boldsymbol{q}\right){}^{K}\tilde{\boldsymbol{\omega}}_{K'|K} - 2\,{}^{K}_{B}\dot{\boldsymbol{q}}\right) \tag{B.24} $$

Where the position vector for each omni-wheel defined in $\{B\}$ frame is derived using SE(3) transformations from the $\{B\}$ frame to each contact point frame $\{C_i\}$.

The $\{C_i\}$ reference frames have to be defined to allow applying the no-slip condition. It is an auxiliary reference frame with its origin at the contact point between the ball and the omni-wheel, the x axis is parallel to the motor axis, and the y axis coincides with the tangential direction of the wheel. It is depicted in Figure B.3.

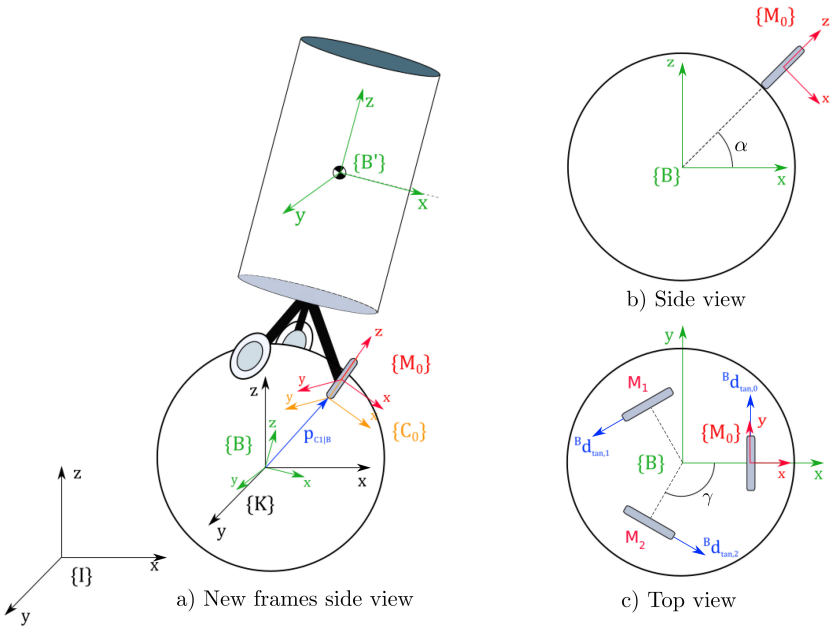


Figure B.3: Motor and point of contact frames, $\{M_i\}$ and $\{C_i\}$ respectively. The angles that define the omni-wheels location are shown in Figure *b*) and *c*).

The SE(3) transformation matrices used are explained in Appendix F. Transformation matrices are used in this transformation instead of quaternions since to define $\{C_i\}$ frame in $\{B\}$ it requires a translation, which is more convenient to be done with transformation matrices, as explained in Appendix D. Furthermore, the rotation required is simple since it does not depend on time, i.e. the $\{C_i\}$ frame does not have relative rotational movement with respect to the $\{B\}$ frame, it just has an offset in the 3D orientation.

The first transformation matrix rotates the $\{B\}$ frame to the position of each omni-wheel by rotating it around the z_B axis $\gamma \cdot i$ degrees, where i is the index of each omni-wheel as the Figure B.3 *c*) shows.

$$ {}^{B}_{\gamma_i}\boldsymbol{T} = \begin{bmatrix} \mathbf{R}_Z(i \cdot \gamma) & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} cos(\gamma \cdot i) & -sin(\gamma \cdot i) & 0 & 0 \\ sin(\gamma \cdot i) & cos(\gamma \cdot i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.25} $$

The next transformation rotates α degrees around each $y_{i,\gamma}$ axis. It places each new

frame with the axes parallel to the axes of the $\{C_i\}$ frame respectively.

$$ {}^{\gamma_i}_{\alpha_i}\boldsymbol{T} = \begin{bmatrix} \mathbf{R}_{Y_i}(\alpha) & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.26} $$

The last transformation is a translation that offsets each axis the ball radius (r_k) in the z_α axis direction, to place it on the surface of the ball at the contact point of each omni-wheel.

$$ {}^{\alpha_i}_{C_i}\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.27} $$

Therefore, concatenating the transformation matrices in the proper order, first transformation is the one on the right side and last one on the left, gives the complete transformation matrix from $\{C\}$ to $\{B\}$ frame,

$$ {}^{B}_{C}\boldsymbol{T} = {}^{B}_{\gamma_i}\boldsymbol{T}\,{}^{\gamma_i}_{\alpha_i}\boldsymbol{T}\,{}^{\alpha_i}_{C_i}\boldsymbol{T} \tag{B.28} $$

A vector defined in the $\{C\}$ frame is transformed to the $\{B\}$ frame by multiplying it by this transformation matrix. To indicate in frame $\{B\}$ the vector position of the contact point, as it corresponds to the origin of the frame $\{C\}$, it requires to apply the transformation matrix to the coordinates of the origin of the frame $\{C\}$ in frame $\{C\}$. The origin is indicated with a null vector with a one appended for matrix matching.

$$ {}^{B}\boldsymbol{p}_{C_i} = \begin{bmatrix} \boldsymbol{I_3} & \mathbf{0}_{3x1} \end{bmatrix} \left({}^{B}_{C}\boldsymbol{T}\ {}^{C}\boldsymbol{p}_{C_i} \right) = \begin{bmatrix} \boldsymbol{I_3} & \mathbf{0}_{3x1} \end{bmatrix} \left({}^{B}_{C}\boldsymbol{T} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \tag{B.29} $$

The identity matrix allows to extract the position vector as follows

$$ {}^{B}\boldsymbol{p}_{C_i} = \begin{bmatrix} \boldsymbol{I_3} & \mathbf{0}_{3x1} \end{bmatrix} \begin{bmatrix} r_k cos(\gamma \cdot i) sin(\theta) \\ r_k sin(\gamma \cdot i) sin(\theta) \\ r_k cos(\theta) \\ 1 \end{bmatrix} = r_k \begin{bmatrix} cos(\gamma \cdot i) sin(\theta) \\ sin(\gamma \cdot i) sin(\theta) \\ cos(\theta) \end{bmatrix} \tag{B.30} $$

The linear velocities obtained through (B.22) for each omni-wheel are tangential to the ball surface, but that does not necessarily imply that are tangential to the

wheel circumference. This resultant velocity direction is possible because the ball-bot is provided with omni-wheels, which allow sliding laterally easily thanks to the small discs or rollers around their circumferences.

The tangential component to the wheel circumference is extracted by taking the y_{C_i} axis velocity direction and it is depicted in Figure B.3.

$$ {}^{B}\boldsymbol{v}_{tanC_i} = {}^{B}\boldsymbol{d}_{tan}\, {}^{B}\boldsymbol{v}_{C_i} \tag{B.31} $$

Where the wheel tangential direction vector defined in the $\{C\}$ frame, which only has a "1" in the y_{C_i} coordinate, is transformed again to the $\{B\}$ frame. The rotation is applied by multiplying the vector with the transformation matrix and, then, removed the last row with the identity matrix.

$$ {}^{B}\boldsymbol{d}_{tan} = \begin{bmatrix} \boldsymbol{I_3} & \boldsymbol{0}_{3x1} \end{bmatrix} \left({}^{B}_{C}\boldsymbol{T} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} -\sin(\gamma \cdot i) \\ \cos(\gamma \cdot i) \\ 0 \end{bmatrix} \tag{B.32} $$

B.7.2 Tangential velocity of the omni-wheels at the contact point

The tangential velocity of the omni-wheels at the contact point is calculated with the cross product between the angular velocity of the omni-wheel and the position vector to the contact point. The omni-wheel motion, a reference frame attached to it, is created. Therefore in this frame, the wheel motion is null because the frame rotates with it. It is x axis points to the wheel shaft direction, and the yz plane is in its radial directions.

To observe the angular velocity of the wheels, the motor frames $\{M_i\}$ are created. It is attached to the body, hence, the axes will rotate in synergy with the rotation of the body. Its x axis coincides with the motor shaft and therefore the wheels rotate around its x axis. They can be seen in the Figure B.3. This way, the angular velocity of the wheels seen from the motor frame is the measurement given by the encoders

$$ {}^{M_i}\boldsymbol{\omega}_{W_i|M_i} = \begin{bmatrix} \dot{\phi}_i \\ 0 \\ 0 \end{bmatrix} \tag{B.33} $$

The linear velocity at the contact point due to the velocity of the wheel is computed

through the cross product.

$$^{M_i}\boldsymbol{\nu}_{C_i|W} = {}^{M_i}\boldsymbol{\omega}_{W_i|M_i} \times {}^{M}\boldsymbol{p}_{C} = {}^{M_i}\boldsymbol{\omega}_{W_i|M_i} \times \begin{bmatrix} 0 \\ 0 \\ -r_w \end{bmatrix} = \begin{bmatrix} 0 \\ r_w\,\dot{\phi}_i \\ 0 \end{bmatrix} \tag{B.34}$$

For the no-slip condition to be applied, we need to define the linear velocity of the wheel in the $\{B\}$ frame. Therefore, we need to transform the linear velocity obtained in the $\{M_i\}$ frames to the $\{B\}$ frame. A transformation matrix that maps from the $\{M_i\}$ frames to the $\{C\}$ has to be concatenated to the already derived matrix to transform from $\{C\}$ to $\{B\}$. $\{M_i\}$ frames are offset r_k distance in the z direction of the $\{C\}$ frame, and, therefore, this transformation matrix is a pure translation

$$^{C}_{M_i}\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_w \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.35}$$

$$^{B}_{M_i}\boldsymbol{T} = {}^{B}_{C_i}\boldsymbol{T}\ {}^{C_i}_{M_i}\boldsymbol{T} \tag{B.36}$$

The velocity can be now transformed to the $\{B\}$ frame. Since the velocity has only a component in the y axis, i.e. tangential to the wheel circumference, the transformation matrix multiplication can be simplified to the value of the velocity in the y axis times the result obtained in Equation B.32.

$$^{B}\boldsymbol{\nu}_{C_i|W} = {}^{B}_{M_i}\boldsymbol{T}\ {}^{M_i}\boldsymbol{\nu}_{C_i|W} = r_w\,\dot{\phi}_i\ {}^{B}\boldsymbol{d}_{tan,i} \tag{B.37}$$

As a remark, the $\{M\}$ frame and the $\{C\}$ frame axes are parallel to each other separated by the radius of the wheel distance. Both frames are attached to the movement of the body and move equally. For this reason, the velocity obtained in (B.34) defined in $\{M\}$ frame is the same in the $\{C\}$ frame and could have been transformed to the $\{B\}$ frame with the matrix transformation obtained in (B.28).

B.7.3 No slip condition

The constraint that links both linear velocities is the no-slip condition, as explained in the previous Section. In this case, it is applied in the way that the omni-wheels

are assumed to rotate without slipping in its tangential direction with the ball. This yields to having zero instantaneous linear velocity at the point of contact between the omni-wheel tangential direction and the ball. The motor rotation is entirely converted into ball tangential translation motion.

Therefore, at the contact point, the tangential velocity of the ball and the tangential velocity of the omni-wheel are equal,

$$ {}^{B}\boldsymbol{d}_{tan,i}\, {}^{B}\boldsymbol{\nu}_{C_i|K} = {}^{B}\boldsymbol{d}_{tan,i}\, {}^{B}\boldsymbol{\nu}_{C_i|W} \tag{B.38} $$

Substituting (B.37) in the previous equation,

$$ {}^{B}\boldsymbol{d}_{tan,i}\, {}^{B}\boldsymbol{\nu}_{C_i|K} = {}^{B}\boldsymbol{d}_{tan,i}\, r_w\, \dot{\phi}_i\, {}^{B}\boldsymbol{d}_{tan,i} \tag{B.39} $$

Isolating the angular velocity of the motor shaft yields to

$$ \dot{\phi}_i = \frac{1}{r_w}\, {}^{B}\boldsymbol{d}_{tan,i}^{T}\, {}^{B}\boldsymbol{\nu}_{C_i|K} \tag{B.40} $$

The angular velocity of the motor shaft is now written in quaternion notation. To finally define this angular velocity in terms of the state variables, it is required a final step of substituting the linear velocity by its formula through the result obtained in (B.23),

$$ \dot{\phi}_i = \frac{1}{r_w}\, {}^{B}\tilde{\boldsymbol{d}}_{tan,i}^{T}\, {}^{B}\tilde{\boldsymbol{v}}_{C_i} = \frac{1}{r_w}\, {}^{B}\tilde{\boldsymbol{d}}_{tan,i}^{T}\, \Gamma\left({}^{B}\tilde{\boldsymbol{p}}_{i,C|B}\right) \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} \left(\Gamma\left({}^{K}_{B}\boldsymbol{q}\right) {}^{K}\tilde{\boldsymbol{\omega}}_{K'|K} - 2\, {}^{K}_{B}\dot{\boldsymbol{q}}\right) \tag{B.41} $$

and the angular velocity of the ball is substituted using (B.9).

$$ \dot{\boldsymbol{\phi}}_{\boldsymbol{i}} = \frac{1}{r_w}\, {}^{B}\tilde{\boldsymbol{d}}_{tan,i}^{T}\, \Gamma\left({}^{B}\tilde{\boldsymbol{p}}_{i,C|B}\right) \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} \left(\Gamma\left({}^{K}_{B}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} - 2\, {}^{K}_{B}\dot{\boldsymbol{q}} \right) \tag{B.42} $$

The angular velocity of the motor shafts has been defined in terms of the state variables.

