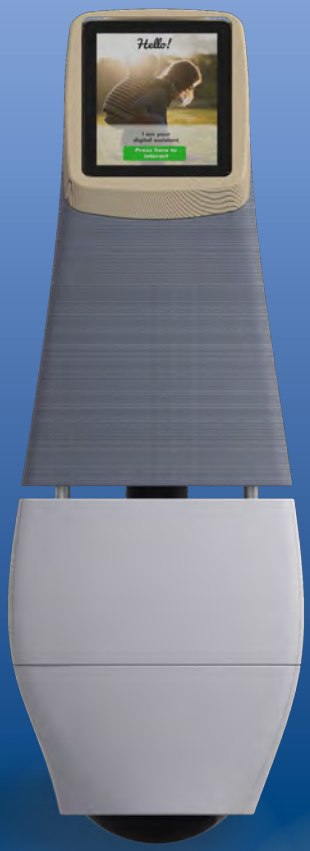
KUGLE

Modelling and Control of a Ball-balancing Robot



Thomas Kølbæk Jespersen

Master Thesis
Control & Automation
April 2019

AALBORG UNIVERSITY
STUDENT REPORT

AALBORG UNIVERSITY
STUDENT REPORT

Control & Automation
Fredrik Bajers Vej 7C
DK-9220 Aalborg Ø
http://es.aau.dk/control

Title:
Kugle - Modelling and Control of a Ball-balancing Robot

Project Period:
September 1st 2018 to April 23rd 2019

Date of Hand-in:
April 23rd 2019

Page Numbers: 274

Supervisor:
Thomas Bak

Co-supervisors:
Karl Damkjær
Rasmus Pedersen

Author:
Thomas Kølbæk Jespersen

Abstract:

This thesis covers the derivation of a non-linear model and sliding mode controller for a ball-balancing robot with three omniwheels. A quaternion-based model is derived using Lagrangian mechanics, with the quaternion and ball position as the generalized coordinates. The quaternion unit norm constraint is enforced with a Lagrange multiplier. The quaternion model is used for sliding mode controller design of an orientation stabilizing controller that considers a quaternion error function based on the desired quaternion and angular velocity reference. Two sliding surfaces are proposed and compared in simulation. The derived controller is verified in both simulation and in practice, using a 16 kg ballbot prototype, Kugle V1. The controller and the necessary estimators, two extended Kalman filters for quaternion and velocity estimation, are implemented in an embedded firmware.The controller is confirmed to work as designed and the simulation model is verified against the practical results, using a Vicon motion capture system. Continuously changing references can be tracked with a tracking error of less than 1° up to 1 Hz. The controller is furthermore tested in a cascaded configuration with first a velocity LQR controller and subsequently a shape-accelerated path-following MPC, which generate quaternion and angular velocity references and enable station-keeping, velocity tracking and path-following. Velocity references up to 1 m/s and 1 rad/s are tested and confirmed trackable in practice. The thesis concludes that it is indeed possible to derive and use a quaternion model for ballbot control even though it complicates the derivation and is deemed unnecessary due to the operating envelope. All material, including the MATLAB code, simulations, ROS drivers and the embedded firmware, is open source and available on GitHub.

Preface

This report documents the Master Thesis work carried out as part of the study program in Control & Automation at Aalborg University.

The work in this thesis is carried out as part of the research project, 'Robot Digital Signage' [1] dealing with human-interaction robots and funded by the Innovation Fund under grant 7076-00051A. As part of this project a ball-balancing robot is to be developed. The purpose of this thesis is to develop a dynamic model of a ball-balancing robot, design a robust controller to keep the balance and make the movements of the robot easily controllable. Furthermore, the work of this thesis has included brainstorming and support during the prototype design process and the implementation of the designed controllers on a working prototype.

To understand the thesis fundamental knowledge of Linear Algebra, Calculus, Non-linear system modelling and Linear and Non-linear controller design are required. Furthermore the reader is expected to know how to model mechanical systems with Lagrangian mechanics and to have a conceptual understanding of sliding mode controllers. Familiarity with Quaternions and their properties when used to describe rotations is useful, although a quick walk-through is provided. Knowledge of Kalman filtering and Model Predictive Control is recommended.

References are made using the IEEE referencing scheme, where numbers in square brackets, e.g., [1], refers to an entry in the bibliography found on page 121. Equations are referred to with the equation identifier in a parenthesis, e.g., (1.2), indicating the chapter and equation number. A nomenclature with a list of notations and abbreviations used in this thesis is found just after the preface. All figures and plots are made in TikZ or a reference is listed.

The thesis is written in the hope that it can serve as a reference for future students and researchers working on ball-balancing robots. A lot of effort has been put into traceability and reproducibility, which is why several appendices are included to show the steps of specific derivations, thereby enabling the reader to trace any specific results. Furthermore, all code from the work has been published, including MATLAB files, Simulink models, Embedded firmware, Gazebo simulation and ROS drivers. The code is publicly available and released under the MIT License on the following GitHub repositories:

`https://github.com/mindThomas/Kugle-MATLAB`	MATLAB code and Simulink models of the model and controller development, including the non-linear Quaternion model, Sliding mode balance controller and ACADO MPC for path following.
`https://github.com/mindThomas/Kugle-Embedded`	Embedded firmware (C++) for the STM32H7 board used on the Kugle V1 prototype, running FreeRTOS enabling balance and velocity control with a Sliding mode controller, LQR and several EKF's while simultaneously communicating with the onboard computer with the ROS driver.
`https://github.com/mindThomas/Kugle-ROS`	ROS driver and ROS launch files.

`https://github.com/mindThomas/Kugle-Gazebo`	Gazebo simulation environment of the high-level navigational properties of the Kugle robot, including sensor layout and the shape-accelerated property (tilt-based acceleration).
`https://github.com/mindThomas/Kugle-Misc`	Code, libraries and other tools built for the Kugle robot project which do not fit in the other repositories, e.g., test code. Includes also a visual test-bench for the ACADO-generated Path-following MPC doing obstacle avoidance.

A series of chronologically ordered YouTube videos from the 8 month on-going development period, including simulations and tests, can be found at `https://www.youtube.com/playlist?list=PLLtE4m3fKcOC_TuErjgOpTiI3abHPWM0x`.

The report is divided in two main parts covering the workflow of the project: The first part in which a quaternion model of the ballbot is derived and the second part in which a non-linear sliding mode balance controller is designed.

Chapter 1 introduces the 'Kugle' research project, what a ball-balancing robot is and the current state-of-the-art. The chapter concludes with the problem formulation of the research topic in this thesis.

Chapter 2 presents the Kugle V1 prototype and its model parameters. The chapter is concluded with a system architecture diagram, listing the elements developed in this thesis.

Chapter 3 is the first chapter of the quaternion model part and it introduces the frames and generalized coordinates. The chapter concludes with a list of model assumptions under which the derived model is valid.

Chapter 4 derives the forward and inverse kinematics of the ballbot, mapping the angular velocities of the motors to the states of the system.

Chapter 5 derives the dynamic quaternion model of the ballbot using Lagrangian mechanics. The chapter concludes with the symbolic structure of the equations of motion describing the system.

Chapter 6 verifies the derived kinematic model through a test with the actual system using a motion tracking system for ground-truth measurements. The chapter concludes with a model summary mentioning that the dynamic model has to be verified in closed-loop.

Chapter 7 is the first chapter of the balance controller part and it presents the control objective and operating envelope. The chapter concludes with a list of specifications which the balance controller is designed for.

Chapter 8 proposes two sliding surfaces designed to stabilize the ballbot and track quaternion and angular velocity references. The chapter concludes with a choice of surface.

Chapter 9 derives the equivalent and switching control laws for the sliding mode balance controller. The chapter concludes with a summary of the derived controller.

Chapter 10 presents simulations of the balance controller applied to the derived quaternion model. The simulations are used to verify the functionality of the chosen sliding surface and to simulate the performance of the sliding mode controller.

Chapter 11 provides a brief overview of the embedded firmware implementation on the Kugle V1 hardware, enabling practical tests of the designed controllers.

Chapter 12 presents the results from several tests of the balance controller running on the prototype while being given different references.

Chapter 13 compares the derived non-linear sliding mode balance controller to an equivalent LQR balance controller through both simulations and practical tests.

Chapter 14 presents a velocity LQR controller designed to stabilize the velocity and position of the ballbot in a cascade configuration with the balance controller.

Chapter 15 summarizes and concludes on the overall results and findings of the thesis related to the problem formulation.

Chapter 16 lists any noticed work or research opportunities that are still left to be done.

I would like to thank a number of people who have encouraged and helped me during the development and writing of this thesis. I am satisfied with the results achieved and I appreciate the support I received from all parties.

It has been an honour to be a part of the Kugle project at Aalborg University managed by Professor Thomas Bak whose support and commitment to the project have astonished me. I am very grateful to have become an integrated part of the project and team, which allowed me a custom prototype on which I could develop and test the work described in this thesis.

I am especially indebted to my co-supervisors Karl Damkjær and Rasmus Pedersen for providing support and brainstorming with me along the way. Your encouragement and constant pressure helped me get this far.

I would like to thank Associate Professor Henrik Schiøler for his help during the modelling phase, including brainstorming and review of my initial model, and Professor Rafal Wisniewski for his support with Lagrangian Mechanics and Quaternion understanding.

I would like to thank the rest of the Kugle-team, including Nicolaj Vinkel Christensen, who designed the electronics and assembled the Kugle V1 prototype used in this thesis.

Finally, I am very grateful to all my friends, family members and especially my girlfriend who supported me during the project and endured my late working hours.

Aalborg, Denmark in April 2019,
Thomas Kølbæk Jespersen

Nomenclature

Table of Notation

The common notations used throughout the report are shown in Table 1 and Table 2.

c	Scalar (non-bold letter)
$\boldsymbol{s}$	Vector (bold small letter)
$\boldsymbol{M}$	Matrix (bold capital letter)
$\mathbf{0}_{3\times4}$	Zero matrix of 3 rows and 4 colums
$\mathbf{I}_3$	Identity matrix of size 3
c_{a}	Text or symbol based subscripts denotes specific variables
s_1	A single numeric subscript denotes an indexed scalar of a vector (indexed from 0)
$m_{1,2}$	Two numeric subscripts denotes an indexed scalar of a matrix, denoted as row comma column (indexed from 0)
$x[k+1]$	Time instance $k+1$ of a time varying variable
$\hat{x}^{k\|k-1}$	Apriori estimate at time k based on measurements up to time $k-1$
$\hat{x}^{k\|k}$	Aposteriori estimate at time k based on measurements up to time k
${}^{\mathrm{A}}_{\mathrm{B}}\mathbf{R} \in \mathbb{R}^{3\times3}$	SO(3) rotation matrix from frame {B} to frame {A}
${}^{\mathrm{A}}_{\mathrm{B}}\mathbf{T} \in \mathbb{R}^{4\times4}$	SE(3) transformation from frame {B} to frame {A}
${}^{\mathrm{A}}\boldsymbol{O}_{\mathrm{B}}$	Origin of frame {B} described in frame {A}
${}^{\mathrm{A}}\boldsymbol{v}$	Vector described in frame {A}
$\underline{\boldsymbol{x}}$	Homogeneous coordinate
$\underline{\mathbf{0}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}}$	Homogeneous zero vector
$\vec{\mathbf{i}}, \vec{\mathbf{j}}, \vec{\mathbf{k}}$	Fundamental Cartesian unit vectors
$\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$	Fundamental unit quaternions
$[\cdots]$	Matrix or vector
$\{\cdots\}$	Set

Table 1: Table of the notations used in the thesis.

${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$	Quaternion describing the rotation of frame {B} with respect to frame {K} with a rotation axis defined in frame {K}
$\boldsymbol{q}^*$	Conjugated quaternion
$\boldsymbol{q} \circ \boldsymbol{p}$	Quaternion multiplication (product operator)
$\boldsymbol{\Phi}(\boldsymbol{q})$	Matrix operator for Left quaternion multiplication: $\boldsymbol{q} \circ$
$\boldsymbol{\Gamma}(\boldsymbol{p})$	Matrix operator for Right quaternion multiplication: $\circ \boldsymbol{p}$
$\wedge$	Quaternion vectorize operator - construct quaternion from 3-dimensional vector
$\vee$	Quaternion devectorize operator - extract 3-dimensional vector part of quaternion
$\mathbf{I}^*$	Quaternion conjugation operator - negates the vector part of the quaternion
$\vec{\boldsymbol{q}}$	Vector part of the quaternion $\boldsymbol{q}$
$\tilde{\boldsymbol{v}}$	3-dimensional vector represented as a quaternion with 0 scalar part
${}^{\mathrm{K}}\boldsymbol{v}_{\mathrm{B}\leftarrow\mathrm{I}}$	Linear velocity of frame {B} relative to frame {I}, described in (transformed to) frame {K}
${}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{I}}$	Angular velocity of frame {B} relative to frame {I}, described in (transformed to) frame {K}
${}^{\mathrm{M}}\boldsymbol{\omega}^{i}_{\mathrm{W}\leftarrow\mathrm{M}}$	Indexed angular velocity of frame {W} relative to frame {M}, described in frame {M} with frame index i, e.g., used for wheel angular velocities
${}^{\mathrm{A}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{B}}$	Linear velocity of point $\boldsymbol{p}$ relative to frame {B}, described in (transformed to) frame {A}
$\boldsymbol{\Omega}(\boldsymbol{v})$	Scew-symmetric matrix of vector $\boldsymbol{v}$
$p(X)$	Probability density functions
$p(X, Y)$	Joint probability density functions
$p(X \mid Y)$	Conditional probability density functions
$\mathbb{E}[X]$	Expectation
$X \sim (\mu,\, \sigma^2)$	Undefined probability distribution with mean μ and covariance σ^2
$X \sim \mathcal{N}(\mu,\, \sigma^2)$	Symbol in front of parentheses indicates type of distribution, e.g., $\mathcal{N}$: Normal distribution.
$\mu = \mathbb{E}[X]$	Scalar Mean
$\sigma^2 = \mathrm{Var}(X)$	Variance
$\bar{\boldsymbol{x}} = \mathbb{E}[X, Y]$	Vectorial Mean
$\boldsymbol{\Sigma} = \mathrm{Cov}[X, Y]$	Covariance
$\hat{\boldsymbol{x}}$	Estimate

Table 2: Table of the notations used in the thesis (contd.)

Abbreviations

The common abbreviations used throughout the report are shown in Table 3.

Ballbot	Ball-balancing Robot
COM	Center of Mass
EKF	Extended Kalman Filter
GUI	Graphical User Interface
HAL	Hardware Layer
HRI	Human-Robot Interaction
LCF	Lyapunov Candidate Function
LSPC	Lightweight Serial Package Communication
MPC	Model Predictive Control
RTOS	Real-Time Operating System
SDK	Software Development Kit
SMC	Sliding Mode Controller
QEKF	Quaternion Extended Kalman Filter
UX	User Experience
VEKF	Velocity Extended Kalman Filter

Table 3: Table of abbreviations used in the thesis.

Interchangeable notions

Other common notions used interchangeably throughout the thesis are listed below to avoid confusion.

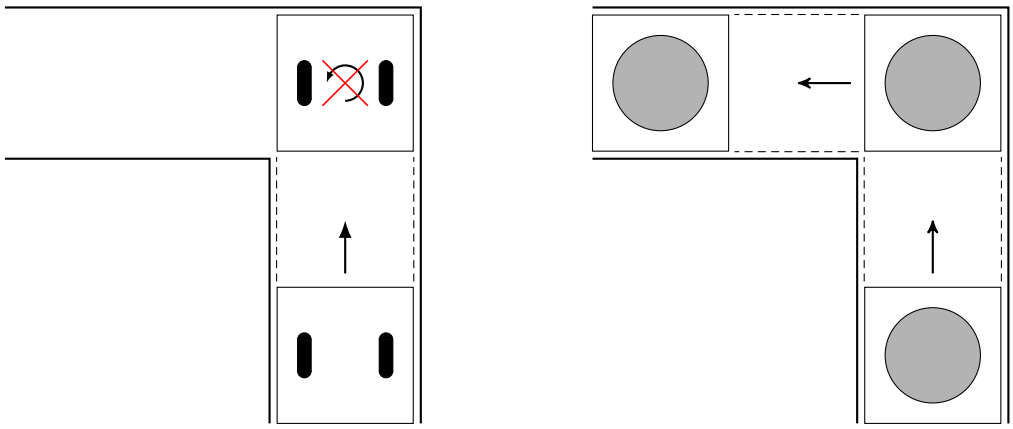
Inclination / Tilt / Angle / Orientation / Attitude / Shape	The rotation of the body with respect to the inertial frame. This rotation defines the resulting acceleration of the ballbot through the shape-space accelerated relationship (non-minimum phase).
Sliding manifold / Sliding surface	The sliding mode controller manifold on which all state trajectories will be after the reaching phase

Table 4: Table of interchangeable notions used in the thesis.

Contents

1 Introduction

Most of the time, mobile robots utilize a simple 2-wheel drive train in a differential-drive configuration since this is robust, easy to model and easy to control. Unfortunately such a drive train has the disadvantage of being non-holonomic and thus requiring the robot to turn to change direction. Depending on the environment, the size and the mechanical design of the robot, this can sometimes be an impossible task, potentially resulting in the robot getting stuck, as illustrated in Figure 1.1.

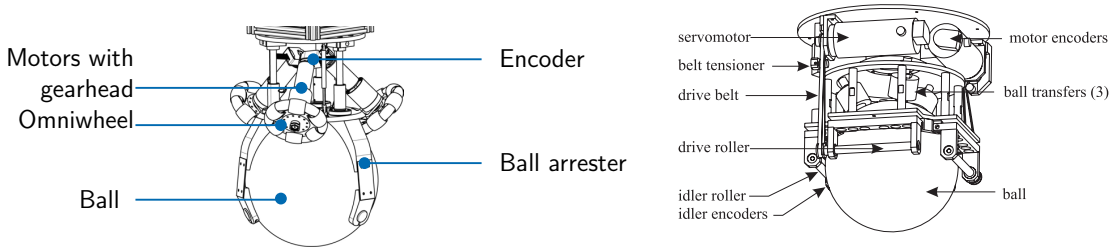


(a) Differential drive robot (b) Ballbot

Figure 1.1: Comparison of the non-holonomic differential drive robot and the holonomic ballbot navigating tight corridors. The holonomic ballbot is capable of navigating in spaces where it is otherwise impossible to turn.

In recent years, ball-balancing robots has received great attention from academia due to their holonomic properties and dynamic movements. Ball-balancing robots, in the following denoted ballbots, has quite a unique way of moving, as they balance on a single ball by applying a torque to the ball through a set of wheels. This make ballbots move in fluent, elegant motions, but more importantly it enables them to move in any desired direction while simultaneously being able to turn around its vertical axis. Being able to move sideways is important to robots interacting with people, since people are continuously making small adjustments to their position when interacting especially during group conversations. This makes the ballbots well-suited for human-robot interaction.

Balancing on a ball, apart from the impressive look and behaviour, also allows a smaller base footprint similar to the size of a human, making the robot applicable to human environments with crowded spaces, tight corridors etc.

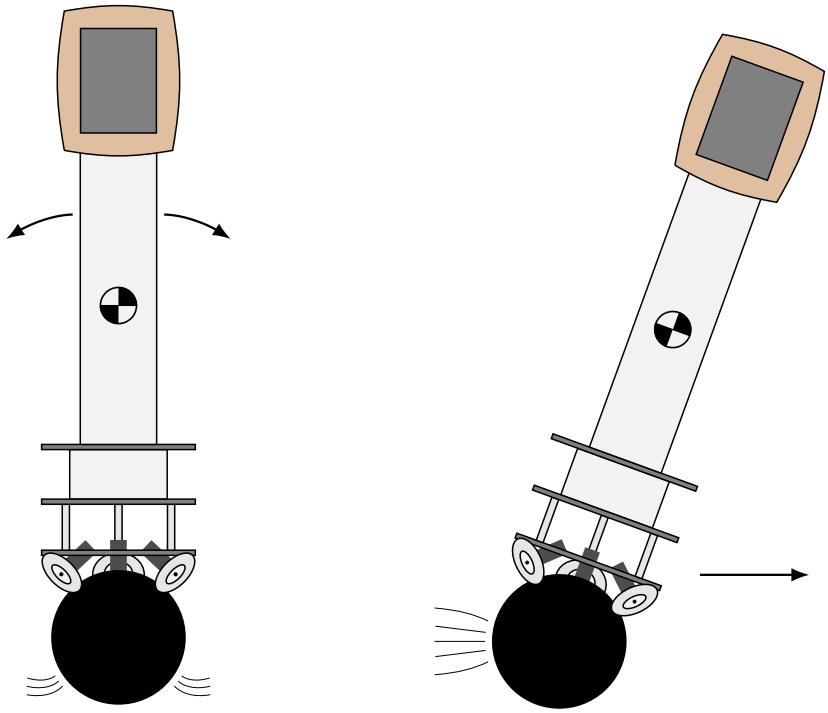


(a) 3 Omnidirectional wheels [2] (b) Inverse mouse-ball drive [3]

Figure 1.2: Ballbot actuation variants

Commonly ballbots consist of three omnidirectional wheels, known as omniwheels, placed in a configuration to span the rotational space of the ball, usually 120 degree spacing and a 45 degree zenith angle on the ball as shown in Figure 1.2(a). An omniwheel is a holonomic wheel that can apply forces in the tangential direction of rotation and allows low-friction motion in the lateral (perpendicular) direction. Other variants include the inverse mouse ball drive shown in Figure 1.2(b) where four drive rollers are placed on the side of the ball to deliver two degrees of freedom.

Similar to a Segway® [4], ballbots are inherently unstable and requires continuous control to keep the balance. Applying a torque to the ball through the motors and wheels will in return apply a counter-torque to the robot body due to Newtons 3rd law. In robotics this type of dynamically stable mobile robots form a special class of underactuated systems where the acceleration of the robot, in this case, is related to inclination of the robot resulting in a non-holonomic dynamic constraint. Keeping a steady state angle of the robot requires a constant torque to be applied to the body, which subsequently results in a constant torque applied to the ball, hence accelerating it, see Figure 1.3. Hence, whenever the robot is falling over, the motors has to counteract the tipping which in return will move the ball. This puts the ballbots in the category of shape-accelerated underactuated balancing systems [5].



(a) Small movements required when standing still

(b) Inclination causes acceleration

Figure 1.3: Shape-accelerated underactuated balancing system

Another way to picture the ballbot motion is to think of the robot being able to move the ball around below the robot body. To keep balanced ballbots constantly try to move the ball below its center of mass, denoted COM. In return the motions of a ballbot is more elegant, fluent and human-like by utilizing this underactuated dynamics, since desired movements would require the robot to tilt, making it look less static.

As described by Lauwers et. al. in [3], the continuous balance allow ballbots to have a narrower wheelbase and higher center of mass without tipping over when compared to wheeled robots of comparable size. Not only is this useful when navigating in crowded spaces and tight corridors but it also gives more freedom in the mechanical design.

Generally ballbots come in a variety of heights. However, the property of a single contact point with the ground makes the ballbots ideal for human-robot interaction, HRI, where a display has to be placed in eye level. The single contact point makes the ballbot less susceptible to changes in the drivable surface, since the balancing works as an active dampener, which is why a screen placed up high would shake less while driving than on a classical 2-wheeled robot without mechanical dampening.

1.1 'Kugle' research project

As described above, a ballbot has several advantages when used in HRI applications which is the main reason to investigate ballbots in this project. This thesis is carried out as part of the 'Robot Digital Signage' [1] project funded by the Innovation Fund with 12.7 million DKK under grant 7076-00051A, with several key partners including Aalborg University who is responsible for the research and development of the robot platform. The other partners include Teknologisk Institut, who is responsible for safety, testing and validation, Combine who is responsible for the UX-design and app development [6], Mapspeople who is responsible for indoor maps API and SDK and 'Det Gode Firma' who is responsible for API development [7]. The aim of the project itself is to develop, demonstrate and commercialize an interactive robotic digital signage solution including a robot that is intuitive to use and is controlled by a standard tablet so that app developers can deploy robot solutions for business applications across sectors. Robotic signage solutions will most likely require interaction with people and thus a highly customizable ballbot has been chosen as the primary robotic platform to be developed. The robot has been named 'Kugle' based on the danish translation of 'ball' [8] [9] [10].

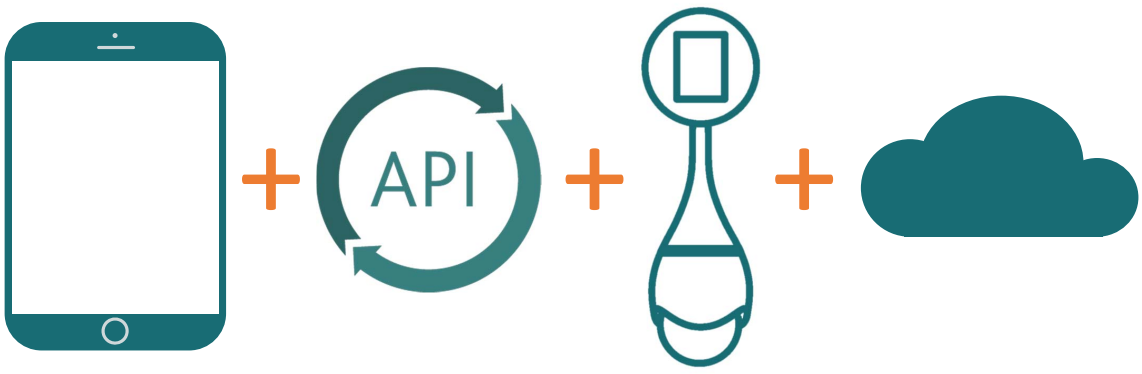


Figure 1.4: Robot Digital Signage with 'Kugle' robot [1]

Kugle is a personal social service robot that is highly customizable, easily deployable and programmable through app development, since Kugle is actually a Bluetooth-enabled device that can be connected to a mobile device like a tablet or phone which it carries around. At first glance, this may not seem very innovative, but this allow app-developers to program the behaviour of the robot.

Thus, companies who want a robot solution need only to contact an app-development house, which usually has a defined workflow with initial brainstorming and need identification processes, artists, back- and front-end developers and user experience experts. Kugle piggybacks onto this process, which results in a much shorter and cheaper development time, a clearer and simpler process, and a more robust product. There are millions of app-developers worldwide, but only thousands of robotic engineers specialized in social robots.

One example of service that Kugle could provide is to offer autonomous path guidance to people in environments such as airports, hospitals or malls [11], see Figure 1.5. In order to provide this service, Kugle must be capable of navigating through a known environment while continuously sensing its surroundings in order to avoid people and other obstacles. Furthermore, Kugle will feature advanced social intelligence making it able to navigate among people and interact with them in a socially acceptable manner, both on a one-to-one basis and in groups.

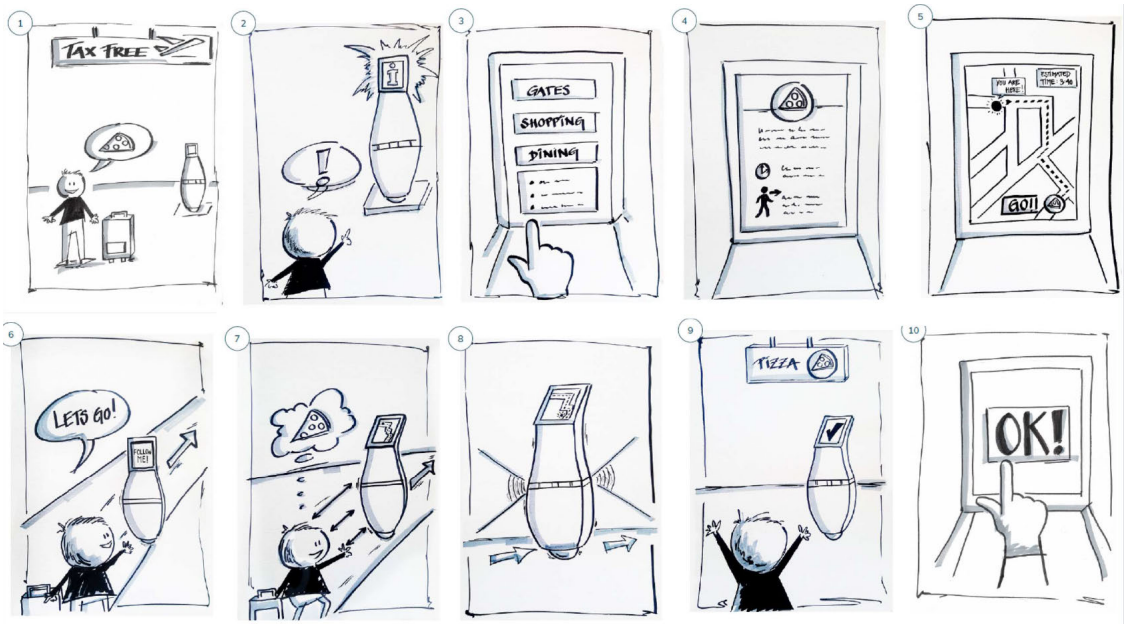
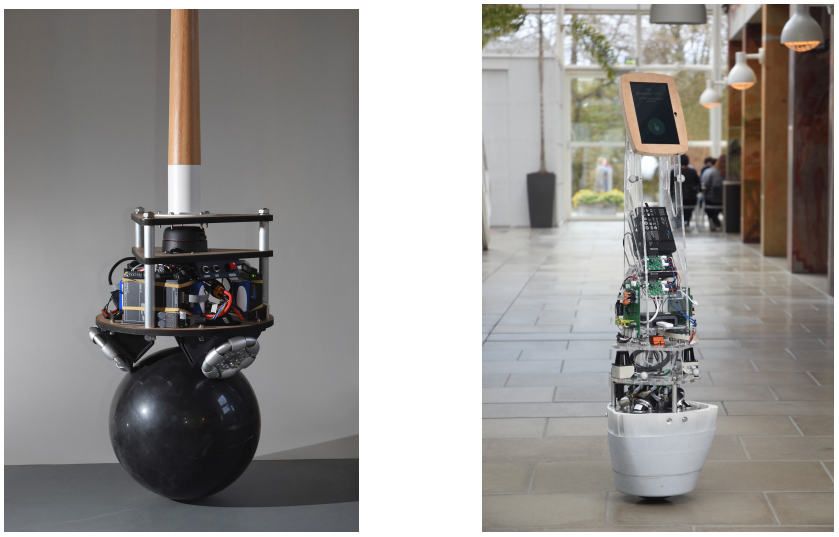


Figure 1.5: Storyboard of usecase example in airport

The purpose of this thesis, as part of the 'Kugle' research project, is to identify, develop and implement the necessary controllers to robustly stabilize and ensure that Kugle stays balanced. Simultaneously, the controllers should track acceleration, velocity and/or position references sent from an onboard computer installed on the robot.

An initial prototype of Kugle has already been built, denoted Kugle V0, shown in Figure 1.6(a). This prototype includes all the necessary mechanics and electronics to make a ballbot and furthermore includes a LiDAR and an onboard computer running ROS [12]. Unfortunately the prototype has some severe flaws including power management, charging of batteries, lack of gearing and lack of microprocessor compute power.

During this thesis a second prototype, Kugle V1, is built, as shown in Figure 1.6(b), including an improved mechanical design, industrial batteries with battery management and two set of IMU's including an expensive research-oriented IMU from Xsens [13]. A detailed description of the parts and properties of Kugle V1 is given in Section 2.1.

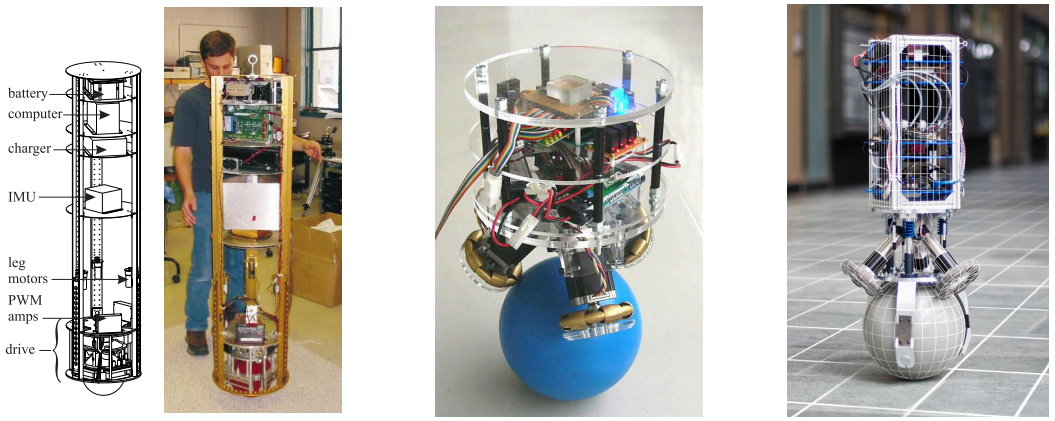


(a) Kugle V0 (b) Kugle V1

Figure 1.6: Kugle prototypes

1.2 Background & State-of-the-art

Within the past 10-15 years several researchers around the world have been working on modelling and controller design for ballbots. However, common to all published papers are the citation of three main contributions. The first ballbot was developed in 2006 at Carnegie Mellon University (CMU) driven by an inverse mouse-ball drive with four DC motors [3]. The second contribution comes from Tohoku Gakuin University (TGU) in 2008 with their 'BallIp' ballbot using an omniwheel configuration driven by stepper motors [14]. Finally, the most recent main contribution comes from Eidgenössische Technische Hochschule Zürich (ETH) in 2010 with their 'Rezero' ballbot utilizing an omniwheel configuration with geared brushless DC motors and with ball arresters to keep the ball in place, increase contact force and minimize slip [2].



(a) Ballbot from CMU [3] (b) 'BallIp' from TGU [14] (c) 'Rezero' from ETH [2]

Figure 1.7: Main ballbot contributions

A comprehensive literature review table including other notable contributions can be found in Appendix A. In the following sections the key observations from this comprehensive review is described. For a deeper literature review of each individual contribution the reader is refered to [15].

1.2.1 Mechanical design

One of the key differences among the existing ballbots are the mechanical design and especially the drive mechanism. Among the reviewed ballbots, all of them balances on a ball with a diameter of 200-250 mm and most of them use a drive configuration with 3 omniwheels spaced evenly, with a zenith angle, being the angle to the vertical axis, of 45 degrees. The main benefit of using the omniwheel configuration is to allow rotation around the vertical axis of the robot. The CMU ballbot [3] is not able to do this since it uses the inverse mouse-ball drive with mechanical rollers on the side.

For those with omniwheels the wheel diameter varies between 48 mm to 120 mm which is likely due to commercial availability of omniwheels, where inexpensive 48-60 mm two-ring versions exists for hobby-use [16] [17], see Figure 1.8(a), and a 10 times more expensive professional 100 mm single-ring version exists in stainless steel [18] or aluminum [19], see Figure 1.8(b).

(a) 60 mm two-ring aluminum omniwheel @ $15/ea. [17]

(b) 100 mm single-ring aluminum omniwheel @ $143/ea. [19]

Figure 1.8: Example of omniwheel types showing the difference between two-ring and one-ring

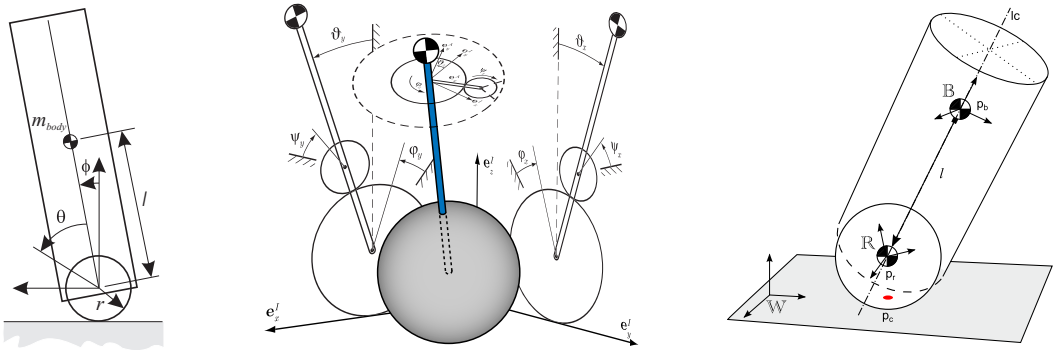
Both stepper motors, brushed and brushless DC motors have been used to actuate the ballbots. Stepper motors are used on the TGU ballbot [14], in a velocity control configuration since actual torque control of stepper motors requires advanced driver electronics [20]. The other ballbots use either brushed or brushless DC motors whose torque is controlled by controlling the current. The ETH ballbot uses a brushless motor with a high gearing ratio of 26:1 increasing the maximum torque to 5 N m to keep the robot of 9.2 kg balanced. In comparison the stepper motors on the TGU robot, with a weight of 12 kg, is only capable of delivering up to 1.3 N m. Overall a configuration with omniwheels and geared brushless DC motors is favoured among all the reviewed ballbots.

The ETH ballbot has a unique mechanical advantage compared to the other ballbots. The team has designed three ball arresters to hold the ball in place and push it up at against the wheels at all times, see Figure 1.2(a). Furthermore, they have installed a mechanical dampening system to reduce vibrations.

Except for these differences every ballbot follow the same design pattern by making a narrow but tall cylindrical structure in which all electronics and sensors are installed. The cylindrical design guards against irregular and non-uniform inertias, which ensures that the center of mass is kept as close as possible to the vertical z-axis.

1.2.2 Modelling

When it comes to the modelling, all the reviewed ballbot models initially simplifies the ballbot into two rigid bodies: a hollow sphere with ground contact, resembling the ball, and a body part either modelled as a cylinder or as a black-box rigid body with known mass, center of mass and inertia, see Figure 1.9(c). In most of the models the wheels are not included as a rigid body but instead it is assumed that the motors provide a coupled torque between the ball and the body. The wheel inertia is, however, still taken into account.



(a) Planar model of CMU ballbot [21]

(b) Projection strategy used for the planar model of ETH ballbot [2]

(c) Coupled 3D model used for INAL ballbot [22]

Figure 1.9: Ballbot modelling types

There exist two ways of modelling a ballbot: either as a planar model where the dynamics of rotation around the individual inertial axes are assumed to be completely decoupled, see Figure 1.9(a) and Figure 1.9(b), and the coupled 3D model, where the system is modelled with coupled dynamics. Obviously the planar models become much simpler and are easier to deal with since the decoupling reduces the dimension of state space. Furthermore, models such as [3] assumes the dynamics for rotation in the x-z plane to be the same as for rotation in the y-z plane, simplifying the dynamics even further. The main advantage of the planar models are the simplicity and human readable equations of motion as shown in (1.1) with the model from [21], which is similar to the one derived in Appendix K.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + D(\dot{q}) = \begin{bmatrix}\tau\\0\end{bmatrix} \quad \text{where} \quad q = \begin{bmatrix}\theta\\\phi\end{bmatrix},$$
$$\alpha = I_{\text{ball}} + (m_{\text{ball}} + m_{\text{body}})r^2, \quad \beta = m_{\text{body}}rl, \gamma = I_{\text{body}} + m_{\text{body}}l^2,$$
$$M(q) = \begin{bmatrix}\alpha & \alpha + \beta\cos(\phi)\\ \alpha + \beta\cos(\phi) & \alpha + \gamma + 2\beta\cos(\phi)\end{bmatrix}, \tag{1.1}$$
$$C(q,\dot{q}) = \begin{bmatrix}0 & -\beta\sin(\phi)\dot{\phi}\\ 0 & -\beta\sin(\phi)\dot{\phi}\end{bmatrix}, \quad G(q) = \begin{bmatrix}0\\ -\frac{\beta g\sin(\phi)}{r}\end{bmatrix}, \quad D(\dot{q}) = \begin{bmatrix}D_c sgn(\dot{\theta}) + D_v\dot{\theta}\\0\end{bmatrix}$$

For the ETH ballbot [2] both a planar and coupled model is derived. The drawing of their planar model, Figure 1.9(b), shows how the decoupled planar models are constructed by projecting the 3D model onto the inertial planes. The authors remarks the complexity of the computer-derived (Mathematica) equations of motion of their Euler-based coupled 3D model, taking up several megabytes of text in its symbolic form. However, the advantage of the coupled models are the ability to capture the change in dynamics when the robot is driving at fast velocities, turning while driving or doing complex maneuvers involving rapid changes in inclination.

The choice of model obviously affects the state representation and while the planar models tend to use either body angle and ball angle, as in Figure 1.9(a), or body angle and ball position, the coupled models exist in several variations. The most common representation of the body orientation is by using the 3-2-1 Euler angles, representing the orientation as a sequence of rotations around the Z-Y-X axes, as used in [2] [23] [24] [25]. This representation is commonly used in robotics due to its intuitive representation and visualization, but it involves several singularities known as gimball lock [26], which is why it is rarely used within aerospace engineering [27] [28]. As an alternative and to avoid the situation with singularities the complete description given by an SO(3) representation, the 3×3 rotation matrices, is used in [29] and [15]. Storing the SO(3) representation as part of the system state results in an overly large state space due to several redundant coordinates. Both papers using the SO(3) representation only carries out simulations and do not implement the controllers on actual hardware. Instead of using rotation matrices another singularity free representation is quaternions, which involves four state variables instead of nine and thus with only one redundant variable. In [22] and [30] a coupled ballbot model is derived using quaternions leading to a simplistic and yet singularity free set of equations of motion. Unfortunately the model and developed controllers are only tested in simulation.

The state of the ball is commonly represented by just the position of the center defined in the intial frame or by the amount of rolling around the x- and y-axis. Some models such as [23] and [31] furthermore allows the ball to rotate around the z-axis by capturing this in an extra state. The others assumes the rotational friction with the ground to be sufficiently big and thus includes a constraint to lock the rotation of the ball around the z-axis (yaw rotation).

When modelling a mechanical system it is essential to define a set of assumptions under which the model is valid. Among the reviewed models the following assumptions are commonly used:

- The ground is flat and level/horizontal
- The ball is rigid, i.e. the radius of the ball is constant
- The body, ball and floor do not deform
- The robot is simplified to a rigid cylinder and the ball to a rigid sphere.
- The ball is homogeneous, i.e. the center of mass is in the center of the sphere resulting in a diagonal inertia tensor with same entries.
- There is no slip between the ball and ground and the ball and drive mechanism (e.g., omniwheels), i.e. the instantaneous velocity at the contact points are always zero.
- The dynamics in the sagittal (x-z) and coronal (y-z) plane are identical.
- The ball does not rotate around the z-axis.
- The body can be modelled as a pendulum that rotates in a spherical joint in the center of the ball, which is a consequence of the no slip condition.
- The actuator dynamics are fast, i.e. the input has no delay.
- The wheels are always in contact with the ball (similar to the no slip condition).
- The contact between the ball and the ground and between the omniwheels and the ball are point contacts.
- The omniwheels apply only a tangential force and allow frictionless lateral motion.
- The distance between the center of mass of the ball and body is constant.
- The center of mass is located above the center the ball when the z-axis of the body is aligned with the inertial z-axis.
- Any non-linear friction components causing discontinuities are neglected.

With regards to the last item, most of the models assumes either no friction forces or only viscous friction between the ball and ground and ball to wheels. However, a few models such as [32], [15] and [33] includes the discontinuous and thus non-linear Coulomb friction and considers it during the controller design.

Deriving the equations of motion of a mechanical system can generally be done using one of two approaches: Newtonion mechanics or Lagrangian mechanics, also known as the Euler-Lagrange method.

In Newtonian mechanics you have to use mainly a rectangular coordinate system and consider all the constraint forces in the modelling process. Lagrange's scheme avoids the considerations of the constraint forces and you can use any set of 'generalized coordinates' like angle, radial distance etc. If, however, the number of generalized coordinates is not the same as the number of degrees of freedom in the system, constraint equations has to be included to reduce the generalized coordinate space.

Lagrangian mechanics have been used to derive all the reviewed ballbot models except for [22] [30], which derives a quaternion-based model using Newtonian mechanics for an inverse mouse-ball drive ballbot. The derivation involves several steps with complicated inversions of symbolic matrices derived from constraint equations. Conversely Lagrangian mechanics involves only one matrix inversion namely the mass matrix.

1.2.3 Control

Given a set of non-linear ordinary differential equations several type of controllers can be used to stabilize the system and track given references. However, only a few controllers and controller structures have been used on ballbots, mostly dominated by PID and LQR. Since both are linear controllers they require the non-linear ODEs to be linearized. When the ballbot is linearized around the upright equilibrium point any coupled effects between the saggital and coronal planes disappear. Therefore, it does not make sense to use a coupled model for linear controller design if the linearization remains the same.

The simplest controller design is found on the TGU ballbot [14] using two PD controllers affected by the inclination of the robot and the wheel angles and velocities. The gains are initially derived using a planar model but is later tuned manually.

In the CMU ballbot [3] and KHU ballbot [33] a cascaded structure shown in Figure 1.10 is used to suppress disturbances and linearization errors. An inner PI controller tracks a desired ball velocity while an outer LQR controller works on the full planar state space keeping the balance. This strategy enables both position control, angle control and velocity control through the LQR controller.

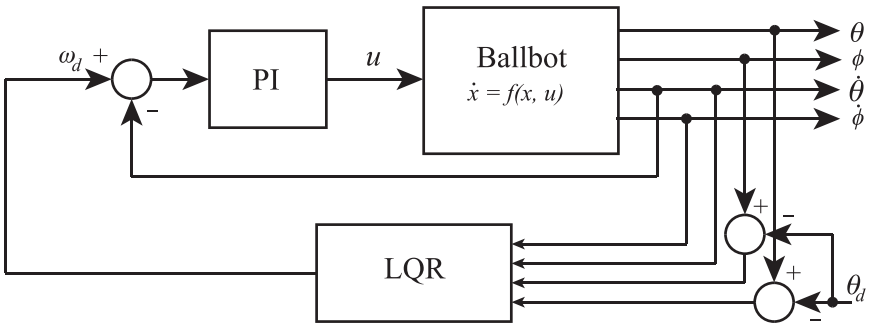


Figure 1.10: Controller structure of CMU ballbot [3]

In the bachelor thesis about the ETH ballbot [2] the authors present both an LQR controller, based on their planar decoupled model, and later a joint LQR working on the complete state space. They conclude that their joint LQR outperforms the decoupled LQR when performing complex manoeuvres involving translational and rotational movement simultaneously, thus coupling the different axes, even though the linearized system matrices of the coupled system shows a clear decoupled structure. The developed LQR controllers are tested with both angle references, position references and velocity references by leaving out certain states in the controller design. In the end they also implement and test a gain-scheduled LQR implementation. By dividing the velocity subspace into discrete regions they can linearize and compute unique LQR gains for each linearized region. The implemented controller performs an online interpolatation between all the computed gains based on the current state estimate. They show how the gain-scheduled controller works but do not conclude whether it improves the performance or operational envelope. However, they remark the curse of dimensionality of gain-scheduled controller, since sufficient discretization of their 10-dimensional state space requires millions of parameters.

Instead of working with precomputed gains another group at ETH continues where the bachelor thesis left, by taking a model predictive controller approach and compute time-varying LQR gains online [34]. Using an efficient implementation of the iLQG approach they are able to implement and run the MPC at 45 Hz with a 5 seconds horizon, enabling an optimal gain-scheduled approach without dealing with the curse of dimensionality.

Digging into the non-linear controller strategies, both computed torque and sliding mode have been suggested. In [23] [35] [15] [36] a sliding mode controller designed on the coupled non-linear ODEs is proposed. Liao et al. from NCHU [35] designs a non-linear sliding mode controller for the CMU ballbot as a hierarchical structure with two inner-layer sliding surfaces, one for inclination/balance and one for position, and an outer sliding surface combining the two. This structure allows for position control of the ballbot. Unfortunately the work is not implemented and has thus never been verified.

On the KHU ballbot [36] a similar hierarchical sliding mode controller for position control is designed and implemented on their human-carrier ballbot, unfortunately with less than satisfactory results including large overshoots and far from perfect station-keeping. In the work from CEES [15] both a non-linear computed torque and sliding mode controller is rigorously derived to control the balance of a ballbot based on his geometric model, given SO(3) orientation references. Similar to Liao et al. he does not carry out the implementation and test of his non-linear controllers and fails implementing a linear controller as well.

With both the ETH [2] and ALTEN [25] ballbot they mention issues with the omniwheels trembling causing them to slip. They both conclude that the trembling seems to be affected by the body angular velocity and the amount of noise in the corresponding state estimates. The problem is mitigated by applying a low-pass filter on either the angular velocity estimate or on the torque output, which is also considered in this thesis as described in Appendix O.2.

1.2.4 Trajectory planning

Due to the unstable nature of a ballbot, planning and executing a desired path is not a simple task, since the non-minimum phase dynamics of the ballbot has to be considered. As mentioned the inclination of the ballbot is related to the resulting acceleration. Following a desired geometric path, hence a sequence of points defined in the inertial frame, is thus a matter of planning a trajectory in time for the inclination of the ballbot.

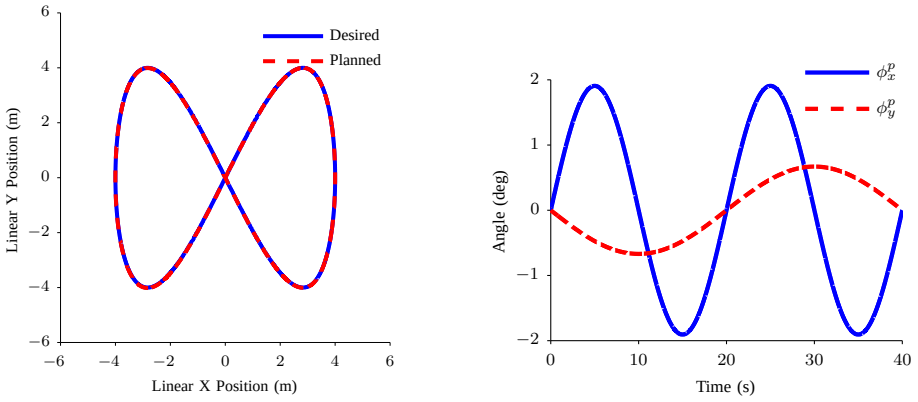
This type of planning strategy is called geometric shape-space planning, since the inclination or orientation of the ballbot defines the shape variables whereof the position of the ballbot defines the space variables. One way to combine optimal planning with optimal control is through Model Predictive Control.

The iLQG-based MPC [34] proposed by another group from ETH is one of the few tested MPC implementations for ballbots, but in this case the MPC do not consider the motion planning part of the movements. They propose an MPC to iteratively compute and adjust the LQR gains of a stabilizing position controller. The combination results in an optimal controller that gradually adjusts its trajectory while moving, resulting in a "swing-in" motion when coming to a stop. However, the only tests carried out involves step and ramp inputs and thus do not consider following a given reference path or trajectory. In a continuation of their work [37] the option of full-state waypoints to be reached, while navigating towards a target position, has been added, but trajectory following is still not tested.

In 2009 the group from CMU carried out an analysis on how to plan an inclination reference trajectory for the ballbot given a desired target position, taking the underactuated dynamics into account [21]. They propose a parametrization of the angle reference trajectory whose parameters are found through an optimization procedure using a cost function that tries to reach a desired position. Similar to the work by ETH this work only considers point-to-point navigation.

Some later work by the same group have continued the development of a general framwork to trajectory planning for shape-accelerated underactuated balancing systems [5], answering the question how shape trajectories of the ballbot can be chosen so that the ball moves along a desired path.

In both [5] and [38] they show how a shape trajectory can be computed through optimization, given a desired acceleration trajectory in the position space, as illustrated in Figure 1.11.

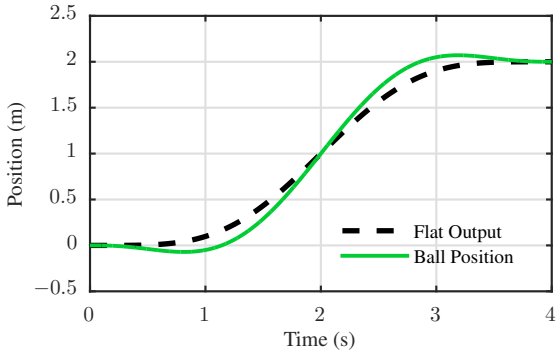


(a) Desired figure-8 trajectory [5]

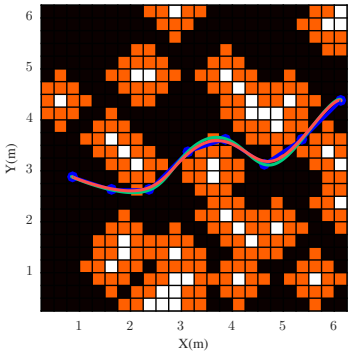
(b) Planned shape trajectories for lean angle of the two axes [5]

Figure 1.11: Shape trajectory planning

In the dissertation by Michael Shomin from CMU a differentially flat output, see Figure 1.12(a), of the ballbot is derived and used for trajectory optimization, enabling arbitrary motion and free constraints to e.g., the position [39]. He shows how his implementation is capable of taking in an optimal kinematic path, e.g., derived from an A* path planning algorithm, and convert this to a dynamically feasible path for the ballbot which is later converted to an angle trajectory given to the balance controller, see Figure 1.12(b).



(a) Desired flat output trajectory and corresponding feasible trajectory



(b) Subsampled waypoints are used to generate an optimal feasible trajectory. Green trajectory shows planned motion of ball and red trajectory shows planned motion of the robot COM.

Figure 1.12: Shape-space planner from CMU [39]

1.3 Scope of thesis

Based on the existing research presented in the previous section it is clear how a lot of effort have been put into the use of planar models and design of linear controllers. The purpose of this thesis should be to contribute with new research to the ballbot community.

1.3.1 Model

In terms of modelling, a coupled model should be derived since a lot of researchers have already derived and used the planar models. The existing coupled models are primarily derived on Euler angles, however, not only are the Euler angles in a risk of falling into a singularity, Euler angles also include discontinuities in regular use e.g., when turning 360 degrees around the z-axis (yaw) where it jumps from $-\pi$ to π [28]. Other models using the SO(3) representation or quaternions have been derived to deal with this problem, but none of these models have been implemented and tested in practice. Unit quaternions have the property of capturing all of the geometry, topology and group structure of three dimensional rotations in the simplest possible way. They provide advantages in the complexity of computations compared to both Euler angles and SO(3). A comparison table that lists the main differences between these three rotation descriptions can be found in Appendix B.

Equations of motion derived from models using Euler angles result in large expressions with long sequences of sine and cosine products, whereof equations of motion from quaternion-based models have simpler derivatives and should therefore be faster to compute and thus also faster to simulate or use actively in the control approach. Hence if simulations are to be carried out for non-linear state estimation or for real-time MPC, a quaternion-based model is more efficient [40]. Based on the above arguments it is decided to use quaternions to represent the orientation of the ballbot. The only downside to quaternions is that they are far more complex to understand and visualize than Euler angles, where people are familiar with roll, pitch and yaw angles.

In [22] a quaternion model was derived using Newtonian mechanics. As mentioned Newtonian mechanics works well for mechanical systems in rectangular coordinate systems. However, when the frames starts to rotate, Newtonian mechanics require artificial constraint forces to tie everything together. The main advantage of Lagrangian mechanics is exactly that constraint

forces do not have to be considered and that equations of motion can be derived blindly as long as the total kinetic and potential energies of the system can be described. Since a ballbot involves both translating and rotating bodies and thus both translating and rotating frames, including all the necessary constraint forces at the right places within a Newtonian mechanics framework will be troublesome and prone to errors. It is therefore decided to use Lagrangian mechanics to derive a quaternion-based coupled model of the ballbot. To the author's knowledge no ballbot model based on quaternions has been derived using Lagrangian mechanics, which is why the work will also be a new contribution to the research.

1.3.2 Controller

Due to the under-actuated nature of the system, it is challenging to design a unified controller. Instead, it is often chosen to design a two-loop control structure to couple the under-actuated states in the outer-loop to the actuated states in the inner-loop. Not many researchers have shown practical success with non-linear controller design for ballbots, but several have proposed especially the sliding mode controller and some also tested it with minimal success. The sliding mode controller has the benefit of being robust against matched disturbances, hence being tolerable against model uncertainties and inaccuracies e.g., due to simplifications, incorrect friction models and external disturbances such as a push. The scope of this thesis is to derive a sliding mode controller for balance control, thus a controller that can take in orientation references and angular velocity references, and most importantly verify its functionality in practice. This should help to identify whether further effort should be put into the design of non-linear controllers for ballbots.

1.3.3 Shape-space planning

To enable station-keeping or to have the ballbot drive with a certain velocity the sliding mode controller should either be extended with a hierarchical structure similar to [36], or another controller should be designed around the sliding mode controller, to set angle and angular velocity references. This other controller could be a linear PID or LQR controller, but it could also be a non-linear model predictive controller.