B.8 Inverse kinematics

The inverse kinematics consists in the mapping from state-space variables, i.e. the generalized coordinates and their derivatives, to actuator variables. The motor axle

angular position is not possible to be derived from the system states, since the mapping between them is non-holonomic. It is non-holonomic since the constraint that relates the rigid bodies is the no-slip condition, which relates velocities. Therefore, the constraint is non-integrable, which means that the integral of the velocities, i.e. the positions, are not related. For this reason, the actuator variables are the angular velocities of the motors and not their angular position.

$$\dot{\boldsymbol{\phi}} = f(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \tag{B.43}$$

The mapping from state variables to actuator variables implies to define it for the angular velocity of each motor shaft. The vector $\dot{\boldsymbol{\phi}}$ from (B.42) is expanded as shown next, where i represents the wheel index as it shown in Figure B.3.

$$\dot{\boldsymbol{\phi}} = \begin{bmatrix} \dot{\phi}_0 \\ \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} = \frac{1}{r_w} {}^{B}\tilde{\boldsymbol{d}}_{tan,i}^{T} \, \Gamma\left({}^{B}\tilde{\boldsymbol{p}}_{C_i|B}\right) \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} \left(\Gamma\left({}^{K}_{B}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} - 2\,{}^{K}_{B}\dot{\boldsymbol{q}} \right) \tag{B.44}$$

For that purpose, the terms in (B.44) that are dependent on the wheel indexes need to be expanded. These terms are grouped in (B.45). They depend on the wheel indexes and not on the state variables, which means that they represent the static, kinematic mapping, i.e. they do not vary with the variation of the state. Conversely, the rest of the terms from (B.44) depends on the states and not on the wheel indexes.

$$\widetilde{\boldsymbol{W}} = \frac{1}{r_w} {}^{B}\tilde{\boldsymbol{d}}_{tan,i}^{T} \, \Gamma\left({}^{B}\tilde{\boldsymbol{p}}_{C_i|B}\right) = \frac{1}{r_w} \begin{bmatrix} 0 & -sin(\gamma \cdot i) & cos(\gamma \cdot i) & 0 \end{bmatrix} \begin{bmatrix} 0 \\ cos(\gamma \cdot i)sin(\alpha) \\ sin(\gamma \cdot i)sin(\alpha) \\ cos(\alpha) \end{bmatrix} \tag{B.45}$$

Computing (B.45) and simplifying yields to

$$\widetilde{\boldsymbol{W}} = \begin{bmatrix} \widetilde{\boldsymbol{W}}_0 \\ \widetilde{\boldsymbol{W}}_1 \\ \widetilde{\boldsymbol{W}}_2 \end{bmatrix} = \frac{r_k}{r_w} \begin{bmatrix} 0 & -cos(\alpha) & 0 & sin(\alpha) \\ 0 & -cos(\gamma)cos(\alpha) & -sin(\gamma)cos(\alpha) & sin(\alpha) \\ 0 & -cos(2\gamma)cos(\alpha) & -sin(2\gamma)cos(\alpha) & sin(\alpha) \end{bmatrix} \tag{B.46}$$

After inserting (B.46) in (B.42), the rest of the terms coincide with the angular

velocity of the ball with respect to the body, (B.44).

$$\dot{\boldsymbol{\phi}} = \widetilde{\boldsymbol{W}}\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\left(\Gamma\left({}^{K}_{B}\boldsymbol{q}\right)\,\frac{1}{r_k}\begin{bmatrix}0\\-\dot{y}\\\dot{x}\\0\end{bmatrix} - 2\,{}^{K}_{B}\dot{\boldsymbol{q}}\right) = \widetilde{\boldsymbol{W}}\;{}^{B}\tilde{\boldsymbol{\omega}}_{K'|B} \tag{B.47}$$

This way, at a static position, i.e. the ball-bot completely vertical and standstill with little wheel angular velocity, the wheel angular velocity vector is defined just in terms of $\tilde{W}$ and the angular velocity of the ball with respect to the inertia frame. This can be seen when applying a standstill state, in which the quaternion is a unit quaternion, i.e. $\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$. When the rotation quaternion has these values is because the orientation of the frame $\{B\}$ is the same as the frame $\{K\}$, there is no inclination of the body. Substituting this, yields to

$$\dot{\boldsymbol{\phi}} \approx \widetilde{\boldsymbol{W}}\,\frac{1}{r_k}\begin{bmatrix}0\\-\dot{y}\\\dot{x}\\0\end{bmatrix} \tag{B.48}$$

B.9 Forward kinematics

Forward kinematics maps the coordinates from the actuator space to the state space. The actuation space of the ball-robot consist of the velocity of the wheels. It is the velocity instead of the position, due to the non-holomonic constraint relating the ball-robot states with the wheel through velocity. Thus, the forward kinematics model consists in defining the translational velocity, $\dot{x}$ and $\dot{y}$, in terms of the wheel angular velocities $\dot{\boldsymbol{\phi}}$, the body orientation and the angular velocity of the body.

$$(\dot{x}, \dot{y}) = f(\dot{\boldsymbol{\phi}}, {}^{K}_{B}\boldsymbol{q}, {}^{K}_{B}\dot{\boldsymbol{q}}) \tag{B.49}$$

The forward kinematics is the opposite of the inverse kinematics. Therefore, it is expected to be derived with the inverse matrix that maps in (B.48) in inverse kinematics. However, since $\tilde{W}$ is not square, it is non-invertible. It can be overcome by removing its first column with zeros and computing with the angular velocity

vector instead of the quaternion of it.

$$\boldsymbol{W} = \widetilde{\boldsymbol{W}} \begin{bmatrix} \mathbf{0}_{1\times 3} \\ \mathbf{I}_3 \end{bmatrix} \tag{B.50}$$

This way, from (B.48) the forward kinematics can be derived by isolating the terms that depend on the states, i.e. the angular velocity of the ball with respect to the body, instead of the angular velocity of the motors that was done to get the inverse kinematics.

$${}^{B}\boldsymbol{\omega}_{K'|B} = \boldsymbol{W}^{-1}\dot{\boldsymbol{\theta}} \tag{B.51}$$

The equation (B.51) needs to be defined in terms of the translational velocity. In (B.21), the angular velocity of the ball can be substituted by (B.9).

$${}^{B}\tilde{\boldsymbol{\omega}}_{K'|B} = {}^{K}_{B}\boldsymbol{q}^{*} \circ \frac{1}{r_k} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} \circ {}^{K}_{B}\boldsymbol{q} - 2\,{}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}_{B}\boldsymbol{q} \tag{B.52}$$

and rearranging the terms,

$${}^{K}_{B}\boldsymbol{q}^{*} \circ \frac{1}{r_k} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} \circ {}^{K}_{B}\boldsymbol{q} = {}^{B}\,\tilde{\boldsymbol{\omega}}_{K'|B} - 2\,{}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}_{B}\boldsymbol{q} \tag{B.53}$$

The quaternion property described in (E.21) and (E.22) is used to multiply both sides of (B.53), by the quaternion to the left and the quaternion conjugate to the right and cancel terms out. Then, the angular velocity of the ball concerning the body can be easily isolated by multiplying by the ball radius.

$$\begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} = r_k\, {}^{K}_{B}\boldsymbol{q} \circ \left({}^{B}\tilde{\boldsymbol{\omega}}_{K'|B} - 2\,{}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}_{B}\boldsymbol{q} \right) \circ {}^{K}_{B}\boldsymbol{q}^{*} \tag{B.54}$$

Finally, rearranging it into (B.49) configuration to isolate each of the components of the ball velocity results in

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = r_k \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} {}^{K}_{B}\boldsymbol{q} \circ \left({}^{B}\tilde{\boldsymbol{\omega}}_{K'|B} - 2\,{}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}_{B}\boldsymbol{q} \right) \circ {}^{K}_{B}\boldsymbol{q}^{*} \tag{B.55}$$

Appendix C

Dynamics

The dynamics of a system describe the relation between the forces applied to the system and its motion. Upon that, the dynamic equation of a system is a mathematical mapping between the forces and the acceleration, and it determines the motion trajectories of the system. If the mapping is done from a known force to an unknown motion is considered forward dynamics. While the mapping in the other way around is called inverse dynamics. In the robotics discipline, it is usually rephrased such that forward dynamics describe the end position of the system given a force, while the inverse dynamics result in the necessary forces to be applied to the system to reach a certain end position of the system. [52] [53]

The dynamic equation of a system can be obtained and expressed through different methods. The method best suited for a system depends on the purpose of the model and the characteristics of the system. However, the most extended methods to describe the behaviour of the system are the Newton-Euler and Euler-Lagrange [54]. A brief overview of both methods is presented bellow:

- **Newton-Euler**: It is derived from the equation of forces from Newton, and the equation of torques from Euler. It provides a dynamic equation for each link of the system analyzed, ending up in a set of differential algebraic equations (DAE). They include the reactions of the links composing the system. Newton-Euler equations are evaluated numerically and recursively with good computational performance.
- **Euler-Lagrange**: It is based in an energy analysis of the system. It analyzes the system composed of multiple bodies as a whole. Euler-Lagrange can be analyzed through symbolic equations. The result of the Euler-Lagrange

method is a closed-form dynamic equation expressed as an Ordinary Differential Equation (ODE). The characteristic form of an ODE follows the structure of (C.1)

[51].

$$\dot{x} = f(x, t) \tag{C.1}$$

It is decided to use Euler-Lagrange to calculate the dynamics due to the final closed-form expression in ODE form. The major advantage is a straightforward calculation of the inverse/forward dynamics as well as a structure fitting the state-space framework used for modern control theory.

C.1 Euler-Lagrange Mechanics

As stated previously, Euler-Lagrange definition of a system is done through the energies involved in it. As the first law of the thermodynamics states, energy can not be created or destroyed in an isolated system, hence, it has to be exchanged between the system and the environment [55]. Upon this definition, it is possible to define two types of energies, conservatives and non-conservatives. The total work done by the conservative forces is zero, that is, they do not change the total amount of energy of the system, i.e. Kinetic Energy and Potential Energy. On the other hand, non-conservative forces produce work, either by introducing energy in the system, i.e. a motor, or by exchanging energy to the environment, i.e. the heat dissipated by friction [56].

The first step to obtain the Euler-Lagrange equations is to construct the Lagrangian of the system, which includes the conservatives energies in it, analyzing them for each of the bodies. The Lagrangian states that the difference between the kinetic and the potential energy of a system must remain constant under no other influence. The Lagranian equation is defined as follows:

$$\mathcal{L} = \mathcal{K} - \mathcal{P} \tag{C.2}$$

Where $\mathcal{K}$ refers to the kinetic energy and $\mathcal{P}$ to the potential energy. For mechanical systems, the kinetic energy can be divided in translational energy and rotational energy (C.3), while the potential energy is composed by the gravitational and the elastic energy (C.4). The terms of the kinetic and the potential energy considered in a system depends on the system properties.

$$K_{trans} = \frac{1}{2}mv^2 \qquad K_{rot} = \frac{1}{2}\omega^T \mathcal{J} \omega \tag{C.3}$$

$$P_{grav} = mgh \qquad P_{elas} = \frac{1}{2}kx^2 \tag{C.4}$$

The ball-robot does not have any component that stores elastic energy. Furthermore, as stated in Section 2.2, it is considered that all the independent parts are rigid bodies, therefore they do not store elastic energy and the elastic potential energy is not considered in the model.

The Euler-Lagrange equation allows to go from the Lagrangian to the equations of motion of the system. It is important to point out that the goal of the Euler-Lagrange equation is to predict the system behaviour. The equations of motion are the terms of the Euler-Lagrange equation collected in such a way that is adequate for computing the states prediction.

Lagrange managed to show that a particle will take a path where the integral of its kinetic energy minus its potential energy, i.e. the Lagrangian, is the lowest [57].

Theory in Calculus of Variations [58] solves this problem, which consists just in optimizing a functional $S[q(t)]$, i.e. finding the values of the system states for which the energy used between t_0 and t_1 is the minimum.

$$S[q(t)] = \int_{t_0}^{t_1} L(t, q, \dot{q}) dt \tag{C.5}$$

The term $q(t)$ represents the system states and $S[q(t)]$ is the action of the system, a functional (function of functions).

To solve it, Calculus of Variations theory finds the minimum or maximum of the functional by doing small changes in the functional, i.e. it changes $q(t)$ by an infinitesimal variation in the coordinates $\eta(t)$ [59] and solves, yielding to

$$\int_{t_1}^{t_2} \left(\frac{\partial L}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{q}} \right) \eta(t) \mathrm{d}t = 0 \tag{C.6}$$

For the left term to be equal to zero, the components inside the parenthesis have to be equal to zero, what yields the Euler-Lagrange equation C.7. This is considered as one of the most important equations that has revolutionized mathematics and

physics [57].

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\chi}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \chi}\right)^T = \mathbf{0} \tag{C.7}$$

Looking at the Euler-Lagrange equation, the units of it can be derived. The first term is going to be non-zero only for the kinetic energy terms as the potential energy does not depend on the velocity. The derivative in velocity of the kinetic energy removes the quadratic term of the velocity, and then the derivative in time yields to acceleration. E.g. for the kinetic energy of a translational movement, it yields to $m\dot{a}$, which represents by the Second Newton Law, a force. This can be also checked with the potential energy, which will give a non-zero term in the second term of (C.7), resulting in $m\dot{g}$, which is the force due to the weight. Therefore, the Euler-Lagrange equation has force units and it is a tool to get balance in forces of the system by defining the balance in terms of energies.

Since the Euler-Lagrange predicts the trajectory of the system, i.e. the future states values, the constraints of the system that limits the possible solutions of the trajectory must be introduced into the Euler-Lagrange equation to be taken into account for the calculation of the trajectory solution. External forces, such as friction and input forces, must be included also due to their influence in defining the trajectory of the system.

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\chi}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \chi}\right)^T + \boldsymbol{D} - \boldsymbol{H}_{hol}^T \boldsymbol{\lambda}_{hol} + \boldsymbol{H}_{non}^T \boldsymbol{\lambda}_{non} = \boldsymbol{Q} \tag{C.8}$$

Where $\boldsymbol{D}$ includes the non-conservative forces arising from friction, $\boldsymbol{H}_{hol}$ and $\boldsymbol{H}_{non}$ refers to the Jacobians of holomonic and non-holomonic constraints respectively, both multiplied by their correspondent Lagrange multiplier. Finally, Q accounts for the input forces to the system. Every term inside C.8 has to be defined in the generalized coordinates.

Solving (C.8) and expanding its terms, the closed-form Euler-Lagrange equation is derived so it will be used to obtain the ODE of the system,

$$\boldsymbol{M}(\chi)\ddot{\chi} + \boldsymbol{C}(\chi, \dot{\chi})\dot{\chi} + \boldsymbol{G}(\chi) + \boldsymbol{D}(\dot{\chi}) - \boldsymbol{H}_{hol}^T(\chi)\boldsymbol{\lambda}_{hol} + \boldsymbol{H}_{non}^T(\chi)\boldsymbol{\lambda}_{non} = \boldsymbol{Q}(\chi, \dot{\chi}, \boldsymbol{u}) \tag{C.9}$$

where:

- $\boldsymbol{M}(\chi)$ is the mass/inertia matrix. It is multiplied by the acceleration, thus, it comes from the velocity related terms in the Lagrange, i.e. from the kinetic energy terms.
- $\boldsymbol{C}(\chi, \dot{\chi})$ is the Coriolis and centrifugal force matrix.
- $\boldsymbol{G}(\chi)$ corresponds to the gravitational matrix, hence, comes from the potential energy of the system. The potential energy varies depending on the altitude, thus, the orientation of the body, therefore, it is influenced by the quaternion constraint.

The closed form Euler-Lagrange equation can be solved to isolate the acceleration. The structure of the state-space model requires to define the derivative of the system states in terms of the states and the inputs. The derivative of the states are $\dot{\chi}$ and $\ddot{\chi}$, represented by $\dot{\boldsymbol{x}}$. Since $\ddot{\chi}$ is obtained from the Euler-Lagrange, another equation in ODE form is introduced to define $\dot{\chi}$, in this case $\dot{\chi} = \dot{\chi}$.

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\chi} \\ -\tilde{\boldsymbol{M}}(\chi)^{-1}\left(\tilde{\boldsymbol{C}}(\chi, \dot{\chi})\dot{\chi} + \tilde{\boldsymbol{G}}(\chi) + \tilde{\boldsymbol{D}}(\dot{\chi})\right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \tilde{\boldsymbol{M}}(\chi)^{-1}\tilde{\boldsymbol{Q}}(\chi, \dot{\chi}, \boldsymbol{u}) \end{bmatrix} \tag{C.10}$$

where

$$\dot{\boldsymbol{x}} = \frac{d}{dt}\boldsymbol{x} = \frac{d}{dt}\begin{bmatrix} \chi \\ \dot{\chi} \end{bmatrix} = \begin{bmatrix} \dot{\chi} \\ \ddot{\chi} \end{bmatrix} \tag{C.11}$$

As a remark, to reach the state-space alike expression, it is necessary to eliminate the Lagrangian multipliers in (C.68). To do so, its effect must be embedded inside the rest of the terms, leading to new matrices denoted as the original matrix with a tilde, as seen in (C.10). This is applied in Section C.4.1 in order to leave the system ready to form the state-space of the system to be used for controlling purposes.

C.2 Energy equations of the ball-robot

The goal of this section is to define the energy equations of each part of the system using the generalized coordinates defined in Section 2.5. As defined previously, for energy purposes, the ball-robot is composed by three subsystems: the ball, the wheels and the body. The mass of the wheels and the motors is included in the body total mass in order to define the body and wheels kinetic energies together. However, the wheels have an additional rotational kinetic energy from the body

one. Furthermore, they have different angular velocities and directions between each other. As a consequence, it is necessary to define a specific energy expression that contemplates their rotational kinetic energy that is not included in the body kinetic energy. The energy equations for each of the subsystems follows (C.3) and (C.4).

C.2.1 Ball energy

From the definition of the kinetic energy, viewed from the inertial frame, the ball energy equation yields to

$$K_k = \frac{1}{2}\left({}^{I}\boldsymbol{v}_{K|I}^{T}\boldsymbol{M}^{I}\boldsymbol{v}_{K|I} + {}^{I}\boldsymbol{\omega}_{K'|I}^{T}\boldsymbol{\mathcal{J}}^{I}\boldsymbol{\omega}_{K'|I}\right) \tag{C.12}$$

The translational velocity of the ball is expressed in generalized coordinates through the use of (B.3) while their angular velocity is substituted by using (B.9).

$$K_k = \frac{1}{2}\left(\begin{bmatrix}\dot{x}\\ \dot{y}\\ 0\end{bmatrix}^{T}\begin{bmatrix}M_k & 0 & 0\\ 0 & M_k & 0\\ 0 & 0 & M_k\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ 0\end{bmatrix} + \frac{1}{r_k}\begin{bmatrix}-\dot{y}\\ \dot{x}\\ 0\end{bmatrix}^{T}\begin{bmatrix}\mathcal{J}_{kx} & 0 & 0\\ 0 & \mathcal{J}_{ky} & 0\\ 0 & 0 & \mathcal{J}_{kz}\end{bmatrix}\frac{1}{r_k}\begin{bmatrix}-\dot{y}\\ \dot{x}\\ 0\end{bmatrix}\right) \tag{C.13}$$

Collecting terms from $\dot{x}$ and $\dot{y}$,

$$K_k = \frac{1}{2}\left(\left(M_k + \frac{\mathcal{J}_{ky}}{r_k^2}\right)\dot{x}^2 + \left((M_k + \frac{\mathcal{J}_{kx}}{r_k^2}\right)\dot{y}^2\right) \tag{C.14}$$

The inertia tensor $\mathcal{J}$ is the result of assuming that the ball is a perfect sphere and has the mass perfectly distributed, hence, its inertia matrix is a diagonal, and $\mathcal{J}_{kx} = \mathcal{J}_{ky} = \mathcal{J}_{kz}$.

As for the potential energy of the ball, in Section 2.2 it is assumed the ball not displace in the z axis direction. Hence, assuming that point as the zero potential energy reference point, the ball possesses no potential energy.

C.2.2 Body energy

The body possesses both kinetic and potential energy. Its kinetic energy is defined as

$$K_b = \frac{1}{2}\left(M_b {}^{I}\boldsymbol{\nu}_{B'|I}^{T} {}^{I}\boldsymbol{\nu}_{B'|I} + {}^{I}\boldsymbol{\omega}_{B'|I}^{T} \begin{bmatrix} \mathcal{J}_{bx} & 0 & 0 \\ 0 & \mathcal{J}_{by} & 0 \\ 0 & 0 & \mathcal{J}_{bz} \end{bmatrix} {}^{I}\boldsymbol{\omega}_{B'|I} \right) \tag{C.15}$$

The angular velocity of the body is expressed in quaternions, which has four parameters, while the inertia matrix is composed by three, hence, it is required to expand the inertia matrix to match the dimensions.

$${}^{B}\widetilde{\mathcal{J}}_b = \begin{bmatrix} 1 & 0 \\ 0 & {}^{B}\mathcal{J}_b \end{bmatrix} \tag{C.16}$$

Substituting the linear velocity of the body (B.15) and the angular velocity (B.16) yields to

$$K_b = \frac{1}{2}\left(M_b \left(\begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + {}^{K}\boldsymbol{\nu}_{B'} \right)^{T} \left(\begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + {}^{K}\boldsymbol{\nu}_{B'} \right) + \left(2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} {}^{K}_{B}\boldsymbol{q} \right)^{T} {}^{B}\widetilde{\mathcal{J}}_b \left(2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} {}^{K}_{B}\boldsymbol{q} \right) \right) \tag{C.17}$$

Expanding and substituting the ball velocity (B.13) yields to

$$\begin{aligned} K_b = \frac{1}{2} M_b \Bigg(\dot{x}^2 + \dot{y}^2 + 2\Pi_{B'}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right) {}^{K}_{B}\boldsymbol{q}^{T} 2\Pi_{B'}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right) {}^{K}_{B}\boldsymbol{q} + 2 \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}^{T} \vee 2\Pi_{B'}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right) {}^{K}_{B}\boldsymbol{q} \Bigg) \\ + \frac{1}{2}\left(2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} {}^{K}_{B}\boldsymbol{q} \right)^{T} {}^{B}\widetilde{\mathcal{J}}_b \left(2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T} {}^{K}_{B}\boldsymbol{q} \right) \end{aligned} \tag{C.18}$$

It is assumed the center of mass of the body to be located in the center of the axis with a symmetrically distributed mass around the axis. Due to the cylindrical representation of the body, and the assumed distribution of masses: $\mathcal{J}_{bx} = \mathcal{J}_{by} \neq \mathcal{J}_{bz}$.