Model predictive control, MPC in short, solves an optimization problem in real-time, thereby yielding similar desirable performance characteristics as an LQR controller. MPC is useful for processes who are difficult to control with standard PID controller, e.g., with large time constants, substantial time delays, inverse response or non-minimum phase, which is the case for the ballbot. It is especially useful for processes that requires planning and iterative replanning, and is thus able to overcome non-holonomic challenges, such as planning a shape-trajectory for the ballbot. One of the main troublemakers in linear control is constraints, e.g., saturation. Constraints on both process variables and manipulated variables can be considered in the online optimization problem solved by the MPC and are thereby accounted for. With a feasible optimization problem, sufficiently long horizons and terminal constraints, overall stability can also be guaranteed. This gives the MPC an intrinsic robustness.

To the authors best knowledge nobody has combined a shape-space planner with model predictive control for use on a ballbot. The question is if the work from [39] can be combined into an online optimization framework for use in closed-loop control of the ballbot.
Given a sliding mode controller capable of tracking angle and angular velocity references, it is decided to develop a model predictive controller for online shape-space planning to make the ballbot follow a desired path. The output of the MPC, being angle and angular velocity references, should be sent straight to the sliding mode controller for tracking.

1.4 Problem formulation

This leads to the following three problems which are the research topic of the thesis.

1.
Can a quaternion-based model of a ball-balancing robot, for use in controller design, be derived using Lagrangian mechanics?

2.
Can a non-linear Sliding mode controller be used to stabilize and control the ball-balancing robot and how does it compare to a traditional LQR?

3.
Can Model Predictive Control be used to control the motion of a ball-balancing robot and make it follow a given path?

The first two problems are covered in the remainder of this written thesis. The third problem, related to shape-space planning using model predictive control, is due to time and space constraints covered briefly in Appendix S which refers to the GitHub repositories including the corresponding MATLAB code, C++ code and Gazebo simulation, see `Kugle-MATLAB/Controllers/MPC`, `Kugle-Misc/src/MPC_Test`, `Kugle-ROS/kugle_mpc` and `Kugle-Gazebo`.

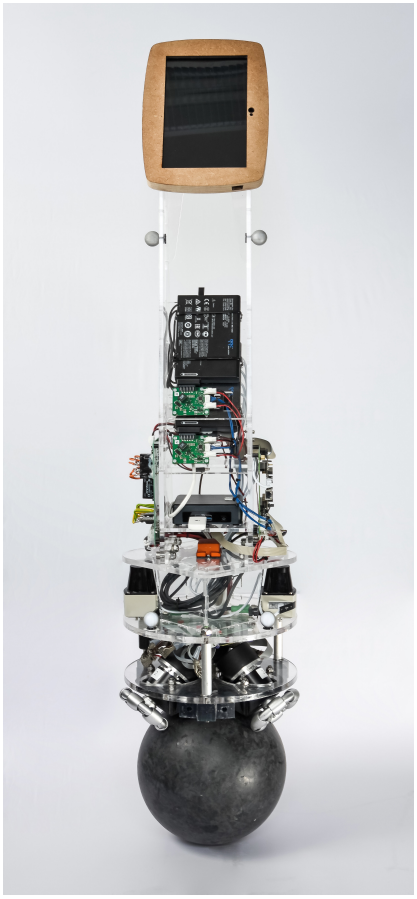
2 System Overview

The purpose of this chapter is to introduce the ballbot hardware, on which the controllers developed in this thesis are tested. An essential part of this thesis is to verify the theory in practice in an efficient and structured way. After presenting the hardware, a proposed system architecture diagram for an autonomous ballbot, is used to illustrate the elements contained within the scope of this thesis.

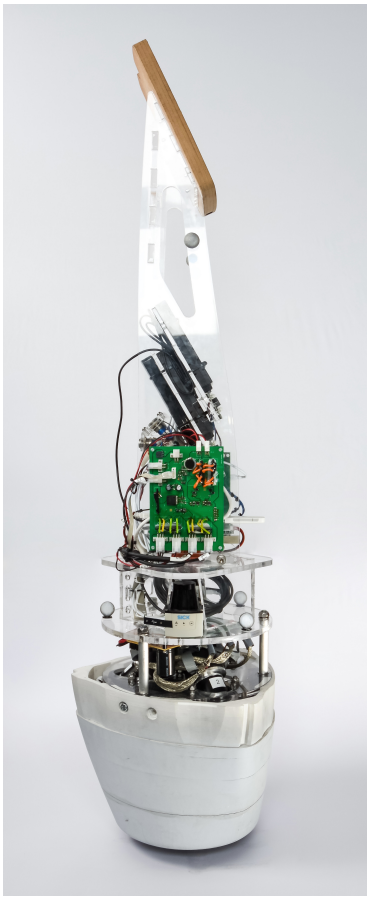
2.1 Kugle V1 prototype

In Section 1.1 two ballbot prototypes within the Kugle research project was mentioned. During the development of Kugle V1 some initial tests were carried out on Kugle V0 to verify some assumptions, estimator design and initial LQR controller design. However, in the remainder of this thesis, only Kugle V1 is used.

Kugle V1 is a ballbot prototype built from an 8-10 mm acrylic sheet structure with a Lenovo tablet at the top, located in eye level. The prototype is built with a layered strategy and with careful attention to keeping the center of mass close to the z-axis, intersecting the center of the ball.



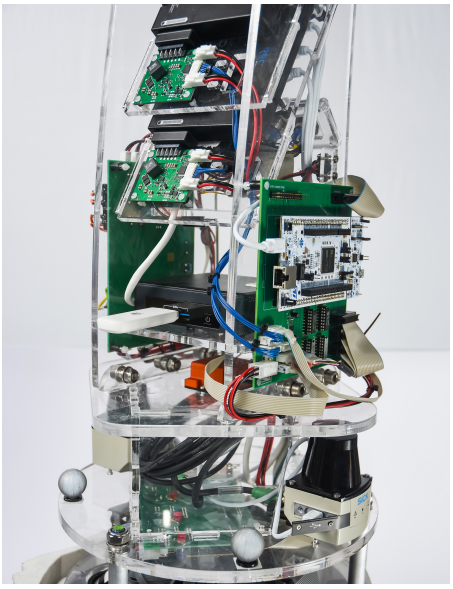
(a) Front without shell. Body y-axis pointing to the left



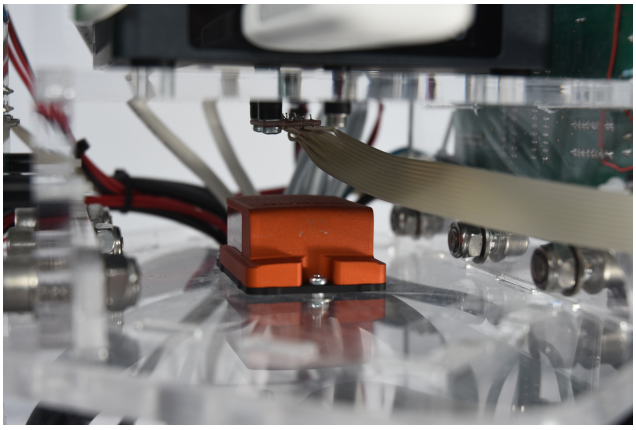
(b) Side with shell. Body x-axis pointing to the left

Figure 2.1: Kugle V1 ballbot prototype

The prototype includes an ARM Cortex based microprocessor running at 400 MHz with 1 MB of RAM, 2 MB of flash and a double precision floating point unit. Furthermore, an Intel NUC onboard computer is installed on which Ubuntu 16.04 and ROS Kinetic is running. Two planar SICK LiDARs with 270° coverage each and connected to the onboard computer, scans the surroundings for localization and obstacle avoidance. In the future one or multiple RGBD cameras will be installed for human interaction and obstacle avoidance.



(a) Visible from the top: batteries, microprocessor board, Intel NUC, Xsens IMU, SICK LiDARs



(b) MPU-9250 in the top and Xsens MTI-200 IMU at the bottom

Figure 2.2: Electronics on Kugle V1 prototype

To measure the orientation two IMUs are installed, an inexpensive MPU-9250 for prototyping and an expensive factory calibrated Xsens MTI-200 with internal bias correction for research and validation, see Figure 2.2(b).

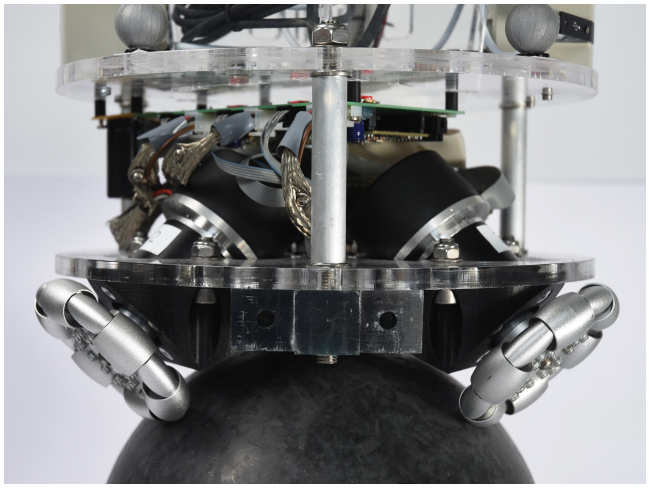


Figure 2.3: Geared brushless DC motors and single-ring omniwheels

Three geared brushless DC motors with quadrature encoders from Maxon constitutes the actuators on which 100 mm single-ring aluminum omniwheels are installed, see Figure 2.3. The three omniwheels are mounted with 120° spacing and a zenith angle of 45°. The motors are driven by three programmable ESCON motor drivers capable of delivering a continuous current of 5 A.

The ball is made from a 4 mm thick spherical polyethylene shell with an outer diameter of 250 mm, coated with 4 mm polyurethane rubber, thus a resulting outer diameter of 258 mm.

The prototype is powered from two 4s3p intelligent lithium-ion batteries connected in parallel, with a nominal capacity of 6600 mAh each. Two power management boards enables easy charging using an external power supply while simultaneously allowing the microprocessor to monitor the charging and discharging process.

A frontpanel with five buttons and LEDs, see Figure 2.4, is installed on the back. Three of the buttons are programmable by the microprocessor with one of them being user reconfigurable. Two buttons are hardwired to control the main power and motor power. Two of the programmable buttons are used for system reset and calibration. The action of the reconfigurable shutdown button is selected through the ROS user-interface, see Figure O.1 and `Kugle-ROS`, and per default the button starts and stops the controller.

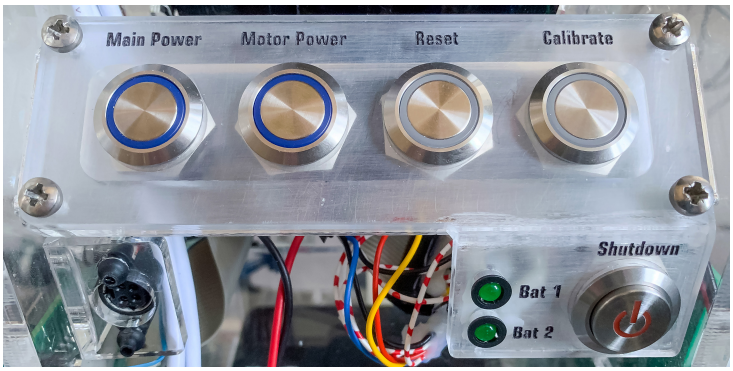


Figure 2.4: Frontpanel with two power buttons, two soft-programmable buttons and one user reconfigurable shutdown button

The hole in both sides close to the top, see Figure 2.1(b), allows a safety thread to be installed and hung onto a solid metal frame for testing. Furthermore, the thread is used as a safety grip while walking with the robot during testing. Since the robot frame is made with acrylic sheets and is thus porous, a lot of care has to be taken while testing to ensure that the robot will not eventually tip over and break.

(a) Motor assembly with shell

(b) Motor assembly without shell

Figure 2.5: Bottom part with and without protective shell

A protective shell covers the bottom part to keep the ball in place. The part does not put any pressure on the ball when the robot is balancing, but if the robot is lifted the shell keeps the ball contained inside the shell. This furthermore helps to ensure that the ball will never roll away from the robot if a wheel slip occurs, although it does not reduce wheel slip since the shell does not push the ball up against the omniwheels.

A table of all the hardware components installed on Kugle V1 can be found in Appendix C.

2.1.1 Mechanical layout

The mechanical layout of Kugle V1 is shown in Figure 2.6 including some dimensions and the frame definition, with origin in the center of the ball.

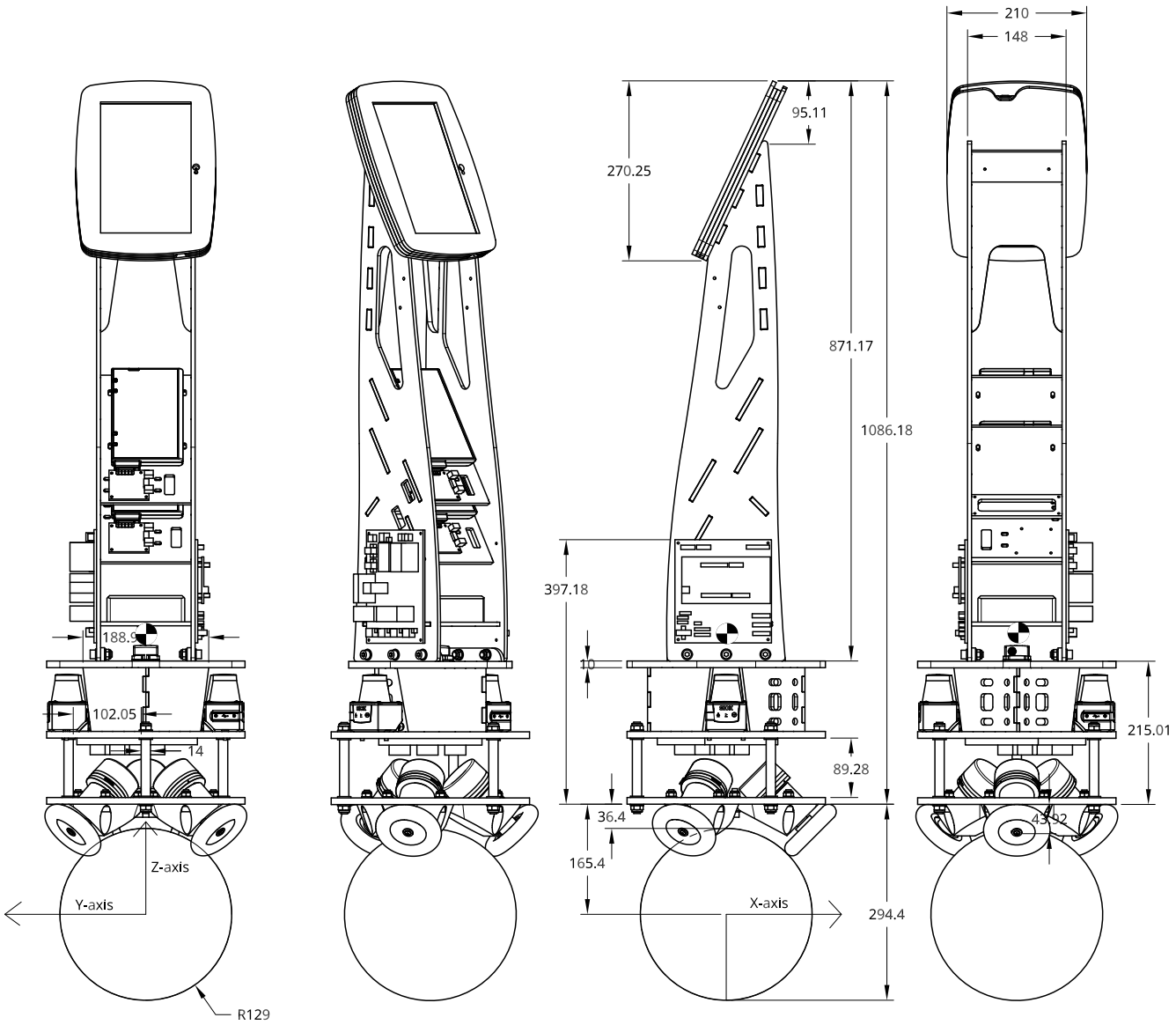


Figure 2.6: Drawing of Kugle V1 including center of mass location (measurements in millimeters)

These mechanical drawings are extracted from the CAD drawing, from which several model parameters including the center of mass and model inertia, see Appendix D, are extracted. A dynamic rendering of the CAD model is shown in Figure 2.7.

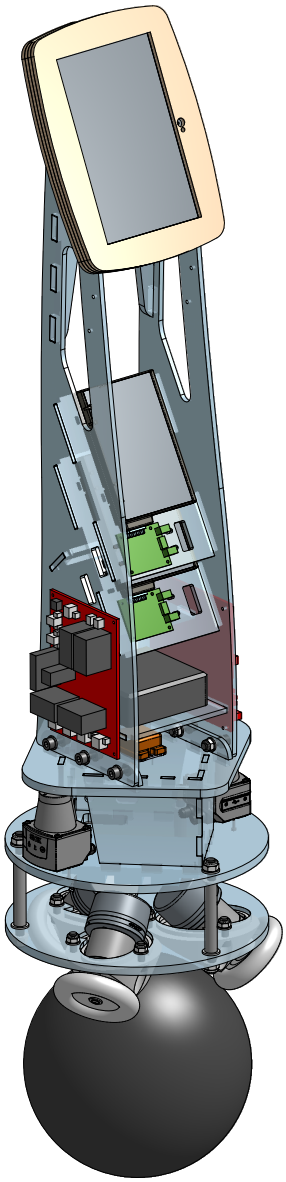


Figure 2.7: 3D CAD model of Kugle V1. Rotate the view/model by opening the PDF in Adobe PDF reader.

Note that the shell shown in Figure 2.5(a), covering the bottom part, is not included in the CAD drawing since it was designed elsewhere. The mass of the shell is, however, included in the body mass listed in the next section.

2.2 Model parameters

When modelling a ballbot several mechanical and electrical parameters are needed. Given the chosen modelling scheme described in Chapter 3, Chapter 4 and Chapter 5, most of the model parameters can be extracted from the CAD drawing while others are extracted from hardware datasheets. Table 2.1 contain the model parameters and specific values for Kugle V1 which are used in the following chapters.

Group	Symbol	Description	Value	
General	g	Gravity	$9.82\ \mathrm{m/s^2}$	
Ball	r_k	Radius of ball	129 mm	
	M_k	Mass of ball	1.478 kg	
	J_k	Inertia of ball, assumed uniform	$15.4 \times 10^{-3}\ \mathrm{kg\,m^2}$	
Body	M_b	Robot body mass (including motors, gears, encoders, wheels, electronics, batteries, tablet etc.) with shell	16.154 kg	
		Robot body mass without shell	14.31 kg	
	${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}$	Center of mass vector with respect to center of ball	$\begin{bmatrix} -0.02 \\ -3.21 \\ 421.2 \end{bmatrix}$ mm	
	l	Distance from center of ball to body center of mass $\left(\left\| {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} \right\|\right)$	421.3 mm	
	J_{bx}	Robot body inertia around x-axis (defined in center of ball)	$4.173\ \mathrm{kg\,m^2}$	$\star$
	J_{by}	Robot body inertia around y-axis (defined in center of ball)	$4.161\ \mathrm{kg\,m^2}$	$\star$
	J_{bz}	Robot body inertia around z-axis (defined in center of ball)	$0.1004\ \mathrm{kg\,m^2}$	$\star$
Kinematics	α	Wheel zenith angle	45°	
	γ	Wheel separation	120°	
Motor	n_{gear}	Gear ratio (4.3 : 1)	13/3	
	J_m	Motor inertia	$1.21 \times 10^{-4}\ \mathrm{kg\,m^2}$	
	J_g	Gear inertia (as seen by motor)	$1.2 \times 10^{-6}\ \mathrm{kg\,m^2}$	
	$\tau_{\max}$	Max output torque	1.9825 N m	$\star$
Wheel	M_w	Mass of omniwheel	270 g	
	r_w	Radius of omniwheel	5 cm	
	J_{ow}	Inertia of omniwheel [2]	$9 \times 10^{-4}\ \mathrm{kg\,m^2}$	
	J_w	Total inertia of wheel (on output shaft), combining both omniwheel, gear and motor inertia	$3.19 \times 10^{-3}\ \mathrm{kg\,m^2}$	$\star$
	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 m / (rad/s)	
	B_{vb}	Viscous (air) friction of body	0 N m / (rad/s)	

Table 2.1: Model parameters for Kugle V1. Values marked with $\star$ are derived in Appendix D.

2.3 System architecture

Given the available hardware components and the chosen scope of thesis, the system architecture shown in Figure 2.8, is proposed. The diagram separates into two parts, based on the tasks suitable for the microprocessor and the onboard computer. Real-time critical tasks, such as the sliding mode balance controller and the estimators needed for this controller, is put in the microprocessor. The estimates needed for the balance controller includes an orientation estimate from the Quaternion Extended Kalman Filter (QEKF), see Appendix M, using the IMU measurements, and a velocity estimate provided by the Velocity Extended Kalman Filter (VEKF), see Appendix N.

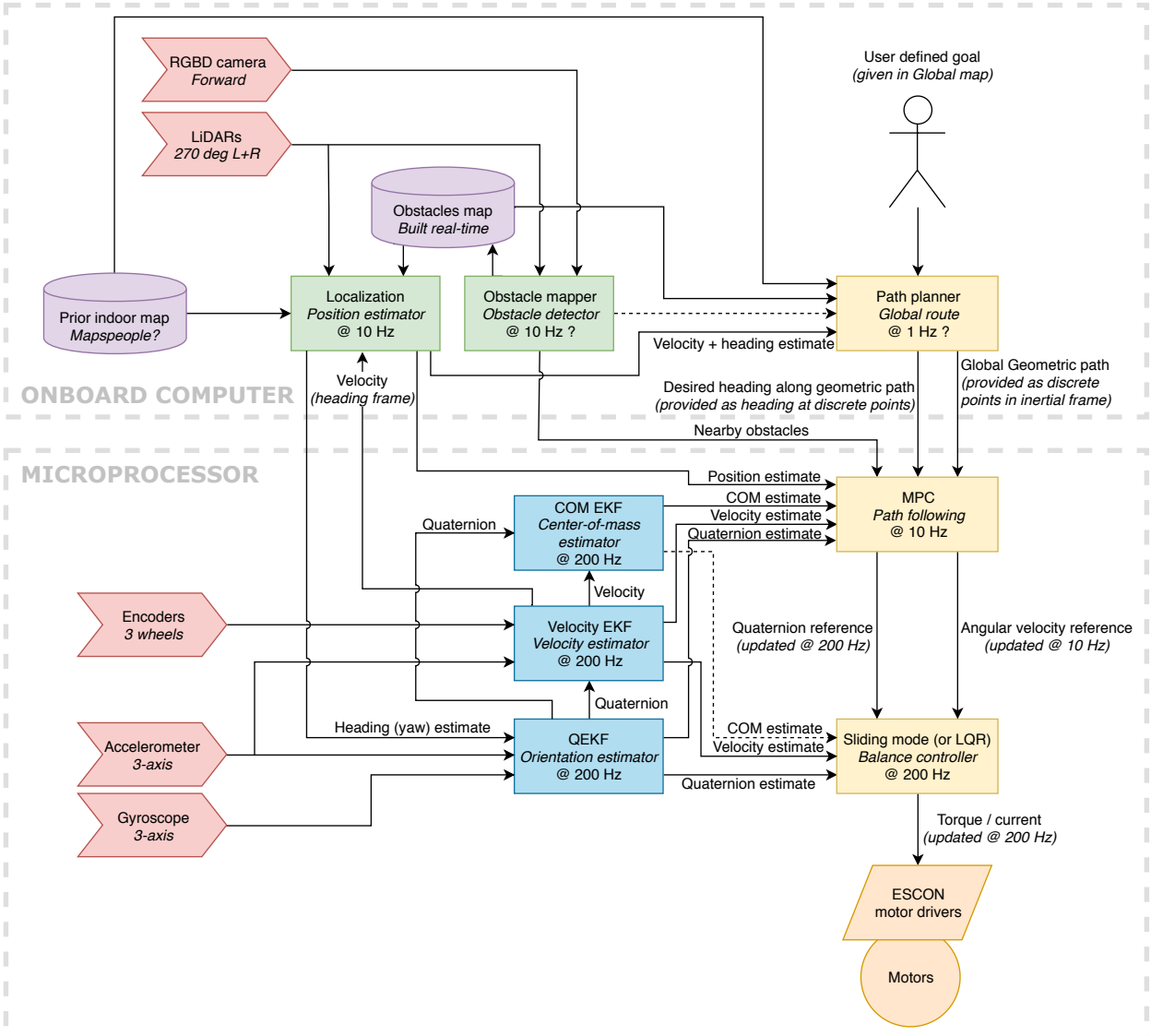


Figure 2.8: Suggested system architecture

As described in Chapter 1 the ballbot is shape-accelerated, which means that the robot will accelerate when the inclination is non-zero. However, the definition of zero inclination is relative to the location of the center of mass and not to the orientation estimate itself. The ballbot is stable whenever the center of mass is right above the center of the ball. However, if a misalignment between the expected center of mass, the orientation estimate and the actual center of mass is present, the ballbot will accelerate, even though the reference to the balance controller is set to upright.

To deal with this kind of alignment issue an initial calibration between the center of mass and orientation estimation frame, can be carried out. However, since both the orientation and acceleration is observable with the available sensors, a center of mass estimator can be developed to estimate the misalignment. Finally, the alignment issue can also be dealt with by including an integral term in a velocity or path following controller, see Chapter 14.

Being able to balance is one thing, but to move the ballbot around it has been decided to use a model predictive controller, MPC, see Appendix S. As a rule of thumb the bandwidth of the MPC should be at least 3 times smaller [41] than the sliding mode controller due to the cascaded structure. This should allow the sliding mode controller to settle to the references given by the MPC and thus avoid instability. Preferably the MPC should be placed in the microprocessor to ensure real-time execution. However, due to the smaller bandwidth and closed-loop stability of the balance controller, the MPC can also be offloaded to the onboard computer.

The MPC controls the position of the ballbot and will thus need an estimate of the current location. This estimate can be computed based on pure localization using the LiDAR sensors and an existing map of the environment, or real-time multimodal SLAM combining both the LiDAR sensors and RGBD camera(s).

Since the scope of this thesis is to derive a sliding mode controller for balance control and a model predictive controller for path following, the estimator design, localization, path planning etc. is outside of the scope. However, to ensure proper state estimation it has been necessary to develop both the quaternion estimator, the QEKF, and the velocity estimator, the VEKF, which are described in Appendix M and Appendix N. An existing localization solution in ROS, the Adaptive Monte Carlo Localization (AMCL) [42], has been utilized for position estimation.

ROS, the Robotic Operating System, is installed and used on the onboard computer. ROS provides a large set of preinstalled libraries, tools and especially message passing capabilities, enabling quick prototyping without having to reinvent the wheel. AMCL [42] is used for localization while 'move_base' [43] is used for path planning and navigation and extended with the MPC, see Appendix S. The embedded microprocessor communicates with the onboard computer over USB and is linked to the ROS infrastructure with a custom ROS driver that has been developed in `Kugle-ROS/kugle_driver`.

This concludes the system overview of a very capable prototype platform for research, development and tests.

Part I

Ballbot Quaternion Model

3 Model Considerations

The first part of the thesis deals with the derivation of a non-linear model of the ballbot using quaternions to describe the orientation. For a mechanical system like the ballbot, the modelling process begins by decomposing the system into a set of rigid bodies from which a set of common frames are defined. The model of a mechanical system generally consists of a kinematic part and a dynamic part, both of which are derived individually for the ballbot in the following chapters. To derive these parts in a practically feasible manner, one normally defines a set of modelling assumptions, most likely simplifying assumptions. After the derivation the model is verified, but due to the unstable open-loop dynamics of the ballbot, it is decided to indirectly verify the dynamics through a closed-loop simulation and test, with the controller. Kinematics are, however, verified. For symbols and notation the reader is referred to the nomenclature on page IV.

In this chapter the rigid bodies and frames used to define the ballbot is presented, followed by a list of modelling assumptions used in the following chapters.

3.1 Frame definitions

It is decided to decompose and simplify the ballbot into two rigid bodies, a ball and a body with three wheels, see Figure 3.1. The ball is assumed to be a uniform spherical shell with known mass and inertia. The body is a simplification of the robot body on top of the ball, hence including motors, wheels, electronics etc., generalized as an object with a known mass, center of mass and inertia tensor, which is assumed to be a diagonal matrix.

Given a world with a fixed inertial frame, $\{I\}$, a set of frames are assigned to the rigid bodies. The origin of all frames except $\{B'\}$ are located in the center of the ball, which thereby defines both the position of the ballbot and the origin of the rotation axes of the body.

- $\{K\}$: the ball frame is attached to the center of the ball and its orientation is always aligned with the inertial frame.
- $\{K'\}$: the rotating ball frame is rigidly attached to the ball, with its origin in the center, and thus rotates with the ball as the ballbot moves.
- $\{H\}$: the heading frame is attached to the center of the ball and is always horizontally aligned, thus the z-axis points in the same direction as the inertial frame z-axis. The x-axis points in the same direction as the projection of the body x-axis onto the xy-plane of the inertial frame, thereby defining the heading/yaw of the body.
- $\{B\}$: the body frame is rigidly attached to the body but with its origin in the center of the ball.
- $\{B'\}$: the body COM frame is rigidly attached to the center of mass of the body.

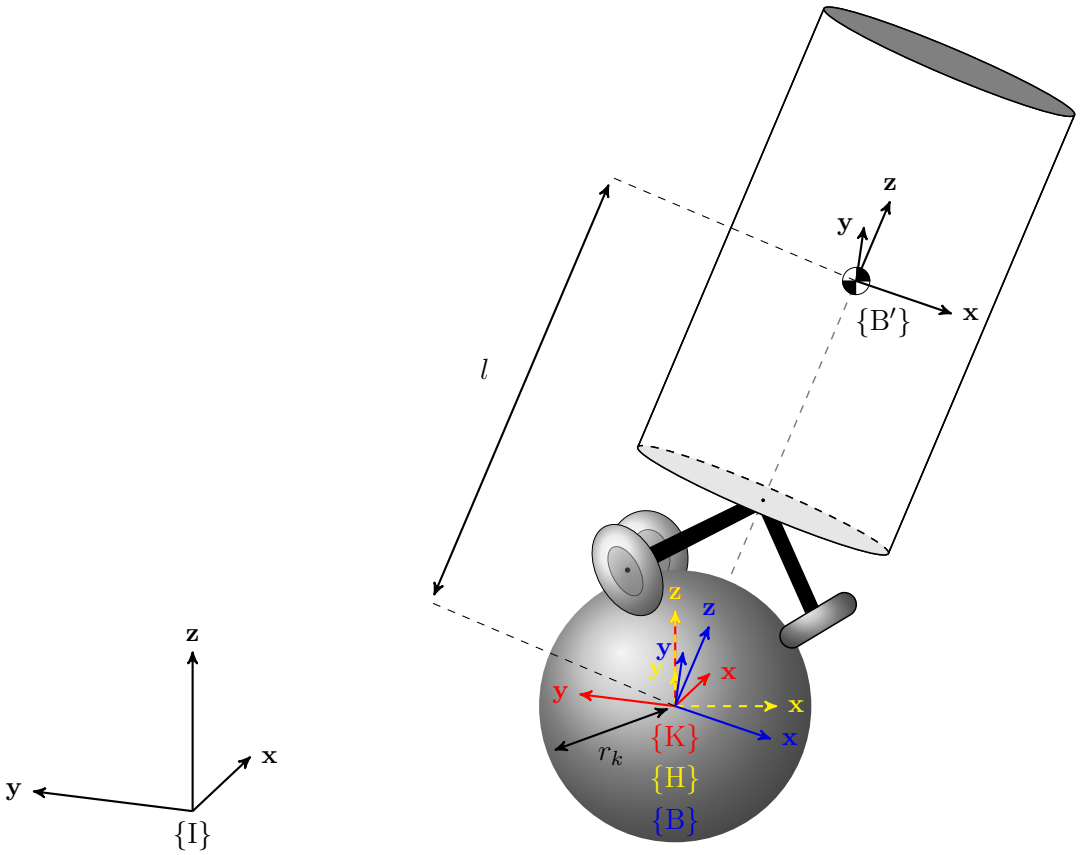


Figure 3.1: Ballbot model frame definitions and rigid body simplification.

The following relations exists between the defined frames

1. $\{I\}$ to $\{K\}$ is only a translation in the xy-plane which defines the position of the ballbot, ${}^{\mathrm{I}}\boldsymbol{O}_{\mathrm{K}} = \begin{bmatrix} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & 0 \end{bmatrix}^{\mathsf{T}}$
2. $\{K\}$ to $\{K'\}$ is only a rotation which defines the rotation of the ball, kinematically linked to the position of the ball
3. $\{K\}$ to $\{B\}$ is only a rotation which defines the orientation of the ballbot, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$
4. $\{B\}$ to $\{B'\}$ is only a translation given by the location of the center of mass, ${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}$

A few other intermediate frames are defined during the model derivation, but the frames listed above are the common frames used throughout the thesis.
The frames are defined relative to each other with an SE(3) transform. An SE(3) transform is a single 4×4 transformation matrix, utilizing the properties of homogeneous coordinates by embedding an SO(3) rotational component and a 3-dimensional translational component.
The rotation matrix, rotating a vector from frame {B} to frame {A} is defined as

$$ {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R} = \begin{bmatrix} {}^{\mathrm{A}}\boldsymbol{i}_B & {}^{\mathrm{A}}\boldsymbol{j}_B & {}^{\mathrm{A}}\boldsymbol{k}_B \end{bmatrix} \in \mathrm{SO}(3) \tag{3.1} $$

Where the column vectors are the unit vectors of frame {B} defined in frame {A}.
The transformation matrix for vector transformation from frame {B} to frame {A} is

$$ {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{T} = \begin{bmatrix} {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R} & {}^{\mathrm{A}}\boldsymbol{O}_{\mathrm{B}} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3) \tag{3.2} $$

With the origin of frame {B} in frame {A} defined as ${}^{\mathrm{A}}\boldsymbol{O}_{\mathrm{B}}$. With the use of homogeneous coordinates transformations can hereby be described with linear matrix multiplications.

$$ {}^{\mathrm{A}}\underline{\boldsymbol{p}} = {}^{\mathrm{A}}_{\mathrm{B}}\mathbf{T}\,{}^{\mathrm{B}}\underline{\boldsymbol{p}} \tag{3.3} $$

where ${}^{\mathrm{B}}\underline{\boldsymbol{p}}$ is the homogeneous coordinates of a point in frame {B} and ${}^{\mathrm{A}}\underline{\boldsymbol{p}}$ is the transformed homogeneous point, now represented in frame {A}.

3.2 Model assumptions

The assumptions under which the ballbot model is derived is listed below, some of which were mentioned in Section 1.2.

1. The ballbot is simplified as two rigid bodies, a ball and a body with three wheels.
2. The ground is flat and level (horizontal).
3. The ball does not deform, i.e. the radius of the ball, r_k, is constant.
4. The ball is homogeneous, i.e. the center of mass is in the center of the sphere resulting in a diagonal inertia tensor with same entries, J_k.
5. The ball does not rotate around the z-axis.
6. The ball only moves in the x-y plane.
7. The body is defined with the center of mass, ${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}$, the mass, M_b, and a diagonal inertia tensor with the elements J_{bx}, J_{by}, J_{bz}.
8. The location of the center of mass is known relative to the body frame, ${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}$.
9. The center of mass is assumed to be constant after initialization.
10. Inertia from the motor, gear and omniwheel are combined into the wheel inertia, J_w.
11. The actuators are torque controlled, delivering a commanded torque on the output shaft.
12. The actuator dynamics are fast, i.e. no delay on the input.
13. The omniwheels applies only a tangential force and allows frictionless motion in the lateral direction.
14. There is no slip between the ball and ground, so that ball rotation is kinematically linked with the ball position.
15. There is no slip between the ball and omniwheels, enabling the body to be modelled as attached with a spherical joint to the center of the ball.
16. The contact between the ball and the ground and between the omniwheels and the ball, are all point contacts.
17. There are no non-linear friction components. Only viscous friction between the ball to ground, B_{vk}, wheel to ball, B_{vm}, and air friction from body rotation, B_{vb}, is considered.

3.3 Generalized coordinates

When modelling with Lagrangian mechanics one have to define an independent set of coordinates with a dimension which reflects the degrees of freedom of the system. A free and unconstrained rigid body has six degrees of freedom. Three to define the position in 3D space and three to define the orientation. The two rigid bodies of the ballbot are, however, linked and thus constrained.

The position of the body is kinematically linked to the center of the ball through a virtual spherical joint, and is thus fully described by the orientation of the body. Therefore, the body has only three degrees of freedom in terms of the orientation.

The rotation and translation of the ball are linked through a kinematic relationship due to the no-slip condition. Rotation of the ball around the z axis is assumed not to occur, and since the ball is assumed to move only in the x-y plane, the 2D position of the center of the ball defines its degrees of freedom.

This gives a total of five degrees of freedom. The chosen generalized coordinates and corresponding derivatives are shown in (3.4).

$$\boldsymbol{\chi} = \begin{bmatrix} {}^{\mathrm{I}}x \\ {}^{\mathrm{I}}y \\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \end{bmatrix} \qquad \dot{\boldsymbol{\chi}} = \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{bmatrix} \tag{3.4}$$

Note that $\boldsymbol{\chi}$ is used to represent the generalized coordinates whereof in most literature $\boldsymbol{q}$ is used. This is chosen to avoid confusion with the quaternion, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, being a part of the generalized coordinates.

Be aware that the three degrees of freedom from the orientation of the body are represented with a quaternion, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$. The generalized coordinates are thereby no longer independent and the set are no longer the minimal set. To deal with a larger set of coordinates than the degrees of freedom in the system, the unit-norm constraint for the quaternion has to be included in the Lagrangian using Lagrange d'Alembert Principle [44] [45]. This is shown in Chapter 5 where the equations of motion are derived.

4 Kinematics

The first step in the process of deriving the non-linear model of the ballbot is to identify the kinematic relationship between motor angular velocities and the system states, as defined by the generalized coordinates in Section 3.3. In the following chapter the kinematics are derived using the principles of moving frames, see Appendix E, and the assumptions defined in Section 3.2. The kinematics are required in the next chapter where the dynamic model is derived, using the kinematic model to relate the energy of the motors and wheels to the system states.

4.1 Ballbot kinematics

The notation used during the derivation of the kinematic model is described in Appendix E. It is important to remind the reader that a velocity is always relative, thus given relatively between two frames. Any velocity is determined by defining a reference frame from where the other frame or point and its velocity is observed. In the following the subscript B←A should be understood as: "when looking on frame {B} from reference frame {A}" [46] [47] [44].

With the convention defined, the ballbot kinematics can be derived. The ballbot kinematics consists of deriving an expression for the individual omniwheel angular velocities based on the system states.

4.1.1 Ball angular velocity

Based on the no-slip condition for the ball together with the assumption that the ball does not rotate around the z-axis, a kinematic relationship is derived between the angular velocity of the ball and the translational velocity of the ball, as part of the systems states. The rotating ball frame, defined as {K'}, allows us to define the angular velocity of the ball as

$$ {}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} = \begin{bmatrix} \omega_{k,x} \\ \omega_{k,y} \\ 0 \end{bmatrix} \tag{4.1} $$

Since the ball frame, {K}, is only translated relative to the inertial frame, the following definitions are all equal

$$ {}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} = {}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{I}} = {}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{I}} \tag{4.2} $$

Note that the position of the center of the ball and the contact point with the ground is coincident in the x-y plane of the inertial frame. Hence the translational velocity of the ball corresponds to the contact point velocity, which in turn is computed as the tangential velocity.

The contact point with the ground is defined within the ball frame.

$$ {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{G}} = \begin{bmatrix} 0 \\ 0 \\ -r_k \end{bmatrix} \tag{4.3} $$

As described in Appendix F.3 the tangential velocity is computed using the cross product with the angular velocity of the ball, ${}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}}$.

$$ {}^{\mathrm{K}}\boldsymbol{v}_{\mathrm{G}\leftarrow\mathrm{K'}} = {}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} \times {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{G}} \tag{4.4} $$

The ball will travel in the opposite direction to the tangential velocity at the contact point.

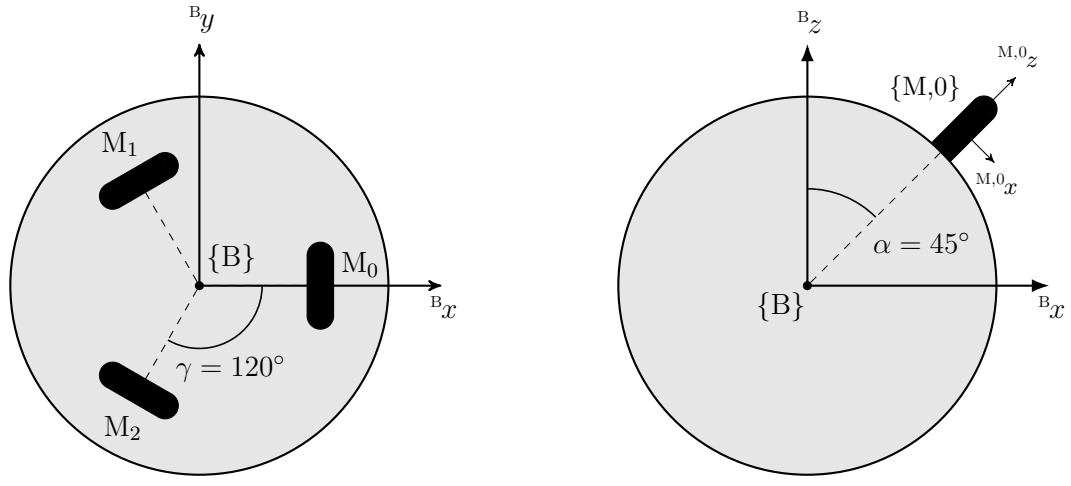
$$ \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{z} \end{bmatrix} = {}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K'}\leftarrow\mathrm{I}} = -\,{}^{\mathrm{K}}\boldsymbol{v}_{\mathrm{G}\leftarrow\mathrm{K}} = r_k \begin{bmatrix} \omega_{k,y} \\ -\omega_{k,x} \\ 0 \end{bmatrix} \tag{4.5} $$

Inverting this relationship yields an equation that defines the angular velocity of the ball with respect to the translational velocity.

$$ {}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} = \frac{1}{r_k} \begin{bmatrix} -\,{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} \tag{4.6} $$

4.1.2 Motor frames

Initially three motor frames, {M,i} for $i = 0, \ldots, 2$, are defined with origin in the center of the omniwheels, rigidly attached to the robot body. The x-axis points in the direction of the shaft, from motor to wheel, and the z-axis points up, see Figure 4.1(b). Motor 0 is aligned such that the shaft points in the direction of the body frame x-axis projection, thus in the forward direction of the robot, see Figure 4.1(a).



(a) Seen from the top

(b) Seen from the side

Figure 4.1: Omniwheel locations in body frame

The contact points of the omniwheels and the motor frames are determined from a sequence of SE(3) transforms.

Starting in the body frame in the center of the ball, the first transform in (4.7) rotates the motor orientation around the z-axis to get the correct direction of each motor based on the wheel separation angle, γ.

$$ {}^{\mathrm{B}}_{\gamma,\mathrm{i}}\boldsymbol{T} = \begin{bmatrix} \mathbf{R}_Z(i\,\gamma) & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \quad \text{for } i = 0,\ldots,2 \tag{4.7} $$

This is followed by a general transform in (4.8) which tilts the motors around the intermediate y-axis by the zenith angle, α.

$$ {}^{\gamma}_{\alpha}\boldsymbol{T} = \begin{bmatrix} \mathbf{R}_Y(\alpha) & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \quad \text{for } i = 0,\ldots,2 \tag{4.8} $$

A final transform in (4.9) moves the frame to the surface of the ball by offsetting it with the radius of the ball in the z-axis direction.

$$ {}^{\alpha}_{\mathrm{C}}\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9} $$

The contact point frames are then computed by combining the transforms:

$$ {}^{\mathrm{B}}_{\mathrm{C,i}}\boldsymbol{T} = {}^{\mathrm{B}}_{\gamma,\mathrm{i}}\boldsymbol{T}\,{}^{\gamma}_{\alpha}\boldsymbol{T}\,{}^{\alpha}_{\mathrm{C}}\boldsymbol{T} \quad \text{for } i = 0,\ldots,2 \tag{4.10} $$

The position of the contact points are given as the origin of the three frames, which is extracted using the homogeneous zero vector, $\underline{\mathbf{0}}$.

$$ {}^{\mathrm{B}}\boldsymbol{p}_{c,i} = {}^{\mathrm{B}}_{\mathrm{C,i}}\boldsymbol{T}\,\underline{\mathbf{0}} \tag{4.11} $$

$$ = r_k \begin{bmatrix} \cos(\gamma\,i)\sin(\alpha) \\ \sin(\gamma\,i)\sin(\alpha) \\ \cos(\alpha) \end{bmatrix} \quad \text{for } i = 0,\ldots,2 \tag{4.12} $$

Furthermore, the tangential direction vectors for positive turning of the omniwheels, points in the direction of the y-axis of the motor frame, see Figure 4.2.

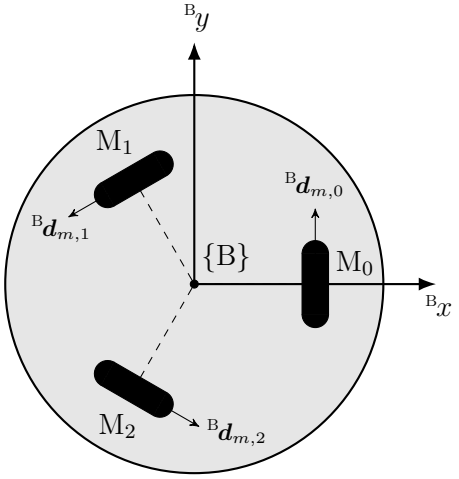


Figure 4.2: Omniwheel tangential directions

These tangential direction vectors are defined in (4.13).

$$
\begin{aligned}
{}^{\mathrm{B}}\boldsymbol{d}_{m,i} &= \begin{bmatrix} \mathbf{I}_3 & 0 \end{bmatrix} {}^{\mathrm{B}}_{\mathrm{C,i}}\boldsymbol{T} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -\sin(\gamma\, i) \\ \cos(\gamma\, i) \\ 0 \end{bmatrix} \quad \text{for } i = 0, \ldots, 2
\end{aligned}
\tag{4.13}
$$

The motor frames in the center of the omniwheels are defined by offsetting the contact point frames with the radius of the omniwheel in the z-axis direction.

$$
{}^{\mathrm{C,i}}_{\mathrm{M,i}}\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_w \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{for } i = 0, \ldots, 2 \tag{4.14}
$$

$$
{}^{\mathrm{B}}_{\mathrm{M,i}}\boldsymbol{T} = {}^{\mathrm{B}}_{\mathrm{C,i}}\boldsymbol{T}\, {}^{\mathrm{C,i}}_{\mathrm{M,i}}\boldsymbol{T} \quad \text{for } i = 0, \ldots, 2 \tag{4.15}
$$

4.1.3 Omniwheel tangential velocity

The angular velocity of each omniwheel with respect to the motor, and thus the angular velocity that can be measured by the encoders, is relative to the now defined motor frames, {M,i}. Each wheel has a frame with an origin coinciding with the motor frame and with the ability to rotate around the x-axis (shaft direction).

$$
{}^{\mathrm{M,i}}_{\mathrm{W,i}}\boldsymbol{T} = \begin{bmatrix} \mathbf{R}_X(\theta_i) & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \quad \text{for } i = 0, \ldots, 2 \tag{4.16}
$$

where the current wheel angles are denoted θ_i. The angular velocity of the wheel is thus defined as ${}^{\mathrm{M}}\boldsymbol{\omega}^{i}_{\mathrm{W}\leftarrow\mathrm{M}}$ with a shaft velocity of $\dot{\theta}_i$, where i defines the motor index from $0, \ldots, 2$.

$$
{}^{\mathrm{M}}\boldsymbol{\omega}^{i}_{\mathrm{W}\leftarrow\mathrm{M}} = {}^{\mathrm{M,i}}\boldsymbol{\omega}_{\mathrm{W,i}\leftarrow\mathrm{M,i}} = \begin{bmatrix} \dot{\theta}_i \\ 0 \\ 0 \end{bmatrix} \tag{4.17}
$$

Note how this angular velocity of the motor shaft is a relative angular velocity of the wheel with respect to the body frame to where the motor is rigidly attached.

The tangential velocity of the omniwheel at the contact point, due to the angular velocity of the wheel, is computed using the cross product, see Appendix F.3.

$$
{}^{\mathrm{C}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{W}} = {}^{\mathrm{M}}\boldsymbol{\omega}^{i}_{\mathrm{W}\leftarrow\mathrm{M}} \times {}^{\mathrm{M}}\boldsymbol{O}_{\mathrm{C}} = {}^{\mathrm{M}}\boldsymbol{\omega}^{i}_{\mathrm{W}\leftarrow\mathrm{M}} \times \begin{bmatrix} 0 \\ 0 \\ -r_w \end{bmatrix} \tag{4.18}
$$

Due to the choice of frames, (4.18) simplifies to

$$ {}^{\mathrm{C}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{W}} = \begin{bmatrix} 0 \\ r_w\,\dot{\theta}_i \\ 0 \end{bmatrix} \tag{4.19} $$

where the contact point frame {C,i} has the same orientation as the motor frame, {M,i}. The tangential velocity is thus in the y-axis direction of the contact point frame. To convert the tangential velocity to the body frame, the vector is rotated. The result corresponds to multiplying the y-axis value with the tangential direction vector from (4.13).

$$ {}^{\mathrm{B}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{W}} = {}^{\mathrm{B}}_{\mathrm{C,i}}\boldsymbol{R}\ {}^{\mathrm{C}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{W}} = r_w\,\dot{\theta}_i\,{}^{\mathrm{B}}\boldsymbol{d}_{m,i} \tag{4.20} $$

4.1.4 Ball tangential velocity

The tangential velocity of the ball at the contact point is computed similarly using the cross product.

$$ {}^{\mathrm{B}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{K}} = {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{B}} \times {}^{\mathrm{B}}\boldsymbol{p}_{c,i} \tag{4.21} $$

Note how this requires the angular velocity of the ball with respect to the body frame, ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{B}}$, and not with respect to the inertial frame, ${}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}}$, as presented in Section 4.1.1. The angular velocity is decomposed according to Appendix E.4.

$$ \begin{aligned} {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{B}} &= {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} + {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K}\leftarrow\mathrm{B}} \\ &= {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} - {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}} \end{aligned} \tag{4.22} $$

where ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}$ is the angular velocity of the body seen from the ball frame but defined in the body frame. Applying the relationship between the quaternion derivative and the body angular velocity from (H.51):

$$ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{4.23} $$

Furthermore, to compute ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}}$ the ball frame angular velocity ${}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}}$ is rotated with the quaternion using (H.40).

$$ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{K}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{4.24} $$

The converted angular velocities are inserted into (4.22).

$$ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{B}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} - 2\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{4.25} $$

The computed angular velocity of the ball frame with respect to the body frame is inserted into (4.21) and the cross product is replaced by quaternion multiplication as described in Appendix H.7.

$$ \begin{aligned} {}^{\mathrm{B}}\tilde{\boldsymbol{v}}^{i}_{\mathrm{C}\leftarrow\mathrm{K}} &= {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{B}} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i} \\ &= \boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i}\Big)\,\boldsymbol{\Phi}\Big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\Big)^{\mathsf{T}}\,\Big(\boldsymbol{\Gamma}\Big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\Big)\ {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{K}} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\Big) \end{aligned} \tag{4.26} $$

4.1.5 Kinematic relationship

With the no slip condition the omniwheels are modelled such that the instantaneous velocity at the contact points of the omniwheels are always zero in the direction tangential to the omniwheel perimeter. Thus the tangential velocity from the omniwheel and the tangential velocity from the ball is assumed to be equal in the direction tangential to the omniwheel. This tangential velocity component is extracted by taking the dot product with the tangential direction vectors, ${}^{\mathrm{B}}\boldsymbol{d}_{m,i}$.

$$ {}^{\mathrm{B}}\boldsymbol{d}_{m,i} \cdot {}^{\mathrm{B}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{W}} = {}^{\mathrm{B}}\boldsymbol{d}_{m,i} \cdot {}^{\mathrm{B}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{K}} \tag{4.27} $$

Using the properties from Appendix F.1 and by inserting (4.20), the equation is reordered to isolate the angular velocity of the motor shaft.

$$\begin{aligned} r_w\,\dot{\theta}_i\ {}^{\mathrm{B}}\boldsymbol{d}_{m,i} \cdot {}^{\mathrm{B}}\boldsymbol{d}_{m,i} &= {}^{\mathrm{B}}\boldsymbol{d}_{m,i} \cdot {}^{\mathrm{B}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{K}} \\ \dot{\theta}_i &= \frac{1}{r_w}\,{}^{\mathrm{B}}\boldsymbol{d}^{\mathsf{T}}_{m,i}\,{}^{\mathrm{B}}\boldsymbol{v}^{i}_{\mathrm{C}\leftarrow\mathrm{K}} \end{aligned} \tag{4.28}$$

As the final step the tangential velocity of the ball at the contact point, (4.26), is inserted leading to the equation for the angular velocity of the motor shafts.

$$\begin{aligned} \dot{\theta}_i &= \frac{1}{r_w}\,{}^{\mathrm{B}}\tilde{\boldsymbol{d}}^{\mathsf{T}}_{m,i}\,{}^{\mathrm{B}}\tilde{\boldsymbol{v}}^{i}_{\mathrm{C}\leftarrow\mathrm{K}} \\ &= \frac{1}{r_w}\,{}^{\mathrm{B}}\tilde{\boldsymbol{d}}^{\mathsf{T}}_{m,i}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i}\right)\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\,{}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{K}} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \end{aligned} \tag{4.29}$$

4.2 Inverse kinematics

Inverse kinematics of a robot transforms the robot state space into actuation space. With regard to the ballbot, the actuation space is the angular velocities of each wheel, rather than the wheel angle itself, due to a non-holonomic mapping between wheel angles and system states. The state space is defined by the system states in Section 3.3 including the position and velocity of the ball and orientation and angular velocity of the body.

The kinematic relationship from (4.29) is simplified, the angular velocity of the ball is substituted with the translational velocity using (4.6) and the equation is partitioned with common terms put towards the right.

$$\dot{\theta}_i = \underbrace{\frac{1}{r_w}\,{}^{\mathrm{B}}\tilde{\boldsymbol{d}}^{\mathsf{T}}_{m,i}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i}\right)}_{\widetilde{\boldsymbol{W}}_i}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \tag{4.30}$$

Note that the part after $\widetilde{\boldsymbol{W}}_i$ does not depend on the index and conversely that the content of $\widetilde{\boldsymbol{W}}_i$ does not depend on the states.

This indicates that $\widetilde{\boldsymbol{W}}_i$ defines the static kinematic mapping.

$$\begin{aligned} \widetilde{\boldsymbol{W}}_i &= \frac{1}{r_w}\,{}^{\mathrm{B}}\tilde{\boldsymbol{d}}^{\mathsf{T}}_{m,i}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i}\right) \\ &= \frac{1}{r_w}\left(\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i}\right)^{\mathsf{T}}\,{}^{\mathrm{B}}\tilde{\boldsymbol{d}}_{m,i}\right)^{\mathsf{T}} \\ &= \frac{1}{r_w}\left({}^{\mathrm{B}}\tilde{\boldsymbol{d}}_{m,i} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}^{*}_{c,i}\right)^{\mathsf{T}} \\ &= -\frac{1}{r_w}\left({}^{\mathrm{B}}\tilde{\boldsymbol{d}}_{m,i} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{c,i}\right)^{\mathsf{T}} \end{aligned} \tag{4.31}$$

Based on (H.10) the mapping can be simplified to

$$\widetilde{\boldsymbol{W}}_i = -\frac{1}{r_w}\left[-{}^{\mathrm{B}}\boldsymbol{d}_{m,i} \cdot {}^{\mathrm{B}}\boldsymbol{p}_{c,i}\,,\quad {}^{\mathrm{B}}\boldsymbol{d}_{m,i} \times {}^{\mathrm{B}}\boldsymbol{p}_{c,i}\right] \tag{4.32}$$

We want to derive a joint mapping whose output is a vector of wheel angular velocities:

$$\dot{\boldsymbol{\theta}} = \begin{bmatrix} \dot{\theta}_0 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{4.33}$$

Therefore, the $\widetilde{\boldsymbol{W}}_i$ elements are stacked into a joint matrix $\widetilde{\boldsymbol{W}}$.

$$\widetilde{\boldsymbol{W}} = \begin{bmatrix} \widetilde{\boldsymbol{W}}_0 \\ \widetilde{\boldsymbol{W}}_1 \\ \widetilde{\boldsymbol{W}}_2 \end{bmatrix} = \frac{r_k}{r_w} \begin{bmatrix} 0 & -\sin(\alpha) & 0 & \cos(\alpha) \\ 0 & -\cos(\gamma)\sin(\alpha) & -\sin(\gamma)\sin(\alpha) & \cos(\alpha) \\ 0 & -\cos(2\gamma)\sin(\alpha) & -\sin(2\gamma)\sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{4.34}$$

Such that the joint wheel angular velocity vector, $\dot{\boldsymbol{\theta}}$, can be computed with

$$\dot{\boldsymbol{\theta}} = \widetilde{\boldsymbol{W}}\, \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \underbrace{\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right)}_{{}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{B}}} \tag{4.35}$$

As an example, when the ballbot is balancing upright and with zero heading, hence with ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$ as a unit quaternion, and is stable with little angular velocity, (4.35) simplifies to

$$\dot{\boldsymbol{\theta}} \approx \widetilde{\boldsymbol{W}} \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} \tag{4.36}$$

4.3 Forward kinematics

As opposed to the inverse kinematics, the forward kinematics transforms the actuation space into the state space. In this case with the ballbot the kinematic actuation space is the angular velocities of the omniwheels due to the non-holonomic mapping. Since the mapping is non-holonomic it is not possible to recover the position and orientation states. The forward kinematics model is thus a mapping from wheel angular velocities, $\dot{\boldsymbol{\theta}}$, body orientation, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, and angular velocity, ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$ to translational velocity, ${}^{\mathrm{I}}\dot{x}$ and ${}^{\mathrm{I}}\dot{y}$.

The forward kinematics is derived by inverting (4.35). This equation can not be inverted directly since the mapping, $\widetilde{\boldsymbol{W}}$ is non-square and thus non-invertible. Instead a reduced version is constructed by removing the first column with zeros and using the angular velocity of the ball defined within the body frame.

$$\dot{\boldsymbol{\theta}} = \boldsymbol{W}\, {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{B}} \tag{4.37}$$

where

$$\boldsymbol{W} = \widetilde{\boldsymbol{W}} \begin{bmatrix} \mathbf{0}_{1\times 3} \\ \mathbf{I}_3 \end{bmatrix} \tag{4.38}$$

such that

$$\widetilde{\boldsymbol{W}} = \boldsymbol{W}\vee \tag{4.39}$$

The new mapping matrix, $\boldsymbol{W}$, is invertible and the ball angular velocity, defined within the body frame, can thus be retrieved from the wheel angular velocities.

$$ {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{B}} = \boldsymbol{W}^{-1}\dot{\boldsymbol{\theta}} \tag{4.40} $$

To extract the translational velocity the ball angular velocity is converted according to (4.25) substituted with (4.6).

$$ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{B}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{y} \\ 0 \end{bmatrix} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} - 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{4.41} $$

Finally, this equation is reordered to yield the translational velocity.

$$ \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} = r_k\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ \left({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{B}} + 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ \right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \tag{4.42} $$

On which the matrix operators for the quaternion multiplication is applied and the inverse mapping is applied.

$$ \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \end{bmatrix} = r_k \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right) \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} \left(\begin{bmatrix} 0 \\ \boldsymbol{W}^{-1}\dot{\boldsymbol{\theta}} \end{bmatrix} + 2\, \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) \tag{4.43} $$

This concludes the derivation of both the forward and inverse kinematics, enabling us to continue with the derivation of the dynamic model.

5 Dynamic Model

Since the ballbot involves both rotating and translating frames it is decided to use Lagrangian mechanics to model the dynamics. Modelling a system with Lagrangian mechanics involves eight steps [44] [48] [45] [49] [50] [51].

1. Define generalized coordinates (preferably independent).
2. Derive kinetic, T, and potential, V, energy functions for all elements (e.g., rigid bodies) in the system.
3. Compute Lagrangian as the differential of all kinetic and potential energy, $\mathcal{L} = T - V$
4. Construct equations of motion using the Euler-Lagrange equation.
5. Apply input forces (and torques) as generalized conservative forces.
6. Add friction components as non-conservative forces. In general they will be added similarly to generalized forces and torques.
7. Include constraints using Lagrange multipliers, e.g., to reduce the space of the generalized coordinates if not independent.
8. Derive the final non-linear ODEs by solving the Lagrange multipliers and reordering the equations of motion.

For mechanical systems, the kinetic energy generally consists of translational and rotational motion, in terms of velocities, see (5.1), and the potential energy mostly consists of gravitational and elastic energy, see (5.2).

$$T_{\text{trans}} = \frac{1}{2}mv^2 \qquad T_{\text{rot}} = \frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{J}\boldsymbol{\omega} \tag{5.1}$$

$$V_{\text{grav}} = mgh \qquad V_{\text{elas}} = \frac{1}{2}kx^2 \tag{5.2}$$

As a remark to the derivation of the potential energy functions, the potential energy with respect to $\dot{\boldsymbol{\chi}}$, should always be zero for mechanical systems.

$$\frac{\partial V}{\partial \dot{\boldsymbol{\chi}}} = 0 \tag{5.3}$$

Given the Lagrangian, the Euler-Lagrange equation defines the equations of motion for the system.

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} + \boldsymbol{D} - \boldsymbol{H}_{\text{hol}}^{\mathsf{T}}\,\boldsymbol{\lambda}_{\text{hol}} + \boldsymbol{H}_{\text{non}}^{\mathsf{T}}\,\boldsymbol{\lambda}_{\text{non}} = \boldsymbol{Q} \tag{5.4}$$

where input forces are included in $\boldsymbol{Q}$, friction forces are included in $\boldsymbol{D}$ and constraints are coupled through the constraint Jacobians, $\boldsymbol{H}_{\text{hol}}$ and $\boldsymbol{H}_{\text{non}}$ with their corresponding Lagrange multipliers, $\boldsymbol{\lambda}_{\text{hol}}$ and $\boldsymbol{\lambda}_{\text{non}}$.

Note that $\boldsymbol{\chi}$ is used to represent the generalized coordinates whereof in most literature $\boldsymbol{q}$ is used. This is chosen to avoid confusion with the quaternion, ${}^{\text{K}}_{\text{B}}\boldsymbol{q}$, being a part of the generalized coordinates.

Input forces has to be defined as generalized forces and thus forces affecting the generalized coordinates. From the principle of an infinitesimal displacement, the following equation defines the relationship between input forces and generalized forces, using the mapping between the input coordinates, affected by the input force, r_i, and the generalized coordinates, χ_j.

$$Q_j = \sum_i F_{\text{ext},i} \frac{\partial r_i}{\partial \chi_j} \tag{5.5}$$

This applies similarly to external torques, e.g., from the motors, where the corresponding generalized torque is included into the Lagrangian by applying the transpose of the geometric Jacobian, which maps the derivatives of the generalized coordinates to the angular velocities of the motors.

If the generalized coordinates are defined as the smallest independent set capable of describing the degrees of freedom of the system, constraints would not be necessary. However, if the generalized coordinates are not independent, each particle in the system will be constrained to a certain trajectory in the generalized coordinate space. In this case the system is said to have constrained generalized coordinates.
In Newtonian mechanics virtual constraint forces are used to handle this, but these are cumbersome to remember, find and define. Examples of constraint forces include the tension of ropes and normal forces applied by surfaces. In the Lagrangian formulation there is no need to worry about defining constraint forces since constraints can be embedded directly into the Lagrangian.

Two type of constraints exist: holonomic constraints, also known as integrable constraints, and non-holonomic constraints. Holonomic constraints can be expressed algebraically and do not constrain the dynamics.

$$h_{\text{hol},j}(\boldsymbol{\chi}) = 0 \tag{5.6}$$

Note that the constraint function is denoted $\boldsymbol{h}(\boldsymbol{\chi})$ whereof in most literature the constraint function is denoted $\boldsymbol{g}(\boldsymbol{\chi})$. This is chosen to avoid confusion between the constraint Jacobian, denoted with a capital letter, and the gravity matrix, $\boldsymbol{G}(\boldsymbol{\chi})$.

When a constraint is holonomic it can be handled in two ways. Either by applying the algebraic relationship of the constraint to the generalized coordinates, thus to generate a reduced set of independent generalized coordinates. Or by embedding the constraint into the Lagrangian, similarly to how non-holonomic constraints are handled. The latter does not modify the generalized coordinates but keeps them as constrained. This requires the Lagrangian to be augmented with Lagrangian multipliers and the Jacobian of the holonomic constraints.

$$\boldsymbol{H}_{\text{hol}}(\boldsymbol{\chi}) = \frac{\partial \boldsymbol{h}_{\text{hol}}}{\partial \boldsymbol{\chi}} \tag{5.7}$$

Non-holonomic constraints cannot be written in a closed-form (algebraic equation) and are instead expressed by including derivatives of the coordinates. Non-holonomic constraints are thus non-integrable and restricts the velocities of the system.

$$h_{\text{non},j}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}, t) = 0 \tag{5.8}$$

If the constraints are non-holonomic the set of generalized coordinates can not be reduced by substitution and the constraints have to be handled by means of Lagrange multipliers [52]. For non-holonomic constraints the Jacobian of the constraints has to be derived with respect to the velocities.

$$\boldsymbol{H}_{\text{non}}(\boldsymbol{\chi}) = \frac{\partial \boldsymbol{h}_{\text{non}}}{\partial \dot{\boldsymbol{\chi}}} \tag{5.9}$$

By reorganizing the Euler-Lagrange equation from (5.4), the resulting equations of motion appears on the form

$$\boldsymbol{M}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \boldsymbol{G}(\boldsymbol{\chi}) + \boldsymbol{D}(\dot{\boldsymbol{\chi}}) - \boldsymbol{H}_{\text{hol}}^{\mathsf{T}}(\boldsymbol{\chi})\,\boldsymbol{\lambda}_{\text{hol}} + \boldsymbol{H}_{\text{non}}^{\mathsf{T}}(\boldsymbol{\chi})\,\boldsymbol{\lambda}_{\text{non}} = \boldsymbol{Q}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}},\boldsymbol{u}) \quad (5.10)$$

where

- $\boldsymbol{M}(\boldsymbol{\chi})$ is the mass/inertia matrix
- $\boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})$ is the Coriolis and centrifugal force matrix
- $\boldsymbol{G}(\boldsymbol{\chi})$ is the gravitational force matrix
- $\boldsymbol{D}(\dot{\boldsymbol{\chi}})$ is the friction/dampening force matrix
- $\boldsymbol{H}_{\text{hol}}(\boldsymbol{\chi})$ is the constraint Jacobian of the holonomic constraints
- $\boldsymbol{H}_{\text{non}}(\boldsymbol{\chi})$ is the constraint Jacobian of the non-holonomic constraints
- $\boldsymbol{\lambda}_{\text{hol}}$ are the Lagrange multipliers used to couple the holonomic constraints to the equations of motion
- $\boldsymbol{\lambda}_{\text{non}}$ are the Lagrange multipliers used to couple the non-holonomic constraints to the equations of motion
- $\boldsymbol{Q}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}},\boldsymbol{u})$ is the external/input force matrix

When the Lagrange multipliers are eliminated from the equations of motion, the number of differential equations will be reduced to reflect the degrees of freedom of the system. The reduced set of equations of motion are on the form

$$\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) + \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}}) = \widetilde{\boldsymbol{Q}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}},\boldsymbol{u}) \quad (5.11)$$

where $\widetilde{\ }$ denotes reduced-set matrices.

These equations of motion, given by a set of ordinary differential equations (ODE), can be reordered to construct the non-linear ODE of the system. Let the joint state vector be defined as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\chi} \\ \dot{\boldsymbol{\chi}} \end{bmatrix} \quad (5.12)$$

The complete non-linear ODE governing the joint state vector is then defined as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{\chi}} \\ \widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\left(\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}},\boldsymbol{u}) - \widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} - \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) - \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}})\right) \end{bmatrix} \quad (5.13)$$

5.1 Generalized coordinates

The generalized coordinates were chosen in Section 3.3 as the position of the center of the ball and the orientation of the body in terms of the quaternion, both defined in the inertial frame.

$$\boldsymbol{\chi} = \begin{bmatrix} {}^{\mathrm{I}}x \\ {}^{\mathrm{I}}y \\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \end{bmatrix} \qquad \dot{\boldsymbol{\chi}} = \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{bmatrix} \tag{5.14}$$

The chosen set of generalized coordinates is not independent, due to the holonomic unit norm constraint on the quaternion. However, it is chosen to keep this set of generalized coordinates rather than reducing the dimension. The holonomic constraint should therefore be embedded into the Lagrangian, which is described in Section 5.6 after the energy equations have been derived.

5.2 Energy equations

The ballbot is simplified as two rigid bodies, a ball and a body, so the energy equations for these two bodies have to be derived. Furthermore, since the motors, gears and omniwheels rotates as well, the kinetic energy in these parts has to be included. The Lagrangian assumes kinetic and potential energy to be defined in the inertial frame. In the following section the energy of the ball, body and wheels are thus derived in the inertial frame.

5.2.1 Ball energy

The kinetic energy of the ball, T_k, is a combination of rotational and translational energy, due to the no-slip assumption and the kinematic relationship between rotation and translation, see Section 4.1.1.

$$T_k = \frac{1}{2}M_k\,{}^{\mathrm{I}}\dot{x}^2 + \frac{1}{2}M_k\,{}^{\mathrm{I}}\dot{y}^2 + \frac{1}{2}\,{}^{\mathrm{K}}\boldsymbol{\omega}^{\mathsf{T}}_{\mathrm{K'}\leftarrow\mathrm{K}}\boldsymbol{J}_k\,{}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{K'}\leftarrow\mathrm{K}} \tag{5.15}$$

where the mass of the ball is assumed to be uniformly distributed in a perfect sphere, which is why the inertia tensor becomes a diagonal matrix.

$$\boldsymbol{J}_k = \begin{bmatrix} J_k & 0 & 0 \\ 0 & J_k & 0 \\ 0 & 0 & J_k \end{bmatrix} \tag{5.16}$$

The kinetic energy equation is simplified by substution with (4.6).

$$T_k = \frac{1}{2}M_k\,{}^{\mathrm{I}}\dot{x}^2 + \frac{1}{2}M_k\,{}^{\mathrm{I}}\dot{y}^2 + \frac{1}{2}\frac{1}{r_k}\begin{bmatrix} -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix}^{\mathsf{T}}\boldsymbol{J}_k\frac{1}{r_k}\begin{bmatrix} -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} \tag{5.17}$$

By reorganizing the final equation for the kinetic energy of the ball is shown in (5.18).

$$T_k = \frac{1}{2}\left(M_k + \frac{J_k}{r_k^2}\right){}^{\mathrm{I}}\dot{x}^2 + \frac{1}{2}\left(M_k + \frac{J_k}{r_k^2}\right){}^{\mathrm{I}}\dot{y}^2 \tag{5.18}$$

The potential energy of the ball, V_k, is zero since the ball is assumed to move in just the x-y plane.

$$V_k = 0 \tag{5.19}$$

5.2.2 Body energy

The body contains both kinetic and potential energy. The kinetic energy, T_b, comes from both translational and rotational movement of the rigid body.

$$T_b = \frac{1}{2} {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{I}}^{\mathsf{T}} {}^{\mathrm{B}}\boldsymbol{J}_b {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{I}} + \frac{1}{2} M_b {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}}^{\mathsf{T}} {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}} \tag{5.20}$$

where the inertia tensor in body frame is modelled as a diagonal matrix.

$${}^{\mathrm{B}}\boldsymbol{J}_b = \begin{bmatrix} J_{bx} & 0 & 0 \\ 0 & J_{by} & 0 \\ 0 & 0 & J_{bz} \end{bmatrix} \tag{5.21}$$

Note that the point of the center of mass coincides with the origin of the body prime frame, ${}^{\mathrm{I}}\boldsymbol{p}_{\mathrm{COM}} = {}^{\mathrm{I}}\boldsymbol{O}_{\mathrm{B'}}$. In the following, this point will be used instead of the frame origin. Note also the relaxed notation of the velocity, ${}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}} = {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{I}} = {}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{B'}\leftarrow\mathrm{I}}$.

The angular velocity of the body seen from the ball frame and defined in the body frame, ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}$, defined in (H.51), is equivalent to the angular velocity of the body seen from the inertial frame, ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{I}}$, since the ball frame does not rotate relative to the inertial frame.

$${}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}} = {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{5.22}$$

The inertia tensor matrix is augmented with an extra row and column, such that the vectorized angular velocity, ${}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}$, having zero scalar part, can be used.

$${}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b = \wedge {}^{\mathrm{B}}\boldsymbol{J}_b \vee \triangleq \begin{bmatrix} 1 & 0 \\ 0 & {}^{\mathrm{B}}\boldsymbol{J}_b \end{bmatrix} \tag{5.23}$$

The added value of 1 has no influence on the energy, since the scalar part of the angular velocity vector is zero, but the value makes the augmented inertia tensor invertible. The updated kinematic energy equation thus become

$$T_b = 2 \left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right)^{\mathsf{T}} {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) + \frac{1}{2} M_b {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{I}}^{\mathsf{T}} {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{I}} \tag{5.24}$$

The velocity of the center of mass point can be derived in two different ways, according to (H.53) and (E.11), both being equivalent.
The position of the center of mass within the ball frame is computed with the quaternion using (H.39).

$${}^{\mathrm{K}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \tag{5.25}$$

where ${}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}}$ is assumed constant as defined in Section 2.2.

Taking the derivative of this rotated point with the quaternion definition, (H.53), yields

$${}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}} = {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* + {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^* \tag{5.26}$$

which can be simplified using the two operators, $\boldsymbol{\Pi}_{\mathrm{COM}}(\boldsymbol{q})$ and $\boldsymbol{\Lambda}_{\mathrm{COM}}(\boldsymbol{q})$, presented in Appendix H.4.

$$\boldsymbol{\Pi}_{\mathrm{COM}}(\boldsymbol{q}) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \right) \mathbf{I}^* \tag{5.27}$$

$$\boldsymbol{\Lambda}_{\mathrm{COM}}(\boldsymbol{q}) = \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \right) \tag{5.28}$$

The velocity of the rotated center of mass thereby simplifies to

$$\begin{aligned} {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}} &= 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\ &= 2\boldsymbol{\Lambda}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2\boldsymbol{\Lambda}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{aligned} \tag{5.29}$$

The velocity in (5.29) is, however, computed with respect to the ball frame, and the velocity needed for the energy equation is with respect to the inertial frame. Since the ball frame do not rotate relative to the inertial frame, the velocity of the center of mass, defined in (4.5), will according to Appendix E be the sum of the velocity described in the ball frame and the velocity of the ball frame with respect to the inertial frame.

$${}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{I}} = {}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{K}} = {}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{K}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{K}} \tag{5.30}$$

Inserting the two velocity definitions, the matrix operators for quaternion multiplications and using the algebraic properties of the dot product, see Appendix F.1, results in a big expansion of the energy equation.

$$\begin{aligned} T_b &= 2\left(\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}}{}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\left(\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) + \frac{1}{2}M_b\left({}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{K}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{K}}\right)^{\mathsf{T}}\left({}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{K}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{K}}\right) \\ &= 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\,\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} + \frac{1}{2}M_b\left({}^{\mathrm{I}}\boldsymbol{v}^{\mathsf{T}}_{\mathrm{K}\leftarrow\mathrm{I}}\,{}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}^{\mathsf{T}}_{\mathrm{COM}\leftarrow\mathrm{K}}\,{}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}} + 2\,{}^{\mathrm{I}}\tilde{\boldsymbol{v}}^{\mathsf{T}}_{\mathrm{K}\leftarrow\mathrm{I}}\,{}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}\right) \end{aligned} \tag{5.31}$$

Note how the velocity of the center of mass has been replaced by its vectorized version, ${}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}$, since the scalar part of zero will leave the result unaffected, but it will simplify the equations. The translational velocity, ${}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}}$, and the simplified velocity of the center of mass, ${}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}$, are substituted into the kinetic energy definition.

$$\begin{aligned} T_b = {} & 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} + \frac{1}{2}M_b\begin{bmatrix}{}^{\mathrm{I}}\dot{x}\\ {}^{\mathrm{I}}\dot{y}\\ 0\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}{}^{\mathrm{I}}\dot{x}\\ {}^{\mathrm{I}}\dot{y}\\ 0\end{bmatrix} \\ & + \frac{1}{2}M_b\left(2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\ & + \frac{1}{2}M_b 2\begin{bmatrix}0\\ {}^{\mathrm{I}}\dot{x}\\ {}^{\mathrm{I}}\dot{y}\\ 0\end{bmatrix}^{\mathsf{T}}2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \end{aligned} \tag{5.32}$$

By reorganizing and applying the properties of the rotated vector derivative operators, as described in Appendix H.4, a simplified expression for the kinetic body energy appears.

$$\begin{aligned} T_b = {} & 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} + \frac{1}{2}M_b\,{}^{\mathrm{I}}\dot{x}^2 + \frac{1}{2}M_b\,{}^{\mathrm{I}}\dot{y}^2 \\ & + 2M_b\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\,{}^{\mathrm{B}}\boldsymbol{p}^{\mathsf{T}}_{\mathrm{COM}}\,{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} + 2M_b\begin{bmatrix}{}^{\mathrm{I}}\dot{x}\\ {}^{\mathrm{I}}\dot{y}\\ 0\end{bmatrix}^{\mathsf{T}}\vee\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \end{aligned} \tag{5.33}$$

Note that ${}^{\mathrm{B}}\boldsymbol{p}^{\mathsf{T}}_{\mathrm{COM}}\,{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} = \left\|{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}\right\|^2$.

The potential energy, V_b, comes from the center of mass, where the distance from the xy-plane to the COM varies with the inclination of the ballbot.

$$V_b = M_b g \, \vec{\mathbf{k}}^{\mathsf{T}} \, {}^{\mathrm{I}}\boldsymbol{p}_{\mathrm{COM}} = M_b g \, \hat{\mathbf{k}}^{\mathsf{T}} \, {}^{\mathrm{I}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \tag{5.34}$$

where the position of the center of mass in the inertial frame is defined as

$${}^{\mathrm{I}}\boldsymbol{p}_{\mathrm{COM}} = {}^{\mathrm{I}}\boldsymbol{O}_{\mathrm{K}} + {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{COM}} = \begin{bmatrix} {}^{\mathrm{I}}x \\ {}^{\mathrm{I}}y \\ 0 \end{bmatrix} + {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{COM}} \tag{5.35}$$

The z-component of the center of mass position is extracted using the dot product with the z-axis unit vector, $\hat{\mathbf{k}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}}$. Since the ball frame is only translated in the xy-plane, the z-axis value of the center of mass position in the inertial frame, is the same as the z-axis value of the position defined in ball frame, (5.25). The position definition from (5.25) is thus inserted into (5.34).

$$V_b = M_b g \, \hat{\mathbf{k}}^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \, {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \tag{5.36}$$

5.2.3 Wheel energy

In the body energy, both the rotational and translational energy from moving the mass and inertia of the motors, gears and omniwheels are included as part of the body mass and body moment of inertia. However, since the wheels can rotate with a different angular velocity than the body, not all energy is captured by the body. The difference in angular velocity, thus the angular velocity of the motor shaft, $\dot{\boldsymbol{\theta}}$, see Section 4.2, which corresponds to the angular velocity of the wheels with respect to the body frame, contributes to some extra kinetic energy.

As listed in Section 2.2 the inertia of the motor, gear and omniwheel have been lumped into a single total inertia for the wheel, J_w. This inertia is used to compute the kinetic energy of a single wheel, T_w, due to its rotation.

$$T_w = \frac{1}{2} \dot{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{J}_w \dot{\boldsymbol{\theta}} \tag{5.37}$$

A joint diagonal inertia tensor is constructed to allow computation of the wheel energies as a single matrix-vector product.

$$\boldsymbol{J}_w = \begin{bmatrix} J_w & 0 & 0 \\ 0 & J_w & 0 \\ 0 & 0 & J_w \end{bmatrix} \tag{5.38}$$

The angular velocity of the wheel shafts were derived as the inverse kinematics, (4.35).

$$\dot{\boldsymbol{\theta}} = \widetilde{\boldsymbol{W}} \, \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \underbrace{\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2 \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right)}_{{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{K'}\leftarrow\mathrm{B}}} \tag{5.39}$$

This angular velocity definition is substituted into the energy definition, (5.37).

$$T_w = \frac{1}{2}\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\begin{bmatrix}0\\-{}^{\mathrm{I}}\dot{y}\\{}^{\mathrm{I}}\dot{x}\\0\end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\begin{bmatrix}0\\-{}^{\mathrm{I}}\dot{y}\\{}^{\mathrm{I}}\dot{x}\\0\end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \tag{5.40}$$

The above equation is expanded further to yield the complete equation for the wheel kinetic energy.

$$\begin{aligned} T_w =& \frac{1}{2}\frac{1}{r_k^2}\begin{bmatrix}0\\-{}^{\mathrm{I}}\dot{y}\\{}^{\mathrm{I}}\dot{x}\\0\end{bmatrix}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\begin{bmatrix}0\\-{}^{\mathrm{I}}\dot{y}\\{}^{\mathrm{I}}\dot{x}\\0\end{bmatrix} + \\ & 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} - \\ & 2\frac{1}{r_k}\begin{bmatrix}0\\-{}^{\mathrm{I}}\dot{y}\\{}^{\mathrm{I}}\dot{x}\\0\end{bmatrix}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{aligned} \tag{5.41}$$

The wheels do not contribute with extra potential energy since the mass of the motors, gears and omniwheels are included in the center of mass of the body.

$$V_w = 0 \tag{5.42}$$

5.3 Euler-Lagrange equation

With the kinetic and potential energies defined, the Lagrangian of the ballbot is computed by combining all the energy terms.

$$\begin{aligned} \mathcal{L} &= T_k + T_b + T_w - V_k - V_b - V_w \\ &= T_k + T_b + T_w - V_b \end{aligned} \tag{5.43}$$

With the Lagrangian, the first part of the Euler-Lagrange equation is computed with (5.44).

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial\mathcal{L}}{\partial\dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} - \left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\chi}}\right)^{\mathsf{T}} = \mathbf{0}_{6\times1} \tag{5.44}$$

Inserting the Lagrangian from (5.43) and noting that the potential energy do not depend on derivative states, yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial\dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial\dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial\dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} - \left(\frac{\partial T_b}{\partial\boldsymbol{\chi}}\right)^{\mathsf{T}} - \left(\frac{\partial T_w}{\partial\boldsymbol{\chi}}\right)^{\mathsf{T}} + \left(\frac{\partial V_b}{\partial\boldsymbol{\chi}}\right)^{\mathsf{T}} = \mathbf{0}_{6\times1} \tag{5.45}$$

The Euler-Lagrange equation results in six differential equations constituting the internal dynamics of the frictionless and unconstrained system without inputs. The differential equations in (5.45) are commonly reorganized into the following form:

$$\boldsymbol{M}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \boldsymbol{G}(\boldsymbol{\chi}) = \mathbf{0}_{6\times1} \tag{5.46}$$

The derivations of the differential equations in this form are found in Appendix I. These equations resemble the initial part of the final equations of motion for the ballbot.

5.4 Input forces

The above equations of motion do not include any exogeneous inputs. To include forces into the Euler-Lagrange equation, the input forces has to affect the generalized coordinates represented by one differential equation each. To include input torques generated by the motors, which affects a combination of the states, Hamiltons principle of generalized forces is applied. The principle relies on the principle of conservation of work, where work corresponds to force multiplied with displacement. Let δr be a displacement along an axis of which the input force or torque is actuating. Conserving work thus requires the following relation to hold

$$\delta r F_{\text{ext}} = \delta \boldsymbol{\chi}^{\mathsf{T}} \boldsymbol{F}_{\text{generalized}} \tag{5.47}$$

By reorganizing the generalized forces are extracted from the input force, using the mapping between the coordinates on which the input force is actuating and the generalized coordinates.

$$\boldsymbol{F}_{\text{generalized}} = \left(\frac{\partial r}{\partial \boldsymbol{\chi}} \right)^{\mathsf{T}} F_{\text{ext}} \tag{5.48}$$

The Jacobian $\frac{\partial r}{\partial \boldsymbol{\chi}}$ thus defines the mapping from the coordinates affected by the input force to the generalized coordinates. Unfortunately due to the non-holonomic relationship between system states and motor shaft angles, as described in Section 4.2, it is not possible to derive a closed-form solution for this Jacobian. However, as we will see this mapping ends up being equal to the same Jacobian using the derivatives.

Assuming that we can write up the mapping from generalized coordinates to input coordinates as

$$r = R(\boldsymbol{\chi}) \tag{5.49}$$

The time derivative of this mapping is taken using the chain rule, see Appendix G.1.

$$\dot{r} = \frac{\partial R}{\partial \boldsymbol{\chi}} \dot{\boldsymbol{\chi}} \tag{5.50}$$

By reorganizing this time derivative it appears how the Jacobian of the time derivative of the mapping is equal to the Jacobian of the mapping itself.

$$\frac{\partial \dot{r}}{\partial \dot{\boldsymbol{\chi}}} = \frac{\partial R}{\partial \boldsymbol{\chi}} = \frac{\partial r}{\partial \boldsymbol{\chi}} \tag{5.51}$$

It is thus possible to use the velocity mapping from generalized coordinates to motor shaft angular velocities, (4.35), to define the Jacobian for mapping the input torques to generalized coordinate space.

$$\boldsymbol{J}_{\tau}\left({}^{\text{K}}_{\text{B}}\boldsymbol{q} \right) = \frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} \underbrace{\frac{\partial \dot{\theta}_0}{\partial {}^{\text{I}}\dot{x}}}_{\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\text{I}}\dot{x}}} & \underbrace{\frac{\partial \dot{\theta}_0}{\partial {}^{\text{I}}\dot{y}}}_{\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\text{I}}\dot{y}}} & \underbrace{\frac{\partial \dot{\theta}_0}{\partial {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}}}_{\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}}} \\ \frac{\partial \dot{\theta}_1}{\partial {}^{\text{I}}\dot{x}} & \frac{\partial \dot{\theta}_1}{\partial {}^{\text{I}}\dot{y}} & \frac{\partial \dot{\theta}_1}{\partial {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}} \\ \frac{\partial \dot{\theta}_2}{\partial {}^{\text{I}}\dot{x}} & \frac{\partial \dot{\theta}_2}{\partial {}^{\text{I}}\dot{y}} & \frac{\partial \dot{\theta}_2}{\partial {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}} \end{bmatrix} \tag{5.52}$$

The columns of this Jacobian is derived from the angular velocity definition, (4.35).

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\text{I}}\dot{x}} = \frac{1}{r_k} \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\boldsymbol{q} \right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\boldsymbol{q} \right) \hat{\mathbf{j}} \tag{5.53}$$

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\text{I}}\dot{y}} = -\frac{1}{r_k} \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\boldsymbol{q} \right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\boldsymbol{q} \right) \hat{\mathbf{i}} \tag{5.54}$$

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}} = -2 \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\boldsymbol{q} \right)^{\mathsf{T}} \tag{5.55}$$

The transpose of the resulting kinematics Jacobian is shown in (5.56).

$$\boldsymbol{J}_{\tau}^{\mathsf{T}}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\right)=\begin{bmatrix}\frac{1}{r_k}\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\right)^{\mathsf{T}}\\-\frac{1}{r_k}\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\right)^{\mathsf{T}}\\-2\end{bmatrix}\boldsymbol{\Phi}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}} \tag{5.56}$$

Since this Jacobian describes how much the motor velocity changes relative to a change in each generalized coordinate derivative, Hamiltons principle can now be applied.

$$\boldsymbol{Q}(\boldsymbol{\chi},\boldsymbol{\tau}_m)=\boldsymbol{J}_{\tau}^{\mathsf{T}}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\right)\boldsymbol{\tau}_m \tag{5.57}$$

where $\boldsymbol{\tau}_m$ is the input torque vector whose elements corresponds to the motor torque delivered by each of the three motors.

The equations of motion is updated to include the input torque

$$\boldsymbol{M}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}}+\boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}}+\boldsymbol{G}(\boldsymbol{\chi})=\boldsymbol{Q}(\boldsymbol{\chi},\boldsymbol{\tau}_m) \tag{5.58}$$

As a remark, please note how the included torque input results in a control-affine system and that the input matrix only depends on the shape variables and not the velocity.

$$\boldsymbol{Q}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}},\boldsymbol{u})=\boldsymbol{Q}(\boldsymbol{\chi})\,\boldsymbol{\tau}_m \tag{5.59}$$

This becomes handy when the sliding mode controller is derived, since a control-affine model makes it easier to derive the inverse dynamics, which are used for equivalent control.

5.5 Friction forces

The equations of motion shown in (5.58) do indeed model the dynamics of the system including input torque, but since no friction is included the input torque will just increase the energy level of the system which will thereafter be kept constant. We say that there is no dissipation of energy, which is obviously incorrect. Examples of common friction types include:

- Viscous, $B\cdot v$ or $B\cdot\omega$
- Coulomb, F_c or τ_c
- Stiction, F_s or τ_s

An example of the friction torque of a motor affected by the three listed friction types, is shown below with a piecewise function.

$$\tau_{\text{friction}}=\begin{cases}B\omega+\operatorname{sgn}(\omega)\,\tau_c & \omega\neq 0\\ \tau_m-\tau_L & |\tau_m-\tau_L|\leq(\tau_c+\tau_s)\\ (\tau_c+\tau_s)\operatorname{sgn}(\tau_m-\tau_L) & |\tau_m-\tau_L|>(\tau_c+\tau_s)\end{cases} \tag{5.60}$$

where τ_m is the generated torque by the motor and τ_L is the load torque by which the motor is loaded. Note how both Coulomb friction and stiction are non-linear friction elements as indicated with the piecewise function including discontinuous switching and the sign function.

Most of the previous ballbot models presented in Section 1.2 includes either no friction forces or only viscous friction forces. In this ballbot model it has been decided to only focus on the viscous

friction, a friction which is linearly proportional to the velocity along the axis it affects. To include such friction, the equations of motion are augmented by a friction matrix which includes the friction components along each of the generalized coordinates.

Three types of friction will be included:

1. Viscous friction from ball to ground friction, $\boldsymbol{D}_K$
2. Viscous friction from body angular velocity, $\boldsymbol{D}_B$, which can thus be seen as air friction.
3. Viscous friction from motor angular velocity, $\boldsymbol{D}_M$, thus a combination of internal motor friction and ball to omniwheel friction.

The friction components are included in the equations of motion by adding a friction force matrix, $\boldsymbol{D}(\dot{\boldsymbol{\chi}})$.

$$\boldsymbol{D}(\dot{\boldsymbol{\chi}}) = \boldsymbol{D}_K(\dot{\boldsymbol{\chi}}) + \boldsymbol{D}_B(\dot{\boldsymbol{\chi}}) + \boldsymbol{D}_M(\dot{\boldsymbol{\chi}}) \tag{5.61}$$

The friction component due to translational movement of the ball is constructed directly from the generalized coordinates, since both ${}^{\mathrm{I}}x$ and ${}^{\mathrm{I}}y$ are in the generalized coordinates and thus independent.

$$\boldsymbol{D}_K(\dot{\boldsymbol{\chi}}) = B_{vk} \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ \mathbf{0}_{4\times 1} \end{bmatrix} = B_{vk} \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \mathbf{0}_{4\times 4} \end{bmatrix} \dot{\boldsymbol{\chi}} \tag{5.62}$$

Since the body angular velocity is not directly present in the generalized coordinates, a similar technique to the Hamiltons principle is used to derive the generalized friction forces due to body angular velocity. First the mapping from generalized coordinates to body angular velocity, (H.51), is used to derive a Jacobian of the mapping.

$${}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{5.63}$$

$$\boldsymbol{J}_{\boldsymbol{\omega}_B}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) = \frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} 0 & 0 & 2\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \tag{5.64}$$

Next the Jacobian of this mapping is used to construct the friction forces in generalized coordinates.

$$\begin{aligned} \boldsymbol{D}_B(\dot{\boldsymbol{\chi}}) &= \boldsymbol{J}_{\boldsymbol{\omega}_B}^{\mathsf{T}} B_{vb}\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \\ &= \boldsymbol{J}_{\boldsymbol{\omega}_B}^{\mathsf{T}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) B_{vb}\, 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\ &= B_{vb} \begin{bmatrix} 0 & 0 & 2\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} 2\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\ &= 4B_{vb} \begin{bmatrix} 0 \\ 0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{bmatrix} = 4B_{vb} \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{bmatrix} \dot{\boldsymbol{\chi}} \end{aligned} \tag{5.65}$$

Finally, the viscous friction in the motor affects the system by subtracting the friction torque from the applied motor torque.

$$\boldsymbol{\tau}_{\text{applied}} = \boldsymbol{\tau}_m - \boldsymbol{\tau}_{\text{friction}} \tag{5.66}$$

The friction torque is computed from the motor angular velocity.

$$\boldsymbol{\tau}_{\text{friction}}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = B_{vm}\dot{\boldsymbol{\theta}} \tag{5.67}$$

Similarly to how the kinematics Jacobian was used in Section 5.4 to convert the motor torque to generalized forces, the kinematics Jacobian is applied to the viscous friction torques.

$$\begin{aligned} \boldsymbol{D}_M(\dot{\boldsymbol{\chi}}) &= \boldsymbol{J}_\tau^\mathsf{T}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\boldsymbol{\tau}_{\text{friction}} \\ &= \boldsymbol{J}_\tau^\mathsf{T}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) B_{vm}\dot{\boldsymbol{\theta}} \end{aligned} \tag{5.68}$$

5.6 Quaternion constraint

The resulting equations of motion are now on the form

$$\boldsymbol{M}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \boldsymbol{G}(\boldsymbol{\chi}) + \boldsymbol{D}(\dot{\boldsymbol{\chi}}) = \boldsymbol{Q}(\boldsymbol{\chi})\,\boldsymbol{\tau}_m \tag{5.69}$$

but as mentioned initially these equations are unconstrained whereof the generalized coordinates are constrained. The quaternion, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, with its four elements, $q_0,\ldots,q_3$, only describes three degrees of freedom and thus has one element too much, a redundant element. The quaternion unit-norm constraint thus has to be applied to the equations of motion [53].

$$\left\|{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right\| = 1 \rightarrow \left\|{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^\mathsf{T}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 1 \tag{5.70}$$

This can be written as a holonomic constraint.

$$h_q(\boldsymbol{\chi}) = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^\mathsf{T}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} - 1 \quad (=0) \tag{5.71}$$

The equations of motion thereby become a set of Differential Algebraic Equations (DAE):

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \boldsymbol{G}(\boldsymbol{\chi}) + \boldsymbol{D}(\dot{\boldsymbol{\chi}}) &= \boldsymbol{Q}(\boldsymbol{\chi})\,\boldsymbol{\tau}_m \\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^\mathsf{T}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} &= 1 \end{aligned} \tag{5.72}$$

However, we are not interested in a set of DAEs but would like a set of ODEs. As described in the beginning holonomic constraints can be included in the Euler-Lagrange equation by using Lagrange multipliers, λ, which ends up corresponding to the reactive forces that ensure the constraints are fulfilled.

5.6.1 Reducing the equations of motion space

The first step requires the equations of motion space to be reduced by embedding the constraint into the equations of motion through a Lagrange multiplier.

$$\boldsymbol{M}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \boldsymbol{G}(\boldsymbol{\chi}) + \boldsymbol{D}(\dot{\boldsymbol{\chi}}) - \boldsymbol{H}_q^\mathsf{T}(\boldsymbol{\chi})\,\lambda = \boldsymbol{Q}(\boldsymbol{\chi})\,\boldsymbol{\tau}_m \tag{5.73}$$

where the constraint Jacobian, $\boldsymbol{H}_q(\boldsymbol{\chi})$, is derived using the properties from Appendix G.3.

$$\boldsymbol{H}_q^\mathsf{T}(\boldsymbol{\chi}) = \left(\frac{\partial h_q(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}}\right)^\mathsf{T} = \begin{bmatrix} 0 \\ 0 \\ 2\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \end{bmatrix} \tag{5.74}$$

The space is reduced by elimination of the Lagrange multiplier. Since the constraint Jacobian is not directly invertible, solving for the Lagrange multiplier requires a few mathematical tricks [54]. Since it is allowed to transform the system with a diffeomorphism transformation, for which

the inverse also exists, we can construct a transformation matrix that when multiplied with the constraint Jacobian, results in the Lagrange multiplier only being present in one row.

$$\overline{\boldsymbol{H}}_q = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*\circ} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \tag{5.75}$$

Pre-multiplying the constraint Jacobian with this diffeomorphism results in a vector with only one non-zero element.

$$\begin{aligned} \overline{\boldsymbol{H}}_q \, \boldsymbol{H}_q^{\mathsf{T}}(\boldsymbol{\chi}) &= \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 2\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \\ \overline{\boldsymbol{H}}_q \, \boldsymbol{\chi} &= \begin{bmatrix} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & 1 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \end{aligned} \tag{5.76}$$

Note how the identity elements are included in the transform to make it invertible, thus to include the position states as shown with the transformed state vector.

The diffeomorphism is applied to the equations of motion by pre-multiplying all terms with $\overline{\boldsymbol{H}}_q$. The reorganized equations of motion is shown below.

$$\overline{\boldsymbol{H}}_q \, \boldsymbol{H}_q^{\mathsf{T}}(\boldsymbol{\chi}) \, \lambda = \overline{\boldsymbol{H}}_q \, \boldsymbol{M}(\boldsymbol{\chi}) \, \ddot{\boldsymbol{\chi}} + \overline{\boldsymbol{H}}_q \, \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \, \dot{\boldsymbol{\chi}} + \overline{\boldsymbol{H}}_q \, \boldsymbol{G}(\boldsymbol{\chi}) + \overline{\boldsymbol{H}}_q \, \boldsymbol{D}(\dot{\boldsymbol{\chi}}) - \overline{\boldsymbol{H}}_q \, \boldsymbol{Q}(\boldsymbol{\chi}) \, \boldsymbol{\tau}_m \tag{5.77}$$

Due to the choice of diffeomorphism, the Lagrange multiplier is given from a single row in the reorganized equations of motion, (5.77).

$$\begin{aligned} \lambda = \frac{1}{2} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \Big(& \overline{\boldsymbol{H}}_q \, \boldsymbol{M}(\boldsymbol{\chi}) \, \ddot{\boldsymbol{\chi}} + \overline{\boldsymbol{H}}_q \, \boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \, \dot{\boldsymbol{\chi}} + \overline{\boldsymbol{H}}_q \, \boldsymbol{G}(\boldsymbol{\chi}) + \overline{\boldsymbol{H}}_q \, \boldsymbol{D}(\dot{\boldsymbol{\chi}}) \\ & - \overline{\boldsymbol{H}}_q \, \boldsymbol{Q}(\boldsymbol{\chi}) \, \boldsymbol{\tau}_m \Big) \end{aligned} \tag{5.78}$$

Eliminating the Lagrange multiplier is now a matter of inserting the solved Lagrange multiplier back into the equations of motion. However, since the Lagrange multiplier is only present in one row, it can be eliminated easily by just removing the corresponding row, row 3, from the equations of motion [53].

$$\widetilde{\boldsymbol{H}} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 1} & \mathbf{I}_3 \end{bmatrix} \overline{\boldsymbol{H}}_q = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{3\times 2} & \begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{I}_3 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \tag{5.79}$$

All the derived model matrices thus have to be pre-multiplied by $\widetilde{\boldsymbol{H}}$ to reduce the equation space and eliminate the Lagrange multiplier.

5.6.2 Constraint enforcement

Pre-multiplying the equations of motion with $\widetilde{\boldsymbol{H}}$ will reduce the equations from six to five, but the generalized coordinates are still unconstrained. The generalized coordinates are constrained by including an extra equation as part of the equations of motion, derived from the second-order time derivative of the holonomic constraint.

$$\begin{aligned} & h_q(\boldsymbol{\chi}) = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} - 1 \quad (= 0) \\ & \qquad\qquad \downarrow \tfrac{\mathrm{d}}{\mathrm{d}t} \\ & \dot{h}_q(\boldsymbol{\chi}) = {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} + {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \quad (= 0) \\ & \qquad\qquad \downarrow \tfrac{\mathrm{d}}{\mathrm{d}t} \\ & \ddot{h}_q(\boldsymbol{\chi}) = 2\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} + 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = 2\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} + 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \quad (= 0) \end{aligned} \tag{5.80}$$

The second-order differential constraint that complies to the unit-norm quaternion constraint is thus:

$$\ddot{h}_q(\boldsymbol{\chi}) = 0 \;\rightarrow\; {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} = -\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{5.81}$$

To include this as another differential equation into the equations of motion, defined by the different matrices, it has to be rewritten to use the generalized coordinate vectors, $\boldsymbol{\chi}$, $\dot{\boldsymbol{\chi}}$ and $\ddot{\boldsymbol{\chi}}$. A transformation matrix to extract the quaternion elements from the generalized coordinate vector is defined as $\boldsymbol{R}$.

$$\boldsymbol{R} = \begin{bmatrix} \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{bmatrix} \tag{5.82}$$

$${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \boldsymbol{R}\,\boldsymbol{\chi} \tag{5.83}$$

$$\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{bmatrix} \tag{5.84}$$

Substituting this transformation into (5.81) yields the extra differential equation

$$\boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}\,\ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{\chi}}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}\,\dot{\boldsymbol{\chi}} = 0 \tag{5.85}$$

which is included side-by-side with the other five equations.

$$\begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi}) \\ \boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} \end{bmatrix}\ddot{\boldsymbol{\chi}} + \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}}) \\ \dot{\boldsymbol{\chi}}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} \end{bmatrix} + \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{G}(\boldsymbol{\chi}) \\ 0 \end{bmatrix} + \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{D}(\dot{\boldsymbol{\chi}}) \\ 0 \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{Q}(\boldsymbol{\chi}) \\ \mathbf{0}_{1\times 3} \end{bmatrix}\boldsymbol{\tau}_m \tag{5.86}$$

5.6.3 Constraint regularization

For controller design purposes it would be sufficient to just include this second order differential constraint in the model as shown in (5.86). Given a system whose initial conditions fulfil the unit-norm constraint and the first order derivative of this, the second order differential constraint would ensure that the quaternion is kept on the unit sphere. However, since the model will also be used for simulation purposes where numerical deviations are to be expected, Baumgartes technique for stabilization of constraints in dynamical systems [55] [56] is embodied. This technique involves an augmentation of the differential constraint equation to include regularization factors on both the algebraic constraint itself and the first order derivative.

As described in [55] a stabilizing constraint law is designed to stabilize $h_q(\boldsymbol{\chi}) \rightarrow 0$. This obviously requires the stabilization of $\dot{h}_q(\boldsymbol{\chi}) \rightarrow 0$ and $\ddot{h}_q(\boldsymbol{\chi}) \rightarrow 0$. The stabilizing constraint law is constructed by combining these three criteria with a regularization gain, β.

$$\ddot{h}_q(\boldsymbol{\chi}) + \beta\,\dot{h}_q(\boldsymbol{\chi}) + \beta^2\,h_q(\boldsymbol{\chi}) = 0 \tag{5.87}$$

Note how this augmented differential constraint equation collapses into the previous, (5.81), if the constraint is fulfilled by $h_q(\boldsymbol{\chi}) = 0$ and $\dot{h}_q(\boldsymbol{\chi}) = 0$. However, if the constraint is not fulfilled, e.g., due to an incorrect initial condition or due to numerical inaccuracies from numerical integration, the regularization terms will drag the quaternion back onto the unit sphere.
Inserting the derivative definitions from (5.80) leads to the regularizing differential equation to include in the equations of motion.

$${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} = -\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} - \beta\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} - \frac{1}{2}\beta^2\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} - 1\right) \tag{5.88}$$

Similarly to before the generalized coordinate vectors has to be substituted into the differential equation by using the transformation matrix, $\boldsymbol{R}$, which extracts the quaternion part.

$$\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\,\ddot{\boldsymbol{\chi}} + \dot{\boldsymbol{\chi}}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\,\dot{\boldsymbol{\chi}} + \beta\,\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\,\dot{\boldsymbol{\chi}} + \frac{1}{2}\,\beta^2\left(\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\,\boldsymbol{\chi} - 1\right) = 0 \tag{5.89}$$

The differential equation is combined with the other five equations of motion to yield the final equations of motion.

$$\underbrace{\begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi})\\ \boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\end{bmatrix}}_{\widetilde{M}(\chi)}\ddot{\boldsymbol{\chi}} + \underbrace{\begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\\ \dot{\boldsymbol{\chi}}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R} + \beta\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\end{bmatrix}}_{\widetilde{C}(\chi,\dot{\chi})}\dot{\boldsymbol{\chi}} + \underbrace{\begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{G}(\boldsymbol{\chi})\\ \frac{1}{2}\beta^2\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\,\boldsymbol{\chi}\end{bmatrix}}_{\widetilde{G}(\chi)} + \underbrace{\begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{D}(\dot{\boldsymbol{\chi}})\\ -\frac{1}{2}\beta^2\end{bmatrix}}_{\widetilde{D}(\dot{\chi})} = \underbrace{\begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{Q}(\boldsymbol{\chi})\\ \boldsymbol{0}_{1\times 3}\end{bmatrix}}_{\widetilde{Q}(\chi)}\boldsymbol{\tau}_m \tag{5.90}$$

5.7 Equations of Motion

This concludes the derivation of the quaternion-based non-linear ordinary differential equations of the ballbot using Lagrangian mechanics.

The final equations of motion, consisting of six coupled differential equations of the control-affine system, are shown in (5.91) with the matrix content shown in (5.92) to (5.96) given the state vector in (5.97). The control input, $\boldsymbol{\tau} = \boldsymbol{\tau}_m$, is the torque vector containing the output shaft torques delivered by the three motors. Note that the gearing ratio is not included since the torques are defined on the output shaft.

$$\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})\,\ddot{\boldsymbol{\chi}} + \widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} + \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) + \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}}) = \widetilde{\boldsymbol{Q}}(\boldsymbol{\chi})\,\boldsymbol{\tau} \tag{5.91}$$

where

$$\widetilde{\boldsymbol{M}}(\boldsymbol{\chi}) = \begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi})\\ \boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\end{bmatrix} \tag{5.92}$$

$$\widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}}) = \begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\\ \dot{\boldsymbol{\chi}}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R} + \beta\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\end{bmatrix} \tag{5.93}$$

$$\widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) = \begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{G}(\boldsymbol{\chi})\\ \frac{1}{2}\beta^2\boldsymbol{\chi}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{R}\,\boldsymbol{\chi}\end{bmatrix} \tag{5.94}$$

$$\widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}}) = \begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{D}(\dot{\boldsymbol{\chi}})\\ -\frac{1}{2}\beta^2\end{bmatrix} \tag{5.95}$$

$$\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi}) = \begin{bmatrix}\widetilde{\boldsymbol{H}}\,\boldsymbol{Q}(\boldsymbol{\chi})\\ \boldsymbol{0}_{1\times 3}\end{bmatrix} \tag{5.96}$$

The state vector is defined as

$$\boldsymbol{x} = \begin{bmatrix} {}_{\mathrm{I}}x \\ {}_{\mathrm{I}}y \\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\ {}_{\mathrm{I}}\dot{x} \\ {}_{\mathrm{I}}\dot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \boldsymbol{\chi} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \dot{\boldsymbol{\chi}} \end{matrix} \tag{5.97}$$

The model results in a set of second order control-affine differential equations, shown in (5.98).

$$\ddot{\boldsymbol{\chi}} = \underbrace{\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\Big(-\widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} - \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) - \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}})\Big)}_{\boldsymbol{f}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})} + \underbrace{\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\,\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi})}_{\boldsymbol{g}(\boldsymbol{\chi})}\boldsymbol{\tau} \tag{5.98}$$

The derived model shows how the ballbot is an underactuated system. An underactuated system is known to be on the form of (5.98) where the rank of the input matrix, $\boldsymbol{g}(\boldsymbol{\chi})$, is smaller than the degrees of freedom [57]. Put differently, the generalized input forces are not able to instantaneously actuate the accelerations of the generalized coordinate space in arbitrary directions.

Since the equations of motion models the second order derivatives, the complete non-linear ODE is assembled by stacking the equations of motion on top of six integrators. The complete non-linear ODE to be used for controller development is shown in (5.99).

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\,\boldsymbol{\tau} = \underbrace{\begin{bmatrix} \dot{\boldsymbol{\chi}} \\ \widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\Big(-\widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\,\dot{\boldsymbol{\chi}} - \widetilde{\boldsymbol{G}}(\boldsymbol{\chi}) - \widetilde{\boldsymbol{D}}(\dot{\boldsymbol{\chi}})\Big) \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} \boldsymbol{0}_{6\times 3} \\ \widetilde{\boldsymbol{M}}(\boldsymbol{\chi})^{-1}\,\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi}) \end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{x})}\boldsymbol{\tau} \tag{5.99}$$

A symbolic model has been derived with MATLAB using the Symbolic Toolbox and has been verified against the derivations shown in this chapter. The symbolic model can be found in the MATLAB GitHub repository listed in the preface.

This concludes the derivation of the quaternion-based non-linear model of the ballbot. Before using this model for controller design, parts of it are verified in the next chapter.

6 Model Verification

Through simulation, the dynamics of the quaternion-based model has been compared with the non-linear Euler-angle model from ETH [2], using their simulation model which has been obtained. The open-loop behaviour of the ballbot constitutes a falling pendulum with the option of applying test torques. With equivalent parameters the two models behave the same.

Due to the unstable dynamics of the ballbot it is deemed unfeasible to verify the model against the actual plant, before applying the controller. Instead the dynamic model will be verified implicitly when the controller is tested and verified.
The kinematics can, however, be verified without applying a controller. In the following chapter the kinematics are verified using a motion capture system. Finally, the modelling part is concluded with a summary of the derived kinematics and dynamics.

6.1 Kinematics verification

The kinematics are verified against close to ground-truth measurements using a Vicon motion capture system [58]. Five Vicon markers are placed on the ballbot with two of them being installed in known screw holes on each side and three on the top-most plate of the bottom part. By placing two markers in the known screw holes and defining the origin of the rigid body in Vicon as the center between these two makers, the Vicon rigid body origin can be converted to the origin of the ballbot, being in the center of the ball, using the known locations extracted from the CAD drawings.
To verify the kinematics the ballbot is moved around manually while encoder measurement and Vicon measurements are recorded. Encoder measurements, in terms of the absolute ticks, are sent from the embedded firmware over USB to the ROS driver running on the onboard computer to be recorded locally. As as post-processing step the numerical derivatives of the encoder ticks are computed and converted to motor angular velocities, $\dot{\boldsymbol{\theta}}$.
Both the position and orientation of the rigid body are captured with Vicon and recorded on an external computer connected to the motion capture system. As a post-processing step the numerical derivatives of the quaternion and position are computed to get the quaternion derivative and inertial frame velocity. The timestamps from both systems are synchronized in a post-processing step through the cross correlation of an angular velocity spike, caused by a fast 90° yaw turn performed before starting the actual test.

The computed translational velocities from the kinematics model is shown in Figure 6.1 together with the actual velocities measured by Vicon. The inputs to the kinematics model, being the motor angular velocities and the inclination angle of the ballbot, is shown on the right. The kinematics-based velocities clearly follows the Vicon ground-truth measurements, even though they seem to be far more noisy than the Vicon velocities, suggesting the need of an estimator, see Appendix N.

During another test, shown in Figure 6.2, the ballbot is held almost upright with inclination angles between ±5°, but with a yaw angle increasing in 60° steps. Since the output of the kinematics is the translational velocity in the inertial frame, this test verifies that the heading of the ballbot affects the kinematics correctly.