Unlike the ball, for the body, the potential energy is not null and it varies with the change of the inclination of the body center of mass height.

$$P_b = M_b g h_b \tag{C.19}$$

For computing the potential energy of the body, the body center of mass height h_b is required, computed from the center of the ball, which has been previously declared as the potential reference point. The z component in the $\{K\}$ reference frame from the body center of mass vector, derived in (B.14), is the body vertical distance from the xy plane. Where $\boldsymbol{k}$ is a vector defined to extract the component z from (B.14).

$$h_b = \boldsymbol{k}^{T\,K}\boldsymbol{p}_{B'} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T \vee \left(\Phi({}^K_B\boldsymbol{q})\,\Gamma({}^K_B\boldsymbol{q})^{T\,B}\tilde{\boldsymbol{p}}_{B'}\right) \tag{C.20}$$

Finally, substituting the (C.20), the potential energy of the body is defined.

$$P_b = M_b g \vec{\boldsymbol{k}}^T \vee \left(\Phi({}^K_B\boldsymbol{q})\,\Gamma({}^K_B\boldsymbol{q})^{T\,B}\tilde{\boldsymbol{p}}_{B'}\right) \tag{C.21}$$

C.2.3 Wheel energy

As previously mentioned, the mass of the wheels is included in the body mass, therefore, their potential energy is already included inside the potential energy of the body. In the same way, since there is no relative translational movement from the wheels to the ball, the mass of the wheels can be included in the body to compute the translational kinetic energy together as the translational velocity is the same for both.

$$K_{w_i} = \frac{1}{2}\dot{\boldsymbol{\phi}}_i^T \boldsymbol{\mathcal{J}}_{wi}\dot{\boldsymbol{\phi}}_i \quad \text{for} \quad i = 0, 1, 2 \tag{C.22}$$

The inertias of each of the wheels are composed by the wheel inertia and the motor inertia.

$$\boldsymbol{\mathcal{J}}_{wi} = \begin{bmatrix}\mathcal{J}_{wix} & 0 & 0\\ 0 & \mathcal{J}_{wiy} & 0\\ 0 & 0 & \mathcal{J}_{wiz}\end{bmatrix} \tag{C.23}$$

Substituting the angular velocity of the wheel from (B.47) and expanding the kinetic energy for each of the wheels, the equation is defined.

$$
\begin{aligned}
K_{wi} = & \frac{1}{2r_k^2} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix}^T \mathbf{\Gamma} \left({}^K_B\boldsymbol{q}\right)^T \mathbf{\Phi} \left({}^K_B\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^T \boldsymbol{\mathcal{J}}_{wi} \widetilde{\boldsymbol{W}} \mathbf{\Phi} \left({}^K_B\boldsymbol{q}\right)^T \mathbf{\Gamma} \left({}^K_B\boldsymbol{q}\right) \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix} + \\
& 2 {}^K_B\dot{\boldsymbol{q}}^T \mathbf{\Phi} \left({}^K_B\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^T \boldsymbol{\mathcal{J}}_{wi} \widetilde{\boldsymbol{W}} \mathbf{\Phi} \left({}^K_B\boldsymbol{q}\right)^T {}^K_B\dot{\boldsymbol{q}} - \\
& 2 \frac{1}{r_k^2} \begin{bmatrix} 0 \\ -\dot{y} \\ \dot{x} \\ 0 \end{bmatrix}^T \mathbf{\Gamma} \left({}^K_B\boldsymbol{q}\right)^T \mathbf{\Phi} \left({}^K_B\boldsymbol{q}\right) \widetilde{\boldsymbol{W}} \boldsymbol{\mathcal{J}}_{wi} \widetilde{\boldsymbol{W}} {}^K_B\dot{\boldsymbol{q}}
\end{aligned}
\tag{C.24}
$$

C.3 Euler-Lagrange equation

Once the energy terms are defined, the Lagrangian is constructed.

$$
L = K_k + K_b + K_{w_0} + K_{w_1} + K_{w_2} - P_b \tag{C.25}
$$

The insertion of the Lagrangian in (C.7) leads to a system of six equations corresponding to the Euler-Lagrange equation for each of the generalized coordinates.

Solving the derivatives and collecting the terms yields the first part of the Euler-Lagrange equation (C.26), where only the conservative forces from the system are stated.

$$
\boldsymbol{M}(\chi)\ddot{\chi} + \boldsymbol{C}(\chi, \dot{\chi})\dot{\chi} + \boldsymbol{G}(\chi) = 0_{6\times 1} \tag{C.26}
$$

A mathematical software using symbolic equations is used to solve for the Lagrangian derivatives. This eases the process and avoids mistakes provoked by the process of expanding the terms. From the solution, the terms are collected in three groups: the terms influenced by the second derivative of the generalized coordinates compose the matrix $\boldsymbol{M}$ and are extracted by deriving by $\ddot{\chi}$; the terms influenced by the first derivative of the generalized coordinates form the matrix $\boldsymbol{C}$ and are collected through deriving by $\dot{\chi}$; finally, the rest of the terms are the matrix $\boldsymbol{G}$.

In (C.26) is contemplated the energy balance of the system without considering external forces or energy lost, hence, it is required to include them in the equation.

C.3.1 Input forces

The system is controlled by the use of motors transmitting force into the system and providing the desired motion. However, each of the motors produce a torque in its correspondent plane. To be able to introduce the input forces into the Euler-Lagrange equation each of the torques has to be related to their influence to the system using the same variables. In other words, it is required to map the torque of each motor to the generalized coordinates, to see the combination of system states they affect to. The mentioned map can be achieved by starting from the virtual work principle described by D'Alemberts principle and its relation with virtual displacement [60] [61].

$$\partial \mathcal{W} = \sum_i F_i \partial r \tag{C.27}$$

Where this states that the total virtual work input to the system is the sum of the external forces applied to the system times their consequent displacements in the force coordinate. In the ball-robot system, the external forces i refers to as the linear force generated by the torque of each motor, and ∂r represents the displacement along the motor coordinate.

Using the conservation of work principle and through Hamilton principle of generalized forces, it is possible to equate the virtual work produced by the input force in any coordinate system to forces expressed in the generalized coordinates.

$$\boldsymbol{F}^{(g)}\, \partial \chi = \boldsymbol{F}^{(ng)}\, \partial r \tag{C.28}$$

In which $\boldsymbol{F}^{(g)}$ and $\boldsymbol{F}^{(ng)}$ represent the forces in generalized coordinates and coordinates of the force (non-generalized coordinates) respectively and, χ and r the generalized coordinates and any non-generalized coordinate respectively. Restructuring the equation, it is obtained the expression for the generalized forces.

$$\boldsymbol{F}^{(g)} = \left(\frac{\partial r}{\partial \chi} \right)^T \boldsymbol{F}^{(ng)} \tag{C.29}$$

The mapping between angular velocity of the motor in the motor reference frame

to velocity in the generalized coordinates have been defined in (B.47). However, the principle of virtual work equilibrium (C.28) is defined by positions and not velocities. In (C.29), the jacobian $\frac{\partial r_j}{\partial \chi_j}$ is in charge of mapping the forces from the position in the coordinates of the force to the position in the generalized coordinates. Therefore, to make use of the kinematic relationship from (B.47), it is necessary to make a relation between the position and the velocity coordinates[6].

Considering the existence of a matrix which maps from the motor coordinates r to the generalized coordinates of the robot χ of the kind

$$\boldsymbol{r} = \boldsymbol{R}(\chi) \tag{C.30}$$

The time derivative is applied to both sides of the mapping position equation in order to get one in term of velocities.

$$\frac{\partial \dot{\boldsymbol{r}}}{\partial t} = \frac{\partial \boldsymbol{R}(\chi)}{\partial \boldsymbol{t}} \tag{C.31}$$

The chain rule needs to be applied to the right side to be able to derive it in terms of time since χ depends on time.

$$\frac{\partial \boldsymbol{R}(\chi)}{\partial \boldsymbol{t}} = \frac{\partial \boldsymbol{R}}{\partial \chi} \frac{\partial \chi}{\partial t} = \frac{\partial \boldsymbol{R}}{\partial \chi} \dot{\chi} \tag{C.32}$$

The combination of (C.31) and (C.32) shows that $\frac{\partial \boldsymbol{R}}{\partial \chi}$, i.e. the jacobian of R, is the mapping from the derivative of generalized coordinates to the derivative of the coordinates in which the force is applied.

$$\dot{\boldsymbol{r}} = \frac{\partial \boldsymbol{R}}{\partial \chi} \dot{\chi} \tag{C.33}$$

However, taking partial derivative with respect to $\dot{\chi}$ in both sides yields to

$$\frac{\partial \dot{\boldsymbol{r}}}{\partial \dot{\chi}} = \frac{\partial \boldsymbol{R}}{\partial \chi} \tag{C.34}$$

that shows that the mapping in (C.33) can be done through the Jacobian in terms of velocities of the coordinates in which the force is applied.

Furthermore, from (C.30) taking the derivative with respect to χ in both sides, the relation in (C.35) can be extended to

$$\frac{\partial \boldsymbol{r}}{\partial \chi} = \frac{\partial \boldsymbol{R}}{\partial \chi} \implies \frac{\partial \dot{\boldsymbol{r}}}{\partial \dot{\chi}} = \frac{\partial \boldsymbol{R}}{\partial \chi} = \frac{\partial \boldsymbol{r}}{\partial \chi} \tag{C.35}$$

Here is where the tool derived from Hamilton's principle can be seen, it allows to map a force from the coordinates in which is applied to the generalized coordinates. It is concluded that this mapping could be done through (C.29), but it is more useful to use the relation derived in the steps above, which uses the derivative of the coordinates of the force with respect to the derivative of the generalized coordinates for the mapping, as shown next

$$\boldsymbol{F}^{(g)} = \left(\frac{\partial \dot{\boldsymbol{r}}}{\partial \dot{\chi}}\right)^T \boldsymbol{F}^{(ng)} \tag{C.36}$$

Therefore, applying (C.36), it is possible to define the mapping to convert the motor torque in the motor coordinates to the force applied by the motor in the generalized coordinates. To do so, the gradient of the velocity coordinates of the motor (B.47) with respect to the velocity of the generalized coordinates $\dot{\chi}$ is calculated

$$\boldsymbol{J}_m^T\left({}^K_B\boldsymbol{q}\right) = \frac{\partial \dot{\boldsymbol{\phi}}}{\partial \dot{\chi}} = \begin{bmatrix} \frac{1}{r_k}\boldsymbol{j}^T\Gamma\left({}^K_B\boldsymbol{q}\right)^T \\ -\frac{1}{r_k}\boldsymbol{i}^T\Gamma\left({}^K_B\boldsymbol{q}\right)^T \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^K_B\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^T \tag{C.37}$$

Where the vectors $\boldsymbol{i}$ and $\boldsymbol{j}$ are the unitary direction vectors $\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^T$ and $\begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}^T$ respectively, used to extract the respective coordinate of the quaternion. As mentioned, the mapping is applied to the motor torque to express it in the generalized coordinates, in order to introduce this input force in the Euler-Lagrange equation.

$$\boldsymbol{Q}(\chi, \boldsymbol{\tau}_m) = \boldsymbol{J}_m^T\left({}^K_B\boldsymbol{q}\right)\boldsymbol{\tau}_m \tag{C.38}$$

in which $\boldsymbol{\tau}_m$ represents the input torques from the motors defined in the motor frames.

C.3.2 Friction forces

The input forces previously defined are non-conservative forces that introduce energy into the system. However, there are also non-conservative forces that dissipates energy from the system, these come in the form of friction forces.

Modeling the friction forces of a system is a trade-off to be assessed, the better they are modelled the more precise the system is. Nonetheless, there is a compromise between complexity and precision.

The friction forces can be classified in different types, and the most common ones found in modelling are the viscous, the Coulomb and the Stiction frictions.

The only friction considered in the model is the viscous friction. It is decided to use the viscous friction due to being a linear function with respect the velocity of the system, hence it does not adds more non-linearities to the model while representing the major effect of the frictions of the system.

- Viscous friction arising from the ball in contact with the ground. This friction allows the rotation of the ball. It is denoted as $\boldsymbol{D}_k$.
- Viscous friction from the body with the air denoted as $\boldsymbol{D}_B$.
- Viscous friction in the motors and the wheels, $\boldsymbol{D}_M$.

The viscous friction is a linear relation between the velocity and the viscous coefficient.

$$f(x)_{vis} = B\frac{dx}{dt} \tag{C.39}$$

Where the units of B are, in the SI, $\frac{Ns}{m}$ [62], whereas if it is used in a rotational movement, the viscous coefficient units modifies to Nms.

Viscous friction in the ball

The viscous friction arising from the ball translation enables the rotation movement. Due to the assumption of no slip, the viscous friction from the ball is directly related to the velocity of the generalized coordinates over the plane xy, hence using

(C.39) the friction forces caused by the ball against the ground are defined by the translation velocity of itself.

$$\boldsymbol{D}_k(\dot{\boldsymbol{\chi}}) = B_{vk} \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{0}_{2\times 4} \\ \boldsymbol{0}_{4\times 2} & \boldsymbol{0}_{4\times 4} \end{bmatrix} \dot{\boldsymbol{\chi}} \tag{C.40}$$

Viscous friction in the body

Body friction arises from the contact between the air and the body. The friction originates by linear and rotational velocities, however, the body is modelled as a cylinder, and due to it, it generates an air profile which reduces the friction due to linear velocity and it is assumed to be negligible, for this shake, the model only includes the viscous friction caused by rotation velocity. To make use of the friction force in the Euler-Lagrange it is required to express it in the generalized coordinates (B.16). However, the resulting friction forces are being applied in the body coordinate frame, whereas the Euler-Lagrange equation is expressed in the generalized coordinates. To account for this difference, the principle of work conservation and Hamilton are used, as explained and described in the Section C.3.1 and Equation (C.36). This principles dictates that through the following Jacobian the friction forces in the body coordinates can be related to the generalized coordinates.

$$\boldsymbol{J}_b\left({}^{K}_{B}\boldsymbol{q}\right) = \frac{\partial\, {}^{I}\tilde{\boldsymbol{\omega}}_{B|I}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} 0 & 0 & 2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^T \end{bmatrix} \tag{C.41}$$

Introducing the mapping in (C.39) yields to

$$\begin{aligned} \boldsymbol{D}_b(\dot{\boldsymbol{\chi}}) &= B_{vb} \boldsymbol{J}_b^{T^I} \tilde{\boldsymbol{\omega}}_{B|I} \\ &= B_{vb} \begin{bmatrix} 0 & 0 & 2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^T \end{bmatrix}^T 2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^T {}^{K}_{B}\dot{\boldsymbol{q}} \\ &= 4 B_{vb} \begin{bmatrix} \boldsymbol{0}_{2\times 2} & \boldsymbol{0}_{2\times 4} \\ \boldsymbol{0}_{4\times 2} & \boldsymbol{I}_4 \end{bmatrix} \dot{\boldsymbol{\chi}} \end{aligned}$$

Viscous friction in the motors and wheels

Due to the movement of the motor shaft which is transmitted to the wheels, friction forces are generated. The friction coefficient in this case B_{vm}, is composed by the

addition of two, the friction coefficient from the motor shaft, and the friction coefficient corresponding to the contact between the wheel and the ball. It is possible to do so, since the motor and the wheel rotate at the same angular velocity.