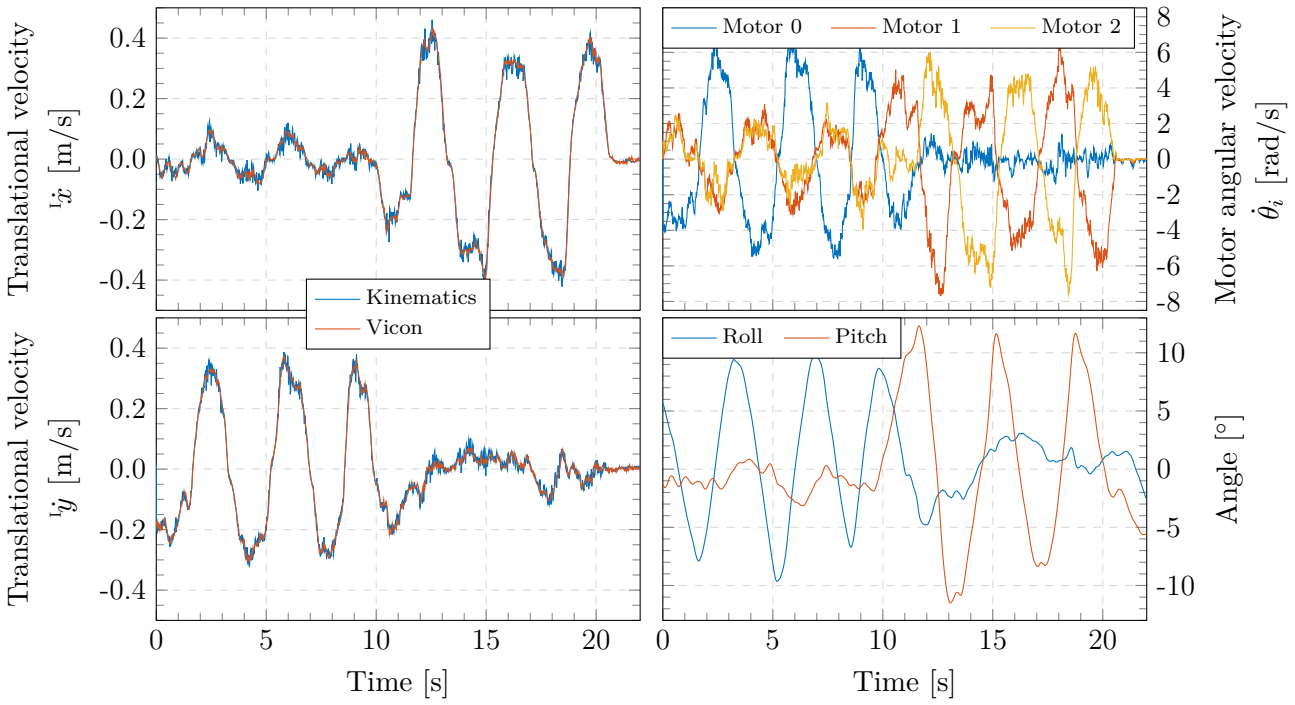


Figure 6.1: Kinematics verification where the ballbot is moved around manually

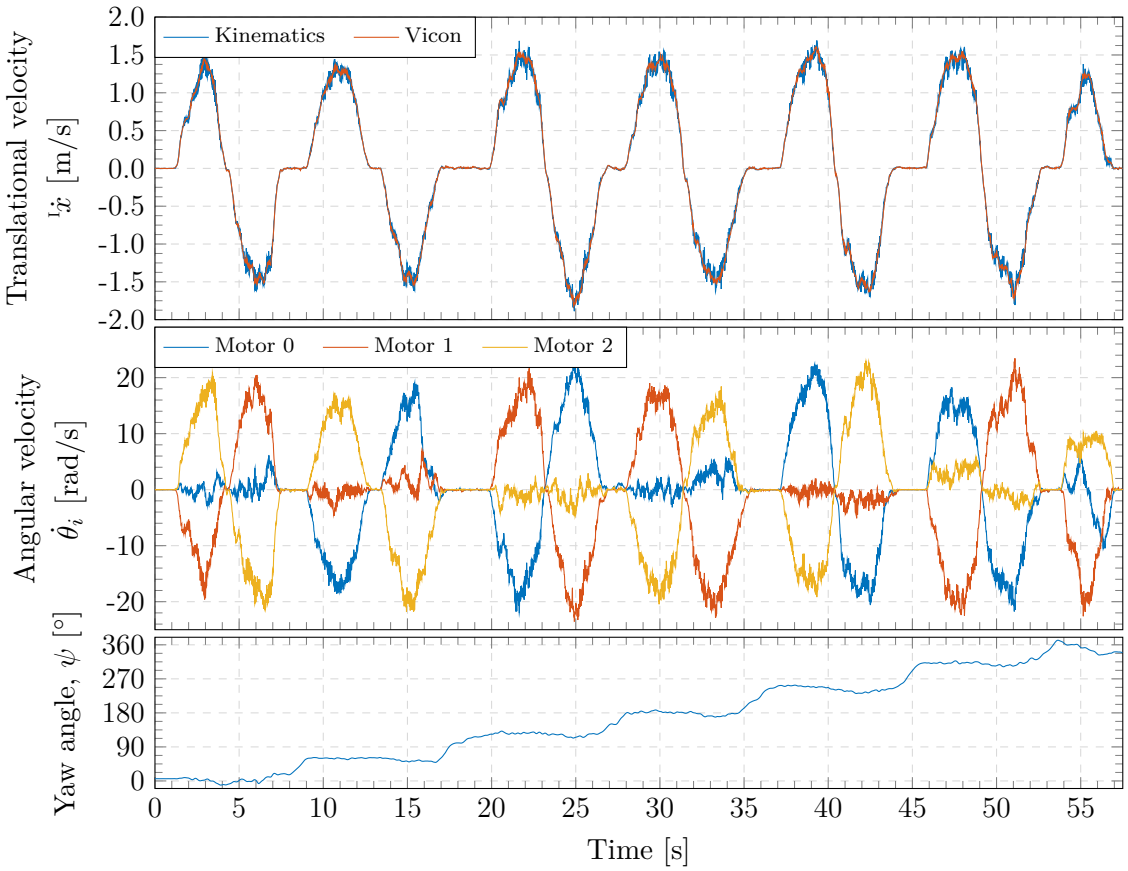


Figure 6.2: Kinematics verification of heading dependency by moving the ballbot manually with inclination angles between ±5° and an increasing yaw angle

The results shown in Figure 6.1 and Figure 6.2 are computed with the forward kinematics from (4.43) using the quaternion and quaternion derivative estimates from the onboard MTI IMU. These estimates are verified to match with the Vicon ground-truth orientation in Appendix M.9.

6.2 Model summary

A non-linear quaternion-based model of a ballbot has been derived using Lagrangian mechanics. The ballbot was separated into two rigid bodies, one for the ball and one for the body, which balances on top of the ball. Six generalized coordinates were chosen to describe the assumed five degrees of freedom of the system. This included the x- and y position of the ball and the four quaternion elements, describing the orientation of the body. Kinetic and potential energy equations were derived for the two bodies and omniwheels using quaternion mathematics. The input forces from the motors, acting on the omniwheels, were included using a derived kinematic model, which describes the mapping between angular velocities of the wheels and the generalized coordinates. Viscous friction forces were applied on the translational velocity, angular velocity of the body and angular velocity of the wheels. Due to the choice of a dependent set of generalized coordinates, the quaternion norm constraint was applied to the equations of motion with a Lagrange multiplier. The result was a control-affine non-linear ODE with 12 states.

Lastly the kinematics was verified against the hardware prototype using a motion capture system. The dynamic model, however, has not been verified due to the unstable dynamics, and will implicitly be verified with the controller.

The next step is to use this quaternion-based model to derive a non-linear sliding mode controller for balance stabilization.

Part II

Balance Controller

7 Design Specification

The second part of the thesis deals with the controller design of a non-linear sliding mode controller for balance control. Sliding mode controller design involves a two-step procedure, see Appendix L. First a sliding manifold is designed in Chapter 8 to fulfil the closed-loop requirements. Next two control laws are designed in Chapter 9 for nominal plant cancellation and disturbance rejection. The controller should fit in the suggested structure shown in Figure 2.8, putting some requirements on the inputs and outputs of the controller. Furthermore, some general performance requirements are defined.

The following chapter serves as a design specification with the purpose of presenting the scope of the balance controller designed in this thesis. The scope includes the control objective, the chosen operating envelope, the desired bandwidth and the desired performance. Note that the design specification do not list any specific requirements since no analytic methods have been used to derive requirements from use cases etc. Instead the design specification is based on a discussion with the Kugle project group, with a starting point in the collection of existing state-of-the-art ballbot projects presented in Section 1.2.

7.1 Control objective

The purpose of the balance controller is to set the torque output, $\boldsymbol{\tau}$, to the three motors driving the omniwheels, so that the ballbot is kept balanced. The controller takes in a desired quaternion reference, ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$, and an angular velocity reference, ${}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}$, to be used as feed-forward to reduce lag and improve reference tracking. The control loop with the balance controller is illustrated in Figure 7.1.

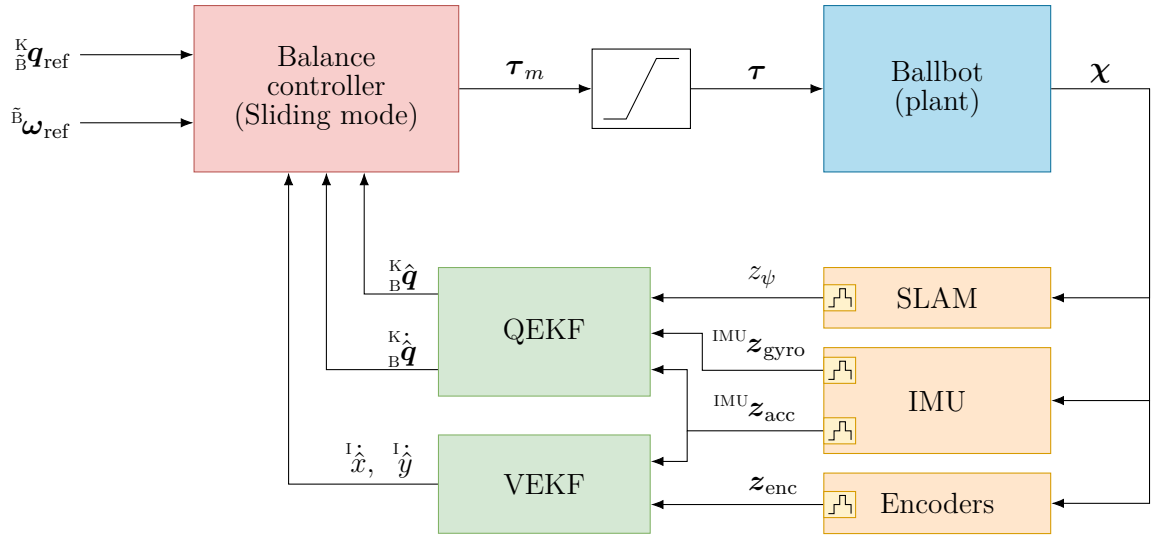


Figure 7.1: Structure of the balance controller loop

The controller should be designed as a non-linear sliding mode controller utilizing the complete coupled non-linear model derived in Chapter 5. Estimates of the states in the state vector, see (5.97), including the quaternion, quaternion derivative and velocity estimate, is provided by corresponding estimators from Appendix M and Appendix N, at the same rate as the controller.

7.2 Operating envelope

Seen from an operating point of view, the system has some behavioural requirements that can be used to specify an operating envelope. Furthermore, the ballbot is also constrained mechanically, leading to e.g., inclination limitations.

1. Maximum translational velocity of 0.5 m/s
2. Linear acceleration limit of 1 m/s^2
3. Maximum heading velocity (rotational, thus the same as yaw angular velocity, $\dot{\psi}$) of 1 rad/s $\approx 60\ ^\circ/s$
4. Angular acceleration limit of 10 rad/s^2
5. Inclination should be kept within $\pm 5^\circ$
    - The acceleration limit of 1 m/s^2 leads to an inclination limit of of 7.9° according to the steady state parts of the linearized model in (Q.19) and (S.10).
    - It is, however, chosen to limit the inclination even further to add some headroom to the linear acceleration limit.
6. Angular velocity on inclination should be kept less than $20\ ^\circ/s$ to avoid sudden, aggressive jumps in the angle and to reduce the risk of wheel-slip.

7.3 Bandwidth considerations

Since the balance controller will be used in a cascaded structure with a model predictive controller generating angular velocity references at a lower rate, see Figure 2.8, the inner closed loop bandwidth should be at least 3 times faster than the outer closed loop bandwidth [41]. This allows the inner loop dynamics to be neglected when designing the outer loop, and the MPC can thus be designed on a simplified model. As another rule of thumb the sample rate should be chosen 10 times faster than the bandwidth [59]. Furthermore, by choosing a high controller sample rate compared to the plant dynamics, the controller can be designed based on the continuous plant instead of discretizing the model.

Unfortunately aliasing, sensor noise, noise from discretization and numerical approximations etc. become prominent at increasing sample rates. The sample rate and filter design should thus be tuned to match the given hardware and estimator design, including the IMU and encoders. The estimators should be tuned to ensure that the bandwidth is 2 to 6 times faster than the controller bandwidth [59].

The sample rate of the sliding mode controller and the corresponding estimators are set to 200 Hz.

7.4 Performance specification

Since the hardware platform is new and untested it is decided not to put any specific performance requirements to the controller. Instead a few general criteria for the desired performance are defined.

1. Reference tracking performance of inclination, equivalent to roll and pitch, should be more aggressive than heading.
   - It is desirable to have a smooth heading performance and a longer reaching time is acceptable.
   - However, inclination errors results in acceleration errors, which will quickly cause large position errors.
2. Inclination tracking error should be kept within $\pm 1^\circ$ at a steady state reference.
3. Heading tracking error should be kept within $\pm 5^\circ$
4. Should be able to track inclination steps, ramps and sine wave references with less than 0.1 s lag, given an angular velocity reference and a matching quaternion reference.
5. Should be able to track quaternion references even though no angular velocity reference is provided with an acceptable lag of less than 1 s, thus a reaching time for step references of less than 1 s.

7.5 Velocity controller

Even though the balance controller is supposed to be used with the MPC as illustrated in Figure 2.8, a velocity LQR controller is derived in Appendix Q to stabilize the otherwise unstable translational dynamics during testing, due to external disturbances and misalignment in the center of mass.

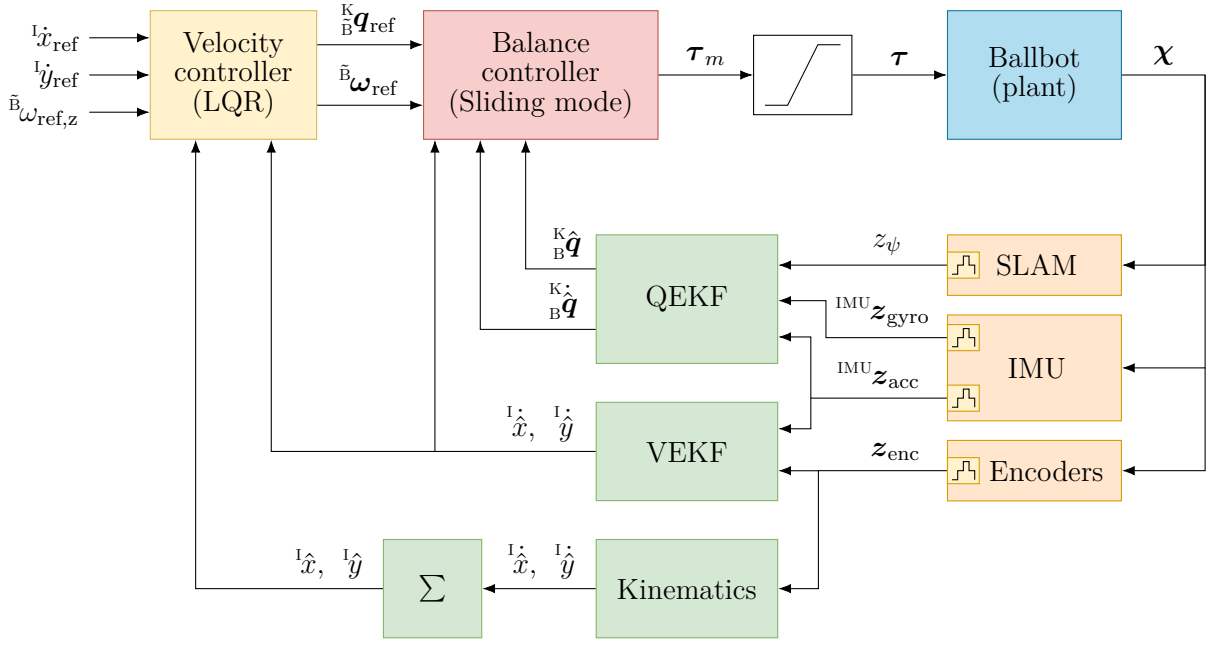


Figure 7.2: Cascade structure of the balance controller and velocity controller

The velocity controller provides the same references to the balance controller as the MPC, namely the quaternion and angular velocity reference, while relying on velocity and position estimates from the VEKF, Appendix N, and numerically integrated kinematics, see Figure 7.2. The velocity controller runs at the same sample rate as the balance controller and estimators, thus 200 Hz.

The controller is designed to operate within the same envelope as the balance controller mentioned above, but the overall design and tuning is outside the scope of this thesis. The velocity controller will, however, be tested in Chapter 14, since it provides valuable results in whether the balance controller works in the cascade configuration with arbitrarily changing references.

This concludes the design specification. In the next chapters the non-linear sliding mode controller is designed to fulfil the above specifications, starting with the sliding manifold design.

8 Sliding Surface Design

As mentioned in Appendix L there is no general method for choosing a sliding surface, also known as a sliding manifold. It is chosen to follow the strategy proposed by Slotine [60], who defines the sliding manifold as a linear combination of the tracking error and its derivative. Since we would like to track a given quaternion reference and angular velocity reference, the quaternion tracking error has to be quantified.

For underactuated systems, such as the ballbot, one would normally have to transform the system to a normal form, also known as regular form, using a global change of coordinates (diffeomorphism) such as to decouple actuated dynamics from the internal dynamics [61]. To simplify the controller design and since the balance controller should only keep the ballbot balanced by tracking the quaternion reference, the translational dynamics of the system is neglected and considered an input disturbance.

8.1 Quaternion tracking error

For a Euclidean error vector, such as for the angular velocity or if the orientation was given by Euler angles, we define the error through a normal subtraction.

$$\boldsymbol{x}_e = \boldsymbol{x} - \boldsymbol{x}_{\text{ref}} \tag{8.1}$$

Note how this reads that if the system is positive away from the reference the error is positive, and the controller should move the system in the negative error direction.

However, for quaternions these operations are not permitted since rotation quaternions are only closed under quaternion multiplication. The same applies for rotation matrices as illustrated in the error arithmetic table, Table 8.1.

Operation	**Euclidean**	**Quaternion**	**Rotation matrix (DCM)**
Summation	$\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b}$	$\boldsymbol{q}_c = \boldsymbol{q}_a \circ \boldsymbol{q}_b$	$\boldsymbol{R}_c = \boldsymbol{R}_a \boldsymbol{R}_b$
Subtraction (e.g., error)	$\boldsymbol{e} = \boldsymbol{a} - \boldsymbol{b}$	$\boldsymbol{q}_e = \boldsymbol{q}_a \circ \boldsymbol{q}_b^*$	$\boldsymbol{R}_e = \boldsymbol{R}_a \boldsymbol{R}_b^\mathsf{T}$

Table 8.1: Arithmetic for different representations

Note that both the quaternion and rotation matrix involves non-commutative operations and the order therefore matters. Subtraction of a reference quaternion is done by multiplication with its conjugate, which negates the rotation vector and thus rotates in the opposite direction.

The conjugated quaternion can be either left-multiplied or right-multiplied, leading to the same type of reference subtraction but represented in two different frames.

If the reference quaternion is right-multiplied it leads to a quaternion error defined in the ball frame, equivalent to the inertial frame.

$$ {}^{\mathrm{K}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} = \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{8.2} $$

If the reference quaternion is left-multiplied it leads to a quaternion error defined in the body frame.

$$ {}^{\mathrm{B}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{8.3} $$

In both cases the error quaternion collapses to the current orientation if the reference quaternion is set to the unit quaternion, ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}$.

To understand the difference between the two error quaternions and the definition of the error frame, an example is given. Euler angles are used to construct the quaternions for interpretability. Let the reference be defined with a yaw of 180° and a roll of 20°.

$$ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} = \boldsymbol{q}_z(180°)\boldsymbol{q}_x(20°) = \begin{bmatrix} 0 & 0 & \sin\left(\frac{20°}{2}\right) & \cos\left(\frac{20°}{2}\right) \end{bmatrix}^{\mathsf{T}} \tag{8.4} $$

Let the current orientation be close to the reference with a yaw of 180° and a roll of 15°.

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \boldsymbol{q}_z(180°)\boldsymbol{q}_x(15°) = \begin{bmatrix} 0 & 0 & \sin\left(\frac{15°}{2}\right) & \cos\left(\frac{15°}{2}\right) \end{bmatrix}^{\mathsf{T}} \tag{8.5} $$

There is thus only a 5° difference on roll, requiring the body to tilt slightly more to the right while keeping the current heading. The two quaternion errors are computed as:

$$ {}^{\mathrm{K}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} = \begin{bmatrix} \cos\left(\frac{5°}{2}\right) & \sin\left(\frac{5°}{2}\right) & 0 & 0 \end{bmatrix}^{\mathsf{T}} \tag{8.6} $$

$$ {}^{\mathrm{B}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{5°}{2}\right) & -\sin\left(\frac{5°}{2}\right) & 0 & 0 \end{bmatrix}^{\mathsf{T}} \tag{8.7} $$

Since the error is defined as positive when being positive away from the reference, the system has to move in the negative direction of the error. The inertial quaternion error, ${}^{\mathrm{K}}\boldsymbol{q}_e$, thus states that the system has to rotate $-5°$ around the inertial frame x-axis, but this has to be converted to an according motion that the body can make on top of the ball. This is exactly what the body quaternion error defines, ${}^{\mathrm{B}}\boldsymbol{q}_e$, by stating that the system should rotate 5° around the body x-axis.

The difference become more distinct at other heading angles. Let the reference be defined with a yaw of 90° and similarly a roll of 20°. Again let the current orientation be close to the reference with a yaw of 90° and a roll of 15°.

$$ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} = \boldsymbol{q}_z(90°)\boldsymbol{q}_x(20°) = \begin{bmatrix} \cos\left(\frac{90°}{2}\right)\cos\left(\frac{20°}{2}\right) \\ \cos\left(\frac{90°}{2}\right)\sin\left(\frac{20°}{2}\right) \\ \sin\left(\frac{90°}{2}\right)\sin\left(\frac{20°}{2}\right) \\ \sin\left(\frac{90°}{2}\right)\cos\left(\frac{20°}{2}\right) \end{bmatrix}, \quad {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \boldsymbol{q}_z(90°)\boldsymbol{q}_x(15°) = \begin{bmatrix} \cos\left(\frac{90°}{2}\right)\cos\left(\frac{15°}{2}\right) \\ \cos\left(\frac{90°}{2}\right)\sin\left(\frac{15°}{2}\right) \\ \sin\left(\frac{90°}{2}\right)\sin\left(\frac{15°}{2}\right) \\ \sin\left(\frac{90°}{2}\right)\cos\left(\frac{15°}{2}\right) \end{bmatrix} \tag{8.8} $$

The 5° difference on roll is thus the same as in the previous example, requiring the body to tilt to the right. The two quaternion errors are computed as:

$$ {}^{\mathrm{K}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} = \begin{bmatrix} \cos\left(\frac{5°}{2}\right) & 0 & -\sin\left(\frac{5°}{2}\right) & 0 \end{bmatrix}^{\mathsf{T}} \tag{8.9} $$

$$ {}^{\mathrm{B}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{5°}{2}\right) & -\sin\left(\frac{5°}{2}\right) & 0 & 0 \end{bmatrix}^{\mathsf{T}} \tag{8.10} $$

Note how the quaternion error defined in the body frame is exactly the same as before, since it is the same motion that is required to correct the 5° difference on roll. However, the quaternion error in inertial frame is now very different since the 5° difference on roll requires a correction by rotating 5° around the inertial frame y-axis-

The quaternion error defined in the body frame is more intuitive and useful for controller design, since the error rotation axis will be referenced to the body in where the actuation occur. The quaternion error hereby become independent of the current heading, and only depends on the orientation error, even making it possible to do linear controller design. It is therefore decided to use the body quaternion error from (8.3).

8.1.1 Shortest rotation

The above example deals only with a small error, which is expected to be the case since the error of roll and pitch has to be close to zero since the ballbot would otherwise accelerate quickly and likely fall over. However, the heading/yaw error can possibly be up to 180°. In a linear heading controller one would usually include a wrapping mechanism as shown in (8.11), which wraps the yaw error between $-\pi$ to π.

$$\begin{aligned} \psi_e &= \psi - \psi_{\text{ref}} \\ \psi_{e,\text{wrapped}} &= \big((\psi_e + \pi) \bmod 2\pi\big) - \pi \end{aligned} \tag{8.11}$$

Such wrapping results in a discontinuity at $\psi_e = 180°$ where the heading direction changes from turning in one direction to another.
The quaternion error includes this wrapping mechanism internally as part of the scalar value, and when doing control with quaternions one can avoid the discontinuity if the error quaternion is handled properly. From the definition of the quaternion, (H.36), the scalar part of the error quaternion corresponds to the cosine of half the error angle, θ_e.

$$q_{e,0} = \cos\left(\frac{\theta_e}{2}\right) \tag{8.12}$$

The scalar value will therefore be unique for error angles up till 360°, which allows the scalar part to be used as an indicator of whether the error quaternion defines a short or long rotation. If the scalar part is positive, the error quaternion defines the short rotation from the current orientation to the desired, $\theta_e < 180°$. If the scalar part is negative, the error quaternion defines a long rotation, $\theta_e > 180°$.

In case of control applications one would always want the short rotation error quaternion, to get the shortest distance to the desired orientation. In case the scalar value is negative, all elements of the quaternion can just be negated, which results in the same rotation but now as the short distance.

8.2 Sliding surfaces

The vector part of the error quaternion defines the rotation axis with the elements ${}^{\text{B}}e_x$, ${}^{\text{B}}e_y$ and ${}^{\text{B}}e_z$ and a magnitude corresponding to the sine of half the error angle.

$${}^{\text{B}}\vec{\boldsymbol{q}}_e = \begin{bmatrix} q_{e,1} \\ q_{e,2} \\ q_{e,3} \end{bmatrix} = \sin\left(\frac{\theta_e}{2}\right) \begin{bmatrix} {}^{\text{B}}e_x \\ {}^{\text{B}}e_y \\ {}^{\text{B}}e_z \end{bmatrix} \tag{8.13}$$

The rotation axis defines the axis around which the body should be rotated to go from the current orientation to the desired orientation. There is thus an obvious relationship between the desired angular velocity and the rotation axis of the error quaternion.

$$ {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}} \propto -\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.14} $$

The error vector, ${}^{\mathrm{B}}\vec{\boldsymbol{q}}_e$, is therefore useful for control, since the vector direction depicts the axis around which a correction should occur and the magnitude will converge to zero when the orientation is aligned.

8.2.1 Angular velocity surface

The first suggestion of a sliding surface, inspired from the work by Wisniewski [62], employs this relationship between the desired angular velocity and the rotation axis of the error quaternion.

$$ \boldsymbol{s} = {}^{\mathrm{B}}\boldsymbol{\omega}_e + \boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.15} $$

Here the desired angular velocity is replaced with the angular velocity error such as to include an angular velocity reference, ${}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} = {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\tilde{\mathrm{B}}\leftarrow\mathrm{K}}$.

$$ \begin{aligned} {}^{\mathrm{B}}\boldsymbol{\omega}_e &= {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}} - {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} \\ &= 2 \vee {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} - {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} \end{aligned} \tag{8.16} $$

It is crucial that the angular velocity error is defined in the same frame as the quaternion error, in this case chosen as the body frame. Since the system states include the quaternion derivative, ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$, and not the angular velocity, ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}$, the angular velocity is computed from the quaternion derivative with the relationship from (H.51). Substituting into (8.16) yields

$$ \boldsymbol{s} = 2 \vee \left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) - {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} + \boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.17} $$

The reader should note that the frame in which the angular velocity reference is defined, $\{\tilde{B}\}$, do not match with the frame of the angular velocity of the body, $\{B\}$. Since the body frame is supposed to track the reference frame, the difference between defining the angular velocity reference in the current frame and in the reference frame is assumed to be negligible.

When the system slides on this surface, $\boldsymbol{s} = 0$, the orientation error will be governed by the following dynamics

$$ {}^{\mathrm{B}}\boldsymbol{\omega}_e = -\boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.18} $$

which states that the correctional angular velocity, ${}^{\mathrm{B}}\boldsymbol{\omega}_e$, is proportional to the negative direction of how much the current orientation is away from the desired. This dynamics will thus move the quaternion towards the reference by the shortest possible way. As long as $\boldsymbol{K}$ is positive definite, the dynamics is globally asymptotically stable and the system will thus converge to the reference.

$$ \lim_{t\to\infty} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}(t) = {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \tag{8.19} $$

The eigenvalues of $\boldsymbol{K}$ defines the poles of the reduced-order dynamics and since efficient tracking is desired, the poles should all be located on the negative real axis. Thus no complex-conjugated pole pairs. This requires a diagonal gain matrix.

8.2.2 Quaternion derivative surface

Another even simpler sliding surface takes in the quaternion derivative directly and follows the suggested design procedure by Slotine [60] to use the error and error derivative. However, since the degrees of freedom in the orientation and actuation is only three, a similar approach to using the vector part, is employed. Though instead of seeing the rotation axis in the quaternion error as the axis around which a correction should happen, the second sliding surface implements a linear proportional controller on the vector part.

$$\boldsymbol{s} = {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e + \boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.20}$$

When the system slides on the surface, $s = 0$, the orientation error will thus be governed by the following dynamics

$${}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e = -\boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.21}$$

and the sliding manifold will hence drive the vector part of the quaternion error to zero, which results in a unit quaternion.

$$\lim_{t\to\infty} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e(t) = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^{\mathsf{T}} \text{ or } \begin{bmatrix}-1 & 0 & 0 & 0\end{bmatrix}^{\mathsf{T}} \tag{8.22}$$

Since the error quaternion, ${}^{\mathrm{B}}\vec{\boldsymbol{q}}_e$, is constrained on the four-dimensional unit sphere, both the scalar value and its derivative is automatically defined from the constraint.

The derivative of the quaternion error is derived using the principles shown in Appendix H.7.

$${}^{\mathrm{B}}\dot{\boldsymbol{q}}_e = {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\dot{\boldsymbol{q}}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} + {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{8.23}$$

We insert the definition of the quaternion derivative reference, based on the angular velocity reference defined in the body reference frame, ${}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} = \wedge\, {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} = \wedge\, {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\tilde{\mathrm{B}}\leftarrow\mathrm{K}}$.

$${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\dot{\boldsymbol{q}}_{\mathrm{ref}} = \frac{1}{2}\,{}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \tag{8.24}$$

The derivative of the quaternion error is thus

$$\begin{aligned}
{}^{\mathrm{B}}\dot{\boldsymbol{q}}_e &= \frac{1}{2}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}\right)^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} + {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\
&= -\frac{1}{2}\,{}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \circ \underbrace{{}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}_{{}^{\mathrm{B}}\boldsymbol{q}_e} + {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}
\end{aligned} \tag{8.25}$$

Thereby the sliding surface is expanded to

$$\boldsymbol{s} = -\frac{1}{2} \vee {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} + \vee\, {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} + \boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{8.26}$$

By some reorganization the vector part of the quaternion derivative of the system is governed by the following dynamics in the sliding phase:

$${}^{\mathrm{K}}_{\mathrm{B}}\dot{\vec{\boldsymbol{q}}} = \frac{1}{2} \vee \left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \circ {}^{\mathrm{B}}\boldsymbol{q}_e\right) - \vee \left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ \wedge \boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e\right) \tag{8.27}$$

Note the similarities with (8.21) in terms of $-\boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e$ and how the dynamics collapses into (8.28) for a unit quaternion reference and zero angular velocity reference.

$${}^{\mathrm{K}}_{\mathrm{B}}\dot{\vec{\boldsymbol{q}}} = -\boldsymbol{K}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e = -\boldsymbol{K}\,{}^{\mathrm{K}}_{\mathrm{B}}\vec{\boldsymbol{q}} \tag{8.28}$$

Again the eigenvalues of $\boldsymbol{K}$ defines the poles of the reduced-order dynamics, so as long as $\boldsymbol{K}$ is positive definite the dynamics is globally asymptotically stable and the system will converge to the reference. It is similarly recommended to use a diagonal gain matrix.

Note how only the vector part of the quaternion error is used in both of the suggested sliding surfaces. This reduces the order of the differential equations, used in the sliding mode controller design, from four to three, thereby making the system fully actuated and invertible. This furthermore allows for equivalent control design, see Section 9.1, which requires the input matrix to be invertible.

8.3 Comparison

The two sliding surfaces looks very similar and behaves very similarly. Since the error will be close to zero most of the time, the vector part of the quaternion error derivative, ${}^{\text{B}}\dot{\vec{\boldsymbol{q}}}_e$, will be almost equivalent to half of the angular velocity error, ${}^{\text{B}}\boldsymbol{\omega}_e$. To investigate the differences in sliding dynamics, thus the dynamics when the system is on the sliding surface, $s = 0$, a MATLAB simulation is carried out using aggressive gains from Section 12.1. To make the simulation comparable the gains used in the angular velocity sliding surface simulation is scaled with a factor of 2. The quaternion reference is set to the unit quaternion and the angular velocity reference is set to zero. To test the global stability of the manifolds the system is tested with two different initial error quaternions and the error quaternion is computed as both the long and short rotation, see Section 8.1.1.

The first simulation, shown in Figure 8.1, is a comparison between computing the quaternion error as a short or long rotation, with an initial rotation error set to $\phi = 30°$, $\theta = 30°$, $\psi = 30°$. The error is thus fairly small, corresponding to a $\theta_e \approx 46°$. The long rotation is computed by negating all elements. Note how the angular velocity manifold do not diverge but takes the long rotation to reach the reference.

In the second simulation, shown in Figure 8.2, a larger error of $\phi = 30°$, $\theta = 30°$, $\psi = 180°$ is tested. As expected the dynamics of the two sliding surfaces are almost identical when the error is small. Even at large errors the dynamics are almost identical, though the angular velocity surface seem to have a slightly slower response and takes a different path to the reference.

The biggest problem and difference between the two, is how the long error quaternion is not handled automatically with the angular velocity manifold, leading to a detour where the system rotates the long way to reach the reference. The quaternion derivative manifold do not care about the scalar value and thus always takes the shortest rotation.

It is therefore decided to use the quaternion derivative, both due to the simulation results above, but also due to the simplicity in the sliding surface definition.

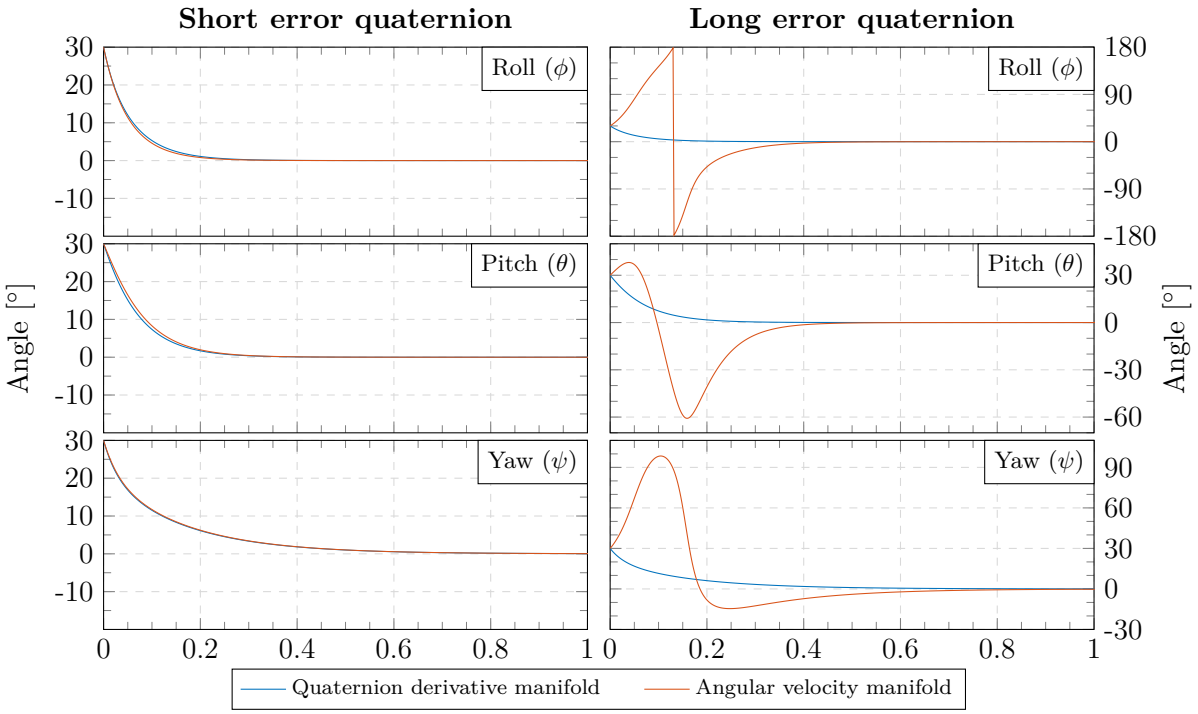


Figure 8.1: Comparison of manifold dynamics when using a short or long error quaternion. Initial error is set to $\phi = 30°$, $\theta = 30°$, $\psi = 30°$.

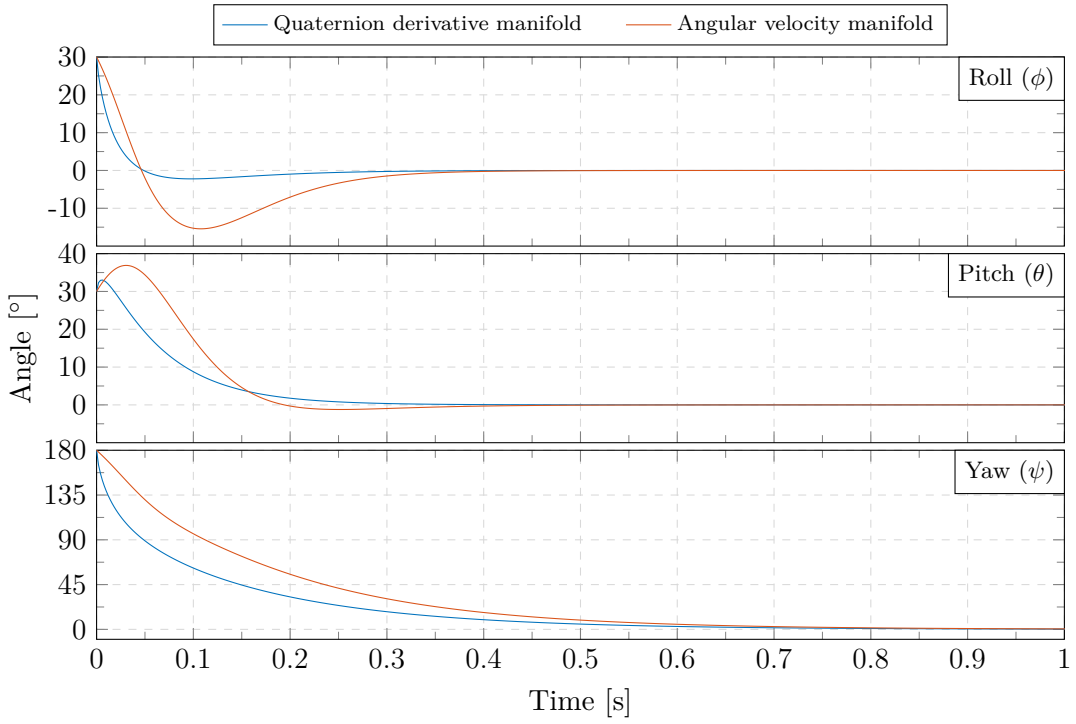


Figure 8.2: Comparison of manifold dynamics with a large initial error of $\phi = 30°$, $\theta = 30°$, $\psi = 180°$.

9 Controller Laws

Both continuous and discrete versions of the sliding mode controller exists, but common practice is to design and implement the continuous version of the controller and run it at a high sample rate [60]. Therefore, the continuous version is only considered in this thesis.

In this thesis no specific uncertainty bounds are defined and the gain parameters will be used as tuning parameters, described in Appendix O. Step 5 of the summarized sliding mode design guidelines in Appendix L.1, will therefore be based on tuning through practical experiments rather than requirements.

The sliding mode control law consists of both an equivalent control part and a switching part.

$$\boldsymbol{\tau}_m = \boldsymbol{\tau}_{\text{equiv}} + \boldsymbol{\tau}_{\text{switch}} \tag{9.1}$$

The equivalent control part cancels out the nominal dynamics of the system while the switching part ensures that the system stays on the sliding surface when disturbances or uncertainties are present. As said in the beginning of Chapter 8 one would usually transform the system into normal form, but in the design of this balance controller it is decided to neglect the translational dynamics and only use the quaternion dynamics. With just the quaternion part of the ODE, extracted in (9.2), the dynamics become fully actuated.

$$\begin{aligned} {}^{\text{K}}_{\text{B}}\ddot{\boldsymbol{q}} &= \boldsymbol{f}_q(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) + \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})\,\boldsymbol{\tau}_m \\ &= \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) + \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})\,\boldsymbol{\tau}_m \end{aligned} \tag{9.2}$$

where $\boldsymbol{f}_q$ and $\boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})$ are extracted from (5.98)

$$\begin{aligned} \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) &= \begin{bmatrix} \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{bmatrix} \boldsymbol{f}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) \\ \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q}) &= \begin{bmatrix} \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{bmatrix} \boldsymbol{g}(\boldsymbol{\chi}) \end{aligned} \tag{9.3}$$

In the following section the two parts of the control law are derived for the two sliding surfaces suggested in Section 8.2. Finally, the stability of the system is proven for both the reaching and sliding phase using the Lyapunov stability criterion.

9.1 Equivalent control laws

Part of the sliding mode controller is to derive an equivalent control law to ensure that a nominal system will keep sliding on the surface, $\dot{s} = 0$, after it has been reached, $s = 0$. One of the most commonly used approaches is computed-torque also known as inverse dynamics, which involves similar steps as Input-State linearization and Feedback linearization. A computed-torque controller requires the system to be on the form shown in (9.4), with an invertible input matrix, $\boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}})$. It thus only works for fully actuated systems. Otherwise the system has to be converted into normal form.

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}})\,\boldsymbol{u} \tag{9.4}$$

Luckily, as stated in the end of Appendix L, by extracting just the quaternion part of the non-linear ODE, (9.2), and combining it with the choice of using just the vector part of the quaternion in the sliding surface, makes the system fully actuated.

$$ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} = \boldsymbol{f}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) + \boldsymbol{g}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})\,\boldsymbol{\tau}_m \tag{9.5} $$

where $\boldsymbol{f}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) \in \mathbb{R}^4$ and $\boldsymbol{g}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) \in \mathbb{R}^{4\times 3}$.
The equivalent control law is derived by substituting this reduced-order dynamics into the derivative of the sliding surface, which should be kept at zero when sliding.

9.1.1 Angular velocity surface

The derivative of the angular velocity sliding surface is

$$ \dot{\boldsymbol{s}} = {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e + \boldsymbol{K}\,{}^{\mathrm{B}}\dot{\tilde{\boldsymbol{q}}}_e \tag{9.6} $$

The angular velocity error is defined in (8.16). The derivative of this error is

$$ \begin{aligned} {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e &= {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} - {}^{\tilde{\mathrm{B}}}\dot{\boldsymbol{\omega}}_{\mathrm{ref}} \\ &= 2 \vee \left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} + {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} \right) - {}^{\tilde{\mathrm{B}}}\dot{\boldsymbol{\omega}}_{\mathrm{ref}} \\ &= 2 \vee {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} - {}^{\tilde{\mathrm{B}}}\dot{\boldsymbol{\omega}}_{\mathrm{ref}} \end{aligned} \tag{9.7} $$

Note how the derivative is simplified since $\vee \left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) = \boldsymbol{0}_{3\times 1}$.

The angular velocity reference is assumed to be slowly varying which is why the derivative, ${}^{\tilde{\mathrm{B}}}\dot{\boldsymbol{\omega}}_{\mathrm{ref}}$, is neglected. The derivative of the quaternion error, derived in (8.25), is substituted into the sliding surface derivative along with the derivative of the angular velocity error.

$$ \dot{\boldsymbol{s}} = 2 \vee {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} - \boldsymbol{K} \vee \left(\frac{1}{2}\, {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} - {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) \tag{9.8} $$

The quaternion dynamics, (9.5), is substituted into the sliding surface derivative equation and the quaternion multiplications are replaced with the matrix operators.

$$ \begin{aligned} \dot{\boldsymbol{s}} = {} & 2 \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} \left(\boldsymbol{f}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) + \boldsymbol{g}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})\,\boldsymbol{\tau}_m \right) \\ & - \frac{1}{2}\boldsymbol{K} \vee \boldsymbol{\Phi}\left({}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \right) \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\ & + \boldsymbol{K} \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{aligned} \tag{9.9} $$

Finally, the equation is reorganized to isolate $\boldsymbol{\tau}_{\mathrm{equiv}} = \boldsymbol{\tau}_m$ for keeping $\dot{\boldsymbol{s}} = 0$ with the nominal model.

$$ \begin{aligned} 2 \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} \boldsymbol{g}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})\,\boldsymbol{\tau}_m = {} & - 2 \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} \boldsymbol{f}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) \\ & + \frac{1}{2}\boldsymbol{K} \vee \boldsymbol{\Phi}\left({}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \right) \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\ & - \boldsymbol{K} \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{aligned} \tag{9.10} $$

Due to the choice of using the reduced-order dynamics and the definition of a three-dimensional sliding surface, matching the dimension of the actuation-space, (9.10) $\boldsymbol{\tau}_m$ can be isolated since $\tilde{\boldsymbol{g}}$ is invertible.

$$ \tilde{\boldsymbol{g}}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}) = 2 \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} \boldsymbol{g}_q({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) \tag{9.11} $$

Thereby the equivalent torque, and thus the equivalent control law for the angular velocity sliding surface, is shown in (9.12).

$$\begin{aligned}\boldsymbol{\tau}_{\text{equiv}} = \tilde{\boldsymbol{g}}^{-1}({}^{\text{K}}_{\text{B}}\boldsymbol{q}) \Bigg(&- 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\text{B}}\boldsymbol{q}\Big)^{\mathsf{T}} \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) \\ &+ \frac{1}{2}\boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\tilde{\text{B}}}\tilde{\boldsymbol{\omega}}_{\text{ref}}\Big)\, \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\boldsymbol{q} \\ &- \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} \Bigg)\end{aligned} \tag{9.12}$$

9.1.2 Quaternion derivative surface

Conversely the derivative of the quaternion derivative sliding surface is

$$\dot{\boldsymbol{s}} = {}^{\text{B}}\ddot{\vec{\boldsymbol{q}}}_e + \boldsymbol{K}\, {}^{\text{B}}\dot{\vec{\boldsymbol{q}}}_e \tag{9.13}$$

The derivative of the quaternion error was derived in (8.23). Taking the time derivative again while neglecting ${}^{\text{K}}_{\tilde{\text{B}}}\ddot{\boldsymbol{q}}_{\text{ref}}$, since the angular velocity reference is assumed to be slowly varying, yields the second derivative of the quaternion error.

$${}^{\text{B}}\ddot{\boldsymbol{q}}_e = 2\, {}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}^*_{\text{ref}} \circ {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} + {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}^*_{\text{ref}} \circ {}^{\text{K}}_{\text{B}}\ddot{\boldsymbol{q}} \tag{9.14}$$

This second derivative from (9.14) is substituted back into the definition of the sliding surface derivative together with the definition from (8.23).

$$\begin{aligned}\dot{\boldsymbol{s}} = &\vee \Big(2\, {}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}^*_{\text{ref}} \circ {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} + {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}^*_{\text{ref}} \circ {}^{\text{K}}_{\text{B}}\ddot{\boldsymbol{q}}\Big) \\ &+ \boldsymbol{K} \vee \Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}^*_{\text{ref}} \circ {}^{\text{K}}_{\text{B}}\boldsymbol{q} + {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}^*_{\text{ref}} \circ {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}\Big)\end{aligned} \tag{9.15}$$

The quaternion dynamics, (9.5), is substituted into the sliding surface derivative equation and the quaternion multiplications are replaced with the matrix operators.

$$\begin{aligned}\dot{\boldsymbol{s}} = &\, 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} \\ &+ \vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \Big(\boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) + \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})\, \boldsymbol{\tau}_m\Big) \\ &+ \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\boldsymbol{q} \\ &+ \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}\end{aligned} \tag{9.16}$$

Finally, the equation is reorganized to isolate $\boldsymbol{\tau}_{\text{equiv}} = \boldsymbol{\tau}_m$ for keeping $\dot{\boldsymbol{s}} = 0$ with the nominal model.

$$\begin{aligned}\vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})\, \boldsymbol{\tau}_m = &- 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} \\ &- \vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) \\ &- \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\boldsymbol{q} \\ &- \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}\end{aligned} \tag{9.17}$$

This equation is inverted using the matrix

$$\tilde{\boldsymbol{g}}({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}) = \vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q}) \tag{9.18}$$

Thereby the equivalent torque, and thus the equivalent control law for the quaternion derivative sliding surface, is shown in (9.19).

$$\begin{aligned}\boldsymbol{\tau}_{\text{equiv}} = \tilde{\boldsymbol{g}}^{-1}({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}) \Bigg(&- 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} \\ &- \vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) \\ &- \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\boldsymbol{q} \\ &- \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} \Bigg)\end{aligned} \tag{9.19}$$

9.1.3 Comparison

The two sliding surfaces appeared very similar in simulation in Section 8.3 and ends up with very similar sliding surface derivatives. To compare the derivatives the quaternion derivative reference definition in the third line of (9.16) is replaced with the angular velocity reference definition. The derivative of the angular velocity surface is shown in (9.20) and the derivative of the quaternion derivative surface is shown in (9.21).

$$\begin{aligned}\dot{\boldsymbol{s}} = {} & 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\text{B}}\boldsymbol{q}\Big)^{\mathsf{T}} \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) \\ & + 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\text{B}}\boldsymbol{q}\Big)^{\mathsf{T}} \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})\, \boldsymbol{\tau}_m \\ & - \frac{1}{2}\boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\tilde{\text{B}}}\tilde{\boldsymbol{\omega}}_{\text{ref}}\Big)\, \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\boldsymbol{q} \\ & + \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}\end{aligned} \tag{9.20}$$

$$\begin{aligned}\dot{\boldsymbol{s}} = {} & 2 \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\dot{\boldsymbol{q}}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} \\ & + \vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \boldsymbol{f}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q},\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}},\, {}^{\text{I}}\dot{x},\, {}^{\text{I}}\dot{y}) \\ & + \vee\boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}} \boldsymbol{g}_q({}^{\text{K}}_{\text{B}}\boldsymbol{q})\, \boldsymbol{\tau}_m \\ & - \frac{1}{2}\boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\tilde{\text{B}}}\tilde{\boldsymbol{\omega}}_{\text{ref}}\Big)\, \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\boldsymbol{q} \\ & + \boldsymbol{K} \vee \boldsymbol{\Phi}\Big({}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}\Big)^{\mathsf{T}}\, {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}}\end{aligned} \tag{9.21}$$

The reader should note how similar the two derivatives are. Actually the main difference is related to how the dynamics of the system enters the derivative with a difference in gain by a factor of 2.

With the angular velocity surface, the dynamics are pre-multiplied with the current quaternion, whereof they are pre-multiplied with the quaternion reference within the quaternion derivative surface. For a constant quaternion reference the quaternion derivative surface will be less affected than the angular velocity surface, where the system dynamics is pre-multiplied with the varying quaternion.
As shown in Section 8.3 the difference was very minimal, but the simulation also seemed to show some extra dynamics in terms of an initial transient period when using the angular velocity surface which was not apparent on the quaternion derivative surface.

9.2 Switching control laws

With just the equivalent control law disturbances will cause the system to move away from the sliding surface. The principle of a sliding mode controller is the switching control law that enforces the system to stay on the sliding surface even when matched disturbances are present. A matched disturbance, $\boldsymbol{\delta}(t, \boldsymbol{x}, \boldsymbol{u})$, see (L.1), is added to the surface derivatives, (9.9) and (9.16), and grouped into a disturbance term, $\boldsymbol{\Delta}$. The input torque, $\boldsymbol{\tau}_m$, is replaced with the joint control

torque defined in (9.1) including both the equivalent torque, $\boldsymbol{\tau}_{\text{equiv}}$, and the switching torque, $\boldsymbol{\tau}_{\text{switch}}$. The resulting sliding surface derivative is shown in (9.22).

$$\dot{\boldsymbol{s}} = \underbrace{\tilde{\boldsymbol{g}}({}^{\text{K}}_{\text{B}}\boldsymbol{q}, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}})\, \boldsymbol{\delta}(t, \boldsymbol{x}, \boldsymbol{u})}_{\boldsymbol{\Delta}} + \tilde{\boldsymbol{g}}({}^{\text{K}}_{\text{B}}\boldsymbol{q}, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}})\, \boldsymbol{\tau}_{\text{switch}} \tag{9.22}$$

To stay on the surface, $\dot{\boldsymbol{s}} = 0$, should still be satisfied. This requires the switching torque to exactly cancel out the disturbance.

$$\boldsymbol{\tau}_{\text{switch}} = -\tilde{\boldsymbol{g}}^{-1}({}^{\text{K}}_{\text{B}}\boldsymbol{q}, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}})\, \boldsymbol{\Delta} \tag{9.23}$$

This will, however, require exact estimation of the disturbance. Instead the following switching control law is proposed.

$$\boldsymbol{\tau}_{\text{switch}} = -\tilde{\boldsymbol{g}}^{-1}({}^{\text{K}}_{\text{B}}\boldsymbol{q}, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}})\, \bar{\boldsymbol{\eta}}\, \text{sgn}(\boldsymbol{s}) \tag{9.24}$$

where $\bar{\boldsymbol{\eta}}$ is a diagonal matrix of the switching gains $\boldsymbol{\eta}$, and the signum function, $\text{sgn}(\boldsymbol{s})$, is an elementwise function defined as

$$\text{sgn}(s) = \begin{cases} +1 & s > 0 \\ 0 & s = 0 \\ -1 & s < 0 \end{cases} \tag{9.25}$$

Inserting this into the definition of the sliding surface derivative yields

$$\dot{\boldsymbol{s}} = \begin{cases} \boldsymbol{\Delta} - \boldsymbol{\eta} & s > 0 \\ \boldsymbol{\Delta} & s = 0 \\ \boldsymbol{\Delta} + \boldsymbol{\eta} & s < 0 \end{cases} \tag{9.26}$$

where the piecewise definitions are elementwise on the vectors. As long as $\boldsymbol{\eta}$ is chosen larger than the upper bound of the disturbance, $\bar{\boldsymbol{\Delta}} = |\boldsymbol{\Delta}|$, the switching law will always be able to drive the system back onto the sliding surface. Similarly if there is no disturbance and the system is on the sliding surface, $\boldsymbol{\tau}_{\text{switch}} = 0$ and $\dot{\boldsymbol{s}} = 0$. For a fast switching controller, $\boldsymbol{\tau}_{\text{switch}}$ will thereby on average be equivalent to (9.23).

9.2.1 Continuous switching law

Since we will be dealing with a mechanical system involving motors and gears with backlash, the discontinuous switching law in (9.23) would result in chattering and thus undesirable vibrations in the system. Furthermore, the large and quick changes in torque are also more likely to make the omniwheels slip on the ball, voiding the assumption of no slip. Instead one can use a continuous switching law, see (L.6), as proposed in [60] [61]. The most common continuous switching law replaces the signum function by the saturation function, making the switching law linear in an ϵ-tube around the origin.

$$\boldsymbol{\tau}_{\text{switch}} = -\tilde{\boldsymbol{g}}^{-1}({}^{\text{K}}_{\text{B}}\boldsymbol{q}, {}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}})\, \bar{\boldsymbol{\eta}}\, \text{sat}\left(\frac{\boldsymbol{s}}{\boldsymbol{\epsilon}}\right) \tag{9.27}$$

where $\bar{\boldsymbol{\eta}}$ is still a diagonal matrix of the switching gains $\boldsymbol{\eta}$ and $\boldsymbol{\epsilon}$ is a vector defining the ϵ-tube for each sliding variable. Note that the fraction $\frac{\boldsymbol{s}}{\boldsymbol{\epsilon}}$ is carried out elementwise. The saturation function, $\text{sat}(\boldsymbol{s})$, is an elementwise function defined as

$$\text{sat}(x) = \begin{cases} x & |x| \leq 1 \\ \text{sgn}(x) & |x| > 1 \end{cases} \tag{9.28}$$

The switching law in (9.27) is used as the final switching law.