The Hamilton principle stated in (C.36) is applied again to map the friction forces to the generalized coordinates, through the Jacobian from (C.37)

$$\boldsymbol{D}_m(\dot{\boldsymbol{\chi}}) = \boldsymbol{J}_m^T B_{vm} \dot{\boldsymbol{\phi}} \tag{C.42}$$

Total viscous friction

The friction forces are combined in a single term and included into the dynamic equation.

$$\boldsymbol{D}(\dot{\boldsymbol{\chi}}) = \boldsymbol{D}_k + \boldsymbol{D}_b + \boldsymbol{D}_m \tag{C.43}$$

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \boldsymbol{G}(\boldsymbol{\chi}) + \boldsymbol{D}(\dot{\boldsymbol{\chi}}) = \boldsymbol{Q}(\boldsymbol{\chi})\tau_m \tag{C.44}$$

Leading to a system of six equations and six variables. However, the ball-robot consists of five degrees of freedom, two in translation and three in rotations. The rotations are described with four variables by the quaternions, which means that there is an extra variable and the system is unconstrained. Hence, it is required to include a constraint in the system to limit the possible trajectories that the equations of motion can lead to.

C.4 Quaternion constraints

The system is over-determined and therefore unconstrained due to fact that quaternion representation requires four parameters to represent three degrees of freedom. A quaternion that represents a rotation must have unit norm. This condition determines the constraint that must be added to the system. Unit norm implies the following relationship

$$\| {}^K_B\boldsymbol{q} \| = 1 \Rightarrow {}^K_B\boldsymbol{q}^T {}^K_B\boldsymbol{q} = 1 \tag{C.45}$$

Which, defined as an holomonic constraint is rewritten as:

$$h_q(\chi) = {}^{K}_{B}\boldsymbol{q}^{T}\,{}^{K}_{B}\boldsymbol{q} - 1 = 0 \tag{C.46}$$

The unit constraint is included alongside the Euler-Lagrange equation.

$$\begin{aligned}\boldsymbol{M}(\chi)\ddot{\chi} + \boldsymbol{C}(\chi,\dot{\chi})\dot{\chi} + \boldsymbol{G}(\chi) + \boldsymbol{D}(\dot{\chi}) &= \boldsymbol{Q}(\chi)\tau_m \\ {}^{K}_{B}\boldsymbol{q}^{T}\,{}^{K}_{B}\boldsymbol{q} - 1 &= 0\end{aligned}$$

With it, the system becomes a set of seven Differential Algebraic Equations (DAE). The goal of the model is to be used to implement control strategies using the state-space of the system, hence it is desired to construct the model in an Ordinary Differential Equation form. In order to do so, it is possible to include the constraint equation embedded in the original Euler-Lagrange equation. The holomonic constraint is embedded through the use of Lagrange multipliers.

C.4.1 Quaternions dynamics

The holomonic constraint is included into the Euler-Lagrange through the Jacobian of the original constraint.

$$\boldsymbol{H}_q^T(\chi) = \frac{\partial h_q(\chi)}{\partial \chi}^T = \begin{bmatrix} 0 \\ 0 \\ 2\,{}^{K}_{B}\boldsymbol{q} \end{bmatrix} \tag{C.47}$$

Then, the Jacobian of the holomonic constraint can be embedded in the Euler-Lagrange equation by multiplying it by a new variable λ, the Lagrange multiplier. The set of Lagrange multiplier with the Jacobian represent the reaction forces in the generalized coordinates.

$$\boldsymbol{M}(\chi)\ddot{\chi} + \boldsymbol{C}(\chi,\dot{\chi})\dot{\chi} + \boldsymbol{G}(\chi) + \boldsymbol{D}(\dot{\chi}) - \boldsymbol{H}_q^T(\chi)\boldsymbol{\lambda} = \boldsymbol{Q}(\chi)\tau_m \tag{C.48}$$

After embedding the constraint, the system consists of six equations and seven variables, hence, it is desired to get rid of the Lagrange multiplier, for that, the

inverse of the constraint Jacobian is defined.

$$\overline{\boldsymbol{H}}_q = \begin{bmatrix} \boldsymbol{I_2} & \boldsymbol{0}_{2\times 4} \\ \boldsymbol{0}_{4\times 2} & \boldsymbol{\Phi}\left({}^{K}_{B}\boldsymbol{q}\right)^T \end{bmatrix} \tag{C.49}$$

Where combining the Jacobian of the constraint with its inverse yields to the identity property E.22, leading to

$$\overline{\boldsymbol{H}}_q \boldsymbol{H}_q^T \boldsymbol{\lambda} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}^T \boldsymbol{\lambda} \tag{C.50}$$

Then, the inverse of the Jacobian is introduced in (C.48) and the Lagrange multiplier is isolated

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \boldsymbol{\lambda} = \frac{1}{2}\left(\overline{\boldsymbol{H}}_q \boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \overline{\boldsymbol{H}}_q \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}} + \overline{\boldsymbol{H}}_q \boldsymbol{G}(\boldsymbol{\chi}) + \overline{\boldsymbol{H}}_q \boldsymbol{D}(\dot{\boldsymbol{\chi}}) + \overline{\boldsymbol{H}}_q \boldsymbol{Q}(\boldsymbol{\chi})\tau_m\right) \tag{C.51}$$

It is observed that the Lagrange multiplier effect affects only to the third equation, while the effect of the constraint is embedded in the system of equations. Hence, it is possible to get rid of the extra variable, the Lagrange multiplier, by multiplying by zero the third equation. A new matrix is defined to couple this with $\overline{\boldsymbol{H}}_q$ and eliminate the third equation.

$$\widetilde{\boldsymbol{H}} = \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{0}_{2\times 1} & \boldsymbol{0}_{2\times 3} \\ \boldsymbol{0}_{3\times 2} & \boldsymbol{0}_{3\times 1} & \boldsymbol{I}_3 \end{bmatrix} \overline{\boldsymbol{H}}_q = \begin{bmatrix} \boldsymbol{I}_2 & & \boldsymbol{0}_{2\times 4} \\ \boldsymbol{0}_{3\times 2} & \begin{bmatrix} \boldsymbol{0}_{3\times 1} & \boldsymbol{I}_3 \end{bmatrix} \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^T \end{bmatrix} \tag{C.52}$$

By introducing the new matrix in the system the Euler-Lagrange equation takes the form of a set of five ordinary differential equations without the Lagrange multiplier.

C.4.2 Kinematic constraint enforcement

The system is still unconstrained and unsolvable due to having five DoF, five equations but six variables after the elimination of the Lagrange multiplier. Therefore,

it is required to include another equation to constrain the system and to obtain a set of equations equal to the set of variables.

The quaternion constraint is in the form of a kinematic relation. To be reused to obtain the sixth equation of the system, it needs to be expressed dynamically. Therefore, the second order derivative of the constraint must be derived.

$$h_q(\chi) = {}^K_B\boldsymbol{q}^{TK}_B\boldsymbol{q} - 1 \tag{C.53}$$

$$\downarrow \frac{d}{dt}$$

$$\dot{h}_q(\chi) = {}^K_B\dot{\boldsymbol{q}}^{TK}_B\boldsymbol{q} + {}^K_B\boldsymbol{q}^{TK}_B\dot{\boldsymbol{q}} = 2^K_B\dot{\boldsymbol{q}}^{TK}_B\boldsymbol{q} \tag{C.54}$$

$$\downarrow \frac{d}{dt}$$

$$\ddot{h}_q(\chi) = 2^K_B\ddot{\boldsymbol{q}}^{TK}_B\boldsymbol{q} + 2^K_B\dot{\boldsymbol{q}}^{TK}_B\dot{\boldsymbol{q}} \tag{C.55}$$

To include the second order derivative of the constraint into the Euler-Lagrange equation it is required to express it in the generalized coordinates to maintain the structure of the ODE. Hence, a new matrix is defined to express it.

$${}^K_B\boldsymbol{q} = \boldsymbol{R}\chi \tag{C.56}$$

$$\boldsymbol{R}\begin{bmatrix}\boldsymbol{0}_{4\times 2}\boldsymbol{I}_4\end{bmatrix} \tag{C.57}$$

$$\boldsymbol{R}^T\boldsymbol{R} = \begin{bmatrix}\boldsymbol{0}_{2\times 2} & \boldsymbol{2\times 4}\\ \boldsymbol{4\times 2} & \boldsymbol{I}_4\end{bmatrix} \tag{C.58}$$

The defined transformation (C.56) and its derivatives are substituted into (C.55) in order to define it in the generalized coordinates

$$\chi^T\boldsymbol{R}^T\boldsymbol{R}\ddot{\chi} + \dot{\chi}^T\boldsymbol{R}^T\boldsymbol{R}\dot{\chi} = \boldsymbol{0} \tag{C.59}$$

Introducing this constraint equation in the Euler-Lagrange and including the transformation from (C.52) the model of the ball-robot is obtained.

$$\begin{bmatrix}\widetilde{\boldsymbol{H}}\boldsymbol{M}(\chi)\\ \chi^T\boldsymbol{R}^T\boldsymbol{R}\end{bmatrix}\ddot{\chi} + \begin{bmatrix}\widetilde{\boldsymbol{H}}\boldsymbol{C}(\chi,\dot{\chi})\\ \dot{\chi}^T\boldsymbol{R}^T\boldsymbol{R}\end{bmatrix}\dot{\chi} + \begin{bmatrix}\widetilde{\boldsymbol{H}}\boldsymbol{G}(\chi)\\ 0\end{bmatrix} + \begin{bmatrix}\widetilde{\boldsymbol{H}}\boldsymbol{D}(\dot{\chi})\\ 0\end{bmatrix} = \begin{bmatrix}\widetilde{\boldsymbol{H}}\boldsymbol{Q}(\chi)\\ \boldsymbol{0}_{1\times 3}\end{bmatrix}\boldsymbol{\tau}_m \tag{C.60}$$

C.4.3 Constraint drift regularization

The model is used for simulation purposes, and hence, it is required to be aware of a phenomenon happening in simulations. Simulations are solved through numerical methods. Numerical methods compute iterative solutions over the change in time. However, due to the continuity of time in physical systems, and the discontinuity of the numerical methods, the results of the numerical methods tend to take distance from the real solution. This effect is called drifting [63].

To account for the drift, Baumgarte's technique is used [64]. Baumgarte states that if the constraint and its derivatives are equal to zero, the sum of each of them remains zero. A proper choice of the value of the coefficients makes the solution bounded and stable, eliminating the drifting

$$\ddot{h} + 2\alpha\dot{h} + \beta^2 h = 0 \tag{C.61}$$

The solution of the previous equations yields to poles of the system, which, expressed in the time domain leads to

$$h = a_1 e^{s_1 t} + a_2 e^{s_2 t} \tag{C.62}$$

where the poles s_1 and s_2 are placed based on the values of α and β [65].

$$s_1, s_2 = -\alpha \pm \sqrt{\alpha^2 - \beta^2} \tag{C.63}$$

From (C.63) can be derived that if $\alpha = \beta$, s_1 and s_2 will always be placed in the left half of the s-plane, also making the system critically damped, which gives the fastest convergence. Applying $\alpha = \beta$ leads to

$$\ddot{h}_q(\chi) + 2\beta\dot{h}_q(\chi) + \beta^2 h_q(\chi) = 0 \tag{C.64}$$

Substituting the constraints, yields to

$$2{}^K_B\boldsymbol{q}^{TK}_B\ddot{\boldsymbol{q}} + 2{}^K_B\dot{\boldsymbol{q}}^{TK}_B\dot{\boldsymbol{q}} + 2\beta 2{}^K_B\boldsymbol{q}^{TK}_B\dot{\boldsymbol{q}} + \beta^2({}^K_B\boldsymbol{q}^{TK}_B\boldsymbol{q} - 1) = 0 \tag{C.65}$$

By expressing the constraint enforcement in the generalized coordinates through (C.57), the constraint enforcement equation is rewritten.

$$\chi^T \boldsymbol{R}^T \boldsymbol{R} \ddot{\chi} + \dot{\chi}^T \boldsymbol{R}^T \boldsymbol{R} \dot{\chi} + 2\beta \chi^T \boldsymbol{R}^T \boldsymbol{R} \dot{\chi} + \frac{1}{2}\beta^2 (\chi^T \boldsymbol{R}^T \boldsymbol{R} \chi - 1) = 0 \tag{C.66}$$

Finally, the constraint enforcement equation is introduced in the ODE system of Euler-Lagrange (C.67)

$$\begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{M}(\chi) \\ \chi^T \boldsymbol{R}^T \boldsymbol{R} \end{bmatrix} \ddot{\chi} + \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{C}(\chi, \dot{\chi}) \\ \dot{\chi}^T \boldsymbol{R}^T \boldsymbol{R} + 2\dot{\chi}^T \boldsymbol{R}^T \boldsymbol{R} \end{bmatrix} \dot{\chi} + \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{G}(\chi) \\ \frac{1}{2}\beta^2 \chi^T \boldsymbol{R}^T \boldsymbol{R} \chi \end{bmatrix} + \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{D}(\dot{\chi}) \\ -\frac{1}{2}\beta^2 \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{Q}(\chi) \\ \boldsymbol{0}_{1\times 3} \end{bmatrix} \boldsymbol{\tau}_m \tag{C.67}$$

Looking for a compact and conventional expression of the Euler-Lagrange equation, such as (C.68) without the constraints, the matrices are denoted as follow

$$\begin{aligned} \widetilde{\boldsymbol{M}}(\chi) &= \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{M}(\chi) \\ \chi^T \boldsymbol{R}^T \boldsymbol{R} \end{bmatrix} \\ \widetilde{\boldsymbol{C}}(\chi, \dot{\chi}) &= \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{C}(\chi, \dot{\chi}) \\ \dot{\chi}^T \boldsymbol{R}^T \boldsymbol{R} + 2\dot{\chi}^T \boldsymbol{R}^T \boldsymbol{R} \end{bmatrix} \\ \widetilde{\boldsymbol{G}}(\chi) &= \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{G}(\chi) \\ \frac{1}{2}\beta^2 \chi^T \boldsymbol{R}^T \boldsymbol{R} \chi \end{bmatrix} \\ \widetilde{\boldsymbol{D}}(\dot{\chi}) &= \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{D}(\dot{\chi}) \\ -\frac{1}{2}\beta^2 \end{bmatrix} \\ \widetilde{\boldsymbol{Q}}(\dot{\chi}) &= \begin{bmatrix} \widetilde{\boldsymbol{H}}\boldsymbol{Q}(\chi) \\ \boldsymbol{0}_{1\times 3} \end{bmatrix} \end{aligned}$$

Giving the final model in Euler-Lagrange form to represent the ball-robot

$$\widetilde{\boldsymbol{M}}(\chi)\ddot{\chi} + \widetilde{\boldsymbol{C}}(\chi, \dot{\chi})\dot{\chi} + \widetilde{\boldsymbol{G}}(\chi) + \widetilde{\boldsymbol{D}}(\dot{\chi}) = \widetilde{\boldsymbol{Q}}(\chi, \dot{\chi}, \boldsymbol{u}) \tag{C.68}$$

Appendix D

Comparison between 3D rotation representations

A rotation in a rigid body movement, unlike a translation, keeps a point fixed. A rotation is simply a progressive radial orientation to a common point.

Even though a rotation has three degrees of freedom according to Euler's rotation theorem, there have been created many different methods that require, for example, three components, four and even nine components to represent it. The advantages and disadvantages of Euler angles, transformation matrices, and quaternions are addressed in this section.

- **Visualization and complexity**: Compared to quaternions, Euler angles are more straightforward and more intuitive; they drive to a straight analysis and control. Quaternions are less intuitive to visualise the rotation they apply, and also less widespread since they are not included in the standard curriculum of modern mathematics. Furthermore, the mathematics and model derivation with quaternions can become more complicated. Since Euler angles are more natural to visualise, even when quaternions are used for the modelling or estimation, the user input or output is often converted to Euler angles.
- **Singularities**: In Euler angles, when the body makes a sharp turn at certain positions, the change tracked in Euler angles can not be continuous. E.g. using the ZYX convention, when the pitch angle approaches $\pm 90°$, the other two axes, x and z, are aligned, producing a singularity called "gimbal lock". Gimbal lock can be noted when moving a telescope [66] [67]. This phenomenon

removes one degree of freedom in the rotation since changes in yaw and roll will result in rotations around the same axis. Thus, it prevents from measuring the 3D orientation properly in the gimbal lock position. For each order of angles convention, there are a series of rotations that produces this phenomenon. Furthermore, Euler angles have discontinuities, e.g. when turning 360 degrees around the z-axis (yaw) [6]. Quaternions representation provides an alternative measurement technique that does not suffer from gimbal lock.

- **Unique representation**: A disadvantage of quaternions is that for a given orientation, there is not a unique quaternion. Conversely, two quaternions represent it, the quaternion and the quaternion with its terms negated, each one at an opposite point on the hyper-sphere, i.e. in the 4-dimensional space, but in the same point in 3 dimensions. This phenomenon is described as if the quaternions "double-cover" space, i.e. the unit quaternions cover the space of rotations twice since they are in the 4-dimensional space. Euler angles require to define the order of the axis rotations. There are thirty-six possible combinations, but since each axis can be used only once, it leads to twelve different formulations. Conversely, the transformation matrix method has a unique representation for a rotation [68].

- **Interpolation**: In Euler angles, the rotation interpolation is not ideal. One approach is to interpolate between each Euler angle independently to produce in-between frames. Thereby the dependencies between the axes are ignored, and the motion specified by such a rotation is frequently ill specified and creates unexpected effects. It is illustrated by there not being a unique path between every two orientations across different coordinate systems. Unlike the Euler angles, interpolating in quaternion space ensures a unique smooth path under all circumstances, since quaternions work with points on a sphere, instead of points on a real projection on a plane [69]. Quaternions have the previously explained disadvantage of covering the space of rotations twice; however, they can not cover the space of rotations once and also provide a proper way to interpolate between paths of rotations. For the transformation matrix, the interpolation is simple.

- **Parameters and constraints**: An orientation in three dimensions has three degrees of freedom, and according to Euler's rotation theorem, it can be represented by three scalar numbers [70]. Euler angles use a sequence of three parameters to represent it, and it uses no constraints, it uses the minimal independent coordinates. The transformation matrices method requires nine parameters to represent a rotation. Therefore they are dependent, and it has six constraints to end up in three degrees of freedom. $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_3$. Quaternions uses four parameters to represent the rotation. Hence it has one constraint.

In this case, it imposes the quaternion requiring to have unit norm, in order to be on the surface of the hyper-sphere. $\boldsymbol{q}^T\boldsymbol{q} = 1$

- **Translation**: Even though it is possible to implement translation using quaternions, they do not provide an easy solution for implementing rotation and translation together as the transformation matrices do.
- **Derivatives calculation**: In terms of calculating the derivative, Euler angles are left behind because of their complexity, since they include several sines and cosines. However, transformation matrices derivatives are easier to derive due to being the cross product between the angular velocity and the current rotation. Quaternions derivative is also simple to derive since it is done in the same way as with time-varying vector functions.
- **Vector rotation and rotation concatenation**: In Euler angles, rotating a vector requires to transform it into a rotation matrix or a quaternion. This transformation involves several sines and cosines operations, computationally inefficient. Concatenating rotations is not simple; they can not be just added. Transformation matrices rotates vectors just by matrix multiplication. Therefore, even though they require more memory because of having nine terms, vector rotation is efficient. To concatenate two rotations is done easily by multiplying the rotation matrices of each rotation. Rotating through quaternions require pre- and post- multiplying and concatenate rotations just quaternion multiplication.

Once the advantages and disadvantages of a general application are explored and understood, they can be assessed for the ball-robot case.

The equations of motion derived in the modelling for the body orientation through Euler angles method yields to three coupled differential equations, through transformation matrices to nine and quaternions yields to four. However, by Euler angles, it results in significant expressions with long sequences of sine and cosine products. Furthermore, with quaternions, the model has simpler derivatives and should be faster to compute and therefore to simulate, also more convenient in case of implementing real-time model-based controllers or non-linear estimators [6].

Besides, quaternions do not require to specify the order of rotations as Euler angles do. On top of that, quaternions do not suffer from gimbal lock. Nonetheless, for this application, since the body is not desired to do sharp movements, implying $\pm 90°$ rotations, this is not a point to consider.

Quaternions require fewer parameters and constraints than transformation matri-

ces. Conversely, more complex mathematics and they can not represent translation in their standard form. However, for the orientation of the body, $\{B\}$ frame, with respect to the $\{K\}$ frame, there is no translation components. Therefore this advantage of transformation matrices does not play a role in this application.

Quaternions are the simplest way to capture all the geometry, topology and group structure of 3D rotations [6]. The same way complex numbers are suitable for 2D rotation, quaternions constitute their analogue extension, but using four dimensions to represent a 3D rotation. Quaternions are widely used for computer graphics, computer vision as well as in different mathematics and physics fields. Therefore, despite using them in the ball-robot orientation is not a must, it is decided to use quaternion representation mainly because the role derivatives of the rotation method play in the modelling and for getting into this algebra field.

Appendix E

Quaternions

In this section, the 3D rotation representation using quaternions is explained, as well as interesting arithmetic properties of the quaternions that will be used during the derivation of the quaternion model.

Quaternion interpolation, as well as complex quaternion mathematics, are beyond the scope of this project. Therefore this appendix includes a brief explanation of the main concepts in quaternions and the quaternion mathematics required to define the ball-robot model, i.e. to define the orientation of a rigid body. In order to get a deep understanding, it is useful this tool [71] that allows to visualise how quaternions work and play with their parameters.

E.1 Introduction

Quaternions were discovered by William Rowan Hamilton in 1843. Hamilton was looking for ways to represent 3D rotations by extending complex numbers (which can be viewed as points on a plane) to higher spatial dimensions because they are suitable for representing 2D rotations.

3D rotations can be defined by three scalar numbers [70], such as Euler angles, but such a representation is non-linear and difficult to work with. An analogy is a two-dimensional map of the earth, and it distorts either angles or areas. However, once the 2D map is wrapped around a 3D sphere, it becomes linear, in a similar way, the 3D space of rotations becomes linear when it is mapped into a 4D hyper-sphere [72]. Hamilton discovered a 4-dimensional division algebra, called quaternions, to represent 3D rotations [73].