9.3 Stability proof

Proving stability of a sliding mode controlled system requires proof of stability of the two phases respectively. Stability can be proven using the Lyapunov stability theorem [61] stating that a system has an asymptotically stable equilibrium point at the origin, $\boldsymbol{x} = \mathbf{0}$, if there exists a function, $V(\boldsymbol{x})$, such that

$$\begin{aligned} V &\in \mathbb{C}^1 \\ V(\boldsymbol{x}) &> 0, \quad \forall \boldsymbol{x} \neq 0 \\ V(\mathbf{0}) &= 0 \\ \dot{V}(\boldsymbol{x}) &< 0, \quad \forall \boldsymbol{x} \neq 0 \\ \lim_{|\boldsymbol{x}| \to \infty} V(\boldsymbol{x}) &= \infty \end{aligned} \tag{9.29}$$

is satisfied within a bounded set Ω containing the origin.

9.3.1 Reaching phase

The reaching phase is commonly proven with a Lyapunov Candidate Function, LCF, constructed from the sliding manifold.

$$V = \frac{1}{2}\boldsymbol{s}^\mathsf{T}\boldsymbol{s} \tag{9.30}$$

This function fulfils the first two criteria, $V \in \mathbb{C}^1$ and $V(\boldsymbol{x}) > 0$. The derivative of this function involves the derivative of the sliding surface.

The collapsed derivative of the sliding surface, (9.26), which includes the joint control law, is inserted.

$$\begin{aligned} \dot{V} &= \boldsymbol{s}^\mathsf{T}\dot{\boldsymbol{s}} \\ &= \boldsymbol{s}^\mathsf{T}\boldsymbol{\Delta} - \left|\boldsymbol{s}^\mathsf{T}\right|\boldsymbol{\eta} \\ &\leq \left|\boldsymbol{s}^\mathsf{T}\right|\left(\bar{\boldsymbol{\Delta}} - \boldsymbol{\eta}\right) \end{aligned} \tag{9.31}$$

where $\left|\boldsymbol{s}^\mathsf{T}\right|$ is the elementwise absolute values of the sliding variables. As long as $\bar{\boldsymbol{\Delta}} - \boldsymbol{\eta} < 0 \rightarrow \boldsymbol{\eta} > \bar{\boldsymbol{\Delta}}$, that is the switching gains are elementwise larger than the upper bound of the disturbance, the system is proven asymptotically stable. Since no assumptions were made on the set it can be chosen as $\Omega = \mathbb{R}^{12}$, making the system globally asymptotically stable.

9.3.2 Sliding phase with angular velocity surface

When the system is in the sliding phase we know that $s = 0$. Hence to prove stability in the sliding phase we only need to verify stability of the reduced-order dynamics, defined by the sliding manifold.
The stability during sliding on the angular velocity surface, (8.15), is shown by proposing a Lyapunov candidate function. Since the quaternion is constrained it would be sufficient to include the vector part in the Lyapunov candidate function.

$$V = \frac{1}{2}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^\mathsf{T}\,{}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{9.32}$$

The derivative is

$$\dot{V} = {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^\mathsf{T}\,{}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e \tag{9.33}$$

Inserting the kinematic relationship for the derivative of the quaternion error based on the angular velocity error

$$\dot{V} = \frac{1}{2} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} \left(\vee {}^{\mathrm{B}}\boldsymbol{q}_e \circ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e \right) \tag{9.34}$$

The dynamics for the angular velocity sliding surface, (8.18), can now be inserted.

$$\begin{aligned} \dot{V} &= \frac{1}{2} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} \left(\vee {}^{\mathrm{B}}\boldsymbol{q}_e \circ \left(\wedge - \boldsymbol{K} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \right) \right) \\ &= -\frac{1}{2} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} \vee \boldsymbol{\Phi}\left({}^{\mathrm{B}}\boldsymbol{q}_e \right) \wedge \boldsymbol{K} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \end{aligned} \tag{9.35}$$

For a diagonal gain matrix, $\boldsymbol{K}$, the derivative simplifies to

$$\dot{V} = -\frac{1}{2} \left(K_x {}^{\mathrm{B}}q_{e,0} {}^{\mathrm{B}}q_{e,1}^2 + K_y {}^{\mathrm{B}}q_{e,0} {}^{\mathrm{B}}q_{e,2}^2 + K_z {}^{\mathrm{B}}q_{e,0} {}^{\mathrm{B}}q_{e,3}^2 \right) \tag{9.36}$$

So as long as the diagonal elements of the gain matrix are positive, $\boldsymbol{K} > 0$, and the scalar value of the quaternion error is positive, ${}^{\mathrm{B}}q_{e,0} > 0$, the system will be asymptotically stable while sliding. This requires that the short rotation error quaternion, see Section 8.1.1, is always used. However, this requirement is only a sufficient but not necessary condition, most likely due to the Lyapunov candidate function not capturing the unit norm constraint of the quaternion. It has been verified through simulation, see Section 8.3, that the angular velocity sliding surface works with both the long and short quaternion error.

9.3.3 Sliding phase with quaternion derivative surface

The stability of the quaternion derivative surface, (8.20), is slightly simpler to prove. Since it is a linear sliding surface the system will be globally asymptotically stable as long as stability is proven for the reaching phase. It is easily shown using the same Lyapunov candidate function as before.

$$V = \frac{1}{2} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \tag{9.37}$$

with the derivative being

$$\dot{V} = {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e \tag{9.38}$$

The dynamics for the quaternion derivative surface, (8.21), is inserted.

$$\begin{aligned} \dot{V} &= {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} \left(-\boldsymbol{K} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \right) \\ &= - {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e^{\mathsf{T}} \boldsymbol{K} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \end{aligned} \tag{9.39}$$

So as long as $\boldsymbol{K}$ is positive definite the system is asymptotically stable. Note that the system will even be globally asymptotically stable, since there is no requirement to the scalar value of the error quaternion. The quaternion derivative sliding surface therefore works with both long and short quaternion errors.

9.4 Controller summary

A non-linear sliding mode controller has been designed to stabilize the rotational dynamics of the ballbot. The control objective is to track a desired quaternion reference and angular velocity reference within a certain operating envelope. The controller design was simplified by neglecting the translational dynamics of the non-linear ODE. This made the system rectangular and thus invertible, thereby allowing inverse dynamics to be applied as equivalent control to cancel the nominal dynamics.

Two sliding surfaces were proposed, relating the quaternion tracking error in the body frame to either the body angular velocity error or to the derivative of the quaternion error. Through simulation and due to simplicity, the sliding surface using the derivative of the quaternion error, was chosen. Next to the equivalent control law a continuous switching law, using the saturation function, was designed to reject matched disturbances. Lastly the stability of the derived controller was proven with two Lyapunov candidate functions, one for the reaching phase and one for the sliding phase.

The next step is to implement the derived controller and test it on the real system.

10 Balance controller simulation

In the following chapter the sliding mode balance controller is simulated and the non-linear model of the ballbot, derived in Chapter 5, is simulated using MATLAB Simulink®. The Simulink model can be found at `Kugle-MATLAB/Simulation/KugleSim_SlidingMode.slx`.

10.1 Simulation diagram

The Simulink model is built as a hierarchy of several referenced models. The top-level model shown in Figure 10.1, follows the structure of the balance controller loop from Figure 7.1 and connects the referenced models together in a control loop.

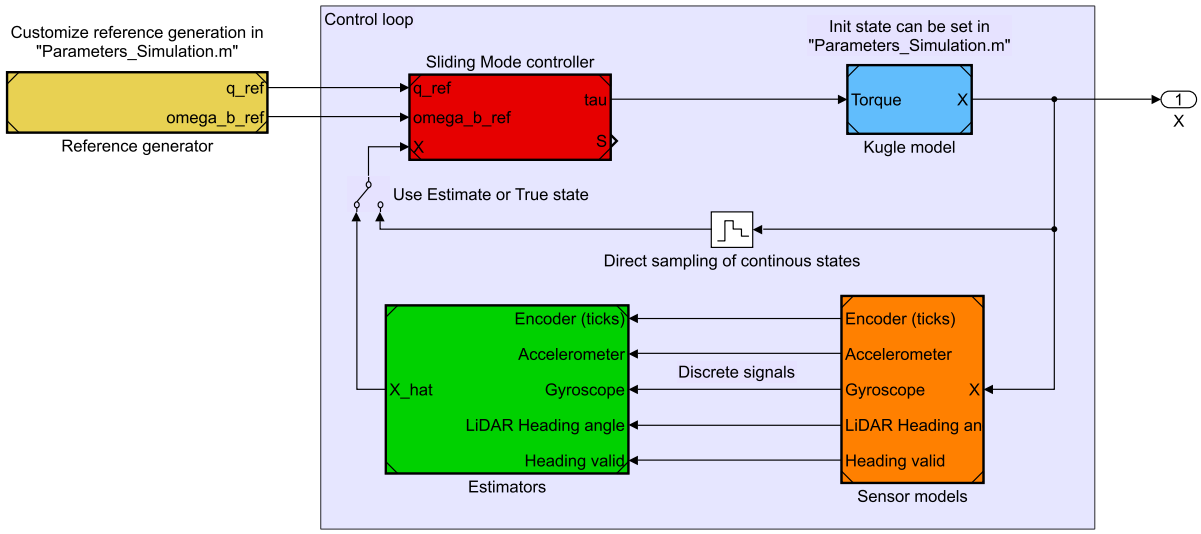


Figure 10.1: Top-level of Simulink model including the ballbot model, sensor models, estimators, sliding mode balance controller and reference generation.

The sensor models are implemented according to the description in Appendix M and Appendix N, and the output is a discrete signal sampled at the sampling rate of $f_s = 200$ Hz. The simulation supports sensor and process noise to be added, but for the simulations in this chapter no noise are added. Furthermore, the accelerometer bias and gyroscope bias are set to zero during simulation. The estimators include the QEKF, Appendix M, for quaternion and quaternion derivative estimation, and the VEKF, Appendix N, for velocity estimation. Furthermore, the estimators include a numerical integration of the kinematics as the position estimate.

All blocks are programmed to automatically load a set of parameters from `Kugle-MATLAB/Parameters` before running the simulation. This enables easy adjustment of simulation parameters without having to dig into specific blocks. As an example the sliding mode controller gains, ϵ-tube and the sliding surface can be changed with the parameters.

Three simulations are carried out, starting with a zero reference simulation, followed by a sine wave reference simulation, next a chirp reference simulation and finally a center of mass misalignment simulation. Furthermore the body mass and inertia is changed in the zero reference simulation to show the influence.

10.2 Simulation parameters

For all simulations carried out in this thesis the following Simulink solver settings are used. Please note that MATLAB 2018a has been used, but later versions are expected to be compatible as well.

- Solver type: Variable-step
- Solver: auto
- Max step size: 1×10^{-2}
- Relative tolerance: 1×10^{-5}
- Min step size: auto
- Absolute tolerance: auto

For the simulations carried out in this chapter the model parameters from Section 2.2 are used, except for the friction values which are set to 0.001 in the simulated model and 0 in the nominal model, used by the sliding mode controller.

The sliding mode controller is configured to use the quaternion derivative manifold and the aggressive set of gains, found through practical tuning on the robot, see Appendix O.1. The aggressive gains are:

$$\begin{aligned} \boldsymbol{K} &= \begin{bmatrix} 15 & 15 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\eta} &= \begin{bmatrix} 6 & 6 & 3 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\epsilon} &= \begin{bmatrix} 0.5 & 0.5 & 0.2 \end{bmatrix}^\mathsf{T} \end{aligned} \tag{10.1}$$

During simulation the output torque is limited to 1.6 N m as reasoned in Appendix O.2, even though the motors are able to deliver up to almost 2 N m as listed in Table 2.1. In every simulation, except for the center of mass influence simulation, the center of mass vector has been aligned with the center of the ball.

$$^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^\mathsf{T} \tag{10.2}$$

Since we are only simulating the balance controller, this alignment limits the translational dynamics to an expected operating range similar to when the velocity or position loop is closed.

10.3 Zero reference with non-zero initial condition

The first simulation tests the balance controller with a non-zero initial condition but with zero references. The quaternion reference is set to the unit quaternion and the angular velocity reference is set to zero. The initial states are set to zero except for the orientation which is set to $\phi = 10°$, $\theta = -10°$ and $\psi = 10°$.

To simulate and verify the robustness against parameter uncertainty, the simulation is run twice. First with the model parameters from Table 2.1 and secondly with a change in the model parameters while the nominal parameters used by the sliding mode controller remain the same. For the changed parameters the body mass, M_b, is increased by 2 kg and the body inertia components, J_{bx}, J_{by} and J_{bz}, are scaled by a factor of 2. The Euler-angle response of the two simulations are shown in Figure 10.2.

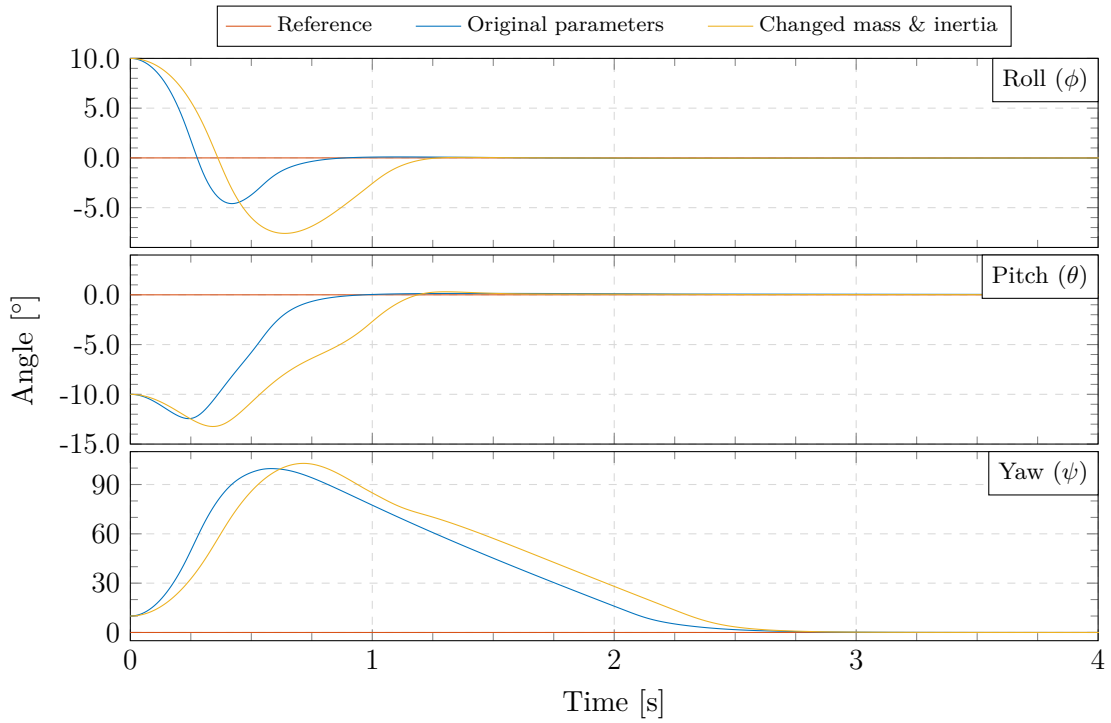


Figure 10.2: Comparison of Euler angles during the zero-reference simulation with non-zero initial conditions using the original parameters or the model with changes mass and inertia.

For both simulations the roll and pitch angles settles quickly to zero, while the yaw angle takes longer to stabilize including a large overshoot to more than 90°. This behaviour can potentially be explained from the smaller gain on yaw and the saturated torque output, see Figure 10.3.

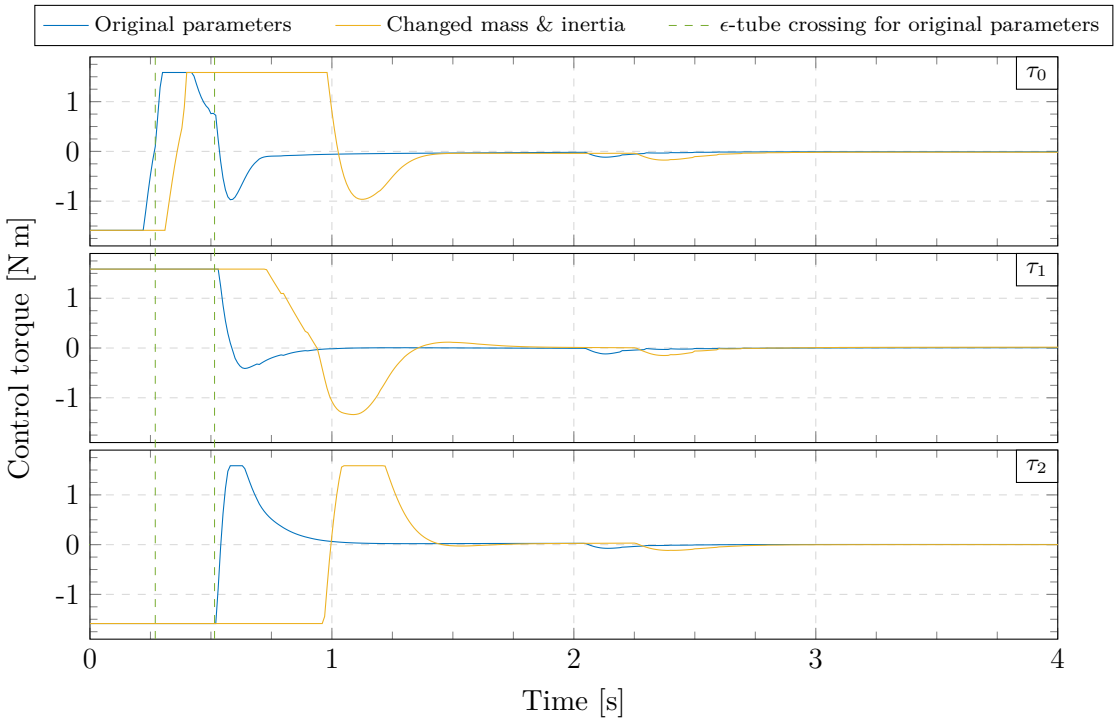


Figure 10.3: Control torque during the zero-reference simulation with non-zero initial conditions. The green vertical lines marks the crossings with the ϵ-tube from the simulation with the original parameters, see Figure 10.4

In the simulation of the model with changed mass and inertia, the torque remains saturated for a much longer time than with the original parameters. This is a consequence of the switching torque which compensates for the changed dynamics. When the saturated torque from Figure 10.3 is compared to the behaviour of the Euler angles in Figure 10.2 it is clear that the initial error on roll and pitch leads to a saturation of the torque, leaving no room to correct the yaw angle. The consequence is that the yaw angle error grows until roll and pitch have almost settled.

The ϵ-tube crossings from the simulation with the original parameters, have been marked with green vertical lines to relate the control torque to the sliding variables shown in Figure 10.4.

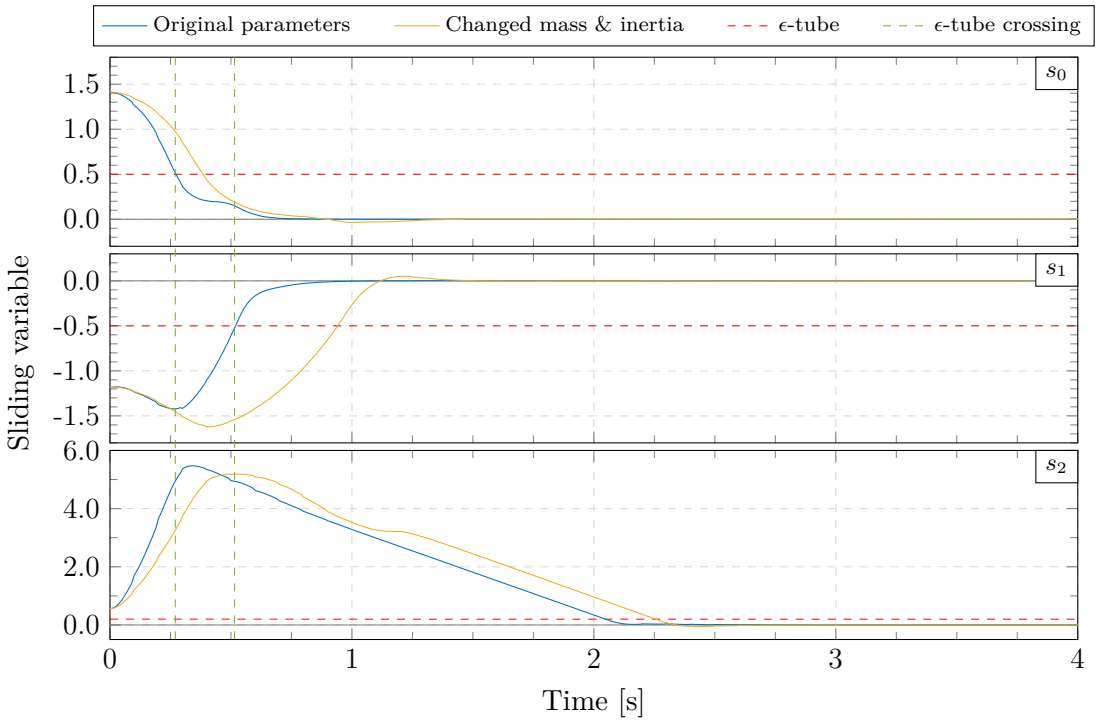


Figure 10.4: Sliding variables during the zero-reference simulation with non-zero initial conditions. The red horizontal lines marks the ϵ-tube and the green vertical lines marks the crossings with the tube from the simulation with the original parameters.

The sliding variable, s_0, seem to dominate the dynamics initially, leading to a counter-intuitive increase in both s_1 and s_2. The disturbance, being the initial condition, is thus bigger than the capable switching torque, $\bar{\boldsymbol{\Delta}} > \boldsymbol{\eta}$, voiding the stability criteria, (9.31). s_0 dominates the dynamics and torque until it reaches the ϵ-tube around 0.3 s. Thereafter s_1, thus the pitch error, starts to decrease and s_2, the yaw error, follows shortly after. After around 0.5 s the second sliding variable, s_1, hits the ϵ-tube and the torque is no longer saturated. From this point an onwards the rate at which the sliding variable s_2 decreases is defined by the switching gain, $\eta_2 = 3$, according to (9.26).

From this simulation the sliding mode controller does not appear robust against the change in model parameters. It is important to note that the sliding mode controller is only robust to matched uncertainties in the sliding phase. Since the initial condition left the controller far from the sliding surface, the changed parameters will affect the dynamics until the sliding surface is reached.

10.4 Sine wave simulation

Therefore, another comparison is made with a sine wave reference of 0.5 Hz and 3° in amplitude. As shown in Figure 10.5 the initial transient behaviour is different until the system settles on the sliding manifold. After settling, the difference between the model with original parameters and the model with changed mass and inertia, is almost indistinguishable.

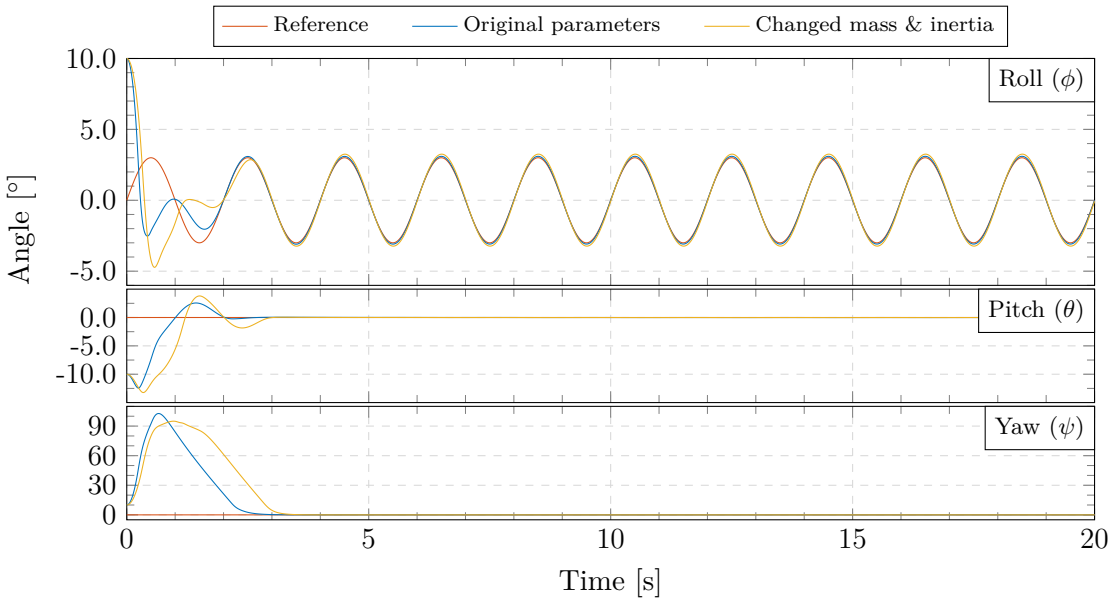


Figure 10.5: Comparison of Euler angles during a simulation with original parameters and with changed mass and inertia. Simulated with a sine wave reference on roll with a frequency of $f = 0.5$ Hz and an amplitude of 3°.

Looking closer at the tracking error in Figure 10.6, reveals the small difference between the two systems. Note the that the small oscillating tracking error is due to oscillations around the sliding manifold within the ϵ-tube.

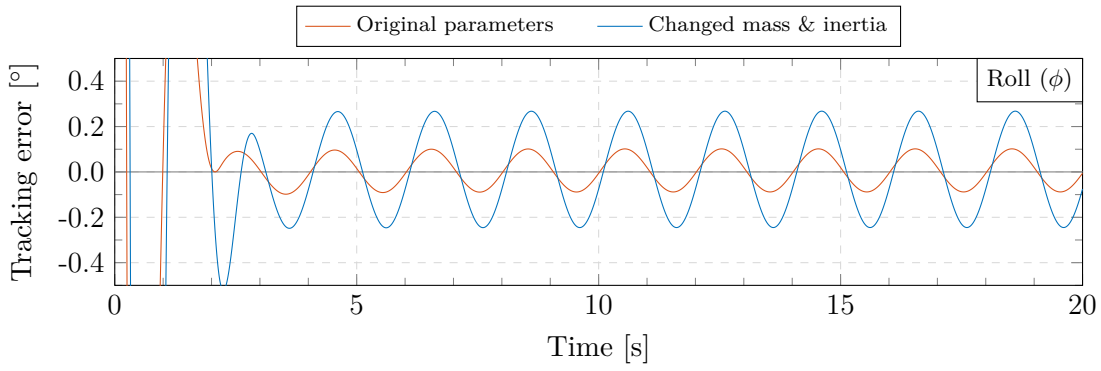


Figure 10.6: Comparison of tracking errors during sine wave simulation

This sine wave simulation is a great example of the benefits in the sliding mode controller, hereby shown to be capable of tracking a sine wave reference indistinguishably from a big parameter uncertainty.

10.5 Chirp simulation

Since the saturation in torque becomes the limiting factor with the sliding mode controller, it is beneficial to identify at what reference frequency the control performance starts to degrade.

A linear chirp signal is a continuous sine wave with an instantaneous frequency, $f(t)$, that increases linearly with time according to (10.3).

$$\begin{aligned} \phi_{\text{ref}}(t) &= a \sin\bigl(2\pi f(t)\bigr) \\ f(t) &= f_0 + kt \end{aligned} \tag{10.3}$$

A simulation using a linear chirp signal for the roll reference angle, is carried out with a frequency starting from $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.05$ Hz/s and an amplitude of $a = 2°$. The Euler angle response of the simulation is shown in Figure 10.7.

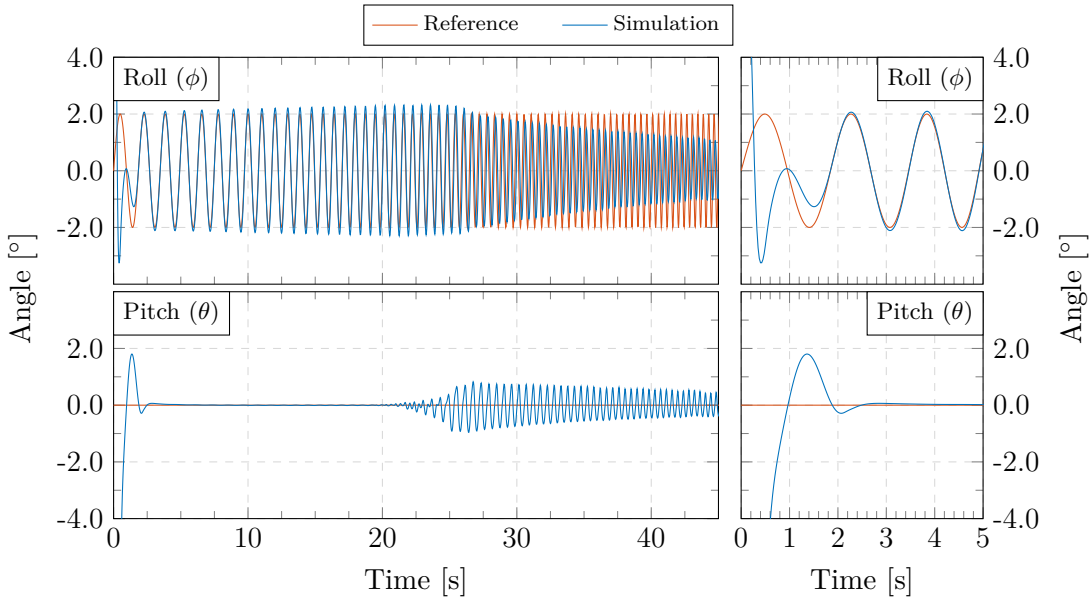


Figure 10.7: Euler angles during the roll chirp simulation with a frequency starting from $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.05$ Hz/s and an amplitude of $\pm 2°$

Similarly to the sine wave simulation in Figure 10.5, the system tracks the sine wave reference after an initial settling period, even while the reference frequency is growing. At a certain point, around 22.5 s, the pitch angle starts to deviate and the amplitude of the system roll angle starts to go down, indicating the bandwidth cap. Looking at the control torque, Figure 10.8, reveals that the bandwidth cap is related to the saturation in torque.

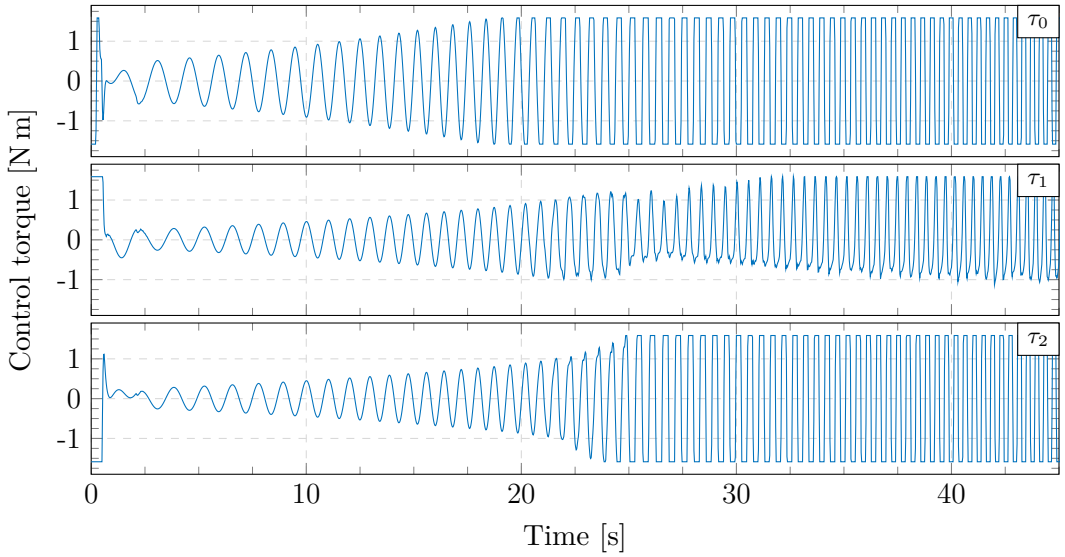


Figure 10.8: Control torque during the roll chirp simulation. Note that saturation starts to occur around 20 s.

At around 20 s, when τ_0 starts to saturate, the frequency of the chirp signal is 1.5 Hz. It is therefore expected that the actual system will have a closed loop bandwidth of 1.5 Hz or less.

10.6 Center of mass influence

Finally, a simulation is made to show the influence of a misaligned center of mass, since all previous tests were carried out with an aligned COM. The problem with the center of mass not being exactly aligned above the center of the ball, is that it leads to a constant translational acceleration due to the torque resulting from gravity affecting the body mass.

A zero reference simulation, similar to the one in Section 10.3, is carried out with the center of mass set to the actual value from Table 2.1 in both the simulated dynamics and as the nominal model parameter. A comparison of the translational velocity with the changed center of mass is shown in Figure 10.9.

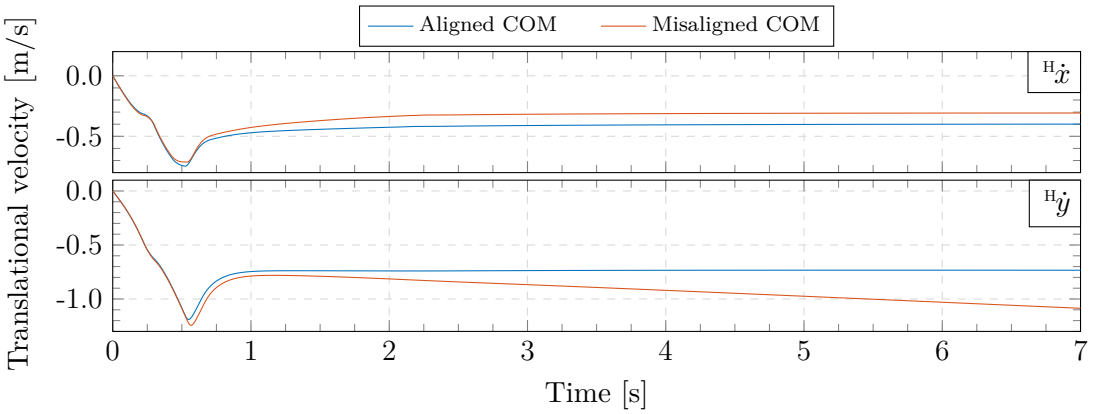


Figure 10.9: Comparison of translational velocity during simulated zero referefence test with aligned COM and misaligned COM as from Table 2.1. The misaligned COM leads to a steady state acceleration.

The misaligned center of mass leads to a steady state acceleration, especially present on the y-axis equal to ${}^{\mathrm{I}}\ddot{\boldsymbol{y}} = -0.0563\ \mathrm{m/s^2}$. This steady state acceleration can be computed using the steady state relationship between orientation, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}_{\mathrm{ref}}$, and linear acceleration, ${}^{\mathrm{H}}\ddot{\boldsymbol{x}}$ and ${}^{\mathrm{H}}\ddot{\boldsymbol{y}}$. This relationship is extracted from the linearized closed loop matrix in Appendix Q.3, similar to how it is done in the MPC, see (S.10). For small misalignments, the misaligned center of mass is equivalent to an offset in the orientation of a model with aligned center of mass. The equivalent orientation offset is computed in (10.4).

$$
{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}_{\mathrm{offset}} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\sin^{-1}\left(-\frac{{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM,y}}}{l}\right) \\ \frac{1}{2}\sin^{-1}\left(\frac{{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM,x}}}{l}\right) \\ 0 \end{bmatrix} \tag{10.4}
$$

From Appendix Q.3 and the linearized model matrix, (S.10), this leads to the steady acceleration computed in (10.5). Note that $c_{qx} = c_{qy}$.

$$
\begin{bmatrix} {}^{\mathrm{H}}\ddot{x} \\ {}^{\mathrm{H}}\ddot{y} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} c_{qx}\sin^{-1}\left(\frac{{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM,x}}}{l}\right) \\ -c_{qy}\sin^{-1}\left(-\frac{{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM,y}}}{l}\right) \end{bmatrix} = 7.265\begin{bmatrix} \sin^{-1}\left(\frac{{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM,x}}}{l}\right) \\ \sin^{-1}\left(\frac{{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM,y}}}{l}\right) \end{bmatrix} = \begin{bmatrix} 0.0357 \times 10^{-2} \\ -0.0553 \end{bmatrix} \mathrm{m/s^2} \tag{10.5}
$$

The computed acceleration is close to the simulation results, where the small difference can be explained from the included friction components.

The influence of a misaligned center of mass makes it difficult to test the balance controller on the actual system. It is therefore expected that manual corrections should be applied while testing the balance controller, as the robot would otherwise run away.

This concludes the simulation of the balance controller, confirming that the derived controller works as intended on the non-linear dynamic model derived in Chapter 5. The dynamic model is, however, still to be verified by testing the controller on the actual plant. In the following chapter the implementation leading up to these tests, is described.

11 Implementation

A lot of work has been put into the implementation on the actual robot. The derived controller and necessary estimators as shown in Figure 2.8, are implemented in firmware on the microprocessor described in Section 2.1, prior to carrying out any tests. In general the focus of the embedded firmware is to make testing and onwards development a breeze. This chapter presents an overview of the implementation focusing mainly on the embedded firmware, which can be found at `Kugle-Embedded`.

11.1 Firmware stack

The Kugle V1 prototype includes a new and fast microprocessor platform for which several low-level drivers are developed to connect the microprocessor to all the external periphirals on the robot. FreeRTOS [63] is used on the microprocessor to enable a multi-threaded processing environment. All code for the microprocessor is implemented as C++ libraries for readability, traceability and ease of debugging. The complete firmware stack, shown in Figure 11.1, has been developed from scratch, including all low-level hardware libraries, periphiral libraries, drivers, all the way up to the application layers. The choice of a multi-threaded object-oriented working environment will hopefully allow future users to quickly expand the features of the embedded firmware without affecting existing applications already running.

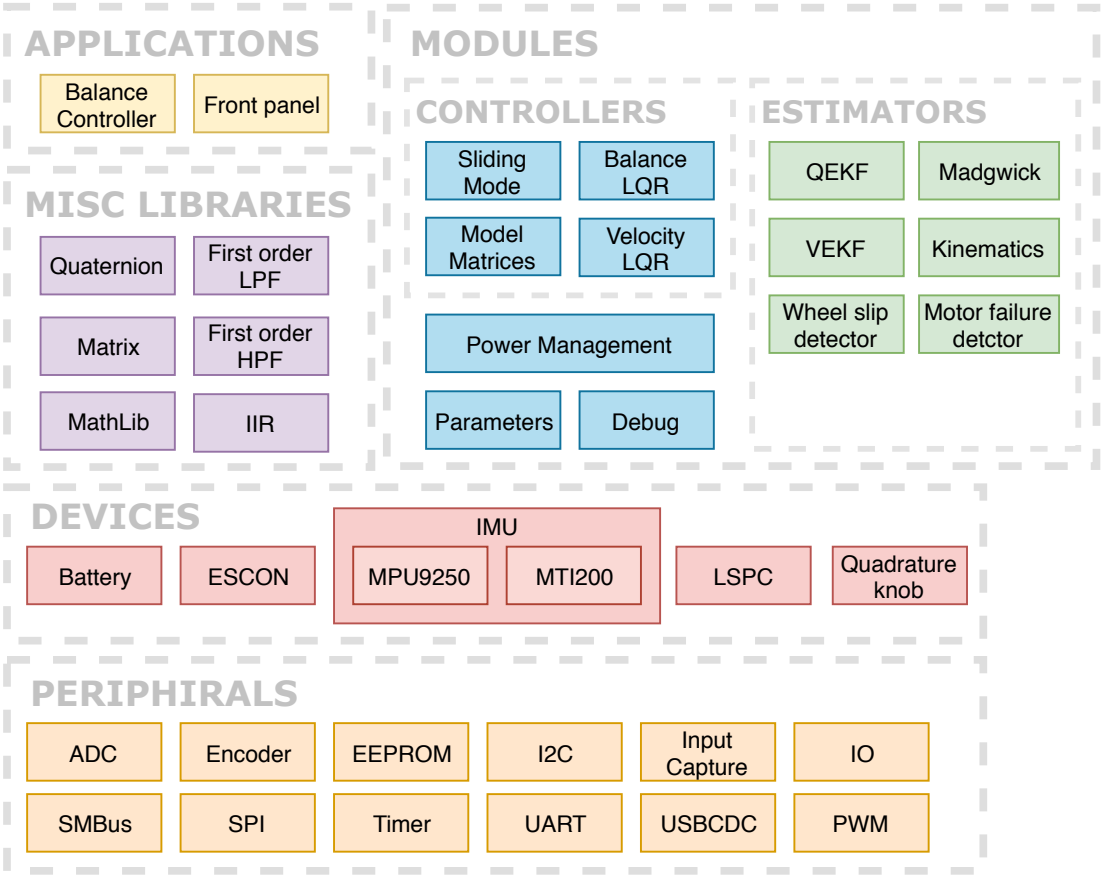


Figure 11.1: Hierarchy of the embedded firmware stack, where low-level elements are in the bottom and high-level elements, such as the balance controller, are in the top.

The bottom of the firmware stack, also known as the hardware layer, HAL, includes all microprocessor-specific libraries with individual classes for each peripheral.

The device drivers include all the necessary drivers for the onboard devices such as the IMU, motor drivers, batteries and the package protocol, LSPC [64], used for the USB communication. LSPC, short for Lightweight Serial Package Communication, is a simple protocol for serial stream interfaces enabling message passing with packages tagged with an ID. The LSPC library thus gives the embedded firmware a *publish/subscribe* mechanism for passing messages to and from the onboard computer. The list of LSPC message types and IDs are defined with the structs in `Kugle-Embedded/KugleFirmware/Libraries/Devices/LSPC/MessageTypes.h`. Both the IMU and LSPC library is templated to allow several interface options. As an example the MPU-9250 supports both I2C and SPI communication, which has been made completely transparent with the MPU-9250 library. In general, device drivers are the only libraries with direct dependency to periphiral libraries.

The module libraries include self contained units which may or may not depend on device drivers. The controller and estimator libraries are all independent classes who only perform mathematical operations and possibly include some time dependent states. Some of these libraries are generated with MATLAB Coder®, such as the QEKF, VEKF and model matrices. The power management library monitors the batteries. The parameters library includes a global list of dynamically modifiable parameters, thereby serving the purpose as a global parameter server. The global parameter server is linked to an LSPC object so that the parameters can be modified from the onboard computer, see Section 11.3. The debug library implements an overload on the `printf` function for passing `printf` messages directly to the onboard computer over USB, using a debug package packed with LSPC.

Some miscellaneous support libraries are developed to make mathematical operations such as matrix multiplication and quaternion operations easier. These math libraries are developed as plain C functions rather than classes, so that they can be called everywhere.

Finally, the application layer include the main applications of the firmware. The characteristic of an application is a thread with a main loop that can be started, stopped or restarted. An application is thus very similar to a service in a Unix system. Two main applications are developed in the embedded firmware, namely the balance controller and the front panel. The balance controller contain all the necessary logic and flow to run the controller and estimators and set the torque output accordingly, which is described in further details in Section 11.2. The front panel updates the user LEDs and monitors the buttons, e.g., handling the shutdown button which can be configured to start and stop the controller.

All libraries are developed with attention to thread-safety and thus include semaphores, mutexes, queues etc. where necessary. As an example the periphiral library for I2C communication, which supports multiple devices through addressing, is developed in such a way that multiple objects pointing to the same hardware I2C periphiral, can be constructed and used independently across multiple threads, see `Kugle-Embedded/KugleFirmware/Libraries/Periphirals/I2C`.

All the C++ libraries from the firmware stack can be found in the Git repository, `Kugle-Embedded/KugleFirmware/Libraries`.

11.2 Controller flow

The main part of the firmware that ties everything together is the control loop inside the balance controller, see `Kugle-Embedded/KugleFirmware/Libraries/Applications/BalanceController/BalanceController.cpp#L318-L925`. The balance controller loop runs at 200 Hz and implements a sequence of tasks shown in Figure 11.2. The flow of the balance controller loop is depicted in six steps where the actions within each step are executed from top to bottom. The coloured boxes indicate the main library used for each action.

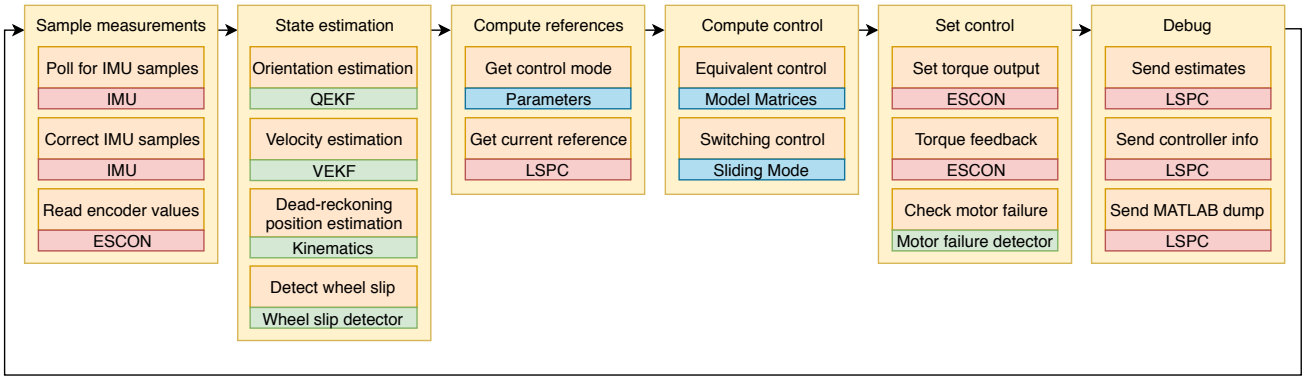


Figure 11.2: Flowchart of balance controller loop

The first step is to collect measurements from the different sensors. These sensor values are used in the second step for state estimation.

In the third step the control mode is defined with a dynamic parameter. The control mode can be set to either balance control, where just the balance controller is running, or set to velocity control, where a cascaded configuration of the balance controller and velocity LQR controller, Appendix Q, is running. Note that in velocity control mode the velocity controller runs in the same loop just prior to computing the sliding mode control. Different type of references for the controllers can be sent over USB through LSPC and are automatically parsed in the control loop.

The fourth step involves computing the control. The depicted actions illustrate the sliding mode balance controller which contains the computation of the equivalent control part followed by the switching part.

Finally, the computed control torque is sent to the ESCON motor drivers and a torque feedback is used to detect motor failures and lift, see Appendix O.2.

As a last step several debug informations are queued for transmission to the onboard computer for ROS topic transmission and logging in general, including a large float array with a raw dump of everything to be processed in MATLAB.

11.2.1 MCU load

To ensure real-time execution the utilization should stay well below 100%. The microprocessor is an ARM Cortex-M7 running at 400 MHz with a full double-precision floating point unit. So even though the Extended Kalman filters are known to be compute intensive with their large matrix inversions, a high load with just the control loop running is not expected.

Under normal operation, thus when running the balance controller application with the velocity controller enabled in station-keeping mode, see Appendix Q, and with no references given, the MCU load for the different tasks running inside the FreeRTOS operating system, is listed in Table 11.1.

Task	**Description**	**Load percentage**
Balance Controller	Balance controller application task	6.2%
LSPC transmitter	Pack and send LSPC packages over USB	3.1%
Xsens UART processing	Parse incoming UART data from Xsens MTI-200 IMU	1.2%
mainTask	CPU load monitoring	0.9%
Front Panel	LED & Button handling and mode switching	0.1%
Debug transmitter	Packs `printf` commands into LSPC debug packages	0.1%
Power Management	Battery monitoring	$< 0.1\%$
LSPC processing	Receive and unpack incoming LSPC packages from USB	$< 0.1\%$
Idle	Remaining and thus available CPU	88.4%

Table 11.1: MCU load of the different tasks running in the embedded firmware

The balance controller application clearly loads the system the most, followed by the LSPC transmitter for sending USB packages with LSPC messages. The MPU-9250 IMU is configured in a poll mode while the Xsens IMU is configured to send samples continuously at 400 Hz over UART, see ↗ `Kugle-Embedded/KugleFirmware/Src/MainTask.cpp#L134`. Parsing this incoming UART data, buffered in a interrupt, is what loads the CPU by another 1.2%. The mainTask takes care of sending CPU load information to the onboard computer while monitoring the general load and real-time execution. The remaining tasks are self-explanatory and their load is almost negligible. Note, however, that the load from the LSPC processing task will rise when references are sent to the embedded board, since these incoming LSPC packages has to be unpacked and handled. Furthermore, the load from debug transmission depends on how much debug information is sent with `printf`.

Altogether the fairly small load of only 11.6% leaves plenty of room for expansion and future development.

11.2.2 Computation time

The 6.2% of the load stems from the balance controller loop running at 200 Hz. Table 11.2 shows a breakdown of this load in terms of the computation time of each action in Figure 11.2 ordered chronologically.

As expected the most time consuming action is the orientation estimation with the quaternion estimator, QEKF. This extended Kalman filter includes ten states and some complicated measurement models involving a lot of computation. However, with 148 µs at 200 Hz there are plenty of room for other actions.

Action	**Description**	**Computation time**
IMU sampling	Poll and correct measurements from IMU	2 µs
Encoder sampling	Read and convert encoder values into angle	1 µs
QEKF	Orientation estimation with EKF	148 µs
VEKF	Velocity estimation with EKF	87 µs
Kinematics	Velocity from kinematics and numerical integration	2 µs
Wheel slip detector	Detect wheel slip based on encoder measurements	2 µs
Reference computation	Compute references based on received set-points	3 µs
Velocity controller	Velocity and position LQR to compute references for balance controller	26 µs
Sliding mode controller	Compute equivalent control and switching control. Uses model matrices and performs a matrix inversion.	85 µs
Set torque	Set torque output. Saturate if necessary. Ramps torque output initially.	6 µs
Send sensors	Pack raw sensor values for LSPC	42 µs
Send estimates	Pack computed estimates for LSPC	27 µs
Send controller info	Pack controller info and computation time	21 µs
Send MATLAB dump	Prepare packed arrays with data for logging	61 µs
Average computation time	Average of total control loop computation time. Computed on microprocessor without overhead.	420 µs

Table 11.2: Computation times for individual actions in the balance controller loop

Note that the average computation time of 420 µs do not seem to match the MCU load from Table 11.1 of 6.2%. 420 µs at a rate of 200 Hz corresponds to a computation time of 84 ms and thus a load of 8.4%.

The MCU load in Table 11.1 are determined with the FreeRTOS system while the computation time in Table 11.2 is determined using an internal microsecond timer. It is thus expected that the actual MCU load is slightly higher than what is listed in Table 11.1.

11.3 ROS support

The Kugle robot is used as part of a ROS framework as described in Section 2.3. An important aspect of the embedded firmware is thus to exchange information with the ROS infrastructure. Using LSPC and a USB connection, packages can be exchanged with the onboard computer. A ROS driver is developed to communicate with the embedded firmware and serves the purpose of bridging ROS topics to according messages, to and from the embedded firmware. Furthermore, a few ROS services are implemented to execute actions such as reboot, calibrate, changing parameters etc. The ROS driver can be found at `Kugle-ROS/kugle_driver`.

When connected the ROS driver automatically populates the *tf* tree [65] with information from the state estimates computed in the embedded firmware. This allows a live view of the current position and orientation of the robot through the *rviz* GUI [66]. A similar view, shown in Figure 11.3, can be achieved with the developed Gazebo simulation, as described in Appendix S.7.

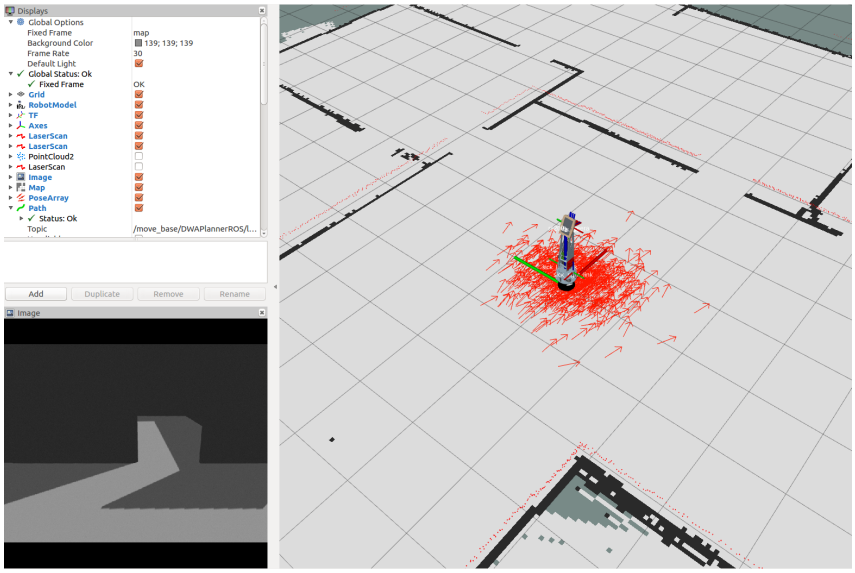


Figure 11.3: RVIZ view of simulated Kugle ballbot. A similar view can be achieved with the actual robot when connected.

Three main elements of the ROS driver made the tests, in the next chapter, a breeze.

1. `printf` debugging into the terminal.
2. Real-time parameter adjustments with the *rqt_reconfigure* GUI.
3. Dump of raw float arrays into MATLAB.

As mentioned in Section 11.1 the debug library creates an overload on the `printf` command so that any debugging inside the embedded firmware is transmitted over USB. In the ROS driver these debug messages are parsed and printed directly in the terminal, thereby enabling transparent `printf`-debugging over USB. Even though the USB JTAG debugger has been used during development of the embedded firmware, this `printf` functionality is valuable to future users. Especially if the users are more familiar with Arduino environments than generic ARM Cortex development.

The dynamic parameters within the global parameter server of the parameters library, is modifiable with dedicated LSPC messages. Since ROS already includes a way to modify parameters dynamically, known as reconfigurable parameters, the ROS driver is developed so that all modifiable parameters of the embedded firmware are published as reconfigurable parameters in the ROS driver. Whenever a reconfigurable parameter is modified in ROS, the ROS driver automatically sends the corresponding LSPC message to modify the parameter in the embedded firmware. With the *rqt_reconfigure* GUI, see Figure O.1, parameter adjustments become very easy, which makes especially tuning a lot faster. The reconfigurable parameters include e.g., the current control mode, sliding surface selection, sliding surface gain, $\boldsymbol{K}$, switching gain, $\boldsymbol{\eta}$, ϵ-tube, estimator tuning parameters, see Appendix M and Appendix N, velocity controller gains, power button mode and flags for enabling reference tests.

Finally, a dump functionality is included in the embedded firmware to enable logging. The dump functionality sends a few fully packed float arrays over USB, dumping everything that can possibly be dumped. The float arrays include raw sensor measurements from all sensors, e.g., both the MPU-9250 and MTI-200 IMU simultaneously, state estimates, computed control outputs, sliding variables, estimation error covariances etc. In the ROS driver these float arrays are automatically dumped into comma separated files, so that they can be loaded into MATLAB easily. The dump arrays are defined in `Kugle-Embedded/KugleFirmware/Libraries/Applications/BalanceController/BalanceController.cpp#L817-L923` while the MATLAB loading script is found at `Kugle-MATLAB/DataProcessing/functions/LoadDump.m`.

This concludes the implementation of the embedded firmware, running the derived controller and estimators and including an easy set of tools for debugging, parameter tuning and logging. The next step is to use this firmware to test the controller.

12 Tests and Results

The following chapter show the results of five tests carried out with the sliding mode balance controller implemented on Kugle V1. The tests are designed to test the reference tracking performance and disturbance rejection of the sliding mode controller. Some of the tests are similar to the simulated ones from Chapter 10, including the zero reference test, the sine wave test and the chirp test. The zero reference test, equal to an upright balance test, is included to verify that the controller is capable of stabilizing the ballbot and how well it is capable of rejecting disturbances in terms of pushes. The last test is a rotating inclination test, where both the roll and pitch reference are changing in a sine wave with increasing frequency.

Assumed ground-truth measurements of the model states are recorded with a Vicon motion capture system [58] introduced in Section 6.1. The test results are discussed along the way.

12.1 Test parameters

The sliding mode controller is configured to use the quaternion derivative sliding manifold as chosen in Section 8.3. Two set of gains, an aggressive and a non-aggressive, have been tested in practice. The gains are found through extensive tuning with the strategy described in Appendix O.1.