Quaternions can be expressed in a "complex number notation", where i, j and k are "the" hypercomplex numbers, i.e. higher-dimensional numbers that have their properties departing of the real and complex numbers. The scalar term a has its origin in quaternions when understood as the mathematical extension of the complex numbers [74].

$$q = a + bi + cj + dk \quad \text{where} \quad a, b, c, d \in \mathbb{R} \tag{E.1}$$

The relationship between i, j and k is very similar to the cross product rules for the unit Cartesian vectors and follow the right hand rule as well.

$$ij = k \quad ji = -k \quad jk = i \quad kj = -i \quad ki = j \quad ik = -j \tag{E.2}$$

The difference between cross product in vectors is stated in Hamilton's expression, which involves abandoning the commutative law

$$i^2 = j^2 = k^2 = ijk = -1 \tag{E.3}$$

Hamilton also recognized that the i, j, and k imaginary numbers could be used to represent the fundamental unit quaternions $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k} \in \mathbb{H}$, four-dimensional. Having the same properties stated above and being also non-commutative in multiplication. Here is where the relationship between the complex numbers world and quaternions can be observed [75].

$$q = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} \quad \text{where} \quad q_0, q_1, q_2, q_3 \in \mathbb{R}, \quad \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k} \in \mathbb{H} \tag{E.4}$$

Quaternions can be treated simply as four-dimensional vectors for the purposes of addition, subtraction and scalar multiplication [75]. According to this, in this project, the quaternions are defined in a vector form rather than in the complex number notation. Therefore, they are represented with a scalar number, q_0 and a vector $\vec{q}$ for the components representing the three orthogonal imaginary parts.

$$\boldsymbol{q} = \begin{bmatrix} q_0 \\ \vec{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad \text{where} \quad q_0 \in \mathbb{R}, \quad \vec{\boldsymbol{q}} \in \mathbb{R}^3 \tag{E.5}$$

To construct a quaternion from a vector is done by adding a zero scalar component

to the vector and it is represented by the operator $\wedge$. The resulting quaternion is denoted by $\tilde{\boldsymbol{q}}$.

$$\tilde{\boldsymbol{q}} = \wedge\vec{\boldsymbol{q}} = \begin{bmatrix} 0_{1\times 3} \\ I_3 \end{bmatrix} \vec{\boldsymbol{q}} = \begin{bmatrix} 0 \\ \vec{\boldsymbol{q}} \end{bmatrix} \tag{E.6}$$

To extract the vector part of a quaternion is done with the operator $\vee$.

$$\vec{\boldsymbol{q}} = \vee\boldsymbol{q} = \begin{bmatrix} 0_{3\times 1} & \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{q} \tag{E.7}$$

E.1.1 Arithmetic of quaternions

As stated before, quaternions can be added and subtracted similar to vector addition or subtraction, meaning that the components of the first quaternion are added to the corresponding components of the second one. The equivalent is true for subtraction [75].

$$\boldsymbol{q} + \boldsymbol{p} = \begin{bmatrix} q_0 + p_0 \\ q_1 + p_1 \\ q_2 + p_2 \\ q_3 + p_3 \end{bmatrix} = \begin{bmatrix} q_0 + p_0 \\ \vec{\boldsymbol{q}} + \vec{\boldsymbol{p}} \end{bmatrix} \tag{E.8}$$

The product of two quaternions results in another quaternion. It is indicated with the quaternion multiplication operator $\circ$. A quaternion multiplication follows a set of common rules as the imaginary parts stated in (E.3) and (E.2). This is useful when using complex number notation. In the case of using vector notation, the multiplication of quaternions uses the dot and the cross product as defined next,

$$\boldsymbol{t} = \boldsymbol{q} \circ \boldsymbol{p} = \begin{bmatrix} q_0 \\ \vec{\boldsymbol{q}} \end{bmatrix} \circ \begin{bmatrix} p_0 \\ \vec{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} q_0 p_0 - \vec{\boldsymbol{q}} \cdot \vec{\boldsymbol{p}} \\ q_0\vec{\boldsymbol{p}} + p_0\vec{\boldsymbol{q}} + \vec{\boldsymbol{q}} \times \vec{\boldsymbol{p}} \end{bmatrix} \tag{E.9}$$

As a remark, if q and p are vectors, or quaternions with zero in the scalar part, a cross product between them is equivalent as the quaternion multiplication between them. Like in matrix multiplication, quaternion multiplication is non-commutative and therefore the order of the terms changes the result. Also, just like the vector cross product can be expressed as the product of a skew-symmetric matrix and a vector, the quaternion multiplication can be transform to the product of the matrix Φ or Γ times the quaternion.

$$\boldsymbol{t} = \boldsymbol{q} \circ \boldsymbol{p} = \boldsymbol{\Phi}(\boldsymbol{q})\boldsymbol{p} = \boldsymbol{\Gamma}(\boldsymbol{p})\boldsymbol{q} \tag{E.10}$$

where, either $\mathbf{\Phi}$, left quaternion multiplication, or $\mathbf{\Gamma}$, right quaternion multiplication, are linear matrix operators that allow converting quaternion multiplication into matrix multiplication.

$$\boldsymbol{t} = \mathbf{\Phi}(\boldsymbol{q})\boldsymbol{p} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_1p_0 + q_0p_1 - q_3p_2 + q_2p_3 \\ q_2p_0 + q_3p_1 + q_0p_2 - q_1p_3 \\ q_3p_0 - q_2p_1 + q_1p_2 + q_0p_3 \end{bmatrix} \tag{E.11}$$

$$\boldsymbol{t} = \mathbf{\Gamma}(\boldsymbol{p})\boldsymbol{q} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_1q_0 + p_0q_1 + p_3q_2 - p_2q_3 \\ p_2q_0 - p_3q_1 + p_0q_2 + p_1q_3 \\ p_3q_0 + p_2q_1 - p_1q_2 + p_0q_3 \end{bmatrix} \tag{E.12}$$

The transpose of this operators is the same as multiplying with the conjugated quaternion

$$\Phi\left(\boldsymbol{q}^*\right) = \Phi(\boldsymbol{q})^T \tag{E.13}$$

$$\Gamma\left(\boldsymbol{q}^*\right) = \Gamma(\boldsymbol{q})^T \tag{E.14}$$

The quaternion conjugate can be computed by negating the vector part of the quaternion. Like in complex numbers, by negating the imaginary part.

$$\boldsymbol{q}^* = \boldsymbol{I}^*\,\boldsymbol{q} = \begin{bmatrix} 1 & \mathbf{0}_{1\times3} \\ \mathbf{0}_{3\times1} & -\boldsymbol{I}_3 \end{bmatrix} \boldsymbol{q} = \begin{bmatrix} q_0 \\ -\vec{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix} \tag{E.15}$$

The inverse of a quaternion, as in matrices, allow to multiply by the quaternion and obtain the identity property. Any matrix multiplied by the identity matrix results in the same matrix. For this to be applied in quaternions, the identity quaternion is called the unit quaternion. It is a quaternion that does not apply any change to

the quaternion which is multiplied by, either if it is left or right multiplication [6].

$$\boldsymbol{q}_{unit} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.16}$$

The inverse quaternion satisfies the identity property.

$$\boldsymbol{q} \circ \boldsymbol{q}^{-1} = \boldsymbol{q}_{unit} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.17}$$

To calculate the magnitude of a quaternion is done as to calculate the module or magnitude of a vector, i.e. with the euclidean norm of the quaternion.

$$\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q}\,\boldsymbol{q}^T} = \sqrt{\boldsymbol{q}^T\boldsymbol{q}} = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2} \tag{E.18}$$

The inverse of a quaternion is easy to compute with the euclidean norm and always exists except for a zero quaternion.

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|} = \frac{\boldsymbol{q}^*}{\boldsymbol{q}^\top\boldsymbol{q}} \tag{E.19}$$

E.1.2 Rotations with quaternions

A three-dimensional rotation possesses three degrees of freedom according to Euler's rotation theorem [70]. When using quaternions to represent rotations, quaternions use four elements to define 3 degrees of freedom, i.e. the rotation is over-determined [6]. A unit norm quaternion provides a convenient mathematical tool for representing orientations and rotations in objects in three dimensions since it forces the rotation to be on the four-dimensional unit sphere [76]. This way, forcing the norm of the quaternion to be equal to one, introduces a constraint that reduces the freedom removing the over-determined problem.

A unit norm quaternion is the same as a unit vector; it has to have a magnitude of exactly one (E.18), i.e. it sits a distance one from the origin. Any quaternion can

be converted to a unit norm quaternion by dividing each of the components by the total magnitude, just like vector normalization [75].

$$\boldsymbol{q}_{\|1\|} = \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|} \tag{E.20}$$

It is important to remark, that quaternions are closed under quaternion multiplication. However, they are not closed under addition and subtraction, i.e. adding or subtracting unit quaternions would lead to a non-unit quaternion. Hence, the resulting quaternion is no longer kept on the 4-dimensional unit sphere.

For unit norm quaternions the inverse of the quaternion becomes the conjugate

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|} = \boldsymbol{q}^* \tag{E.21}$$

As a consequence, the identity property applies for quaternion multiplication of its conjugate

$$\boldsymbol{q}^* \circ \boldsymbol{q} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.22}$$

It is not intuitive to visualise the rotation applied by a quaternion since it is defined in four dimensions. Instead, three Euler angles or a rotation around a particular axis can be used to specify the desired rotation or to read it. Even though quaternions are used for modelling because of the advantages listed in Appendix D, conversions from quaternions to the other representations and vice-versa can be done through the conversion formulas [75].

The rotation applied by a quaternion can be shown by the rotation angle θ and the normalized 3D rotation axis r. The inverse quaternion, which is the conjugate, will apply the same rotation amount and around the same axis, but in the opposite direction.

$$\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ r_x \sin\left(\frac{\theta}{2}\right) \\ r_y \sin\left(\frac{\theta}{2}\right) \\ r_z \sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{E.23}$$

Euler's rotation theorem states that it is possible to represent any 3D rotation with an axis of rotation and the angle, therefore, quaternions can describe any 3D rotation since they encode it [68]. However, one disadvantage of the quaternions enters

here. There are two quaternions to represent the same 3D orientation. The quaternion and the quaternion with all its terms negated. The quaternion would make reaching that orientation by turning the specific amount, and the negate quaternion would make turn the amount plus 360°, representing the same orientation [69]. In other words, any given orientation corresponds to two separate points on the opposite side of a hyper-sphere in four dimensions. It is only needed half of the hyper-sphere since opposite sides of the hyper-sphere represent the same rotation [72]. This phenomenon is known as the unit quaternions "double-cover" the rotations space, and its explanation comes from the angle doubling fact, which is important when seeing the rotation through quaternion multiplication and explains their "sandwich" multiplication kind, as it is explained next [77].

In the ball-robot model, derivation unit norm quaternions are used to represent the rotation of the body with respect to the ball ($\{K\}$ frame). Since it embeds the information of the orientation of the body, i.e. the rotation over a reference frame, the rotation quaternion is called orientation quaternion or attitude quaternion. The quaternion is defined as ${}^{K}_{B}\boldsymbol{q}$ since it encodes the rotation from the body frame $\{B\}$ to the $\{K\}$ frame, i.e. the 3D orientation of the body with respect to the inertial frame. Like transformation matrices, the rotation is represented as how to rotate the final frame $\{K\}$ to the initial frame $\{B\}$ [6]. Since ${}^{K}_{B}\boldsymbol{q}$ has this information encoded, it can be used to rotate an arbitrary vector from the $\{B\}$ frame to the $\{K\}$ frame and vice-versa.

As complex numbers multiplication rotates a two-dimensional vector, the rotation of a vector in 3 dimensions will be accomplished through quaternion multiplication. Quaternion multiplication to a vector requires the vector to be defined in four dimensions, that is accomplished by adding a zero to its scalar part as explained in (E.6) Therefore, for a vector defined in the body frame to be rotated to the inertial frame, the following quaternion multiplication must be applied.

$$ {}^{K}\tilde{\boldsymbol{p}} = {}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\boldsymbol{q}^{*} \tag{E.24} $$

As shown in the above equation, quaternion rotation requires pre- and post- multiplication by the quaternion ("sandwich multiplication"). The first multiplier must be the quaternion and the last one corresponds to its conjugate. This can be applied since they are unit quaternions. The reason for this additional multiplication is keep the magnitude in the resulting vector. In other words, the resulting quaternion must be on the surface of the hyper-sphere, i.e. have unit norm, and this is achieved by the pre- and post- multiplication. As a consequence, the angle rotated by the quaternion must be half of the required one (E.23), since both the quaternion and its inverse multiplication adds rotation. This rotation multiplication character-

istic is due to the double-cover property of the quaternions mentioned above and explained in [77], and the reason is that they represent the 3D rotation by using four dimensions.

The matrix operators (E.10) are applied next to simplify the equations and transforming the quaternion multiplication into matrix multiplication.

$$ {}^{B}\tilde{p} = \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\left({}^{K}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\boldsymbol{q}\right) = \Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\Gamma\left({}^{K}_{B}\boldsymbol{q}\right){}^{K}\tilde{p} \tag{E.25} $$

And for the opposite rotation from (E.24), the order has to be reversed.

$$ {}^{B}\tilde{\boldsymbol{p}} = {}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\boldsymbol{q} = \Phi\left({}^{K}_{B}\boldsymbol{q}\right)\Gamma\left({}^{K}_{B}\boldsymbol{q}\right)^{T}{}^{K}\tilde{p} \tag{E.26} $$

To extract the vector part of a rotated vector

$$ {}^{B}p = \vee\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\Gamma\left({}^{K}_{B}\boldsymbol{q}\right) \wedge {}^{K}\tilde{p} \tag{E.27} $$

The orientation of one vector with respect to another one, or the rotation needed to a vector to be aligned with another, can be calculated by applying the properties of the cross product. The cross multiplication of the two vectors results in a perpendicular vector to the plane form by them, which is the rotation axis.

$$ {}^{K}\boldsymbol{r} = \boldsymbol{p}_{K} \times \boldsymbol{p}_{B} \tag{E.28} $$

The angle between them defined in that plane can be extracted by isolating from the dot product.

$$ \theta = \cos^{-1}\left(\frac{\boldsymbol{p}_{K} \cdot \boldsymbol{p}_{B}}{\|\boldsymbol{p}_{K}\|\,\|\boldsymbol{p}_{B}\|}\right) \tag{E.29} $$

This way, the orientation quaternion, or the rotation, from the frame $\{B\}$ to the frame $\{K\}$ can be derived through (E.28) and (E.29) two vectors from the body and the ball respectively.

$$ {}^{K}_{B}\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ r_{x}\sin\left(\frac{\theta}{2}\right) \\ r_{y}\sin\left(\frac{\theta}{2}\right) \\ r_{z}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{E.30} $$

E.2 Quaternion differentiation

The quaternion differentiation formula connects the angular velocity and the quaternion time derivative.