The aggressive gains, intended for reference testing and tight tracking, are:

$$\begin{aligned} \boldsymbol{K} &= \begin{bmatrix} 15 & 15 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\eta} &= \begin{bmatrix} 6 & 6 & 3 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\epsilon} &= \begin{bmatrix} 0.5 & 0.5 & 0.2 \end{bmatrix}^\mathsf{T} \end{aligned} \tag{12.1}$$

The non-aggressive gains, intended for cascade loop with the velocity controller in Chapter 14, are:

$$\begin{aligned} \boldsymbol{K} &= \begin{bmatrix} 6 & 6 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\eta} &= \begin{bmatrix} 5 & 5 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\epsilon} &= \begin{bmatrix} 0.8 & 0.8 & 0.8 \end{bmatrix}^\mathsf{T} \end{aligned} \tag{12.2}$$

Both gains are tested and compared in the upright balance test and the rotating inclination test. The aggressive gains are too aggressive for the step test, which is why the non-aggressive gains are used instead. On the other hand the aggressive gains achieves desirable tight tracking in the sine wave and chirp test.

In all tests the output torque is limited to 1.6 N m as reasoned in Appendix O.2, even though the motors are able to deliver up to almost 2 N m as listed in Table 2.1.

12.2 Upright balance test

The first test involves a constant zero reference, thus corresponding to an upright unit quaternion and zero angular velocity. As described in Section 10.6 the position of the ballbot is likely to drift due to disturbances and a misaligned center of mass, since the velocity is left uncontrolled.

The upright balance test therefore includes several timestamps where a disturbance is exerted to correct for the drifting position and increasing velocity, see Figure 12.1. Each disturbance are marked with a red dotted vertical line.

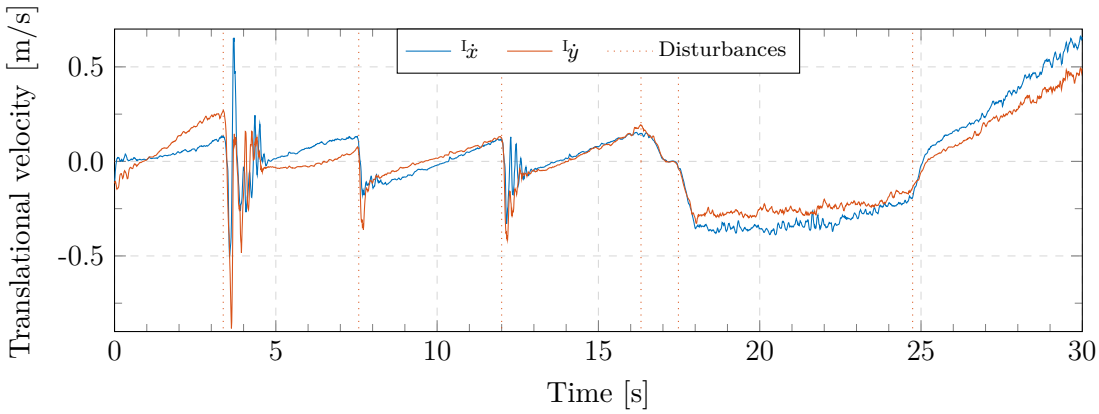


Figure 12.1: Unstable velocity during the upright balance test with aggressive gains. The red dotted vertical lines indicate the timestamps of the correctional disturbances. Note how the velocity is corrected and lowered to zero with each disturbance.

Initially a balance test is carried out with the aggressive gains. The Euler angles of the ballbot during the test is shown in Figure 12.2, with both the ground-truth angles from the Vicon system and the Euler angles, computed from the quaternion estimate from the QEKF, see Appendix M. The controller is indeed capable of stabilizing the ballbot, even though the ballbot accelerates due to small deviations in the inclination and probably due to the misalignment in the center of mass. Note also how the controller quickly reacts and compensates for every disturbance, causing almost no change in angle.

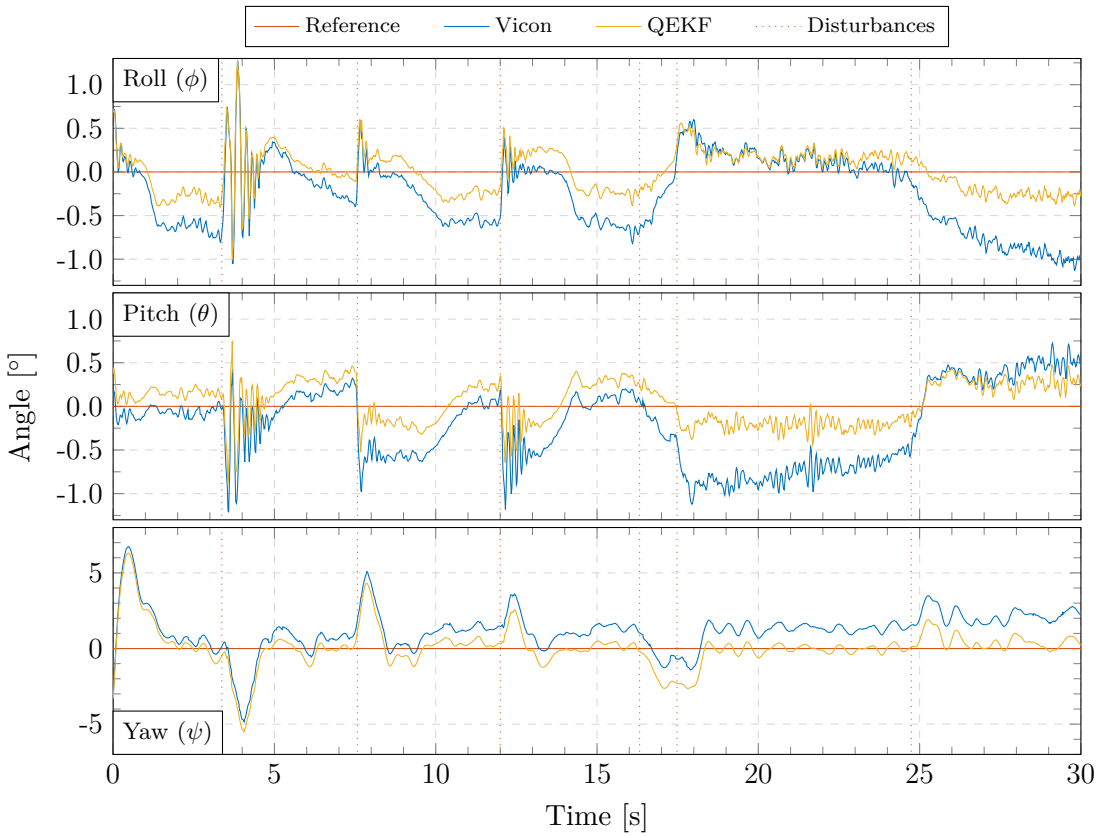


Figure 12.2: Upright balance test with aggressive gains. The dotted vertical lines indicate the timestamps of the correctional disturbances. The QEKF angles show the internal estimate of the orientation used by the sliding mode controller.

The estimated orientation from the QEKF has an estimation error of around 0.5°, which is consistent with the tests in Appendix M. With the aggressive gains the sliding mode controller is able to keep the estimated inclination within ±0.4° of the reference, while the actual inclination deviates up to 1° from the reference. In the following tests the Vicon orientation is only shown, since the reference tracking performance of the complete system, including the observers, should be evaluated against the ground-truth.

When the controller is started the system has a non-zero inclination but a zero quaternion derivative, and is thus away from the sliding manifold, see Section 8.2.2. During the reaching phase, shown in Figure 12.3 for s_0, the sliding mode controller tries to enforce sliding by pushing the system onto the sliding manifold. Since a continuous switching law is used, see Section 9.2.1, the controller pushes the system into an ϵ-tube around the manifold.

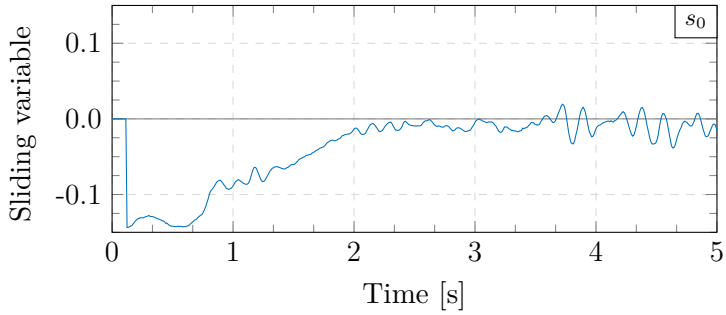


Figure 12.3: Sliding variable s_0 during reaching phase just after the controller is turned on

In Figure 12.4 all the sliding variables, $\boldsymbol{s}$, from the test in Figure 12.2, reveals how the system is pushed away from the manifold at each disturbance and how it quickly finds its way back again.

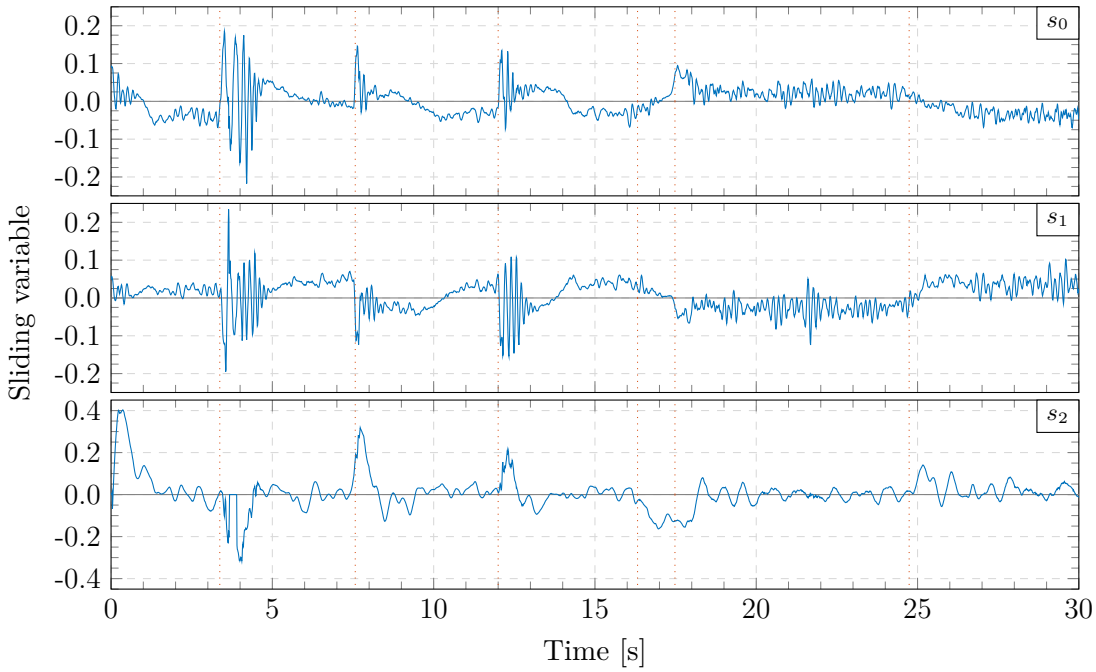


Figure 12.4: Sliding variables during upright balance with aggressive gains

When the system is disturbed, indicated with the dotted lines, the switching law kicks in immediately. This is especially noticeable on the control torque which saturates immediately. A zoomed plot of the control torque near the disturbance at 12 seconds, is shown in Figure 12.5.

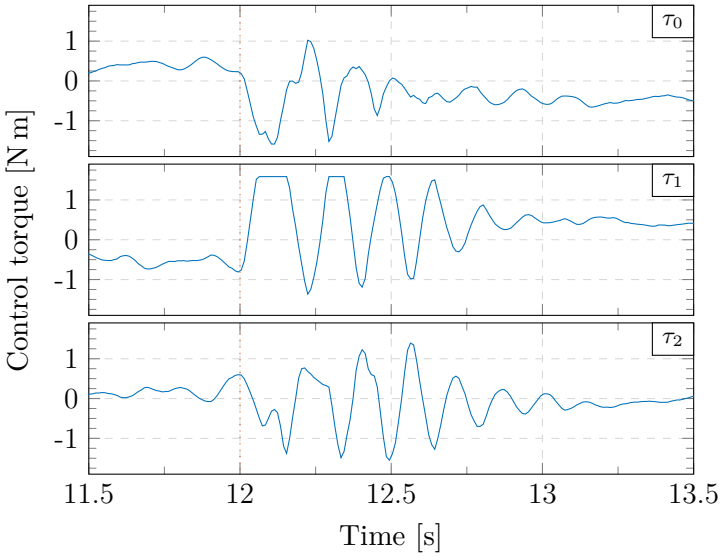


Figure 12.5: Zoomed plot of the control torque from when the system is disturbed. Note how the torque, τ_1, saturates immediately at the disturbance.

12.2.1 Gain comparison

The upright balance test is carried out with both of the gains mentioned in Section 12.1. A comparison of the Euler angles from both tests is shown in Figure 12.6.

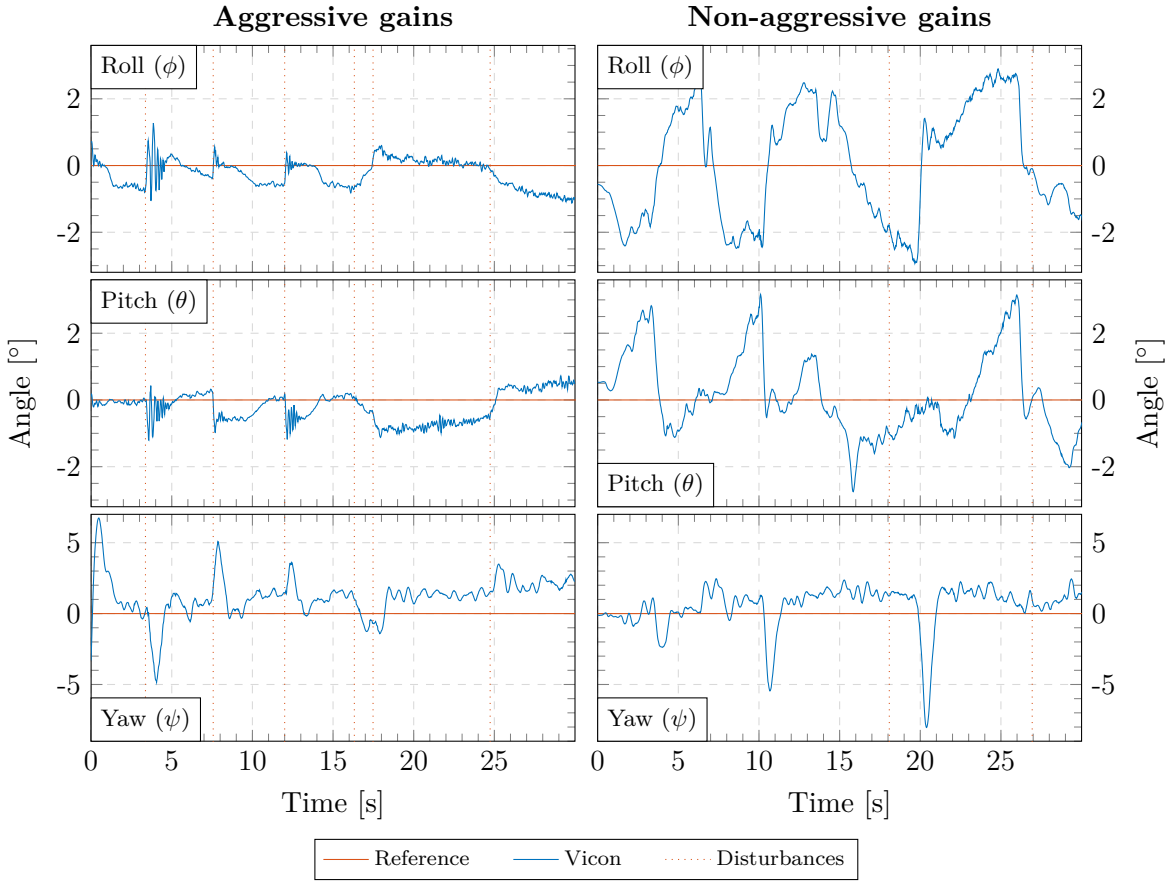


Figure 12.6: Comparison of Euler angles using aggressive and non-aggressive sliding mode controller gains

With the non-aggressive gains the system is clearly slower and the upper bound of the tracking error is also larger. Looking at just this gain comparison, one would definitely prefer to use the aggressive gains to keep the tracking error down. Unfortunately, when used with the velocity controller in Chapter 14, the aggressive gains leads to overly aggressive reference tracking, resulting in the ballbot becoming too vulnerable to disturbances. When the cascaded loop is closed, see Figure 7.2, any external disturbances affecting the velocity controller would also indirectly affect the balance controller through the references. With the aggressive gains these rapid changes in reference would lead to wheel slip and eventually the ballbot falling.

12.3 Step test

Similarly the non-aggressive gains are used in the step test, since the aggressive gains leads to controller saturation and wheel slip resulting in the ballbot falling off the ball. The step test involves a reference sequence of nine Euler angle steps, defined in the order of roll, pitch, yaw, (ϕ, θ, ψ).

1. $(0°, 0°, 0°)$
2. $(5°, 0°, 0°)$
3. $(0°, 0°, 0°)$
4. $(0°, 5°, 0°)$
5. $(0°, 0°, 0°)$
6. $(-5°, 5°, 0°)$
7. $(0°, 0°, 0°)$
8. $(0°, 0°, 45°)$
9. $(0°, 0°, 0°)$

The angular velocity reference is set to zero during all steps. The references are changed with an interval of 2 seconds except for the last two steps involving yaw, where an interval of 4 seconds is used, see Figure 12.7. The test is implemented at `Kugle-Embedded/KugleFirmware/Libraries/Applications/BalanceController/BalanceController.cpp#L999-L1046`.

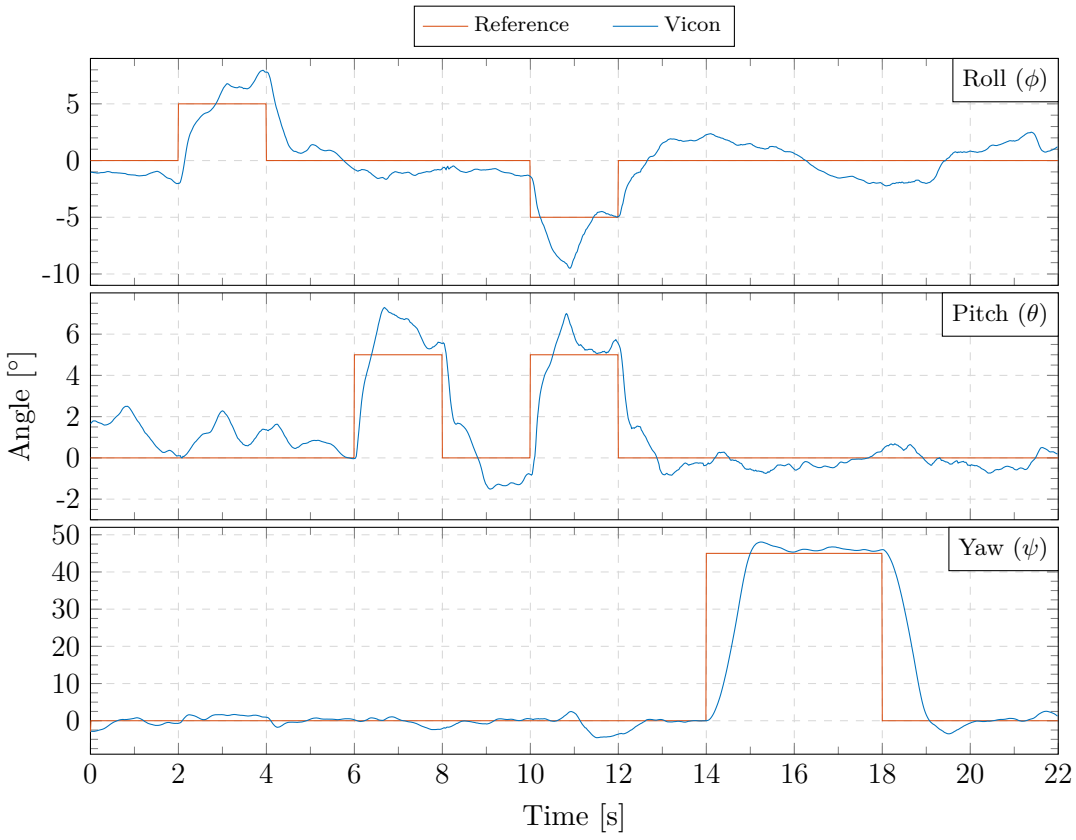


Figure 12.7: Reference step test with sliding mode controller using non-aggressive gains and the quaternion derivative sliding manifold

The controller is capable of tracking the step references but seem to overshoot. The overshooting is likely due to the sudden momentum gained from the step change in angle reference, while lacking any change in the angular velocity reference.
The step on yaw works flawlessly though, with a minimal overshoot of less than 5°. Due to the way the sliding mode controller works any size of yaw reference steps can be carried out without worrying about saturating the error, as one would usually do with a linear controller.
Unfortunately the same can not be said for roll and pitch. As long as the change in roll and pitch references are kept sufficiently small, in this case a step of 5°, and as long as the gains are not too aggressive, the system is capable of tracking the steps. However, it comes at a cost of saturating the torque output every time the reference is changed, which increases the likelihood of wheel slippage.

It is therefore not recommended to use step references on roll and pitch. Instead, a smooth reference signal, consisting of a matched pair of angular velocity references and quaternion references, should always be used. This leads up to the next test with a continuous sine wave reference.

12.4 Sine test

To test how well the controller tracks a continuous reference with matching angular velocity and quaternion references, a sine wave test is carried out. With a continuously changing reference the aggressive gains can be used again, ensuring a more tight reference tracking.

A sine wave of 0.5 Hz and ±3° amplitude is set as the roll reference and the corresponding derivative, being a cosine, is set as the angular velocity reference. The reference signals, along with the assumed ground-truth measurements from Vicon, are shown in Figure 12.8 and Figure 12.9.

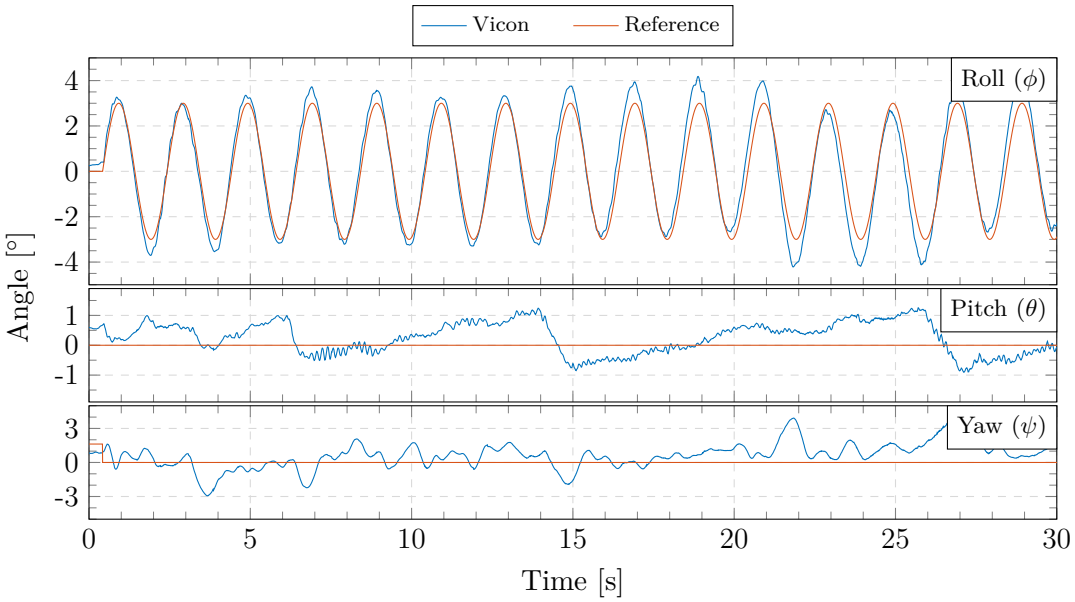


Figure 12.8: Euler angles during sine wave test with a 0.5 Hz ±3° sine wave reference on roll

The angular velocity references indirectly serve as a feed-forward term through the equivalent control law, Section 9.1, and thereby helps the controller to achieve accurate tracking with minimal lag, as seen in Figure 12.8. The controller tracks the angle references nicely with only minor deviations at the peaks, which suggests a less perfect tracking of the angular velocity reference.

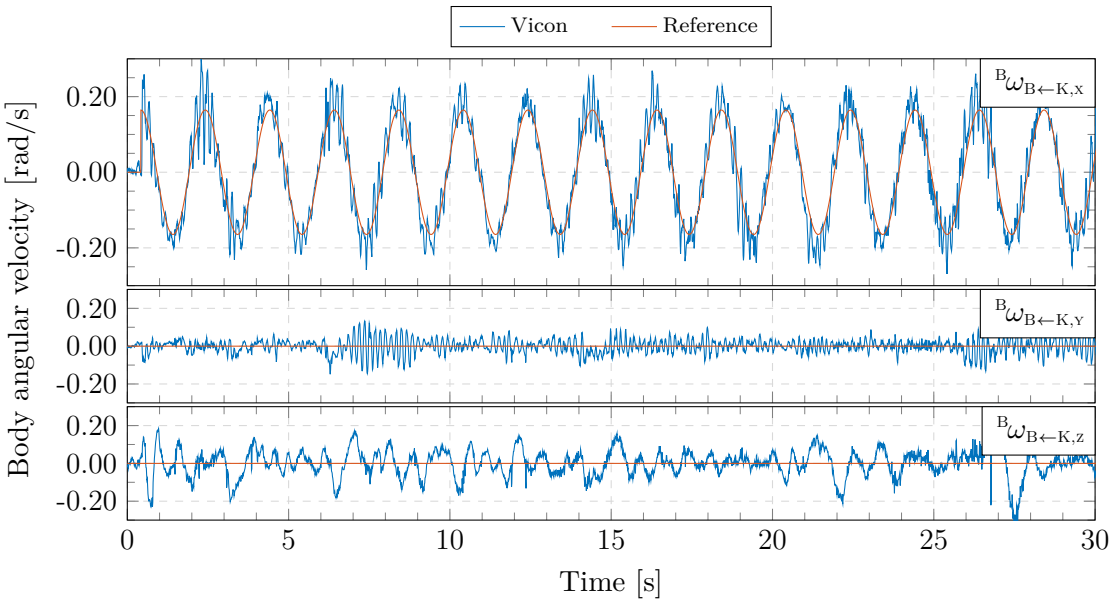


Figure 12.9: Angular velocities during sine wave test with a ±3°, 0.5 Hz sine wave reference on roll

Zooming in on the angular velocity reveals some distinct oscillations around the reference, with a frequency of approximately 7 Hz. In Figure 12.10 the estimated angular velocity from the QEKF is also shown to give the reader an idea of the smoothing, and thus also the small lag, that is introduced by the quaternion estimator. However, the lag in the QEKF is so small that it is not to blame when it comes to the less perfect tracking at the peaks.

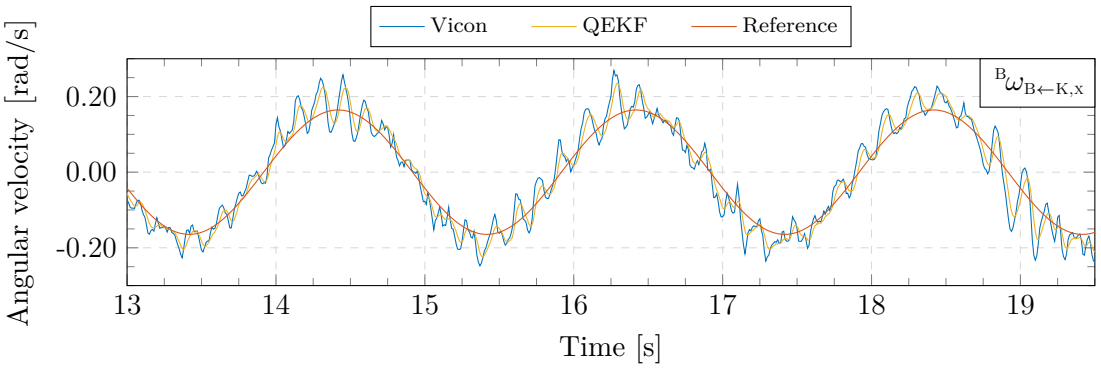


Figure 12.10: Zoom of angular velocities during sine wave test to illustrate the oscillations around the reference

The oscillations around the reference are likely due to unmodelled dynamics excited by the switching torque from the sliding mode controller and the amplitude of the oscillations is likely related to the ϵ-tube. The amplitude is, however, so small that it is barely visible during the test, but one can hear it.

12.5 Chirp test

A way to identify the closed loop bandwidth of a system, without sending an impulse or a step, which excites all frequencies at once, is to use the continuously changing chirp signal, as described in Section 10.5. A chirp test is carried out on the roll angle with a frequency starting from $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.02$ Hz/s and an amplitude of ±2°. The test is implemented at `Kugle-Embedded/KugleFirmware/Libraries/Applications/BalanceController/BalanceController.cpp#L1047-L1078`. The Euler angle response is shown in Figure 12.11.

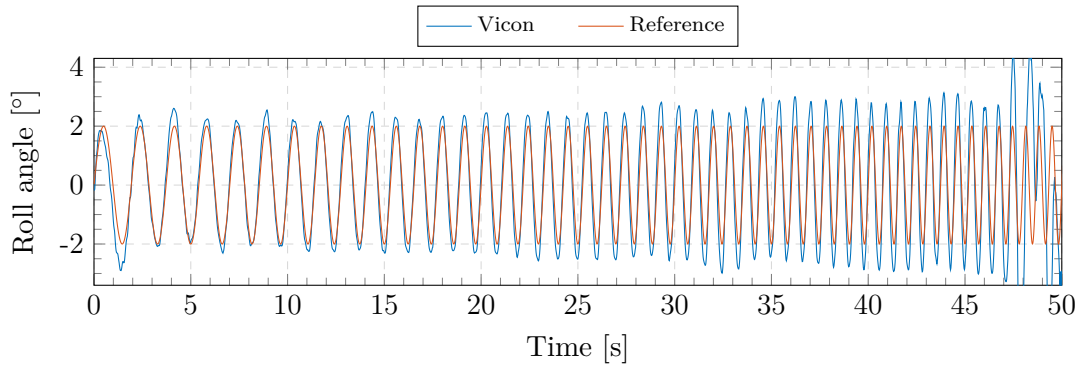


Figure 12.11: Euler angles during the roll chirp test with a frequency starting from $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.02$ Hz/s and an amplitude of $\pm 2°$

After an initial settling period the controller tracks the angle reference well up to a certain point. A chirp signal is known to have a derivative whose amplitude increase linearly with time. In this case the derivative corresponds to the angular velocity reference, shown in red in Figure 12.12.

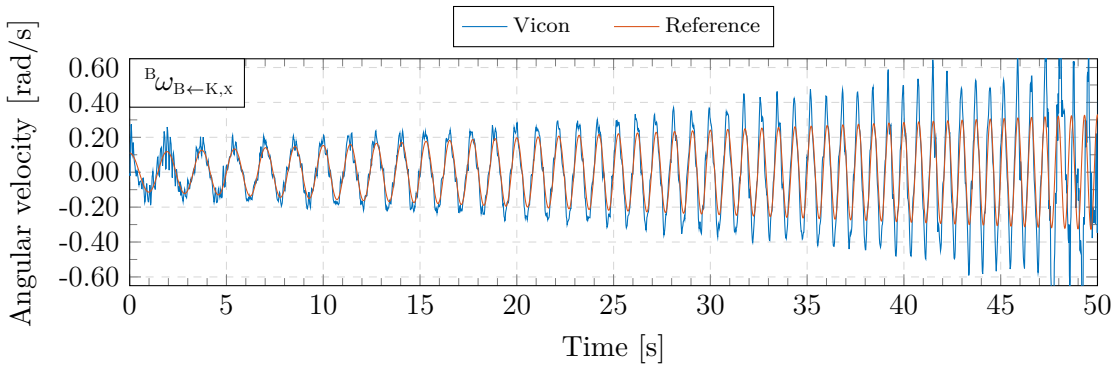


Figure 12.12: Angular velocities during the roll chirp test with a frequency starting from $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.02$ Hz/s and an amplitude of $\pm 2°$

The controller is capable of tracking the chirp signal, but overshoots more and more on the angular velocity, leading to large overshoots on the angle near the end of the test, see Figure 12.13.

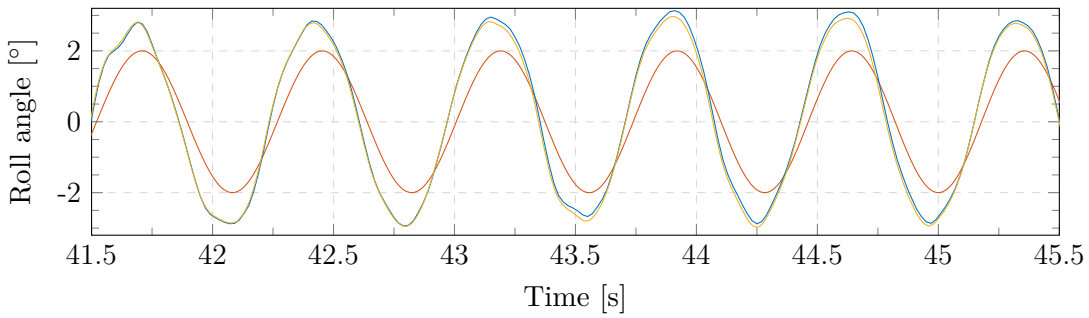


Figure 12.13: Zoom of chirp test near the end where the reference signal is changing with a frequency of nearly 1.5 Hz

The zoomed plot in Figure 12.13 shows the response near the end where the reference frequency approaches 1.5 Hz. Here the controller greatly overshoots and seem to have lost control authority at the peaks, likely due to saturations in torque, see Figure 12.14. Since the necessary torque can be assumed proportional to the angular acceleration of the reference signal, a chirp signal like this will require a torque that increases in a squared relation to the frequency. The motors will thus quickly saturate.

At a certain point though, the controller looses tracking completely, due to the ESCON motor drivers limiting the output to protect the motors. This is revealed by looking at the commanded control torque and the delivered torque feedback from the ESCON drivers, see Figure 12.14.

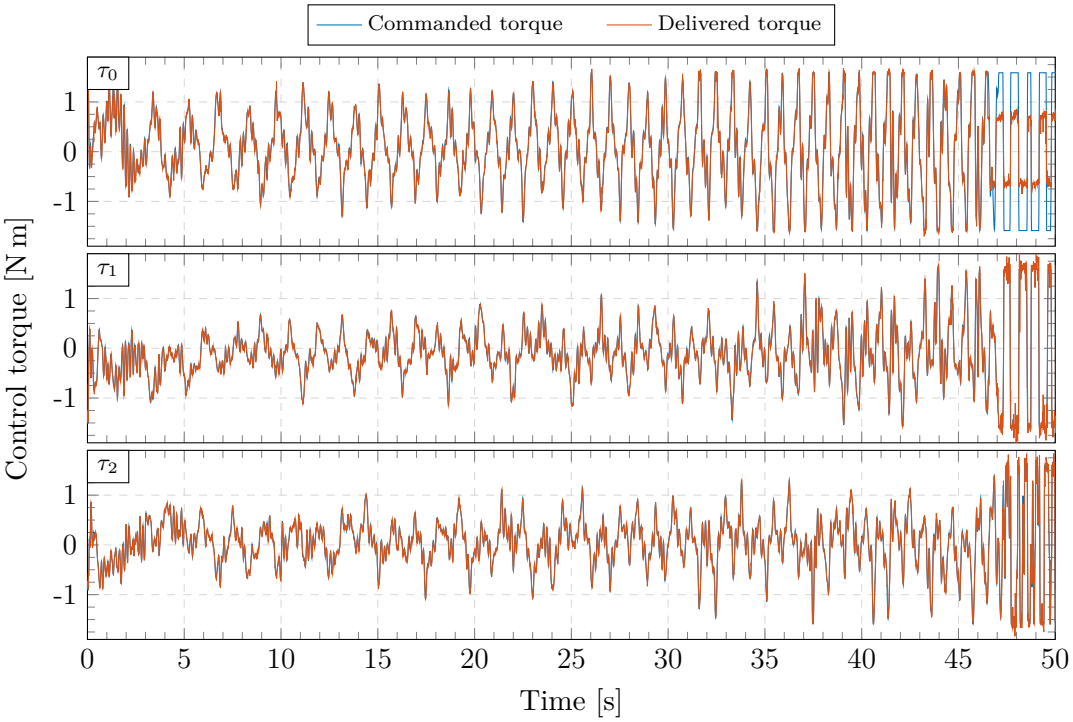


Figure 12.14: Comparison of command torque and delivered torque during the chirp test, revealing the ESCON motor drivers limiting the output near the end

The bandwidth of the system is thus close to 1.5 Hz, where the torque output is completely saturated and the motor drivers are "running hot". This result is consistent with the simulation in Section 10.5, which also indicated a bandwidth of 1.5 Hz.

12.6 Rotating inclination test

As a final test the reference tracking performance with the aggressive and non-aggressive gains are compared through a rotating inclination test, where both the roll and pitch angle are changing simultaneously. The quaternion reference is tilted to an inclination of 3° and is thereafter rotated around the inertial z-axis, while keeping the heading constant. This gives a swaying motion of the robot, since the center of mass will be moving around in a circle. Similarly to how the frequency was increased in the chirp test, the angular velocity of the rotation around the z-axis, starts from zero and increases with 0.02 rounds per second squared.

The test is implemented at `Kugle-Embedded/KugleFirmware/Libraries/Applications/BalanceController/BalanceController.cpp#L1079-L1121`. The comparison is shown in Figure 12.15.

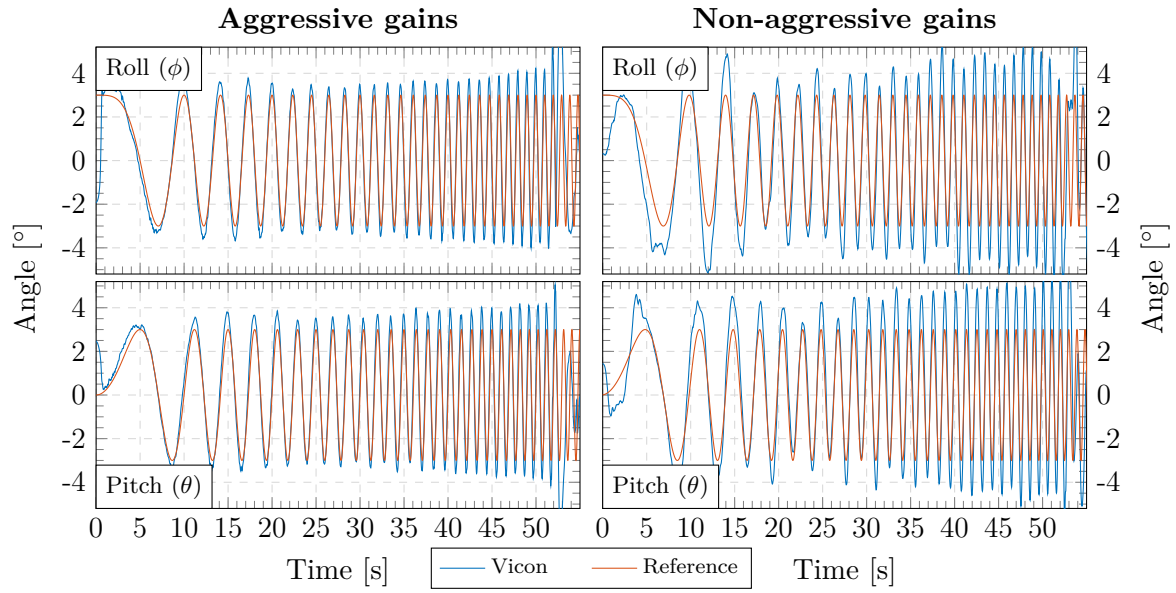


Figure 12.15: Comparison of Euler angles during the rotating inclination test with the aggressive and non-aggressive sliding mode controller gains

Yet again the balance controller shows how it is capable of tracking the continuously changing angle reference, in this case where both roll and pitch are changing simultaneously. The corresponding angular velocity reference is derived from the derivative of the quaternion reference and shown with the compared responses in Figure 12.16.

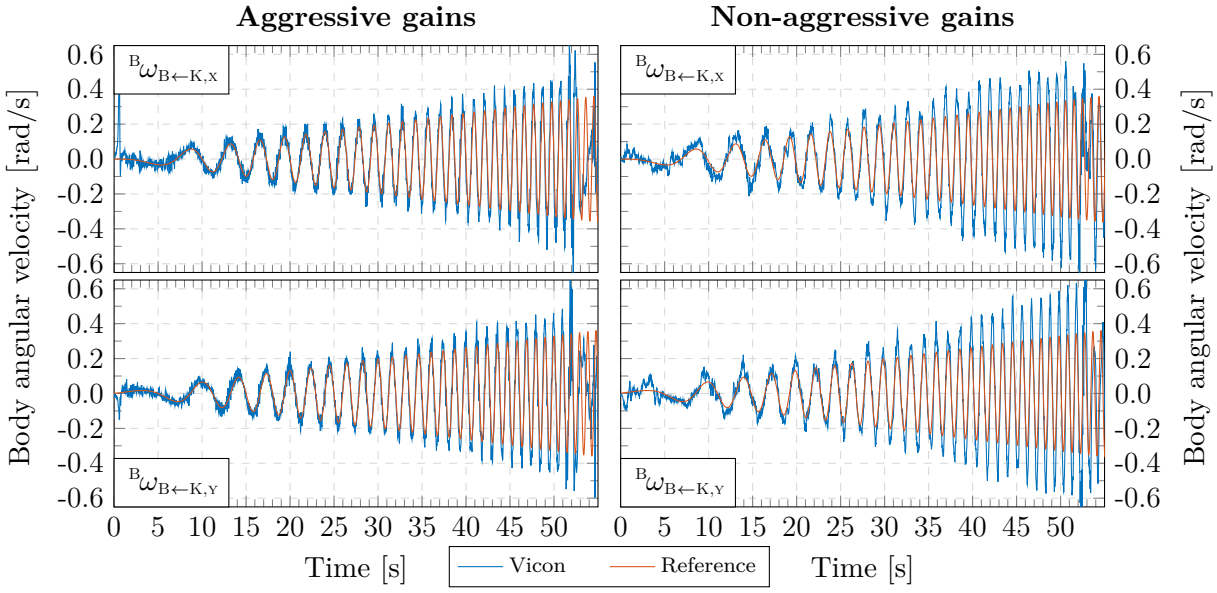


Figure 12.16: Comparison of angular velocity response during the rotating inclination test with the aggressive and non-aggressive sliding mode controller gains

From the results in Figure 12.15 and Figure 12.16 it is clear that the tracking of both the angle and angular velocity references is much better with the aggressive gains. This is also clear from the tracking error comparison shown in Figure 12.17, where the tracking is almost twice as good with the aggressive gains than with the non-aggressive gains.

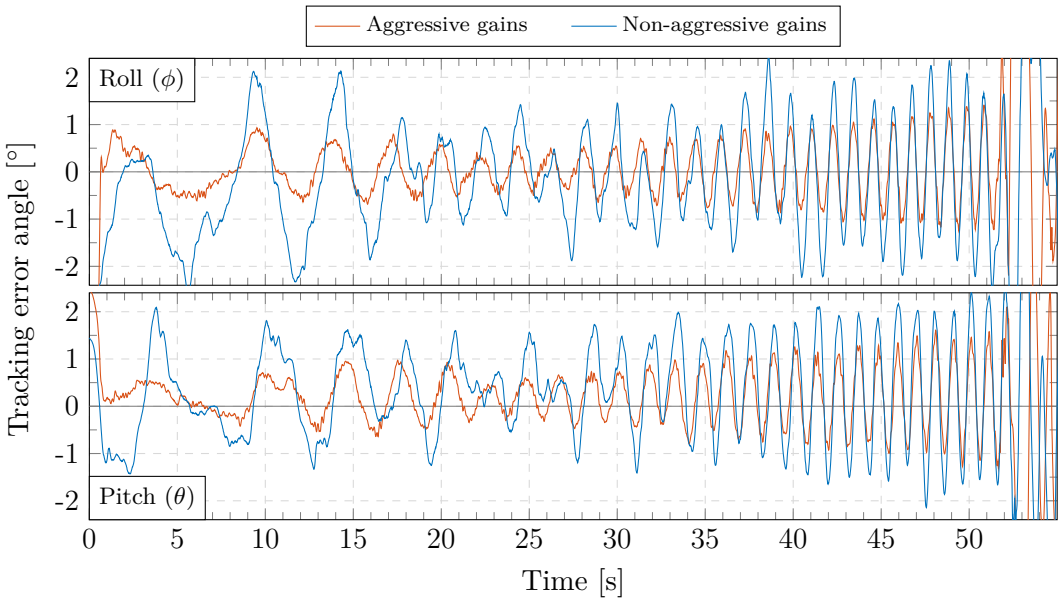


Figure 12.17: Comparison of tracking error during the rotating inclination test with the aggressive and non-aggressive sliding mode controller gains

However, both gains lead to the system "failing" around the same angular velocity, happening close to $t = 50$ s, which corresponds to the angular velocity references changing with 1 Hz. Again the limitation is due to the motor torque saturating and the motors "running hot". Thus when both roll and pitch references are changing simultaneously the bandwidth seems to be limited even further to around 1 Hz.

12.7 Summary

This concludes the tests and confirms that the sliding mode balance controller, using the quaternion model and the quaternion derivative sliding manifold, works as expected and is capable of tracking a pair of desired quaternion and angular velocity references. With a zero upright reference the controller is capable of keeping the inclination between $\pm 1°$ using the aggressive gains and $\pm 2.5°$ using the non-aggressive gains. The controller is capable of tracking small steps on roll and pitch, though with large overshoots. Step references are tracked nicely on yaw with minimal overshoot of approximately 5°. However, on roll and pitch, step references should in general be avoided. In terms of a continuous reference signal, the balance controller with aggressive gains is capable of tracking a single sine wave reference with an amplitude of 3° on the roll axis up till 1.5 Hz. At this frequency the motor torque is completely saturated and further limitations occur, since the ESCON motor drivers limit the output to protect the motors. The bandwidth of 1.5 Hz for a sine wave reference on roll is consistent with the simulation results. When both the roll and pitch references are changing with an amplitude of 2°, the protection kicked in at a frequency of 1 Hz. This suggests that the closed loop bandwidth of the system is more likely 1 Hz than 1.5 Hz. Any reference for roll and pitch should therefore be kept smooth and below these saturating limits.

This concludes the tests of the sliding mode balance controller, confirming that the controller works as designed on the actual robot, Kugle V1, thereby also verifying the non-linear model.

13 Balance LQR comparison

One part of the problem formulation in Section 1.4 questions how the non-linear sliding mode controller compares to a regular optimal linear controller, LQR. A stabilizing LQR is therefore designed in Appendix P, based on a linearized model around the upright equilibrium. This chapter contains a simulation comparison, showing the benefits of the non-linear controller, and an actual test of the LQR controller, where the performance is compared to the upright balance test from Section 12.2. The performance of the balance LQR is compared to the sliding mode controller using the aggressive gains and the quaternion derivative sliding surface. The simulated balance LQR can be found at `Kugle-MATLAB/Simulation/KugleSim_LQR.slx`.

13.1 Simulation comparison

A sine wave simulation, similar to the one in Section 10.4, is carried out with the LQR balance controller. The same references and initial conditions are used so that the simulated response can be compared directly with the sliding mode controller response from Figure 10.5. The comparison of the Euler angles is shown in Figure 13.1.

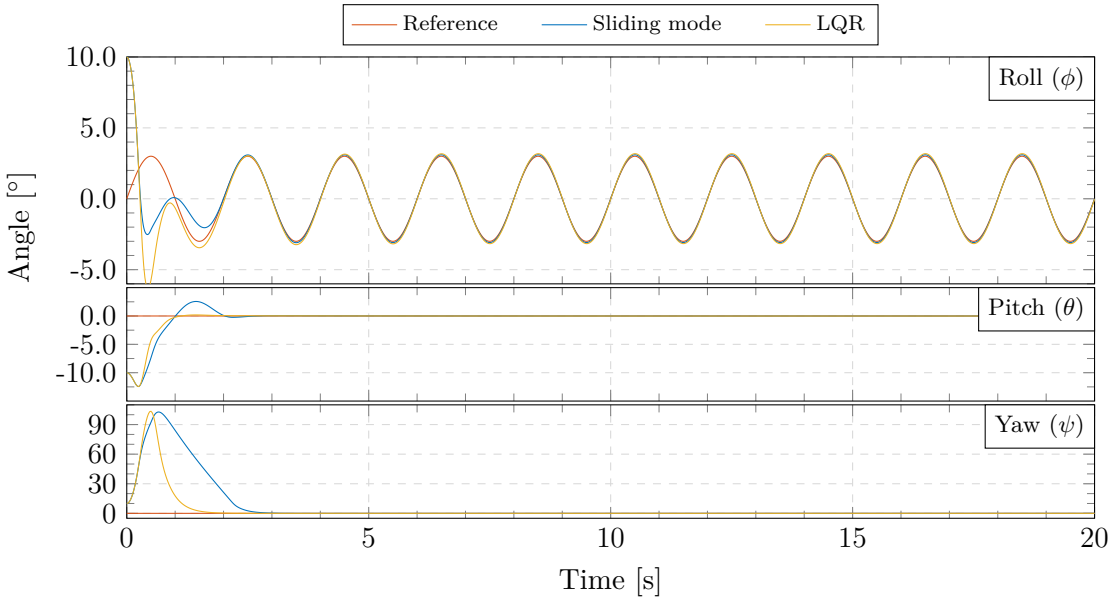


Figure 13.1: Comparison of Euler angles during the sine wave simulation of the sliding mode and the LQR balance controller

The initial response of the LQR balance controller seems slightly better on pitch and yaw than the sliding mode controller, which both has a overshoot and lags a lot on yaw as described in Section 10.3. This can potentially be explained from the chosen sliding mode switching gains, where especially the gain on yaw is chosen smaller than the gain on roll and pitch. Except for this noticeable difference initially, the LQR controller tracks the sine wave reference almost as good as the sliding mode controller.

The tracking error of the roll angle, shown in Figure 13.2, reveals that the tracking with the sliding mode controller is about twice as good as with the LQR controller. However, this could just be a matter of tuning.

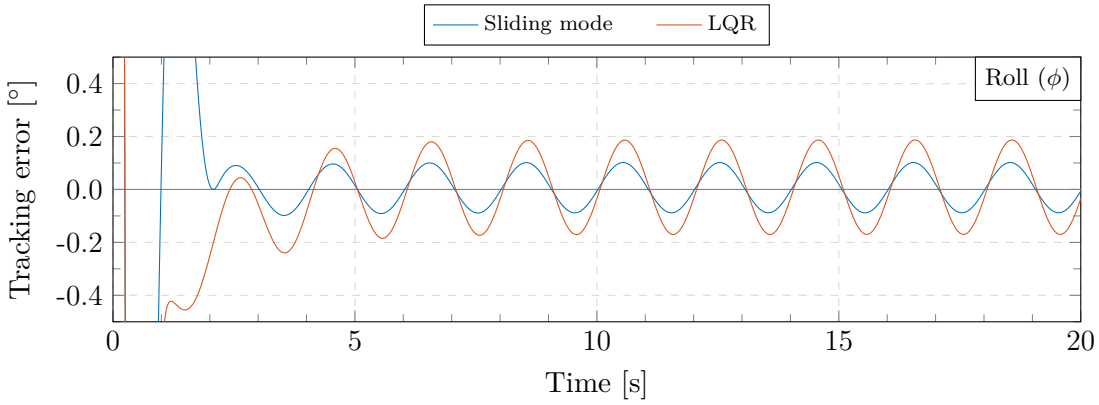


Figure 13.2: Comparison of the roll tracking error during the sine wave simulation of the sliding mode and the LQR balance controller

A chirp simulation, similar to the one in Section 10.5, is carried out to test the bandwidth and verify if the system is limited to the same frequency, as with the sliding mode controller. The roll angle reference is set to a sine of $\pm 2°$ in amplitude, starting at a frequency of $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.05$ Hz/s. The simulated response is shown in Figure 13.3.

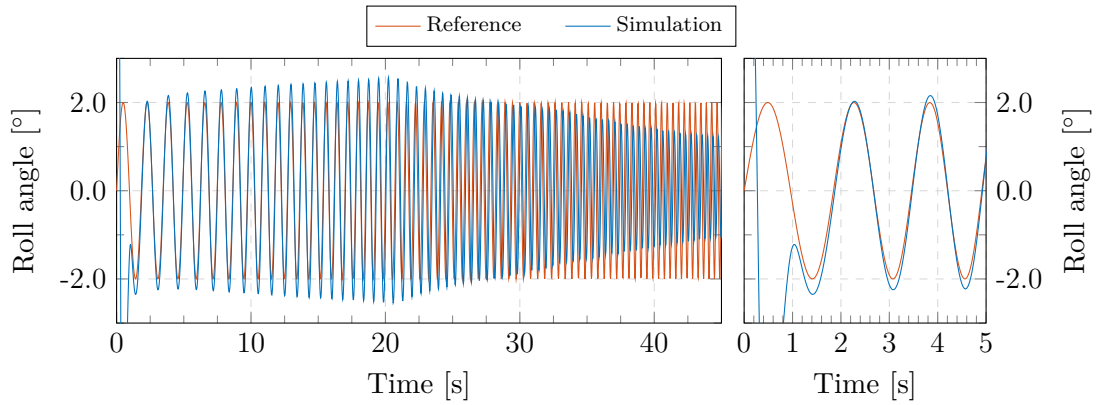


Figure 13.3: Euler angles during the roll chirp simulation with a frequency starting from $f_0 = 0.5$ Hz, increasing with a rate of $k = 0.05$ Hz/s and an amplitude of $\pm 2°$

The response is similar to Figure 10.7, with an overshoot that increases until the torque saturates at 20 s, equal to the same frequency of 1.5 Hz.

From these results the LQR controller is deemed to perform just as good as the sliding mode controller. So for a nominal system, where the actual system parameters match the controller design, the LQR performs just as good.

13.1.1 Changed mass and inertia

However, if the mass and inertia are changed, as they were in Section 10.3, adding 2 kg to the body mass, M_b, and scaling the body inertia with a factor of 2, the LQR controller is not able to stabilize the plant with the same initial conditions. With the changed parameters the initial conditions can be reduced to reach a point where the LQR controller is again able to stabilize the system. With an initial condition of $\phi = 2°$, $\theta = -2°$ and $\psi = 2°$ the LQR controller is able to stabilize the changed system. A similar sine wave simulation is then carried out with the sliding mode balance controller and the LQR balance controller, both simulated with the changed body mass and body inertia. The comparison of the roll tracking error is shown in Figure 13.4.

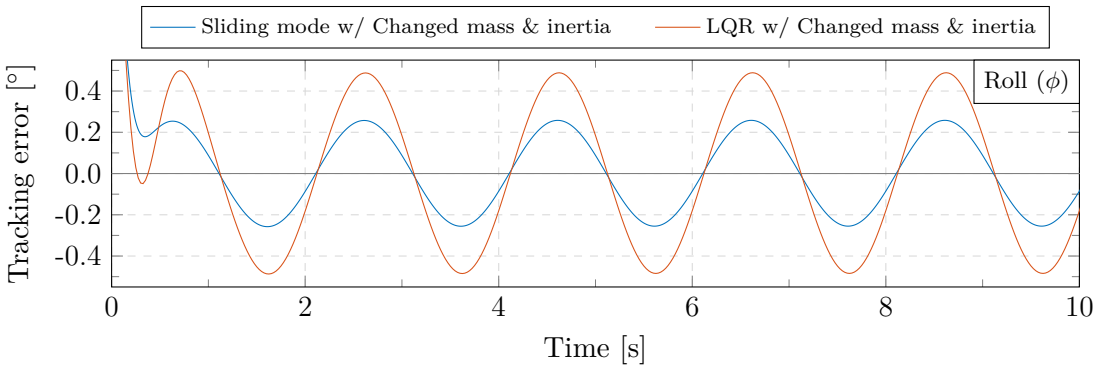


Figure 13.4: Comparison of tracking errors during sine wave simulation with changed mass and inertia.

With the changed parameters the tracking error seems to increase proportionally for both controllers. Here it is important to note that with the ϵ-tube the sliding mode controller effectively becomes a linear controller when the sliding variable is inside the ϵ-tube. This possibly explains the relationship between the increased tracking error.

13.2 Test comparison

The LQR controller is implemented in the embedded firmware alongside with the sliding mode controller, so that a software switch can be used to switch between the controllers. The code for the LQR controller can be found at `Kugle-Embedded/KugleFirmware/Libraries/Modules/Controllers/LQR/LQR.cpp`.

A balance test, similar to the upright balance test in Section 12.2, is carried out by setting the quaternion reference to the upright unit quaternion and the angular velocity reference to zero. A comparison of the LQR controller and the sliding mode controller with aggressive gains, is shown in Figure 13.5.