The quaternion time derivative is computed through the elementwise derivative. The unit quaternion is a point on the 4 dimensional sphere storing the orientation from one of the frames with respect to the other. Therefore its derivative is the tangential velocity to the sphere in the point.

$$ {}^{K}_{B}\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{E.31} $$

However, the angular velocity defines how fast the point moves around the perimeter of the circle, i.e. the relative angular velocity between the two frames, defined in three dimensions. There exists a kinematic relationship between how the quaternion evolves and the angular velocity, such as there exists a relation between linear velocity and the angular velocity.

Since quaternions are non-commutative, the derivative of the quaternion can be calculated with the angular velocity relative to the first frame or to the second one just varying the order of the terms.

$$ {}^{K}_{B}\dot{\boldsymbol{q}} = \frac{1}{2}\,{}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{\omega}}_{B|K} = \frac{1}{2}\,{}^{K}\tilde{\boldsymbol{\omega}}_{B|K} \circ {}^{K}_{B}\boldsymbol{q} \tag{E.32} $$

The angular velocity of the quaternion can be derived isolating it from the above equation by multiplying both sides by ${}^{K}_{B}\boldsymbol{q}^{*}$. Then, it is simplified by using the matrix operator.

$$ {}^{B}\tilde{\boldsymbol{\omega}}_{B|K} = 2\,{}^{K}_{B}\boldsymbol{q}^{*} \circ {}^{K}_{B}\dot{\boldsymbol{q}} = 2\,\Phi\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\,{}^{K}_{B}\dot{\boldsymbol{q}} \tag{E.33} $$

It can be rotated to the $\{K\}$ frame to be defined in that frame by quaternion rotation

$$ {}^{K}\tilde{\boldsymbol{\omega}}_{B|K} = {}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{\omega}}_{B|K} \circ {}^{K}_{B}\boldsymbol{q}^{*} = 2\,{}^{K}_{B}\dot{\boldsymbol{q}} \circ {}^{K}_{B}\boldsymbol{q}^{*} = 2\,\Gamma\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\,{}^{K}_{B}\dot{\boldsymbol{q}} \tag{E.34} $$

The derivative of an equation with quaternions is derived the same way as in the case of time-varying vector functions.

The derivative of a vector rotation applies the product rule, taking into account that quaternion multiplications are matrix multiplications with matrix functions defined from the quaternion.

$$ {}^{K}\dot{\tilde{\boldsymbol{p}}} = \frac{d}{dt}\left({}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\boldsymbol{q}^{*}\right) = \left({}^{K}_{B}\dot{\boldsymbol{q}} \circ {}^{B}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\boldsymbol{q}^{*}\right) + \left({}^{K}_{B}\boldsymbol{q} \circ {}^{B}\dot{\tilde{\boldsymbol{p}}} \circ {}^{K}_{B}\boldsymbol{q}^{*}\right) + \left({}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\dot{\boldsymbol{q}}^{*}\right) \quad \text{(E.35)} $$

The above equation simplifies to only two factors in case the vector ${}^{B}\tilde{\boldsymbol{p}}$ is a constant vector within a rotating frame. For example, in the case of the position vector for the body center of mass, in which the vector has a constant magnitude and point in a constant direction defined in the frame $\{B\}$, but this frame in which is defined, rotates with respect to the frame $\{K\}$. In this case, ${}^{B}\dot{\tilde{\boldsymbol{p}}}$ is zero because the position vector is constant, and the above equation simplifies to,

$$ \begin{aligned} {}^{K}\dot{\boldsymbol{p}} &= \wedge\left(\left({}^{K}_{B}\dot{\boldsymbol{q}} \circ {}^{B}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\boldsymbol{q}^{*}\right) + \left({}^{K}_{B}\boldsymbol{q} \circ {}^{B}\tilde{\boldsymbol{p}} \circ {}^{K}_{B}\dot{\boldsymbol{q}}^{*}\right)\right) \\ &= \wedge\left(\boldsymbol{\Phi}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right)\boldsymbol{\Phi}\left({}^{B}\tilde{\boldsymbol{p}}\right)\boldsymbol{I}^{*} + \boldsymbol{\Gamma}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right)^{T}\boldsymbol{\Gamma}\left({}^{B}\tilde{\boldsymbol{p}}\right)\right){}^{K}_{B}\boldsymbol{q} \\ &= \wedge\left(\boldsymbol{\Phi}\left({}^{K}_{B}\boldsymbol{q}\right)\boldsymbol{\Phi}\left({}^{B}\tilde{\boldsymbol{p}}\right)\boldsymbol{I}^{*} + \boldsymbol{\Gamma}\left({}^{K}_{B}\boldsymbol{q}\right)^{T}\boldsymbol{\Gamma}\left({}^{B}\tilde{\boldsymbol{p}}\right)\right){}^{K}_{B}\dot{\boldsymbol{q}} \end{aligned} \quad \text{(E.36)} $$

To simplify the equation of vector rotation two new operators are used, $\boldsymbol{\Pi}_{p}$ and $\boldsymbol{\Lambda}_{p}$. The subscript indicates for which vector is defined.

$$ \boldsymbol{\Pi}_{p}(\boldsymbol{q}) = \boldsymbol{\Phi}(\boldsymbol{q})\boldsymbol{\Phi}\left({}^{B}\tilde{\boldsymbol{p}}\right)\boldsymbol{I}^{*} \quad \text{(E.37)} $$

$$ \boldsymbol{\Lambda}_{p}(\boldsymbol{q}) = \boldsymbol{\Gamma}(\boldsymbol{q})\boldsymbol{\Gamma}\left({}^{B}\tilde{\boldsymbol{p}}\right) \quad \text{(E.38)} $$

Thus, the angular velocity is simplified to

$$ {}^{K}\dot{\boldsymbol{p}} = \left(\vee\boldsymbol{\Pi}_{p}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right) + \vee\boldsymbol{\Lambda}_{p}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right)\right){}^{K}_{B}\boldsymbol{q} = 2 \vee \boldsymbol{\Pi}_{p}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right){}^{K}_{B}\boldsymbol{q} \quad \text{(E.39)} $$

Due to the structure of the operators, the following relationships are fulfilled

$$ \boldsymbol{\Pi}_{p}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right){}^{K}_{B}\boldsymbol{q} = \boldsymbol{\Lambda}_{p}\left({}^{K}_{B}\dot{\boldsymbol{q}}\right){}^{K}_{B}\boldsymbol{q} = \boldsymbol{\Pi}_{p}\left({}^{K}_{B}\boldsymbol{q}\right){}^{K}_{B}\dot{\boldsymbol{q}} = \boldsymbol{\Lambda}_{p}\left({}^{K}_{B}\boldsymbol{q}\right){}^{K}_{B}\dot{\boldsymbol{q}} \quad \text{(E.40)} $$

In the partial derivatives of quaternions the new matrix operations show a more direct derivative.

$$ \frac{\partial}{\partial \boldsymbol{p}}\boldsymbol{q} \circ \boldsymbol{p} = \frac{\partial}{\partial \boldsymbol{p}}\boldsymbol{\Phi}(\boldsymbol{q})\boldsymbol{p} = \boldsymbol{\Phi}(\boldsymbol{q}) \quad \text{(E.41)} $$

For the Euler-Lagrange equations definition is useful to define the derivative of the vector rotation with respect to the quaternion. (E.15) and (E.36) are used for the derivation.

$$\begin{aligned}\frac{\partial}{\partial {}^K_B\boldsymbol{q}}\left({}^K_B\boldsymbol{q}\circ {}^B\tilde{\boldsymbol{p}}\circ {}^K_B\boldsymbol{q}^*\right) &= \frac{\partial}{\partial {}^K_B\boldsymbol{q}}\left(\boldsymbol{\Gamma}\left({}^K_B\boldsymbol{q}\right)^T\boldsymbol{\Gamma}\left({}^B\tilde{\boldsymbol{p}}\right){}^K_B\boldsymbol{q}\right)+\frac{\partial}{\partial {}^K_B\boldsymbol{q}}\left(\boldsymbol{\Phi}\left({}^K_B\boldsymbol{q}\right)\boldsymbol{\Phi}\left({}^B\tilde{\boldsymbol{p}}\right)\boldsymbol{I}^*{}^K_B\boldsymbol{q}\right)\\ &= \boldsymbol{\Gamma}\left({}^K_B\boldsymbol{q}\right)^T\boldsymbol{\Gamma}\left({}^B\tilde{\boldsymbol{p}}\right)+\boldsymbol{\Phi}\left({}^K_B\boldsymbol{q}\right)\boldsymbol{\Phi}\left({}^B\tilde{\boldsymbol{p}}\right)\boldsymbol{I}^* = \boldsymbol{\Lambda}\left({}^K_B\boldsymbol{q}\right)+\boldsymbol{\Pi}\left({}^K_B\boldsymbol{q}\right)\end{aligned} \tag{E.42}$$

The vector part of $\boldsymbol{\Pi}_p$ and $\boldsymbol{\Lambda}_p$ coincides due to construction, which makes this simplify to

$$\frac{\partial}{\partial {}^K_B\boldsymbol{q}}\vee\left({}^K_B\boldsymbol{q}\circ {}^B\tilde{\boldsymbol{p}}\circ {}^K_B\boldsymbol{q}^*\right) = \vee\boldsymbol{\Lambda}\left({}^K_B\boldsymbol{q}\right)+\vee\boldsymbol{\Pi}\left({}^K_B\boldsymbol{q}\right) = 2\vee\boldsymbol{\Lambda}\left({}^K_B\boldsymbol{q}\right) = 2\vee\boldsymbol{\Pi}\left({}^K_B\boldsymbol{q}\right) \tag{E.43}$$

Appendix F

Transformation matrix

The group of all transformations in the 3D Cartesian space is SE(3). Transformations consist of a rotation and a translation and can be represented in multiple different ways. In this case, transformation matrices form is addressed, also called homogeneous representation of a transformation [78].

A transformation matrix can be used to transform a vector defined in the frame $\{B\}$ coordinates to a frame $\{A\}$

$$ {}^{A}\boldsymbol{p} = {}^{A}_{B}\boldsymbol{T}\,{}^{B}\boldsymbol{p} = {}^{A}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{p} + {}^{A}\boldsymbol{t}_{B} \tag{F.1} $$

where, the point is described in homogeneous coordinates and ${}^{A}_{B}\boldsymbol{T}$ is the $4x4$ transformation matrix. ${}^{A}t_{B}$ is the translation vector that express the frame $\{B\}$ origin in frame $\{A\}$ to show the translation needed to make their origins coincide. ${}^{A}_{B}\boldsymbol{R}$ is the rotation matrix where the column vectors of frame $\{B\}$ are defined in frame $\{A\}$ and rotates a vector defined in frame $\{B\}$ to frame $\{A\}$.

$$ {}^{A}_{B}\boldsymbol{T} = \begin{bmatrix} {}^{A}_{B}\boldsymbol{R} & {}^{A}\boldsymbol{t}_{B} \\ \boldsymbol{0_{1x3}} & 1 \end{bmatrix} \tag{F.2} $$

F.1 Rotation matrices

The group of rotations in the 3D Cartesian space is SO(3).

To visualise 3D rotations the easiest way is by the axis-angle form. According to Euler's rotation theorem [70] any arbitrary rotation can be defined by an axis of

rotation normalized and an angle that describes the amount of rotation, i.e. by 3 scalar numbers [79].

Each rotation matrix is a rotation about a single coordinate axis [80].

The rotate a point or a reference frame about the z-axis (yaw) by an angle ϕ is done through the following rotation matrix

$$\mathbf{R}_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{F.3}$$

The rotate about the y-axis (pitch) by an angle χ

$$\mathbf{R}_y(\chi) = \begin{pmatrix} \cos\chi & 0 & \sin\chi \\ 0 & 1 & 0 \\ -\sin\chi & 0 & \cos\chi \end{pmatrix} \tag{F.4}$$

and around the x-axis (roll) by an angle ψ

$$\mathbf{R}_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \tag{F.5}$$

Three coordinate rotations in sequence can describe any rotation. Thus, knowing the value of Euler angles (ϕ, χ, ψ) rotated from frame $\{B\}$ to frame $\{A\}$ and concatenating this rotation matrices in the proper order, the complete rotation matrix can be obtained.

$$^{A}_{B}\boldsymbol{R} = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \tag{F.6}$$