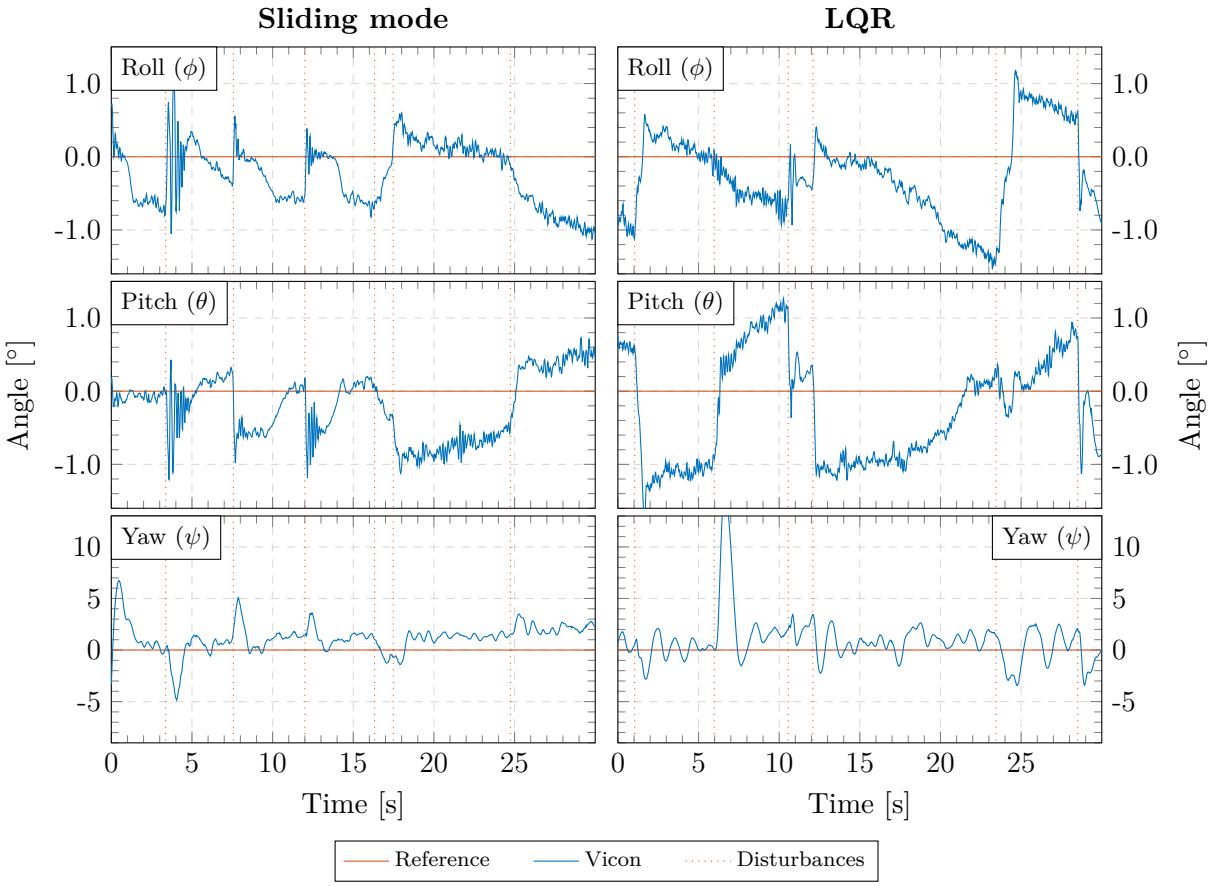


Figure 13.5: Comparison of the non-linear sliding mode controller and LQR controller during the upright balance test with disturbance pushes

13.3 Performance conclusion

The results in this chapter shows how the sliding mode controller compares to an equivalent LQR controller. The results shows an almost identical tracking performance between the sliding mode controller and LQR controller. With regard to whether the sliding mode controller is better than the LQR controller, depends on the actual operating conditions, which requires more extensive tests on the actual system with e.g., changing references and variations in the model parameters and disturbances.

One thing is clear though. Since the inclination operating envelope is fairly small, as the desired accelerations of the ballbot is bounded and should be kept small, the system will always be within close proximity of the linearization point in terms of angle. Velocity dependent terms will potentially influence the dynamics when driving faster. However, at the speeds tested in this thesis and during normal station keeping, a well-tuned LQR controller for a single ballbot, will perform just as well or even better than a generally tuned non-linear sliding mode controller.

14 Velocity Controller

As a final chapter it is investigated how well the balance controller performs in a cascade configuration with an LQR-based velocity controller as described in Section 7.5. A similar situation would be the case when the MPC controller, described in Appendix S, is combined with the balance controller. This chapter thus serves as a validation of the cascaded structure shown in Figure 7.2, illustrated with the MPC in Figure 2.8.

One of the advantages of the ballbot, namely being a shape-accelerated robot, is also one of the main disadvantages when it comes to testing. In all previous tests of the balancing controller, the position and velocity had to be corrected by pushing the robot, acting as an external disturbance. Without a controller closing the loop around the translational dynamics, involving at least the velocity, the ballbot has an inherent run-away problem if e.g., the center of mass is not aligned exactly. The run-away problem with the COM was simulated and described in Section 10.6, where it was shown that even if the balance controller is able to achieve perfect tracking, a misaligned COM leads to an acceleration in a given direction due to gravity pulling it down.

When the COM is aligned it is positioned right above the center of the ball. To keep a constant velocity or to hold the position, the inclination and thus the quaternion reference sent to the balance controller, should be adjusted accordingly until the alignment is achieved. This is where the velocity controller, derived in Appendix Q, comes into play. The simulation of the velocity controller can be found at `Kugle-MATLAB/Simulation/KugleSim_VelocityLQR.slx` while the implementation can be found at `Kugle-Embedded/KugleFirmware/Libraries/Modules/Controllers/VelocityLQR/VelocityLQR.cpp`.

14.1 Objective

The velocity controller is a linear optimal controller, LQR, designed on the closed loop dynamics including the sliding mode balance controller, see Appendix Q.2. The controller is designed to stabilize the velocity to a given reference, ${}^{\text{I}}\dot{x}_{\text{ref}}$ and ${}^{\text{I}}\dot{y}_{\text{ref}}$, and includes an integral term, equivalent to a position error feedback, that enables the controller to capture and compensate for COM misalignments. Since the integral term is equivalent to position error it can also be used for position control and thus station keeping. The controller outputs an angular velocity reference for the x- and y- body axes, which are then integrated to the quaternion equivalent of a roll and pitch reference. Thereby it leaves the heading angle, ψ, and angular velocity, ${}^{\tilde{\text{B}}}\omega_z$, as a degree of freedom. Note that both the angular velocity reference output and the roll and pitch references are saturated to avoid unreasonably large references. Since the controller is based on linearized dynamics it operates in the heading frame, rotating both the position error and velocity error into the heading frame, see also Appendix R.

14.1.1 Mode switching

Depending on the velocity reference the controller automatically switches between station-keeping mode and velocity control mode. Initially when the controller is turned on the integral term is enabled to stabilize the position of the ballbot by adjusting the inclination accordingly. Whenever a non-zero velocity reference is set the controller automatically switches into velocity control mode where the error is no longer integrated but the integral term is included in the control law. When the velocity reference goes back to zero the current position is stored as a position reference and the controller goes into station-keeping mode. In station-keeping mode the position error is added on top of the integral term to thereby hold the position.

14.2 Gain selection

When using the sliding mode balance controller in the cascade configuration with the velocity controller, it has been necessary to lower the sliding mode controller gains to make it less aggressive, as mentioned Appendix O.1. Initially the controller was designed and tested with the aggressive gains, but even with lowered gains in the velocity controller, the closed loop system became very sensitive to external disturbances e.g., a push. The system performance was greatly improved by decreasing the sliding mode controller gains and increasing the velocity controller gains and finally adding a low-pass filter to the angular velocity reference output, before integrating it to roll and pitch angles.

Decreasing the gain of the inner controller in a cascaded configuration is, however, against normal practice, which recommends a high inner loop gain to achieve a high inner loop bandwidth. However, in this specific setup, with the sliding mode balance controller and the problems with wheel slippage at rapid changes in the direction of torque, optimal behaviour was achieved by lowering the inner loop gains. As a consequence of the lowered gains the balance controller achieves less perfect tracking, which can lead to oscillatory behaviour in the outer loop involving an integral term. To limit these oscillations the integral term of the velocity controller is only updated right after being turned on until the very first movements.

Not only does this velocity LQR enable velocity control and station-keeping of the ballbot, but in general the controller enables proper closed-loop testing of the balance controller without the otherwise inevitable disturbances, see Figure 12.1. In the remainder of this chapter two tests are carried out of the velocity controller, one test for station-keeping and another test for velocity control using a wireless joystick to set the reference. Due to the cascaded configuration it is important to emphasize that the following tests serves as a joint test of both the velocity controller and balance controller, extending the reference tracking tests from Chapter 12 to the arbitrary continuously changing references set by the velocity controller.

14.3 Station-keeping test

The first test is a station-keeping test where the controller stabilizes the position of the ballbot by adjusting the inclination accordingly. The position response is shown in Figure 14.1 including a few times where the controller is disturbed by manually moving the robot away.

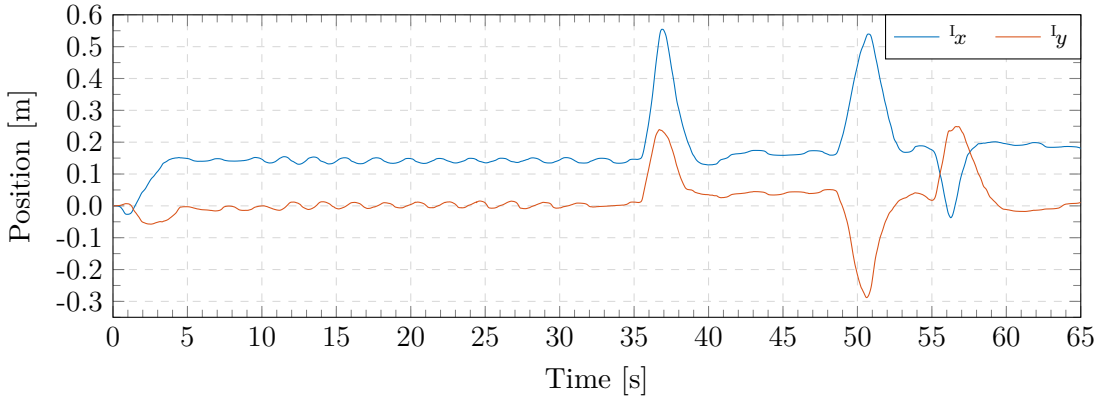


Figure 14.1: Velocity controller in station-keeping mode with non-aggressive sliding mode controller gains. The position stabilizes during initialization and is then held constant even when manually disturbed at 35 s, 48 s and 55 s

It takes a short while for the controller to stabilize after being started. After around 5 s the controller is stabilized and goes into a small position limit cycle of up to ±1 cm. The generated roll and pitch angle references are shown in Figure 14.2. By comparison with the actual inclinations it appears that the limit cycles are likely due to the less aggressive sliding mode balance controller which does not track the reference exactly, leading to a small lag.

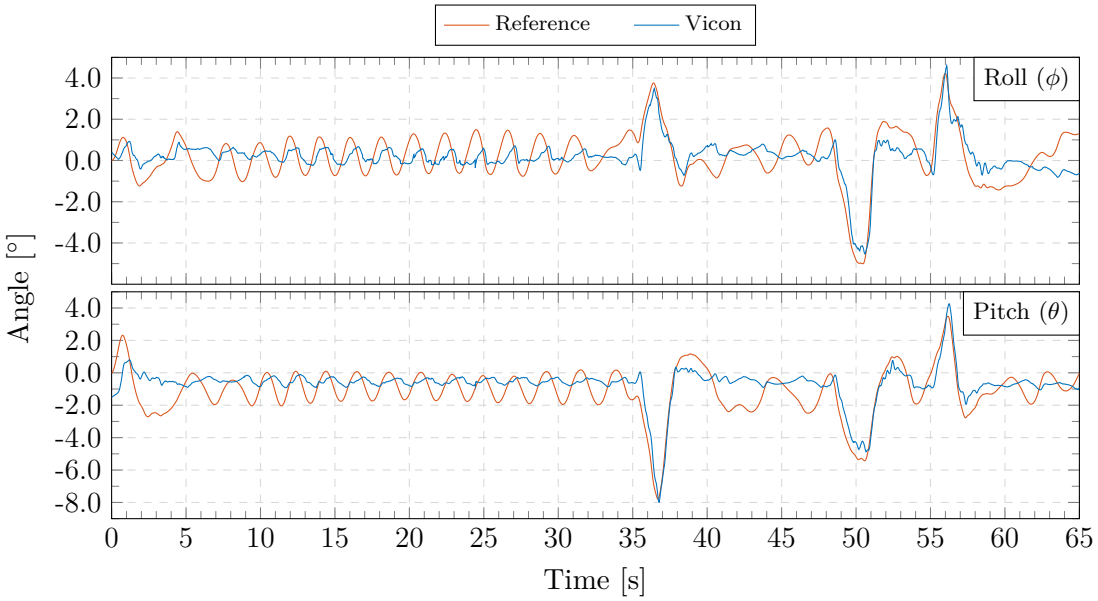


Figure 14.2: Angle reference output of the velocity controller compared to the actual inclination of the ballbot during station-keeping test with non-aggressive sliding mode controller gains.

The limit cycles in position are noticeable but yet so small that they do not affect the overall performance, except for increasing the energy consumption. In general the limit cycles make the ballbot look alive. When disturbed the velocity controller reacts quickly and the balance controller tracks the compensating angle references well. This stabilizes the ballbot which finds its way back without overreacting.

14.3.1 Aggressive gain comparison

Even though the aggressive gains are not used with the velocity controller, a careful test is made to see how the aggressive gains affect the limit cycles. The compared position responses during the limit cycles are shown in Figure 14.3. With the aggressive gains the limit cycles become faster in frequency and slightly smaller in amplitude.

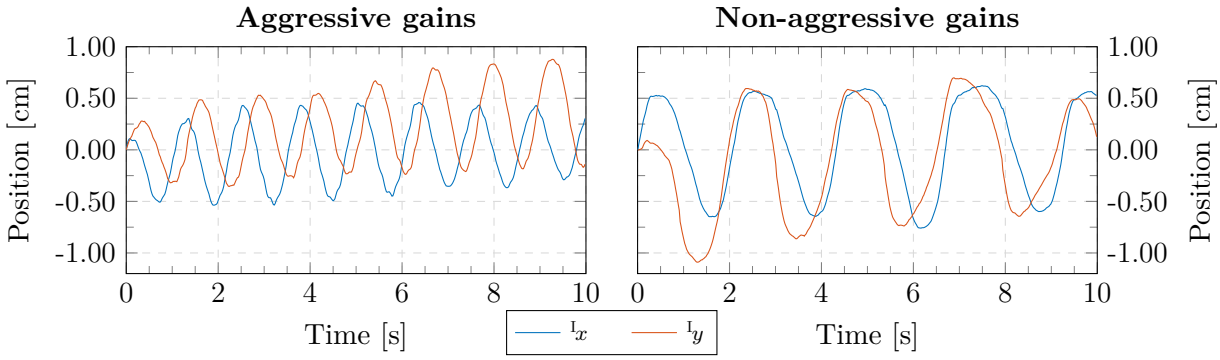


Figure 14.3: Comparison of position limit cycle in station-keeping mode when using aggressive and non-aggressive sliding mode controller gains.

14.4 Velocity test

The main purpose of the velocity controller is to track a given translational velocity and yaw angular velocity reference. The second test goes through a sequence of velocity references, set manually with a wireless joystick, see Figure 14.4. Note that the velocity references are both set and visualized in the heading frame.

Starting with a step on the yaw angular velocity, the balance controller overshoots slightly but keeps a steady rate while keeping the translational velocity close to zero. For the translational velocity steps in the sequence, right, left, forward, backward, the velocity controller clearly lags behind and it takes a while for the system to stabilize on the velocity reference.

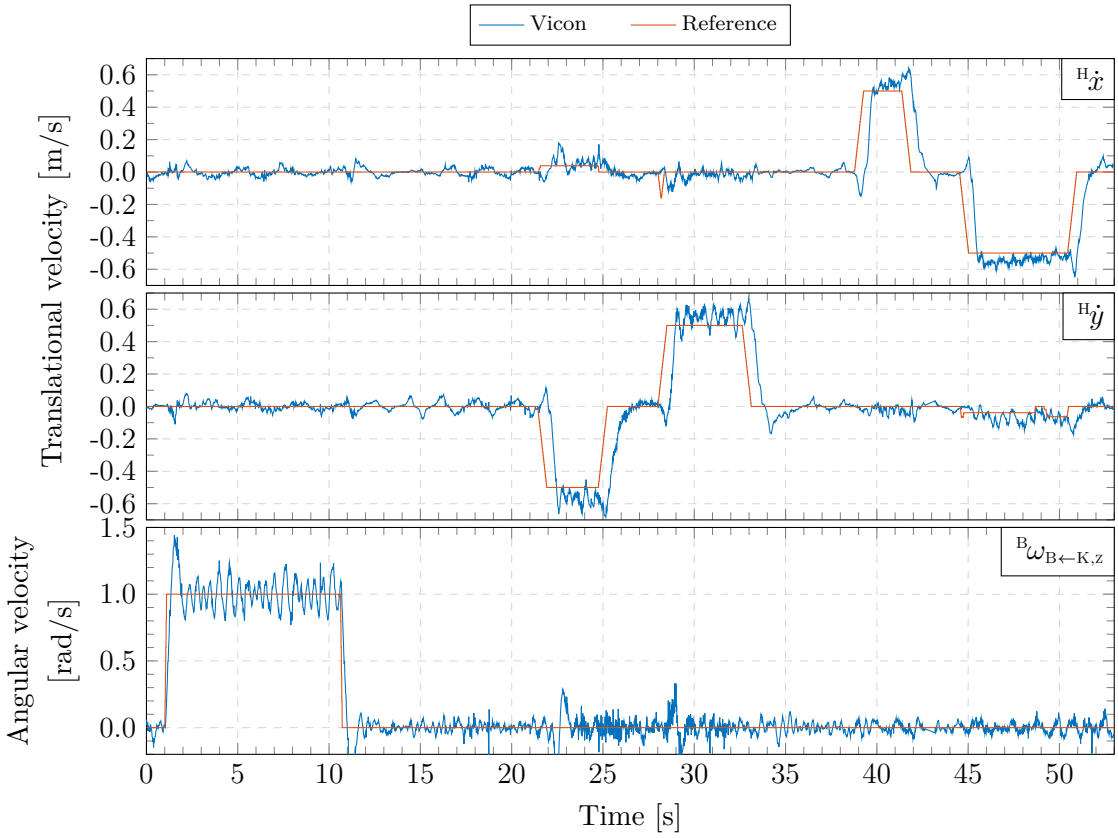


Figure 14.4: Translational heading velocity during velocity test with references are given through a joystick in the sequence: turn, right, left, forward, backward

The lag is mainly due to the non-minimum phase characteristics of the ballbot, which is revealed by zooming in on the translational velocity plot, see Figure 14.5. The non-minimum phase characteristic requires the ballbot to go in the opposite direction of the intended direction of acceleration every time the velocity reference changes. This leads to a fairly slow response time of the velocity.

The slow closed loop response is, however, expected for a non-minimum phase system like this and for a system where the inner loop controller is less aggressive and do not achieve perfect tracking.

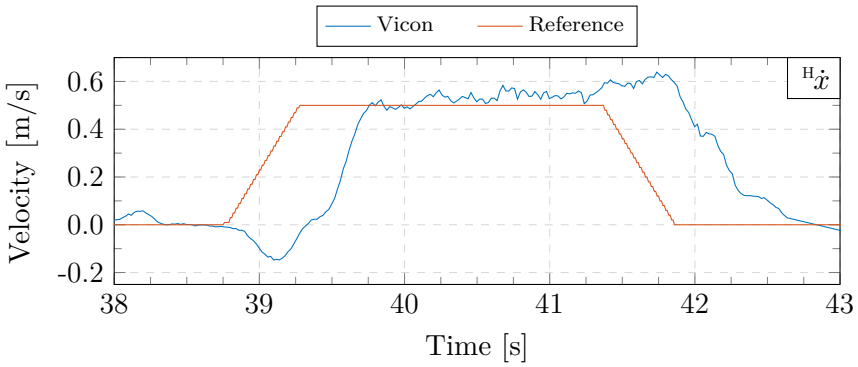


Figure 14.5: Zoomed plot of translational velocity in the forward direction (x-axis of heading frame) reveals both the non-minimum phase characteristic and the big lag

14.4.1 Driving in a circle

Finally, a circle test, shown in Figure 14.6, is carried out to test the translational and rotational velocity of the ballbot. The velocity references are limited to the operating envelope specified in Section 7.2, thus a maximum translational velocity of 0.5 m/s and a maximum angular velocity of 1 rad/s.

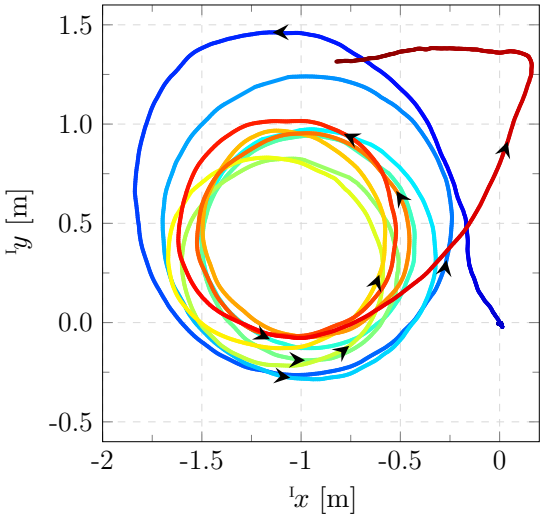


Figure 14.6: 2D plot of the Vicon-based ballbot position during the circle test. The trajectory has been colour-coded with time for distinguishability, going from blue, to green, to red.

Again the references are set and visualized in the heading frame in Figure 14.7. When driving fast while turning, the controller does not achieve as good tracking on the velocity. Even though the velocity reference along the y-axis is non-zero this reference is tracked poorly.

The controller is capable of reaching the defined operating limits and achieves sufficient tracking to make the ballbot drive smoothly, even at fast velocities.
Looking at how well the balance controller tracks the angle references while driving fast, see Figure 14.8, reveals how the tracking performance of the balance controller seems to degrade with the velocity.

This suggests the need of adaptive gains in the sliding mode controller, e.g., an adaptive switching gain, $\boldsymbol{\eta}$, or an adaptive ϵ-tube, as mentioned in Appendix L. Furthermore, the inclusion of friction components in the nominal model should help to cancel out the increased friction torque from driving. This will, however, require that the friction coefficients are estimated first.

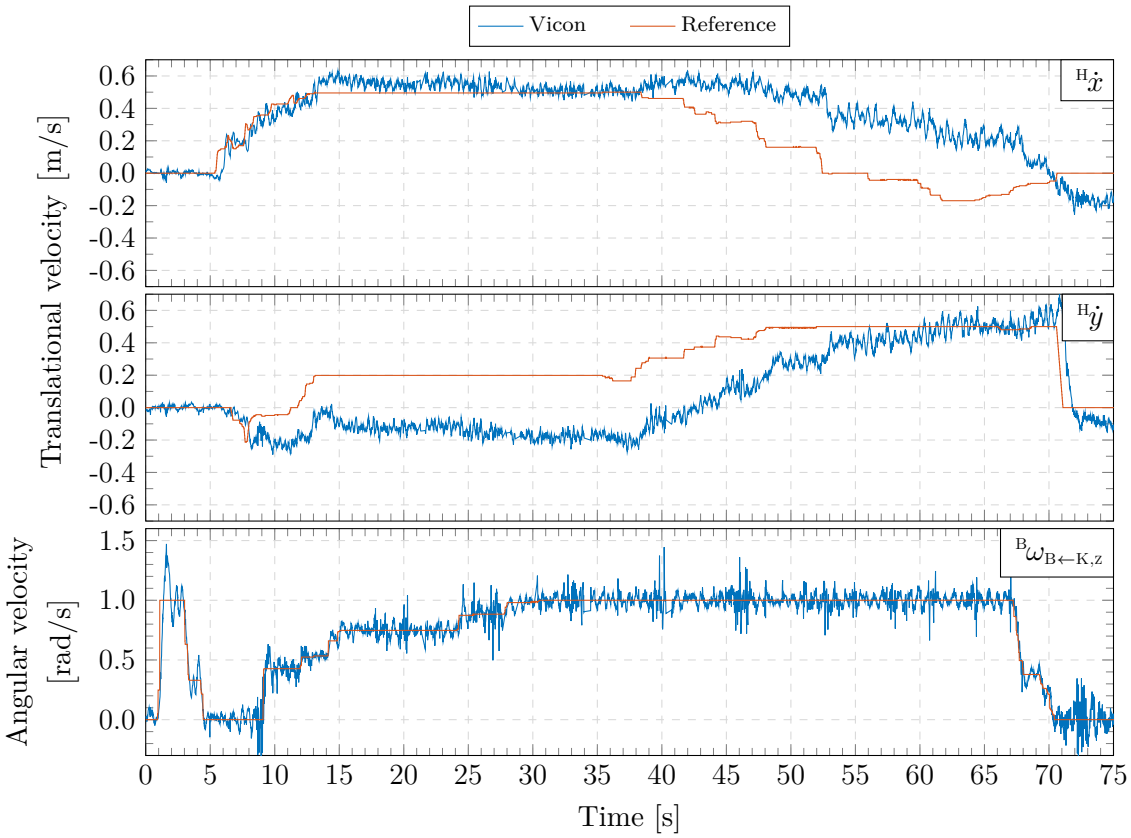


Figure 14.7: Translational and angular velocity during circle test

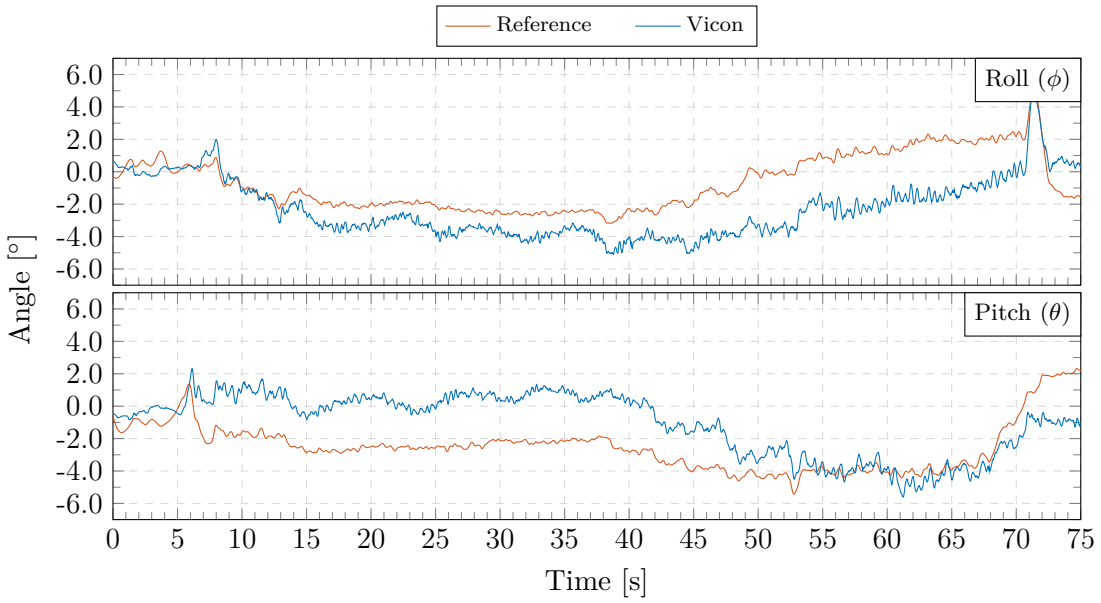


Figure 14.8: Euler angle comparison during the circle test, illustrating how the balance controller tracking error seems to increase with velocity.

This concludes the tests of the velocity controller and verifies that the balance controller is capable of stabilizing the system within the operating envelope but with a tracking error that seems to increase with velocity.

15 Conclusion

The purpose of this thesis was to investigate if it is possible to derive a quaternion-based model of a ballbot using Lagrangian mechanics and use it for the design of a non-linear sliding mode controller for stabilization of the balance and orientation. Both the model and controller design were tailored towards an actual prototype, the 16 kg Kugle V1, including geared brushless motors, single-ring omniwheels, Xsens IMU and SICK LiDARs.

The ballbot was modelled as two rigid bodies using six generalized coordinates to describe the five degrees of freedom in the system, namely the orientation of the body and the position of the ball. The quaternion model was derived by defining three kinetic and one potential energy function, which are combined into the Lagrangian. Motor torques were included as input forces using the Jacobian of the inverse kinematics, which had been verified with the prototype using a motion capture system. Viscous friction forces were included on the translational velocities and angular velocities. Since the generalized coordinates were dependent, the quaternion unit norm constraint was applied to the equations of motion through a Lagrange multiplier. The result was a control-affine non-linear ODE with 12 states derived as a symbolic model.

A design specification narrowed down the scope of the sliding mode balance controller to an operating envelope of 0.5 m/s and 1 rad/s, and an inclination tracking error below 1° and a heading tracking error below 5°.

The sliding mode controller was designed in two steps, a sliding surface design and a control law design, both based on the continuous model. The sliding surface was defined based on the quaternion tracking error in the body frame. Two surfaces were proposed, one relating the body angular velocity to the quaternion error, and another relating the quaternion error to its derivative. The quaternion derivative surface was chosen due to simplicity and performance. The control law consisted of an equivalent control law and a continuous switching law. The quaternion scalar value and translational dynamics were neglected to make the model invertible. The equivalent control was designed to cancel the nominal model using inverse dynamics. A continuous switching law with the saturation function and an ϵ-tube was chosen to reduce wheel-slippage due to rapid changes in torque. This resulted in a sliding mode controller that converges to a linear controller inside the tube.

The balance controller was verified through simulation with the non-linear model, sensor models and estimators. In simulation the balance controller was capable of tracking angle references with less than 0.1° error on a nominal system. From a roll angle chirp test the simulation revealed how the bandwidth of the system was limited to 1.5 Hz, mainly due to saturations in torque.

An object-oriented embedded firmware was implemented on the STM32H7 ARM Cortex-M7 microprocessor of the Kugle V1 prototype, including all necessary controllers, estimators and device drivers. The controllers and estimators were implemented at a sample rate of 200 Hz. Since the quaternions embed the sine and cosine conversions internally, the quaternion-based symbolic model could quickly be evaluated, leading to a control loop computation time of 420 µs on the microprocessor.

The implementation was verified in practice through a series of tests. With zero references the controller was capable of stabilizing the ballbot with inclination deviations kept below 1°. The step test revealed that step references on roll and pitch should generally be avoided, since they resulted in large overshoots and saturation in torque likely to cause wheel-slip. Step references on yaw worked well with minimal overshoot of less than 5°. The sine wave test confirmed that the controller worked with a changing reference, with a tracking error of less than 1°. Oscillations of 7 Hz were, however, present around the reference, probably due to unmodelled dynamics. The roll angle chirp test resulted in motor saturation at a frequency of approximately 1.5 Hz, similar to the simulation results. The rotating inclination test, equivalent to a chirp test on both roll and pitch, revealed how the bandwidth was limited to 1 Hz when both references were changing simultaneously.

The sliding mode controller was compared to an LQR balance controller which yielded almost identical performance. The small difference could probably be explained by tuning. The conclusion was that a well-tuned LQR controller performs just as well or even better than the generally-tuned sliding mode controller with an ϵ-tube. To give a conclusive answer on whether a sliding mode controller is better than an LQR controller for ballbots, more extensive tests, especially across multiple plants, will have to be carried out.

Finally, a velocity controller was designed to stabilize the velocity and position of the ballbot, simultaneously compensating for a misaligned center of mass. From a test with the ballbot driving in a circle, the operating limits of 0.5 m/s and 1 rad/s were confirmed to work in practice. However, it appeared that the tracking error of the balance controller grew with the velocity, probably due to the less aggressive gains.

The overall conclusion to the problem formulation is thus that it is possible to derive a quaternion-based model of a ballbot using Lagrangian mechanics. It is also shown that the model can be used for non-linear controller design using sliding mode. Due to space and time constraints the derivation of the shape-accelerated path-following MPC is put in the appendix, where simulations verified that the MPC was capable of following a given reference path, leaving time along the path as a degree of freedom. Even though the quaternions simplify some of the equations making the evaluation faster, the added complexity and the fairly narrow operating envelope, close to upright, makes it less attractive to use quaternions in ballbot models. Instead it is recommended to use an Euler-angle model since the roll and pitch angles would never come close to the singularities.

All code is available on GitHub and the hope is thus that this work will contribute towards new ballbot prototypes, but mostly that the work will be useful to future students and researchers dealing with ball-balancing robots.

16 Future Work

Even though this thesis is a long deep-dive into the non-linear model and controller design of a ball-balancing robot, it is far from complete. Some of the topics, problems and concerns that were noted during the project period are mentioned below as possible future work. Some of these come from observations while others come from theory.

The first thing to try would definitely be to test the simulated MPC on the actual robot and verify if the simplifying assumptions used in the MPC model hold in practice.

As regards the model, the assumption that the ball does not rotate around the vertical axis due to friction has been proven wrong several times during development. Whenever the controller tries to rotate around the yaw axis, the generated torque will be split between the ball and body, according to Newton's third law of motion. Since the ball has only one contact point with the ground, the Coulomb friction and stiction around the vertical axis is fairly limited, and the ball will thus spin up. Since the inertia of the body is more than six times larger than the inertia of the ball, the ball will accelerate six times faster than the body resulting in the ball spinning up quickly. This is yet another reason that the yaw gains of the sliding mode controller are tuned less aggressive than the roll and pitch gains, since a fast spinning ball makes wheel-slip more likely to occur. The model should thus be extended to include the ball rotation around the inertial z-axis. This should be a fairly simple task but obviously requires an extension of the state space and some considerations when it comes to applying the inverse dynamics.

Related to the chattering problems, oscillations around the reference and wheel-slip due to rapid changes in torque, one can consider putting a rate limitation on the torque output. This will, however, require careful tuning of the sliding mode controller, which is why other options such as integral sliding mode, ISMC, the super-twisting algorithm or higher order sliding mode controllers, HOSM, should be considered [67] [68] [69]. The benefit of these adaptations is a far more continuous control signal, thus with low excitation of the unmodeled dynamics.

The switching gains and ϵ-tube of the sliding mode controller should also be made time varying, so that the gains are made more aggressive while driving and less aggressive when the controller is doing station-keeping. In terms of navigation, which is also related to the model predictive controller, a ballbot should be compliant when it drives around. This again requires that the gains are adaptive to prevent undesired and uncontrollable movement if the ballbot accidentally rolls into an object, causing a large disturbance. Similarly, if anybody starts pushing the robot, it should drive away smoothly instead of aggressively bouncing back.

The continuous switching law in the sliding mode controller was chosen as the common saturation function even though several other options exist. It is worth exploring the benefits of choosing other smooth switching laws, such as the tanh function or $\frac{s(t)}{|s(t)|+\lambda}$.

During the velocity controller tests it was noted how the tracking performance of the balance controller degraded with increasing velocity. This is probably due to a combination of the chosen gains and the nominal model parameters used to cancel the dynamics in the equivalent control law. Especially the friction components, currently set to zero in the nominal model, should be determined.

In general a more extensive motor parameter identification should be carried out since it is unknown how much the motor-current controller, within the ESCON motor drivers, affects the closed loop performance. Furthermore, in terms of simulation, the motor model should be extended to include both backlash and Coulomb friction.

When it comes to the velocity controller, another option that should be investigated is to combine the velocity controller into the sliding mode controller as a hierarchical sliding mode. By augmenting the sliding surface to include the velocity error, it is be possible to put both velocity tracking and balance stabilization into the same sliding mode controller. Some initial tests have already been carried out but with limited success.

The quaternion estimator, the QEKF, is derived and implemented with the brute-force normalization method. Even though it is shown to yield decent estimation accuracy, the estimator should be changed to a multiplicative version that takes care of the unit norm constraint correctly. Furthermore, the assumption of negligible body accelerations in the accelerometer sensor model probably does not hold well. One of the key relationships with the ballbot is the shape-accelerated property, hence any inclination will also lead to an acceleration. Therefore, the only time when the sensor model is valid is when the ballbot is perfectly upright as it would otherwise accelerate. The lack of acceleration in the sensor model probably leads to a bias in the angle estimate. It has also been observed how the acceleration from a sudden push affects the gyroscope bias estimate incorrectly.

Finally, it should be noted that the center of mass is clearly observable from the states and sensor measurements due to the shape-accelerated relationship between inclination and acceleration. Instead of dealing with a misaligned COM with the integral term of the velocity controller it should be moved out in an estimator, as shown in the system architecture diagram, so that the nominal model in the sliding mode controller can also benefit from this estimate [70].

Bibliography

[1] Aalborg University. Robot Digital Signage, February 2018. URL http://vbn.aau.dk/da/projects/robot-digital-signage(f0c6a2d4-c8de-423b-86b5-e7c65d134fa0).html.

[2] Corsin Gwerder and Péter Fankhauser. *Modeling and Control of a Ballbot*. PhD thesis, ETH Zürich, 2010. URL http://hdl.handle.net/20.500.11850/154271.

[3] T.B. Lauwers, G.A. Kantor, and R.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.*, pages 2884–2889, Orlando, FL, USA, 2006. IEEE. ISBN 978-0-7803-9505-3. doi: 10.1109/ROBOT.2006.1642139. URL http://ieeexplore.ieee.org/document/1642139/.

[4] Segway. URL http://www.segway.com.

[5] Umashankar Nagarajan. Dynamic Constraint-based Optimal Shape Trajectory Planner for Shape-Accelerated Underactuated Balancing Systems. In *Robotics: Science and Systems 2010*, page 8, 2010. URL http://www.roboticsproceedings.org/rss06/p31.pdf.

[6] Lars Bojer Kanstrup. Revolutionerende Robotsamarbejde. *Combine*, February 2018. URL https://combine.dk/nyheder/combine-i-revolutionerede-robotsamarbejde/.

[7] Uffe Koch. Danske Roboteksperter på vild mission: Sådan kan du i 2020 bygge dine egne robotter. *Det Gode Firma*, January 2018. URL http://detgodefirma.dk/danske-roboteksperter-paa-vild-mission-saadan-kan-du-2020-bygge-dine-egne-robotter/.

[8] Søren Østergaard. Kuglerobot kan få nordjysk erhvervseventyr til at rulle. *Nordjyske*, January 2018. URL https://nordjyske.dk/plus/kuglerobot-kan-faa-nordjysk-erhvervseventyr-til-at-rulle/5153d5c7-6d40-4972-8f08-e6ab7f7d6e46.

[9] Lars Bojer Kanstrup. Udvikles i Aalborg: Fremtidens robot skal spare samfundet millioner. *migogaalborg*, February 2018. URL https://migogaalborg.dk/udvikles-i-aalborg-fremtidens-robot-skal-spare-samfundet-millioner.

[10] Electronic Supply. Millioner til kuglerobot. *Electronic Supply*, January 2018. URL https://www.electronic-supply.dk/article/view/579658/millioner_til_kuglerobot.

[11] Esben Hardenberg. Vejviser, vagt og videoafspiller: Nu kommer hjælperobotterne. *DR*, April 2018. URL https://www.dr.dk/nyheder/viden/teknologi/vejviser-vagt-og-videoafspiller-nu-kommer-hjaelperobotterne.

[12] Laurids Hovgaard. Jysk robot skal servere apps på et sølvfad. *Ingeniøren*, February 2018. URL https://ing.dk/artikel/jysk-robot-skal-servere-apps-paa-soelvfad-210730.

[13] Xsens. URL https://www.xsens.com/.

[14] Masaaki Kumagai and Takaya Ochiai. Development of a robot balancing on a ball. In *2008 International Conference on Control, Automation and Systems*, pages 433–438, Seoul, South Korea, October 2008. IEEE. ISBN 978-89-950038-9-3 978-89-93215-01-4. doi: 10.1109/ICCAS.2008.4694680. URL http://ieeexplore.ieee.org/document/4694680/.

[15] Cees Verdier. *Geometric Control of an Under-Actuated Balancing Robot*. PhD thesis, TU Delft, 2015. URL http://resolver.tudelft.nl/uuid:30f06d51-fba8-4be5-a6e1-38ec964fef51.

[16] 48mm Omniwheel (compatible with Servos and Lego Mindstorms NXT). URL https://www.robotshop.com/uk/48mm-omniwheel-compatible-servos-lego-mindstorms-nxt.html.

[17] 60mm Aluminum Omni Wheel - RobotShop. URL https://www.robotshop.com/uk/60mm-aluminum-omni-wheel.html.

[18] 100mm stainless steel rollers omni wheel for ball balance ballbot | US $143.0. URL https://www.aliexpress.com/item/100mm-stainless-steel-rollers-omni-wheel-for-ball-balance-ballbot-14183/32717239477.html.

[19] 4inch(100mm) single aluminum omni wheel | US $143.0. URL https://www.aliexpress.com/item/4inch-100mm-single-aluminum-omni-wheel-14179/32442408244.html.

[20] Field Oriented Control. URL https://www.trinamic.com/technology/std-technologies/field-oriented-control/.

[21] U. Nagarajan, G. Kantor, and R.L. Hollis. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *2009 IEEE International Conference on Robotics and Automation*, pages 3743–3748, Kobe, May 2009. IEEE. ISBN 978-1-4244-2788-8. doi: 10.1109/ROBOT.2009.5152624. URL http://ieeexplore.ieee.org/document/5152624/.

[22] Ali Nail Inal. *3D DYNAMIC MODELING OF A SPHERICAL WHEELED SELF-BALANCING MOBILE ROBOT*. Master's Thesis, Bilkent University, August 2012. URL https://www.semanticscholar.org/paper/3D-DYNAMIC-MODELING-OF-A-SPHERICAL-WHEELED-MOBILE-

Inal/f50041b0117f6ed53b59e171e0bed0fc884f00d9.

[23] A. Lotfiani, M. Keshmiri, and M. Danesh. Dynamic analysis and control synthesis of a spherical wheeled robot (Ballbot). In *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 481–486, Tehran, February 2013. IEEE. ISBN 978-1-4673-5811-8 978-1-4673-5809-5 978-1-4673-5810-1. doi: 10.1109/ICRoM.2013.6510154. URL http://ieeexplore.ieee.org/document/6510154/.

[24] JIAMIN WANG. *RESEARCH & DEVELOPMENT OF Q-BALLER - A SPHERICAL WHEELED ROBOT*. PhD thesis, University of Missouri-Columbia, May 2017. URL https://mospace.umsystem.edu/xmlui/bitstream/handle/10355/63386/research.pdf?sequence=1.

[25] Koos van der Blonk. *Modeling and Control of a Ball-Balancing Robot*. PhD thesis, University of Twente, July 2014. URL https://essay.utwente.nl/65559/1/vanderBlonk_MSc_EEMCS.pdf.

[26] Peter I. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Number v. 73 in Springer Tracts in Advanced Robotics. Springer, Berlin, 2011. ISBN 978-3-642-20143-1 978-3-642-20144-8. OCLC: ocn754988355.

[27] Karsten Groÿekatthöfer and Zizung Yoon. Introduction into quaternions for spacecraft attitude representation. Technical report, Technical University of Berlin, Department of Astronautics and Aeronautics, May 2012. URL http://www.tu-berlin.de/fileadmin/fg169/miscellaneous/Quaternions.pdf.

[28] Philipp Allgeuer and Sven Behnke. Fused Angles and the Deficiencies of Euler Angles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5109–5116, Madrid, October 2018. IEEE. ISBN 978-1-5386-8094-0. doi: 10.1109/IROS.2018.8593384. URL https://ieeexplore.ieee.org/document/8593384/.

[29] Aykut C. Satici, Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. Intrinsic Euler-Lagrange dynamics and control analysis of the ballbot. In *2016 American Control Conference (ACC)*, pages 5685–5690, Boston, MA, USA, July 2016. IEEE. ISBN 978-1-4673-8682-1. doi: 10.1109/ACC.2016.7526560. URL http://ieeexplore.ieee.org/document/7526560/.

[30] Ali Nail Inal, Omer Morgul, and Uluc Saranli. A 3D Dynamic Model of a Spherical Wheeled Self-Balancing Robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5381–5386. IEEE, October 2012. doi: 10.1109/iros.2012.6385689.

[31] Ronnie Dessort Enrico Pellegrini, Klaus J. Diepold and Heiko Panzer. 3D-Modeling of a Robot Balancing on a Ball. Technical report, TUM, August 2011. URL https://mediatum.ub.tum.de/doc/1081938/1081938.pdf.

[32] Umashankar Nagarajan, George Kantor, and Ralph Hollis. The ballbot: An omnidirectional balancing mobile robot. *The International Journal of Robotics Research*, 33(6):917–930, May 2014. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364913509126. URL http://journals.sagepub.com/doi/10.1177/0278364913509126.

[33] Pham Dinh Ba, Kim Hyung, Kim Jaejun, and Lee Soon-Geul. Balancing and Transferring Control of a Ball Segway Using a Double-Loop Approach. *IEEE Control Systems*, 38(2):15–37, April 2018. ISSN 1066-033X, 1941-000X. doi: 10.1109/MCS.2017.2786444. URL https://ieeexplore.ieee.org/document/8319541/.

[34] Michael Neunert, Farbod Farshidian, and Jonas Buchli. Adaptive Real-time Nonlinear Model Predictive Motion Control. page 6, 2014. doi: 10.1.1.713.5860. URL http://www.adrlab.org/archive/p_14_mlpc_mpc_rezero.pdf.

[35] 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*, pages 2951–2955, Chofu, August 2008. IEEE. ISBN 978-4-907764-30-2. doi: 10.1109/SICE.2008.4655168. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4655168.

[36] Dinh Ba Pham and Soon-Geul Lee. Aggregated Hierarchical Sliding Mode Control for a Spatial Ridable Ballbot. *International Journal of Precision Engineering and Manufacturing*, 19(9):1291–1302, September 2018. ISSN 2234-7593, 2005-4602. doi: 10.1007/s12541-018-0153-5. URL http://link.springer.com/10.1007/s12541-018-0153-5.

[37] Michael Neunert, Cedric de Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1398–1404, Stockholm, May 2016. IEEE. ISBN 978-1-4673-8026-3. doi: 10.1109/ICRA.2016.7487274. URL http://ieeexplore.ieee.org/document/7487274/.

[38] Umashankar Nagarajan and Ralph Hollis. Shape space planner for shape-accelerated balancing mobile robots. *The International Journal of Robotics Research*, 32(11):1323–1341, September 2013. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364913495424. URL http://journals.sagepub.com/doi/10.1177/0278364913495424.

[39] Michael Shomin. *Navigation and Physical Interaction with Balancing Robots*. PhD Thesis, Carnegie Mellon University, October 2016. URL https://www.ri.cmu.edu/wp-content/uploads/2018/01/Navigation-and-Physical-Interaction-with-Balancing-Robots-ilovepdf-compressed.pdf. CMU-RI-TR-16-58.

[40] A. Alaimo, V. Artale, C. Milazzo, and A. Ricciardello. Comparison between Euler and quaternion

parametrization in UAV dynamics. In *11TH INTERNATIONAL CONFERENCE OF NUMERICAL ANALYSIS AND APPLIED MATHEMATICS 2013: ICNAAM 2013*, pages 1228–1231, Rhodes, Greece, 2013. doi: 10.1063/1.4825732. URL http://aip.scitation.org/doi/abs/10.1063/1.4825732.

[41] Thomas E. Marlin. *Process Control: Designing Processes and Control Systems for Dynamic Perfomance*. McGraw-Hill, Boston, 2nd ed edition, 2000. ISBN 978-0-07-039362-2. URL http://pc-textbook.mcmaster.ca/Marlin-Ch14.pdf.

[42] Amcl - ROS Wiki. URL http://wiki.ros.org/amcl.

[43] Move_base - ROS Wiki. URL http://wiki.ros.org/move_base.

[44] Keith R. Symon. *Mechanics*. Addison-Wesley, 3rd edition edition. ISBN 978-0-201-07392-8. URL https://www.amazon.com/Mechanics-3rd-Keith-R-Symon/dp/0201073927.

[45] Jerry Ginsberg. *Engineering Dynamics*. Cambridge: Cambridge University Press, 3rd edition edition. ISBN 978-0-521-88303-0. doi: 10.1017/CBO9780511805899.

[46] Franz Hover and Michael Triantafyllou. Kinematics of moving frames. URL https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/course-text/MIT2_017JF09_ch09.pdf.

[47] Thomas Bak. Modeling of Mechanical Systems. February 2002.

[48] Alain J Brizard. *AN INTRODUCTION TO LAGRANGIAN MECHANICS*. First edition edition, July 2007. ISBN 978-981-4623-64-3. URL https://www.worldscientific.com/worldscibooks/10.1142/9321.

[49] C Karen Liu and Sumit Jain. A Quick Tutorial on Multibody Dynamics. Technical report, Georgia Institute of Technology. URL https://www.cc.gatech.edu/~karenliu/RTQL8_files/dynamics.pdf. GIT-GVU-15-01-1.

[50] C. T. Whelan. The Principles of Dynamics. Technical report, Cambridge, 2000. URL https://www.maths.cam.ac.uk/sites/www.maths.cam.ac.uk/files/pre2014/studentreps/ftp/podv0.1.pdf.

[51] David Tong. The Lagrangian Formalism. URL http://www.damtp.cam.ac.uk/user/tong/dynamics/two.pdf.

[52] Giancarlo Genta. Equations of Motion in the Configuration and State Spaces. In *Introduction to the Mechanics of Space Robots*. Springer Netherlands, Dordrecht, 2012. ISBN 978-94-007-1795-4 978-94-007-1796-1. doi: 10.1007/978-94-007-1796-1. URL http://link.springer.com/10.1007/978-94-007-1796-1.

[53] Shinichi Hirai. Analytical Mechanics: Rigid Body Rotation. URL http://www.ritsumei.ac.jp/se/~hirai/edu/2017/analyticalmechanics/handout/rotation.pdf.

[54] Firdaus E. Udwadia and Aaron D. Schutte. An Alternative Derivation of the Quaternion Equations of Motion for Rigid-Body Rotational Dynamics. *Journal of Applied Mechanics*, 77(4):044505, 2010. ISSN 00218936. doi: 10.1115/1.4000917. URL http://AppliedMechanics.asmedigitalcollection.asme.org/article.aspx?articleid=1420974.

[55] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1(1):1–16, June 1972. ISSN 00457825. doi: 10.1016/0045-7825(72)90018-7. URL https://linkinghub.elsevier.com/retrieve/pii/0045782572900187.

[56] Uri M. Ascher, Hongsheng Chin, Linda R. Petzold, and Sebastian Reich. Stabilization of Constrained Mechanical Systems with DAEs and Invariant Manifolds. *Mechanics of Structures and Machines*, 23(2): 135–157, January 1995. ISSN 0890-5452. doi: 10.1080/08905459508905232. URL http://www.tandfonline.com/doi/abs/10.1080/08905459508905232.

[57] Reza Olfati-Saber. *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles*. Ph.D. Thesis, Massachusetts Institute of Technology, February 2001. URL https://dspace.mit.edu/handle/1721.1/8979.

[58] Vicon Vero | Power. Flexibility. Motion Capture | VICON. URL https://www.vicon.com/products/camera-systems/vero.

[59] Gene F. Franklin, J. David Powell, Abbas Emami-Naeini, and H. S. Sanjay. *Feedback Control of Dynamic Systems*. Always Learning. Pearson, Boston, Mass., 7. ed., global ed edition, 2015. ISBN 978-1-292-06890-9 978-0-13-349659-8. OCLC: 931664403.

[60] J.-J. E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, N.J, 1991. ISBN 978-0-13-040890-7.

[61] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, N.J, 3rd ed edition, 2002. ISBN 978-0-13-067389-3.

[62] Rafal Wisniewski. *Satellite Attitude Control Using Only Electromagnetic Actuation*. Ph.D. Thesis, Aalborg University, Department of Control Engineering, December 1996. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.467.832&rep=rep1&type=pdf.

[63] FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions. URL https://www.freertos.org/.

[64] Karl Damkjær Hansen. LSPC: Lightweight Serial Package Communication. URL https://github.com/kdhansen/LSPC.

[65] Tf - ROS Wiki. URL http://wiki.ros.org/tf.
[66] Rviz - ROS Wiki. URL http://wiki.ros.org/rviz.
[67] Giorgio Bartolini, Antonella Ferrara, Arie Levant, and Elio Usai. On second order sliding mode controllers. In K.D. Young and Ü. Özgüner, editors, *Variable Structure Systems, Sliding Mode and Nonlinear Control*, volume 247, pages 329–350. Springer London, London, 1999. ISBN 978-1-85233-197-9. doi: 10.1007/BFb0109984. URL http://www.springerlink.com/index/10.1007/BFb0109984.
[68] Ming-Lei Tseng and Min-Shin Chen. Chattering reduction of sliding mode control by low-pass filtering the control signal. *Asian Journal of Control*, 12(3):392–398, February 2010. ISSN 15618625, 19346093. doi: 10.1002/asjc.195. URL http://doi.wiley.com/10.1002/asjc.195.
[69] Yongping Pan, Chenguang Yang, Lin Pan, and Haoyong Yu. Integral Sliding Mode Control: Performance, Modification, and Improvement. *IEEE Transactions on Industrial Informatics*, 14(7):3087–3096, July 2018. ISSN 1551-3203, 1941-0050. doi: 10.1109/TII.2017.2761389. URL https://ieeexplore.ieee.org/document/8064665/.
[70] Bhaskar Vaidya, Michael Shomin, Ralph Hollis, and George Kantor. Operation of the ballbot on slopes and with center-of-mass offsets. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2383–2388, Seattle, WA, USA, May 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139516. URL http://ieeexplore.ieee.org/document/7139516/.
[71] Umashankar Nagarajan, Anish Mampetta, George A. Kantor, and Ralph L. Hollis. State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot. In *2009 IEEE International Conference on Robotics and Automation*, pages 998–1003, Kobe, May 2009. IEEE. ISBN 978-1-4244-2788-8. doi: 10.1109/ROBOT.2009.5152681. URL http://ieeexplore.ieee.org/document/5152681/.
[72] Masaaki Kumagai and Takaya Ochiai. Development of a Robot Balanced on a Ball - First Report, Implementation of the Robot and Basic Control. *Journal of Robotics and Mechatronics*, 22(3):348–355, June 2010. ISSN 1883-8049, 0915-3942. doi: 10.20965/jrm.2010.p0348. URL https://www.fujipress.jp/jrm/rb/robot002200030348.
[73] Lionel Hertig, Dominik Schindler, Michael Bloesch, C. David Remy, and Roland Siegwart. Unified state estimation for a ballbot. In *2013 IEEE International Conference on Robotics and Automation*, pages 2471–2476, Karlsruhe, Germany, May 2013. IEEE. ISBN 978-1-4673-5643-5 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6630913. URL http://ieeexplore.ieee.org/document/6630913/.
[74] J. L. Meriam and L. G. Kraige. *Engineering Mechanics*. J. Wiley, New York, 7th ed edition, 2012. ISBN 978-0-470-61473-0.
[75] MIT. Moment of Inertia of a Spherical Shell. URL http://web.mit.edu/8.01t/www/materials/modules/guide15Appendix.pdf.
[76] Gearhead: Mass inertia – maxon Support. URL https://support.maxonmotor.com/hc/en-us/articles/360006129633-Gearhead-Mass-inertia.
[77] Bong Wie. Attitude Dynamics and Control. In *Space Vehicle Dynamics and Control*, AIAA Education Series. American Institute of Aeronautics and Astronautics, Reston, VA, 2nd ed edition, 2008. ISBN 978-1-56347-953-3. URL http://renaissance.ucsd.edu/courses/mae207/wie_chap5.pdf. OCLC: ocn232786911.
[78] Cross product - Wikipedia. URL https://en.wikipedia.org/wiki/Cross_product.
[79] Andrew Hanson. *Visualizing Quaternions*. Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann ; Elsevier Science [distributor], San Francisco, CA : Amsterdam ; Boston, 2006. ISBN 978-0-12-088400-1.
[80] Reza N. Jazar. *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. Springer, New York, 2nd ed edition, 2010. ISBN 978-1-4419-1749-2. OCLC: ocn465370325.
[81] 3Blue1Brown . Quaternions and 3d rotation, explained interactively, . URL https://www.youtube.com/watch?v=zjMuIxRvygQ.
[82] Ben Eater. Visualizing quaternions, an explorable video series. URL https://eater.net/quaternions.
[83] 3Blue1Brown . What are quaternions, and how do you visualize them? A story of four dimensions., . URL https://www.youtube.com/watch?v=d4EgbgTm0Bg.
[84] Understanding Quaternions3D Game Engine Programming. URL https://www.3dgep.com/understanding-quaternions/.
[85] James Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, October 2006. URL https://www.astro.rug.nl/software/kapteyn-beta/_downloads/attitude.pdf.
[86] Basile Graf. Quaternions and dynamics. *arXiv:0811.2889 [math-ph]*, November 2008. URL http://arxiv.org/abs/0811.2889.
[87] Joan Solà. Quaternion kinematics for the error-state Kalman filter. *arXiv:1711.02508 [cs]*, November 2017. URL http://arxiv.org/abs/1711.02508.
[88] Ashwin Narayan. How to Integrate Quaternions | Ashwin Narayan. URL https://www.ashwinnarayan.com/post/how-to-integrate-quaternions/.

[89] Michael S Andrle and John L Crassidis. Geometric Integration of Quaternions. page 10. URL http://ancs.eng.buffalo.edu/pdf/ancs_papers/2013/geom_int.pdf.

[90] Nikolas Trawny and Stergios I Roumeliotis. Indirect Kalman Filter for 3D Attitude Estimation. Technical Report, University of Minnesota, Multiple Autonomous Robotic Systems Laboratory, March 2005. URL http://mars.cs.umn.edu/tr/reports/Trawny05b.pdf.

[91] I. Bar-Itzhack, J. Deutschmann, and F. Markley. Quaternion normalization in additive EKF for spacecraft attitude determination. In *Navigation and Control Conference*, New Orleans,LA,U.S.A., August 1991. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1991-2706. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19920004874.pdf.

[92] Yuri Shtessel, Christopher Edwards, Leonid Fridman, and Arie Levant. *Sliding Mode Control and Observation*. Control Engineering. Springer New York, New York, NY, 2014. ISBN 978-0-8176-4892-3 978-0-8176-4893-0. doi: 10.1007/978-0-8176-4893-0. URL http://link.springer.com/10.1007/978-0-8176-4893-0.

[93] Stanislaw H. Zak and Raymond A. DeCarlo. A Quick Introduction to Sliding Mode Control and Its Applications. 2008. URL https://www.semanticscholar.org/paper/A-Quick-Introduction-to-Sliding-Mode-Control-and-1-DeCarlo-Zak/fa8b3d31508a983e3037fb82505a8b014ae1dc4d.

[94] Mirza Tariq Hamayun, Christopher Edwards, and Halim Alwi. Integral Sliding Mode Control. In *Fault Tolerant Control Schemes Using Integral Sliding Modes*, volume 61, pages 17–37. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32236-0 978-3-319-32238-4. doi: 10.1007/978-3-319-32238-4_2. URL http://link.springer.com/10.1007/978-3-319-32238-4_2.

[95] Xie Zheng, Xie Jian, Du Wenzheng, and Cheng Hongjie. Nonlinear Integral Sliding Mode Control for a Second Order Nonlinear System. *Journal of Control Science and Engineering*, 2015:1–7, 2015. ISSN 1687-5249, 1687-5257. doi: 10.1155/2015/218198. URL http://www.hindawi.com/journals/jcse/2015/218198/.

[96] Matteo Rubagotti, Antonio Estrada, Fernando Castanos, Antonella Ferrara, and Leonid Fridman. Integral Sliding Mode Control for Nonlinear Systems With Matched and Unmatched Perturbations. *IEEE Transactions on Automatic Control*, 56(11):2699–2704, November 2011. ISSN 0018-9286. doi: 10.1109/TAC.2011.2159420. URL http://ieeexplore.ieee.org/document/5873128/.

[97] Heide Brandtstadter. *Sliding Mode Control of Electromechanical Systems*. Ph.D. Thesis, Technical University of Munich, January 2008. URL https://d-nb.info/996708243/34.

[98] Claudio Vecchio. *Sliding Mode Control: Theoretical Developments and Applications to Uncertain Mechanical Systems*. Ph.D. Thesis, University of Pavia, 2008. URL https://www.tesionline.it/tesi/35561/Sliding-Mode-Control%3A-theoretical-developments-and-applications-to-uncertain-mechanical-systems.

[99] Govert Monsees. *Discrete-Time Sliding Mode Control*. PhD Thesis, TU Delft, 2002. URL https://repository.tudelft.nl/islandora/object/uuid%3A0eef3afb-f627-412a-9233-c96d28f859bd. ISBN 90-77017-83-6.

[100] Introduction of Sliding Mode Control - Multidimensional Sliding Modes, 2012.

[101] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, Zurich, June 2011. IEEE. ISBN 978-1-4244-9862-8 978-1-4244-9863-5 978-1-4244-9861-1. doi: 10.1109/ICORR.2011.5975346. URL http://ieeexplore.ieee.org/document/5975346/.

[102] J.L. Marins, Xiaoping Yun, E.R. Bachmann, R.B. McGhee, and M.J. Zyda. An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 4, pages 2003–2011, Maui, HI, USA, 2001. IEEE. ISBN 978-0-7803-6612-1. doi: 10.1109/IROS.2001.976367. URL http://ieeexplore.ieee.org/document/976367/.

[103] Angelo Maria Sabatini. Kalman-Filter-Based Orientation Determination Using Inertial/Magnetic Sensors: Observability Analysis and Performance Evaluation. *Sensors*, 11(10):9182–9206, September 2011. ISSN 1424-8220. doi: 10.3390/s111009182. URL http://www.mdpi.com/1424-8220/11/10/9182.

[104] OlliW . IMU Data Fusing: Complementary, Kalman, and Mahony Filter, . URL http://www.olliw.eu/2013/imu-data-fusing/.

[105] R. Mahony, T. Hamel, and J.-M. Pflimlin. Complementary filter design on the special orthogonal group SO(3). In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 1477–1484, Seville, Spain, 2005. IEEE. ISBN 978-0-7803-9567-1. doi: 10.1109/CDC.2005.1582367. URL http://ieeexplore.ieee.org/document/1582367/.

[106] Sebastian O H Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical report, April 2010. URL http://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf.

[107] Li Wang, Zheng Zhang, and Ping Sun. Quaternion-Based Kalman Filter for AHRS Using an Adaptive-Step Gradient Descent Algorithm. *International Journal of Advanced Robotic Systems*, 12(9):131, September 2015. ISSN 1729-8814, 1729-8814. doi: 10.5772/61313. URL

http://journals.sagepub.com/doi/10.5772/61313.
[108] Dan Simon. *Optimal State Estimation: Kalman, H [Infinity] and Nonlinear Approaches*. Wiley-Interscience, Hoboken, N.J, 2006. ISBN 978-0-471-70858-2. OCLC: ocm64084871.
[109] M. Zamani, J. Trumpf, and R. Mahony. Nonlinear Attitude Filtering: A Comparison Study. *arXiv:1502.03990 [cs]*, February 2015. URL http://arxiv.org/abs/1502.03990.
[110] Lionel Hertig, Dominik Schindler, Michael Bloesch, C. David Remy, and Roland Siegwart. Unified state estimation for a ballbot. In *2013 IEEE International Conference on Robotics and Automation*, pages 2471–2476, Karlsruhe, Germany, May 2013. IEEE. ISBN 978-1-4673-5643-5 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6630913. URL http://ieeexplore.ieee.org/document/6630913/.
[111] F Landis Markley. Multiplicative vs. Additive Filtering for Spacecraft Attitude Determination. page 8, NASA's Goddard Space Flight Center. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20040037784.pdf.
[112] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Number 33 in Space Technology Library. Springer, New York, 2014. ISBN 978-1-4939-0801-1. OCLC: ocn882605422.
[113] Thomas Bak. *Spacecraft Attitude Determination*. Ph.D. Thesis, Aalborg University, December 1999. URL http://vbn.aau.dk/en/publications/spacecraft-attitude-determination(d1086a60-0034-11da-b4d5-000ea68e967b).html. ISBN 87-90664-03-5.
[114] Robert C. Leishman and Timothy W. McLain. Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation. *Journal of Aerospace Information Systems*, 12(12):728–744, December 2015. ISSN 2327-3097. doi: 10.2514/1.I010236. URL http://arc.aiaa.org/doi/10.2514/1.I010236.
[115] James M. Maley. Multiplicative Quaternion Extended Kalman Filtering for Nonspinning Guided Projectiles:. Technical report, Defense Technical Information Center, Fort Belvoir, VA, July 2013. URL http://www.dtic.mil/docs/citations/ADA588831.
[116] Kasper Vinther, Kasper F Jensen, Jesper A Larsen, and Rafael Wisniewski. INEXPENSIVE CUBESAT ATTITUDE ESTIMATION USING QUATERNIONS AND UNSCENTED KALMAN FILTERING. page 13.
[117] E. Kraft. A quaternion-based unscented Kalman filter for orientation tracking. In *Sixth International Conference of Information Fusion, 2003. Proceedings of The*, pages 47–54, Cairns, Queensland, Australia, 2003. IEEE. ISBN 978-0-9721844-4-1. doi: 10.1109/ICIF.2003.177425. URL http://ieeexplore.ieee.org/document/1257247/.
[118] Yuhong Yang, Junchuan Zhou, and Otmar Loffeld. Quaternion-based Kalman Filtering on INS/GPS. page 8.
[119] M. Verhaegen and P. Van Dooren. Numerical aspects of different Kalman filter implementations. *IEEE Transactions on Automatic Control*, 31(10):907–917, October 1986. ISSN 0018-9286. doi: 10.1109/TAC.1986.1104128. URL http://ieeexplore.ieee.org/document/1104128/.
[120] F. Ferraris, I. Gorini, U. Grimaldi, and M. Parvis. Calibration of three-axial rate gyros without angular velocity standards. *Sensors and Actuators A: Physical*, 42(1-3):446–449, April 1994. ISSN 09244247. doi: 10.1016/0924-4247(94)80031-6. URL http://linkinghub.elsevier.com/retrieve/pii/0924424794800316.
[121] Rqt_reconfigure - ROS Wiki. URL http://wiki.ros.org/rqt_reconfigure.
[122] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. Wiley, Chichester, 2. ed., reprint edition, 2009. ISBN 978-0-470-01168-3. OCLC: 837462409.
[123] Oscar Mauricio Agudelo Mañozca. Introduction to Model Predictive Control (MPC), May 2017. URL http://homes.esat.kuleuven.be/~maapc/static/files/CACSD/Slides/all_slides_mpc.pdf.
[124] Moritz Diehl. Embedded Optimization for Model Predictive Control of Mechatronic Systems, July 2016. URL https://www.ethz.ch/content/dam/ethz/special-interest/conference-websites-dam/dlmc-2016-dam/documents/Slides/Diehl_slides.pdf.
[125] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. ACADO toolkit - An open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, May 2011. ISSN 01432087. doi: 10.1002/oca.939. URL http://doi.wiley.com/10.1002/oca.939.
[126] ACADO Toolkit, . URL http://acado.github.io/.
[127] ACADO Toolkit User's Manual, Version 1.2.1beta, January 17, 2014. URL http://acado.sourceforge.net/doc/pdf/acado_manual.pdf.
[128] Hans Joachim Ferreau. qpOASES User's Manual, Version 3.0 (December 2014). URL https://www.coin-or.org/qpOASES/doc/3.0/manual.pdf.
[129] David Ariens. ACADO for Matlab User's Manual, Version 1.0beta - v2022 (June 2010). URL http://acado.sourceforge.net/doc/pdf/acado_matlab_manual.pdf.
[130] Rien Quirynen. ACADO Simulink interface example, . URL https://sourceforge.net/p/acado/discussion/general/thread/2873192d/. ACADO_Simulink.zip.
[131] Sébastien Gros, Mario Zanon, Rien Quirynen, Alberto Bemporad, and Moritz Diehl. From linear to nonlinear MPC: Bridging the gap via the real-time iteration. *International Journal of Control*, pages 1–19,

September 2016. ISSN 0020-7179, 1366-5820. doi: 10.1080/00207179.2016.1222553. URL https://www.tandfonline.com/doi/full/10.1080/00207179.2016.1222553.

[132] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, October 2011. ISSN 00051098. doi: 10.1016/j.automatica.2011.08.020. URL https://linkinghub.elsevier.com/retrieve/pii/S0005109811003918.

[133] ACADO Tutorials, . URL http://acado.sourceforge.net/doc/html/db/d4e/tutorial.html.

[134] Mario Zanon and Moritz Diehl. Embedded Optimization for Nonlinear Model Predictive Control - Exercise 4. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.456.7445&rep=rep1&type=pdf.

[135] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Solving Optimal Control Problems with ACADO Toolkit, February 2009. URL http://acado.sourceforge.net/doc/pdf/acado_introtalk.pdf.

[136] Rien Quirynen. Swing-up of an inverted pendulum (PART 1). November 2014. URL http://www.syscop.de/files/2014ws/acado/exercise1.pdf.

[137] Adriaen Verheyleweghen and Christoph Backi. Introduction to Optimization and Optimal Control using the software packages CasADi and ACADO, October 2016. URL https://static1.squarespace.com/static/55ddc2cfe4b0bddc4add8891/t/582dcbd737c58112cc1aa934/1479396314416/Introduction+to+Optimization+and+Optimal+Control+using+the+software+packages+CasADi+and+ACADO.pdf.

[138] Moritz Diehl, Boris Houska, and Hans Joachim Ferreau. ACADO TOOLKIT - AUTOMATIC CONTROL AND DYNAMIC OPTIMIZATION. URL https://www.vehicular.isy.liu.se/Edu/Courses/NumericalOptimalControl/Slides/20110615LinkopingAcado.pdf.

[139] Rien Quirynen. ACADO Code Generation tool, . URL http://www.syscop.de/files/2015ss/numopt/ACADO_codegen.pdf.

[140] *Robot Operating System (ROS)*. Springer Berlin Heidelberg, New York, NY, 2017. ISBN 978-3-319-54926-2.

[141] Fernando A. C. C. Fontes. Model Predictive Control of Nonholonomic Vehicle Formations, March 2011. URL https://hal.inria.fr/inria-00585716/document.

[142] Timm Faulwasser. *Optimization-Based Solutions to Constrained Trajectory-Tracking and Path-Following Problems*. PhD Thesis, Otto-von-Guericke-Universität Magdeburg, October 2012. URL https://www.researchgate.net/publication/259672465_Optimization-based_solutions_to_constrained_trajectory-tracking_and_path-following_problems.

[143] A. Pedro Aguiar, Dragan B. Dačić, João P. Hespanha, and Petar Kokotović. Path-following or reference tracking? *IFAC Proceedings Volumes*, 37(8):167–172, July 2004. ISSN 14746670. doi: 10.1016/S1474-6670(17)31970-5. URL https://linkinghub.elsevier.com/retrieve/pii/S1474667017319705.

[144] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-Based Autonomous Racing of 1:43 Scale RC Cars. *Optimal Control Applications and Methods*, 36(5):628–647, September 2015. ISSN 01432087. doi: 10.1002/oca.2123. URL http://arxiv.org/abs/1711.07300.

[145] R. Frezza, A. Beghi, and A. Saccon. Model predictive for path following with motorcycles: Application to the development of the pilot model for virtual prototyping. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, pages 767–772 Vol.1, Nassau, Bahamas, 2004. IEEE. ISBN 978-0-7803-8682-2. doi: 10.1109/CDC.2004.1428754. URL http://ieeexplore.ieee.org/document/1428754/.

[146] Mathias Hauan Arbo, Esten Ingar Grøtli, and Jan Tommy Gravdahl. On Model Predictive Path Following and Trajectory Tracking for Industrial Robots. *arXiv:1703.02279 [cs]*, March 2017. URL http://arxiv.org/abs/1703.02279.

[147] Michael S. Floater and Tatiana Surazhsky. Parameterization for Curve Interpolation. In *Studies in Computational Mathematics*, volume 12, pages 39–54. Elsevier, 2006. ISBN 978-0-444-51844-6. doi: 10.1016/S1570-579X(06)80004-2. URL https://linkinghub.elsevier.com/retrieve/pii/S1570579X06800042.

[148] Kendall Atkinson. Modelling a Road Using Spline Interpolation. Technical report, The University of Iowa, February 2002. URL https://homepage.divms.uiowa.edu/~atkinson/ftp/road_surface.pdf.

[149] Steven M. . Dubins Curves, 2006. URL http://planning.cs.uiuc.edu/node821.html.

[150] Whitman College. Arc length and curvature. URL https://www.whitman.edu/mathematics/calculus_online/section13.03.html.

[151] Mohsen Madi. CLOSED-FORM EXPRESSIONS FOR THE APPROXIMATION OF ARCLENGTH PARAMETERIZATION FOR BÉZIER CURVES. In *Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 1, 33–41*, page 9, 2004. URL http://zbc.uz.zgora.pl/Content/2570/HTML/5madi.pdf.

[152] Marcelo Walter and Alain Fournier. Approximate Arc Length Parametrization. page 8, 1996. URL https://www.visgraf.impa.br/sibgrapi96/trabs/pdf/a14.pdf.

[153] Hongling Wang, Joseph Kearney, and Kendall Atkinson. Arc-Length Parameterized Spline Curves for Real-Time Simulation. *Curve and Surface Design*, Saint-Malo 2002:387–396. URL http://homepage.cs.uiowa.edu/~kearney/pubs/CurvesAndSurfacesArcLength.pdf. ISBN 0-9728482-0-7.

[154] Navigation - ROS Wiki. URL http://wiki.ros.org/navigation.

Part III

Appendices

A Literature Review Table

The following three pages contain the literature review tables that compares the Kugle V1 prototype used in this thesis with eleven state-of-the-arts ball-balancing robots.

Abbreviation	CMU	TGU	ETH	NCHU	PRISMA	INAL	TUM	UMC	ALTEN	CEES	KHU	AAU
Year	2006-2014	2008-2010	2010-2014	2008	2016	2012	2011	2017	2014	2015	2018	2019
Reference(s)	[3] [71] [21] [38] [32] [23]	[14] [72]	[2] [73] [34]	[35]	[29]	[22] [30]	[31]	[24]	[25]	[15]	[33] [36]	
Wheel configuration	Inverse mouse-ball drive, 2 drive rollers + 2 idle rollers	3 omniwheels, 40° zenith, 120° spacing	3 omniwheels, 45° zenith, 120° spacing	Inverse mouse-ball drive, 2 drive rollers + 2 idle rollers	N/A	Inverse mouse-ball drive	3 omniwheels, 45° zenith, 120° spacing	4 omniwheels, 30° zenith, 90° spacing	3 omniwheels, 45° zenith, 120° spacing	3 omniwheels, 45° zenith, 120° spacing	3 omniwheels, 65.5° zenith, 120° spacing	3 omniwheels, 45 ° zenith, 120 ° spacing
Wheel diameter	12.7 mm roller	100 mm omniwheel	120 mm omniwheel	Unknown	N/A	N/A	N/A	48 mm omniwheel	100 mm omniwheel	100 mm omniwheel	200 mm omniwheel	100 mm omniwheel
Motor type	DC motor	Stepper motor	200 W Brushless DC motor (EC-4pole 30) with 26:1 gearing (GP 42 C)	Brushless DC motor	N/A	N/A	N/A	DC motor	Brushless DC motor with gearing	Brushless DC motor with gearing	Brushless DC motor with gearing	100 W Brushless DC motor (EC-flat 60) with 4.3:1 gearing (GP 52 C)
Maximum torque	> 20 N	1.3 Nm	5 Nm	N/A	N/A	N/A	N/A	2.5 Nm	3.75 Nm	Unknown	> 30 Nm	2 Nm (limited by motor drivers)
Body mass	1. 45 kg 2. 51.7 kg	12 kg	9.2 kg	45 kg	N/A	N/A	N/A	6.4 kg	12.87 kg	12.97 kg	68 kg	16.2 kg
Ball mass	2.5 kg	3.6 kg	2.29 kg	7.3 kg	N/A	N/A	N/A	1.5 kg	3.2 kg	3.07 kg	19.6 kg	1.48 kg
Height	160 cm	150 cm	100 cm	150 cm	N/A	N/A	N/A	42 cm	> 80 cm	Unknown	Unknown	138 cm
Ball diameter	1. 200 mm hydroformed steel sheel with 3.2 mm urethane coating 2. 190.5 mm aluminum shell with 12.7 mm urethane coating	200 mm bowling ball coated with liquid rubber spray	200 mm bowling ball coated with liquid rubber spray	220 mm	N/A	N/A	N/A	200 mm	230 mm	230 mm	440 mm	250 mm ball (made of 4 mm polyethylene) with a 4 mm polyurethane rubber coating
Body diameter	400 mm	Unknown	283 mm	300 mm	N/A	N/A	N/A	400 mm	283 mm	Unknown	Unknown	298 mm
COM height	690 mm	Unknown	339 mm	Unknown	N/A	N/A	N/A	214 mm	405 mm	400 mm	380 mm	421 mm
Model type	1. Planar decoupled model 2. Coupled model with Euler angles	Planar decoupled model	1. Planar decoupled model 2. Coupled 3D model using Euler angles	Coupled 2.5D model (heading and tilt/inclination)	Intrinsic coordinate free coupled model	Coupled 3D model using Quaternions	Coupled 3D model using Euler angles including yaw rotation of ball	Coupled 3D model	1. Planar decoupled model 2. Coupled twist-based 3D model using Euler angles	Coupled 3D model using screw theory (and SO(3) states)	Planar decoupled model	Coupled model
Way of modelling	1. Lagrangian Mechanics 2. Lagrangian Mechanics + Lagrange multiplier to reduce generalized coordinate space	Kinematics only	Lagrangian Mechanics	Lagrangian Mechanics	Lagrangian Mechanics (with variation of Lagrangian)	Newtonian Mechanics	Lagrangian Mechanics + Constraint equation and Lagrange multiplier to reduce generalized coordinate space	Lagrangian Mechanics	Lagrangian Mechanics	Lagrangian Mechanics	Lagrangian Mechanics	Lagrangian Mechanics

Abbreviation	CMU	TGU	ETH	NCHU	PRISMA	INAL	TUM	UMC	ALTEN	CEES	KHU	AAU
Model states	1. Body inclination angle (around x/y axes) + ball rolling angle (around x/y axes) + derivatives 2. Body Euler angles + derivatives	Virtual planar wheel velocity + travelling velocity	1. Body inclination angles + ball rolling angles (x+y axes only) + derivatives 2. Body Euler angles + ball rolling angles + derivatives	Heading + inclination (3-1 Euler) + position + derivatives	Rotation matrix of body and ball (rotation around z-axis not allowed) + angular velocities	Position of ball and body + orientation of ball and body described with quaternions (constraint locks yaw motion of ball) + linear and angular momenta (proportional to derivatives)	Position of ball (x+y) + yaw angle of ball (rotation around z-axis) + Body Euler angles + derivatives	Position of ball (x+y) + Body Euler angles + derivatives	Position of ball (x+y) + Body Euler angles + derivatives	Rotation matrix of body and ball (rotation around z-axis not allowed) + angular velocities	Ball position (x+y) + Body inclination angles (Tait-Bryan angles) + derivatives	Inertial position + quaternion + derivatives
Friction included	Viscous friction between wheel-ground and wheel-body Coulomb friction added in some of the models	No	No	No	No	Viscous friction between wheel-ground	Viscous spin friction included between ball-ground (for yaw rotation of ball only)	Translational and Angular viscous friction	No	Both viscous and Coulomb friction added between ball-ground and ball-body	Both viscous and Coulomb friction between ball-ground and ball-body	Viscous friction between ball-floor (in x+y direction only) and wheel-ball and body-air
Controller structure	1. Decoupled linear cascade 2. Coupled non-linear controller	Decoupled linear SISO controller	1. Decoupled linear MIMO controller 2. Coupled linear MIMO controller	Hierarchical non-linear controller	Coupled linear controller	Coupled non-linear controller	Coupled linear MIMO controller	Decoupled linear MIMO controller	Coupled linear MIMO controller	Coupled non-linear controller	1. Decoupled linear cascade 2. Hierarchical non-linear controller (but still decoupled)	Coupled non-linear controller
Controller type	1. PI inner loop on ball velocity + LQR outer loop with full state feedback 2. Sliding mode for balance control	PI on model states	LQR	Sliding mode with 2 inner sliding surfaces and 1 combining sliding surface	PD on rotation matrix error and angular velocity error	PD controller on desired roll and pitch Computed torque controller on desired translational torque	Linear full-state feedback controller through Pole-placement	1. LQR controller on model states 2. PID controller on model states for velocity control	LQR controller on model states	1. Computed torque on SO(3) or S^2 error function 2. Sliding mode on SO(3) or S^2 error function	1. PI + Feedforward inner loop on translational velocity + LQR outer loop with full state feedback 2. Hierarchical Sliding mode with an inner sliding surface on position and an outer sliding surface combining the inner and the angle	LQR on Velocity states of model Sliding mode on Quaternion states of model
Controller output	1. LQR sets ball velocity references + PI sets motor torque 2. Sliding mode sets motor torque	Motor velocity reference	Torque	Torque	Torque	PD controller sets translational (xy) torque references Non-linear Computed torque controller sets motor torque	Torque	Torque	Torque	Torque	1. LQR sets translational velocity references + PI sets virtual motor torque 2. Torque	LQR sets angle references Sliding mode sets motor torque

Abbreviation	CMU	TGU	ETH	NCHU	PRISMA	INAL	TUM	UMC	ALTEN	CEES	KHU	AAU
Max. tested inclination	< 1 deg	< 3 deg (8 deg in second paper)	17 deg	N/A	N/A	N/A	N/A	Unknown	< 2 deg	N/A	< 3 deg	5 deg
Max. tested velocity	< 0.1 m/s	0.3 m/s (0.6 m/s in second paper)	3.5 m/s	N/A	N/A	N/A	N/A	Unknown	Not tested (Only station-keeping tested)	N/A	< 0.9 m/s	1.0 m/s
Other notes		Parameters listed for tall/first version		Simulation only	Mostly a modelling paper Does not allow yaw movement	Simulation only	Simulation only Includes yaw angle/rotation of ball		In coorperation with ALTEN Mechatronics	Simulation only since he does not even get linear controller to work In coorperation with ALTEN Mechatronics, hence same prototype as the ALTEN report	Capable of carrying a person Uses 4 control moment gyroscopes (similar to reaction-wheels) to help with stabilization and disturbance rejection	

B Rotation Description Comparison

Criteria	Euler angles	SO(3)	Quaternion
Parameters	3	9	4
Constraints	None	Six constraints, $\boldsymbol{R}^\mathsf{T}\boldsymbol{R} = \mathbf{I}_3$	One constraint, $\boldsymbol{q}^\mathsf{T}\boldsymbol{q} = 1$
Ambiguity	12 different formulations	Unique	Two possible unit quaternions
Singularities	Gimbal lock	No	No
Numerical stability	High, due to no constraints	Low, several constraints to fulfil, regularization recommended	Decent, only one constraint, regularization can be carried out by normalizing
Calculating derivatives	Complicated, involves a lot of sines and cosines	Easy, see (E.10)	Easy, see (H.49)
Equations of motion	Three coupled differential equations	Nine coupled differential equations (six if 3rd axis is calculated from cross product)	Four coupled differential equations/states
Interpolation	Complicated	Simple	Efficient
Vector transformation	Requires a rotation matrix or quaternion to be computed through several sine/cosine operations	3×3 matrix multiplication	Quaternion multiplication since the sine/cosine of the angle is stored internally
Concatenation	Difficult, not just addition	3×3 matrix multiplication	Quaternion multiplication

Table B.1: Rotation description comparison table

C Hardware Components

Group	Type	Part	Description
Actuation	Motor	Maxon EC 60 flat *(412819)*	100 W 3-phased brushless DC motor, 12 V nominal, 2.85 N m stall torque, 6000 RPM max speed, 60 mm diameter, 84% efficiency
	Gear	Maxon GP 52 C *(223081)*	Planetary gear, 4.3 : 1 reduction (13/3 absolute reduction), 52 mm diameter, 91% efficiency
	Encoder	Maxon Encoder MILE *(421988)*	Quadrature encoder, 4096 counts pr. revolution (thus 16384 edges pr. rev.)
	Omniwheel	Single-ring omniwheel [19]	100 mm single-ring aluminum omniwheel, 28 mm width, 270 g weight, 6 small rollers + 6 big rollers, load capacity up to 5 kg
	Motor driver	ESCON 50/5 *(438725)*	DC Brushed/Brushless Motor driver module, max 50 V, max 5 A continuous, max 15 A stall, for up to 250 W motors, configurable over USB
Sensors	IMU	Xsens MTI-200 VRU *(MTi-200-VRU-2A8G4-DK)*	9-axis IMU with 450 deg/s gyroscope, factory calibrated, includes proprietary attitude estimator (roll, pitch and unreferenced yaw)
	IMU	MPU-9250 *(SEN-13762)*	Sparkfun IMU breakout board
	LiDAR	SICK TiM571	2D LiDAR scanner, 270° scan, 0.33° resolution, up to 25 m range, 15 Hz scan frequency
Power	Battery	RRC2024	4s3p lithium-ion, 14.4 V nominal, up to 16 V, 6600 mAh nominal capacity, internal protection circuitry
	Power management	RRC-PMM240	Charge and discharge monitoring, SMBus controllable
Processing	Microprocessor	NUCLEO-H743ZI	STM32H743ZI processor board, ARM Cortex-M7, 400 MHz, 2 MB flash, 1 MB RAM (864 KB user SRAM), double precision FPU, USB Host, hardware quadrature decoder, includes onboard USB debugger
	Onboard computer	Intel NUC i3 *(NUC7I3DNKE)*	Intel Core i3-7100U, dual-core, 8 GB RAM, 512 GB SSD
	Tablet	Lenovo Tab 4 10 LTE *(TB-X304L)*	10.1 inch display, 16 GB flash, 2 GB RAM, Quad-core 1.4 GHz Cortex-A53, Android 7.0

Table C.1: List of hardware components on Kugle V1

D Model Parameter Computation

This chapter explains how certain model parameters have been derived mathematically or from the CAD model.

D.1 Ball inertia

The inertia of the ball is approximated with the model of a sphere. A sphere has a uniform inertia and the inertia tensor is thus a diagonal matrix with equal entries. The ball is defined by a mass of $M_k = 1.478\,\text{kg}$, an outer radius $r_k = 129\,\text{mm}$ and a shell thickness of 8 mm consisting of a 4 mm inner plastic shell and a 4 mm outer rubber coating, thus an inner diameter of $r_i = r_k - 8\,\text{mm} = 121\,\text{mm}$.
The inertia of a hollow sphere is [74]:

$$J_{\text{hollow}} = \frac{2}{3} M_k r_k^2 = 16.4 \times 10^{-3}\,\text{kg}\,\text{m}^2 \tag{D.1}$$

The inertia of a solid sphere is [74]:

$$J_{\text{solid}} = \frac{2}{5} M_k r_k^2 = 9.8 \times 10^{-3}\,\text{kg}\,\text{m}^2 \tag{D.2}$$

While the inertia of a sphere shell with an inner radius r_i and outer radius r_k is [75]:

$$J_{\text{shell}} = \frac{2}{5} M_k \frac{r_k^5 - r_i^2}{r_k^3 - r_i^3} = 15.4 \times 10^{-3}\,\text{kg}\,\text{m}^2 \tag{D.3}$$

Since the ball is a uniform sphere shell, it is decided to use the inertia of the sphere shell for the ball.

$$J_k = J_{\text{shell}} = 15.4 \times 10^{-3}\,\text{kg}\,\text{m}^2 \tag{D.4}$$

Note that this assumes the density of the plastic shell and the rubber coating to be the same. Furthermore, the inertia is almost equal to the inertia of a hollow sphere with the average radius.

$$J_k \approx \frac{2}{3} m \left(\frac{r_k + r_i}{2} \right)^2 = 15.4 \times 10^{-3}\,\text{kg}\,\text{m}^2 \tag{D.5}$$

It is decided to use an inertia of $J_k = 15.4 \times 10^{-3}\,\text{kg}\,\text{m}^2$ for the ball.

D.2 Wheel inertia

The wheel inertia, J_w, listed in Table 2.1 is a combination of the inertia from the omniwheel, J_{ow}, gearbox, J_g and motor J_m. The wheel inertia should be described relative to the output shaft, but both the gear inertia and the motor inertia is described relative to the internal motor shaft, thus before the gearing. The combined inertia thus has to be computed by taking the gearing ratio into account [76].

$$\omega_{\text{motor}} = n_{\text{gear}}\,\omega_{\text{output}} \tag{D.6}$$

For a gearbox with no loss the energy is conserved and since inertia is related to energy and kinetic energy is proportional to the squared velocity, the inertia is scaled with the squared gearing ratio.

$$\begin{aligned} E_{\text{output}} &= E_{\text{motor}} \\ J_{\text{output}}\,\omega_{\text{output}}^2 &= J_{\text{motor}}\omega_{\text{motor}}^2 \end{aligned} \tag{D.7}$$

Inserting the definition of the motor angular velocity, (D.6), yields the squared relationship.

$$\begin{aligned} J_{\text{output}}\,\omega_{\text{output}}^2 &= J_{\text{motor}} n_{\text{gear}}^2 \omega_{\text{output}}^2 \\ J_{\text{output}} &= J_{\text{motor}}\, n_{\text{gear}}^2 \end{aligned} \tag{D.8}$$

The combined wheel inertia is therefore computed as:

$$J_w = J_{ow} + n_{\text{gear}}^2 \left(J_g + J_m \right) = 3.20 \times 10^{-3}\ \text{kg}\,\text{m}^2 \tag{D.9}$$

D.3 Body inertia

From the CAD drawing of the Kugle V1 designed in OnShape, the inertia tensor of the body, (D.10), is extracted.

$${}^{\text{B'}}\boldsymbol{J}_b = \begin{bmatrix} 1.306 & 6.440 \times 10^{-5} & 3.173 \times 10^{-2} \\ 6.440 \times 10^{-5} & 1.295 & -2.373 \times 10^{-4} \\ 3.173 \times 10^{-2} & -2.373 \times 10^{-4} & 1.002 \times 10^{-1} \end{bmatrix} \text{kg}\,\text{m}^2 \tag{D.10}$$

This inertia tensor is defined in the body center of mass frame, $\{B'\}$, with the axes aligned with the body frame. However, the inertia used in the modelling chapter, Chapter 5, has to be defined with the same origin around which the rotation of the body occurs. Since the ball origin has been chosen as the rotation origin, the inertia tensor from (D.10) has to be translated. Inertia tensor translation can be performed using the generalized Parallel Axis Theorem [74] using the inner and outer product of the translation vector. In this case the translation vector is the origin of the center of mass frame defined in the body frame, which, due to the frame definition in Section 3.1, corresponds to the center of mass vector, ${}^{\text{B}}\boldsymbol{O}_{\text{B'}} = {}^{\text{B}}\boldsymbol{p}_{\text{COM}}$.

$$\begin{aligned} {}^{\text{B}}\boldsymbol{J}_b &= {}^{\text{B'}}\boldsymbol{J}_b + M_b \left({}^{\text{B}}\boldsymbol{O}_{\text{B'}}^{\mathsf{T}}\, {}^{\text{B}}\boldsymbol{O}_{\text{B'}} \mathbf{I}_3 - {}^{\text{B}}\boldsymbol{O}_{\text{B'}}\, {}^{\text{B}}\boldsymbol{O}_{\text{B'}}^{\mathsf{T}} \right) \\ &= \begin{bmatrix} 4.173 & 6.332 \times 10^{-5} & 3.187 \times 10^{-2} \\ 6.332 \times 10^{-5} & 4.161 & 2.159 \times 10^{-2} \\ 3.187 \times 10^{-2} & 2.159 \times 10^{-2} & 1.004 \times 10^{-1} \end{bmatrix} \text{kg}\,\text{m}^2 \end{aligned} \tag{D.11}$$

Notice some small off-diagonal elements, indicating that the robot is not completely symmetrical. Note especially the elements related to the inertia around the z-axis, where rotations around the other two axes starts to play a role. This would be the benefit of including the complete inertia tensor in the model. However, for now only the diagonal elements are used, assuming the model to be symmetrical.

The inertia tensor is therefore simplified by only including the diagonal elements, thus the inertia around the x-, y- and z-axes which we denote J_{bx}, J_{by} and J_{bz}.

$$\begin{aligned} J_{bx} &= 4.173\ \text{kg}\,\text{m}^2 \\ J_{by} &= 4.161\ \text{kg}\,\text{m}^2 \\ J_{bz} &= 0.1004\ \text{kg}\,\text{m}^2 \end{aligned} \tag{D.12}$$

E Kinematics of Moving Frames

This appendix introduces the notation for linear and angular velocities used in this thesis. It is important to remind the reader that a velocity is always relative, thus given relatively between two frames. Any velocity is determined by defining a reference frame from where the other frame or point and its velocity is observed. In the following the subscript B←A should be understood as: "when looking on frame {B} from reference frame {A}". [46] [47] [44]

E.1 Linear velocity

The linear (or translational) velocity of a frame {B} describes the speed and direction in which the origin of a frame is moving relative to a reference frame {A}.

$$\boldsymbol{v}_{\mathrm{B}\leftarrow\mathrm{A}} = {}^{\mathrm{A}}\boldsymbol{v}_{\mathrm{B}\leftarrow\mathrm{A}} = {}^{\mathrm{A}}\dot{\boldsymbol{O}}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.1}$$

This defines the linear velocity of frame {B} when seen from frame {A} and described along the axes of frame {A}. Note that without any left superscript the velocity is assumed to be defined in the reference frame, thus in this case frame {A}.
The reason for having the left superscript is to be able to distinguish between the reference frame and the frame in which the vector is described. As an example a rotation will only change the frame in which the velocity is described. It will not affect the reference frame.

$${}^{\mathrm{B}}\boldsymbol{v}_{\mathrm{B}\leftarrow\mathrm{A}} = {}^{\mathrm{B}}_{\mathrm{A}}\boldsymbol{R}\,{}^{\mathrm{A}}\boldsymbol{v}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.2}$$

Swapping the observed frame and the reference frame corresponds to negating the velocity vector.

$${}^{\mathrm{A}}\boldsymbol{v}_{\mathrm{A}\leftarrow\mathrm{B}} = -\,{}^{\mathrm{A}}\boldsymbol{v}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.3}$$

E.2 Angular velocity

A rotational velocity of a frame {B} describes the speed and direction around which a frame is rotating relative to a reference frame {A}.

$$\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} = {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.4}$$

This defines the angular velocity of frame {B} when seen from frame {A} and described around the axes of frame {A}. Without any left superscript the velocity is described in the reference frame, in this case frame {A}.
Similarly to the linear velocity, the reference frame and the frame in which the angular velocity vector is described can be different, e.g., due to a rotation.

$${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} = {}^{\mathrm{B}}_{\mathrm{A}}\boldsymbol{R}\,{}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.5}$$

The observed frame and the reference frame can be swapped by swapping sign of the angular velocity vector.

$${}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{A}\leftarrow\mathrm{B}} = -\,{}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.6}$$

Note that an angular velocity of a frame referenced to itself is 0, also even if it is represented in a different frame.

$${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{A}\leftarrow\mathrm{A}} = \mathbf{0} \tag{E.7}$$

E.3 Linear velocity of a point in a moving frame

When a moving point within a moving frame is observed from a different frame, the linear velocity has to be computed while taking the moving frame into account. This thus applies to both arbitrary points in a moving frame and to the origin of the frame itself, except that for the origin parts of the equation collapses due to the point being $\mathbf{0}_{3\times1}$.
Lets have a point defined in frame {B}, ${}^{\mathrm{B}}\boldsymbol{p}$ and let the velocity of this point relative to frame {B} and described in frame {B} be denoted as ${}^{\mathrm{B}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{B}}$. Applying the SE(3) transform from (3.2), including rotation and translation, the point is transformed into frame {A}.

$$ {}^{\mathrm{A}}\boldsymbol{p} = {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{T}\,{}^{\mathrm{B}}\boldsymbol{p} = {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\boldsymbol{p} + {}^{\mathrm{A}}\boldsymbol{O}_{\mathrm{B}} \tag{E.8} $$

The derivative of this equation, hence the velocity of the point described in frame {A} is computed using the product rule, see Appendix G.2.

$$ {}^{\mathrm{A}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{A}} = {}^{\mathrm{A}}_{\mathrm{B}}\dot{\boldsymbol{R}}\,{}^{\mathrm{B}}\boldsymbol{p} + {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{B}} + {}^{\mathrm{A}}\dot{\boldsymbol{O}}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.9} $$

Note how this involves the derivative of the rotation matrix which is the cross product between the angular velocity and the current rotation [77].

$$ {}^{\mathrm{A}}_{\mathrm{B}}\dot{\boldsymbol{R}} = {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \times {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R} \tag{E.10} $$

Applying the skew-symmetric matrix definition from Appendix F.2.1 and inserting into (E.9):

$$ \begin{aligned} {}^{\mathrm{A}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{A}} &= \left({}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \times {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R} \right) {}^{\mathrm{B}}\boldsymbol{p} + {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{B}} + {}^{\mathrm{A}}\dot{\boldsymbol{O}}_{\mathrm{B}\leftarrow\mathrm{A}} \\ &= \boldsymbol{\Omega}({}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}})\,{}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\boldsymbol{p} + {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\dot{\boldsymbol{p}}_{\leftarrow\mathrm{B}} + {}^{\mathrm{A}}\dot{\boldsymbol{O}}_{\mathrm{B}\leftarrow\mathrm{A}} \end{aligned} \tag{E.11} $$

This defines the kinematic relationship between the linear velocity of a point defined in a moving frame but observed from another. Depending on whether the moving frame is only rotating (${}^{\mathrm{A}}\dot{\boldsymbol{O}}_{\mathrm{B}\leftarrow\mathrm{A}} = \mathbf{0}_{3\times1}$) or only translating (${}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} = \mathbf{0}_{3\times1}$), equation (E.11) simplifies accordingly.

E.4 Angular velocity in a rotating frame

The observed angular velocity of a frame is not affected by translational movement of neither the observed frame nor the reference frame. However, if the reference frame is rotating or the observed frame is within another frame that is rotating, the resulting observed angular velocity has to be computed from a sum of angular velocities, taking all the frames into account.
Lets have a frame {C} inside of frame {B}, with an angular velocity, ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{C}\leftarrow\mathrm{B}}$, relative to frame {B}. To compute the angular velocity of frame {C} with respect to an outer frame {A}, the relative angular velocity between frame {A} to {B}, ${}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}}$, has to be known.

$$ {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{C}\leftarrow\mathrm{A}} = {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{C}\leftarrow\mathrm{B}} + {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.12} $$

Note that the above equation assumes all angular velocities to be described within the resulting reference frame, here frame {A}. Since this is usually not the case, the angular velocities might have be to be rotated accordingly.

$$ {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{C}\leftarrow\mathrm{A}} = {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{C}\leftarrow\mathrm{B}} + {}^{\mathrm{A}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{A}} \tag{E.13} $$

F Vector Operations

This appendix serves as a look-up table for common vector operations used mainly for the model derivation.

F.1 Dot product

Get the projection of one vector onto another. Hence it captures the amount that one vector is pointing in the same direction as the other vector.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \, \|\boldsymbol{b}\| \cos(\theta) \tag{F.1}$$

where θ is the angle between the two vectors.

If the vectors are column vectors the dot product can be calculated as a vector multiplication

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^\mathsf{T} \boldsymbol{b} \tag{F.2}$$

The squared norm of a vector corresponds to the dot product of the vector itself

$$\boldsymbol{a} \cdot \boldsymbol{a} = \boldsymbol{a}^\mathsf{T} \boldsymbol{a} = \|\boldsymbol{a}\|^2 \tag{F.3}$$

F.1.1 Algebraic properties

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a} \tag{F.4}$$

$$(c\boldsymbol{a}) \cdot \boldsymbol{b} = c(\boldsymbol{a} \cdot \boldsymbol{b}) = \boldsymbol{a} \cdot (c\boldsymbol{b}) \tag{F.5}$$

$$\boldsymbol{a} \cdot (\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \cdot \boldsymbol{c} \tag{F.6}$$

$$(\boldsymbol{a} + \boldsymbol{b}) \cdot (\boldsymbol{c} + \boldsymbol{d}) = \boldsymbol{a} \cdot \boldsymbol{c} + \boldsymbol{a} \cdot \boldsymbol{d} + \boldsymbol{b} \cdot \boldsymbol{c} + \boldsymbol{b} \cdot \boldsymbol{d} \tag{F.7}$$

$$(\boldsymbol{a} + \boldsymbol{b}) \cdot (\boldsymbol{a} + \boldsymbol{b}) = \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{b} \cdot \boldsymbol{b} + 2\boldsymbol{a} \cdot \boldsymbol{b} \tag{F.8}$$

Which is then the same as

$$(\boldsymbol{a} + \boldsymbol{b})^\mathsf{T} (\boldsymbol{a} + \boldsymbol{b}) = \boldsymbol{a}^\mathsf{T} \boldsymbol{a} + \boldsymbol{b}^\mathsf{T} \boldsymbol{b} + 2\boldsymbol{a}^\mathsf{T} \boldsymbol{b} \tag{F.9}$$

For two orthogonal vectors

$$\boldsymbol{a} \cdot \boldsymbol{b} = 0 \tag{F.10}$$

For two parallel vectors

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|a\| \, \|b\| \tag{F.11}$$

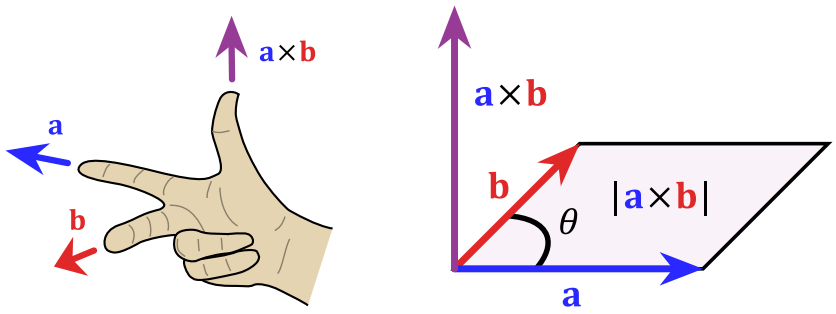
F.2 Cross product

The cross product can be used to extract the perpendicular vector to two 3-dimensional vectors, thus equivalent to the normal vector, $\boldsymbol{n}$, of the plane spanned by the two vectors. The resulting collection of now three 3-dimensional vectors will all be orthogonal to each other.

$$\boldsymbol{a} \times \boldsymbol{b} = \|\boldsymbol{a}\| \, \|\boldsymbol{b}\| \sin(\theta) \, \boldsymbol{n} \tag{F.12}$$

where θ is the angle between the two vectors.

The direction of the cross product vector is equivalent to the z-axis vector within the right-hand rule.



(a) Right-hand-rule applied to the cross-product

(b) Norm of Cross-product equivalent to parallelogram area

Figure F.1: Cross product illustrations [78]

Since the normal vector has unit-length, the norm of a cross product is:

$$\|\boldsymbol{a} \times \boldsymbol{b}\| = \|\boldsymbol{a}\| \, \|\boldsymbol{b}\| \sin(\theta) \tag{F.13}$$

The norm of the cross product can be interpreted as the area of the parallelogram spanned by the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, see Figure F.1(b). The squared norm is also related to the dot product through

$$\|\boldsymbol{a} \times \boldsymbol{b}\|^2 = \|\boldsymbol{a}\|^2 \, \|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2 \tag{F.14}$$

F.2.1 Cross product as matrix multiplication

The cross product can be computed as matrix multiplication with the skew-symmetric matrix representation of one of the vectors.

$$\begin{aligned} \boldsymbol{a} \times \boldsymbol{b} = \Omega(\boldsymbol{a}) \, \boldsymbol{b} &= \begin{bmatrix} 0 & -a_2 & a_1 \\ a_2 & 0 & -a_0 \\ -a_1 & a_0 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \\ &= \Omega(\boldsymbol{b})^\mathsf{T} \, \boldsymbol{a} = \begin{bmatrix} 0 & b_2 & -b_1 \\ -b_2 & 0 & b_0 \\ b_1 & -b_0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \end{aligned} \tag{F.15}$$

F.2.2 Algebraic properties

$$\boldsymbol{a} \times \boldsymbol{a} = \mathbf{0}_{3\times 1} \tag{F.16}$$

$$\boldsymbol{a} \times \boldsymbol{b} = -(\boldsymbol{b} \times \boldsymbol{a}) \tag{F.17}$$

$$\boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a} \times \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{c} \tag{F.18}$$

$$(c\boldsymbol{a}) \times \boldsymbol{b} = \boldsymbol{a} \times (c\boldsymbol{b}) = c(\boldsymbol{a} \times \boldsymbol{b}) \tag{F.19}$$

Since the resulting vector from the cross product is orthogonal to both vectors, the dot product with the same vectors will always be zero.

$$(\boldsymbol{a} \times \boldsymbol{b}) \cdot \boldsymbol{a} = 0 \tag{F.20}$$

$$(\boldsymbol{a} \times \boldsymbol{b}) \cdot \boldsymbol{b} = 0 \tag{F.21}$$

For two parallel vectors

$$\boldsymbol{a} \times \boldsymbol{b} = \mathbf{0}_{3\times 1} \tag{F.22}$$

The cross-product is rotation invariant

$$(\boldsymbol{R}\boldsymbol{a}) \times (\boldsymbol{R}\boldsymbol{b}) = \boldsymbol{R}\,(\boldsymbol{a} \times \boldsymbol{b}) \tag{F.23}$$

F.2.3 Differentiation

The product rule, see Appendix G.2, also applies to the cross product

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{a} \times \boldsymbol{b}) = \frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} \times \boldsymbol{b} + \boldsymbol{a} \times \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}t} \tag{F.24}$$

F.3 Tangential velocity as cross product

Similarly to how the tangential velocity of a wheel can be determined by multiplying the angular velocity with the radius of the wheel, this can be generalized to compute the tangential velocity vector, $\boldsymbol{v}$, from the angular velocity vector, $\boldsymbol{\omega}$.

$$\boldsymbol{v} = \boldsymbol{\omega} \times \boldsymbol{r} \tag{F.25}$$

where $\boldsymbol{r}$ is a constant position vector from the frame origin to the position where the tangential velocity should be determined, $\boldsymbol{v}$ is the tangential velocity vector with origin in end of $\boldsymbol{r}$ and $\boldsymbol{\omega}$ is the angular-velocity vector.
This can be generalized further by using the frame definitions. Let a contact point, ${}^{\mathrm{S}}\boldsymbol{p}_{\mathrm{C}}$, defined within a shaft frame {S}, be used to denote the origin of a contact point frame {C} as shown in (F.26).

$${}^{\mathrm{S}}_{\mathrm{C}}\boldsymbol{T} = \begin{bmatrix} \mathbf{I}_3 & {}^{\mathrm{S}}\boldsymbol{p}_{\mathrm{C}} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{F.26}$$

Furthermore, let a rotating frame whose origin is coinciding with the shaft frame be defined as {W}. Now the tangential velocity defined in the contact point frame is computed with the cross product.

$$^{\mathrm{C}}\dot{\boldsymbol{p}}_{\leftarrow \mathrm{W}} = {}^{\mathrm{S}}\boldsymbol{\omega}_{\mathrm{W}\leftarrow \mathrm{S}} \times {}^{\mathrm{S}}\boldsymbol{p}_{\mathrm{C}} \tag{F.27}$$

where $^{\mathrm{C}}\dot{\boldsymbol{p}}_{\leftarrow \mathrm{W}}$ is to be read as the linear velocity of the contact point when seen from the wheel frame, described in the contact point frame.

Since the orientation of the contact point frame is the same as the shaft frame, as defined in (F.26), the tangential velocity is the same when described in the shaft frame, $^{\mathrm{S}}\dot{\boldsymbol{p}}_{\leftarrow \mathrm{W}} = {}^{\mathrm{C}}\dot{\boldsymbol{p}}_{\leftarrow \mathrm{W}}$.

$$^{\mathrm{S}}\dot{\boldsymbol{p}}_{\leftarrow \mathrm{W}} = {}^{\mathrm{S}}\boldsymbol{\omega}_{\mathrm{W}\leftarrow \mathrm{S}} \times {}^{\mathrm{S}}\boldsymbol{p}_{\mathrm{C}} \tag{F.28}$$

G Differential Calculus

Note that the numerator layout notation, as shown in (G.1) and (G.2), is used for partial derivatives, here illustrated with $f \in \mathbb{R}$, $\boldsymbol{g} \in \mathbb{R}^m$ and $\boldsymbol{x} \in \mathbb{R}^n$.

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{G.1}$$

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix} \tag{G.2}$$

G.1 Chain rule

Derivative of inner function

$$\frac{\partial}{\partial \boldsymbol{x}} f\left(g(\boldsymbol{x})\right) = \frac{\partial f}{\partial g}\frac{\partial g}{\partial \boldsymbol{x}} \tag{G.3}$$

G.2 Product rule

Derivative of products

$$\frac{\partial}{\partial \boldsymbol{x}} \left(f(\boldsymbol{x})\, g(\boldsymbol{x})\right) = \frac{\partial f}{\partial \boldsymbol{x}} g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}} \tag{G.4}$$

G.3 Derivative of dot product

Computing the derivative of the dot product is almost as treating it as a product.

$$\frac{\partial}{\partial \boldsymbol{x}} \left(\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x})\right) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \cdot \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{f}(\boldsymbol{x}) \cdot \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \tag{G.5}$$

Since the dot product is commutative we can freely swap the order such as to put the two Jacobians on the right-most side.

$$\frac{\partial}{\partial \boldsymbol{x}} \left(\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x})\right) = \boldsymbol{g}(\boldsymbol{x}) \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x}) \cdot \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \tag{G.6}$$

This enables us to write the derivative as vector multiplications.

$$\frac{\partial}{\partial \boldsymbol{x}} \left(\boldsymbol{f}(\boldsymbol{x})^\mathsf{T} \boldsymbol{g}(\boldsymbol{x})\right) = \boldsymbol{g}(\boldsymbol{x})^\mathsf{T} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x})^\mathsf{T} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \tag{G.7}$$

For the dot product with the same vector:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{x}} \left(\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x})\right) = \frac{\partial}{\partial \boldsymbol{x}} \left(\boldsymbol{f}(\boldsymbol{x})^\mathsf{T} \boldsymbol{f}(\boldsymbol{x})\right) &= \boldsymbol{f}(\boldsymbol{x})^\mathsf{T} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x})^\mathsf{T} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \\ &= 2\boldsymbol{f}(\boldsymbol{x})^\mathsf{T} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \end{aligned} \tag{G.8}$$

Dot product with the same vector with matrix constant in between.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{x}}\left(\boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{C}\boldsymbol{f}(\boldsymbol{x})\right) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\cdot\boldsymbol{C}\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{C}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \\ &= \boldsymbol{C}\boldsymbol{f}(\boldsymbol{x})\cdot\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{C}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \\ &= \left(\boldsymbol{C}\boldsymbol{f}(\boldsymbol{x})\right)^{\mathsf{T}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{C}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \\ &= \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{C}^{\mathsf{T}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{C}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \\ &= \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}}\left(\boldsymbol{C}^{\mathsf{T}}+\boldsymbol{C}\right)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\end{aligned} \tag{G.9}$$

G.4 Derivative of matrix equations

Taking the partial derivative of equations with matrix multiplications, e.g., $\boldsymbol{A}\boldsymbol{x}$

$$\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A} \tag{G.10}$$

$$\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A} = \boldsymbol{A}^{\mathsf{T}} \tag{G.11}$$

The latter is proven by looking at the partial derivative as a column vector of elementwise derivatives. Let $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ and $\boldsymbol{x}\in\mathbb{R}^{m}$ such that the matrix vector product, $\boldsymbol{y}^{\mathsf{T}} = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}$, with $\boldsymbol{y}\in\mathbb{R}^{n}$. Let $\boldsymbol{a}_j$ denote the j-th column of $\boldsymbol{A}$ and $y_j = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{a}_j = \boldsymbol{a}_j^{\mathsf{T}}\boldsymbol{x}$ denote the j-the element of the matrix vector product.

$$\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A} = \frac{\partial \boldsymbol{y}^{\mathsf{T}}}{\partial \boldsymbol{x}} \triangleq \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix}\frac{\partial y_1}{\partial \boldsymbol{x}}\\ \vdots \\ \frac{\partial y_n}{\partial \boldsymbol{x}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{a}_1^{\mathsf{T}}\\ \vdots \\ \boldsymbol{a}_n^{\mathsf{T}}\end{bmatrix} = \boldsymbol{A}^{\mathsf{T}} \tag{G.12}$$

where the vector-vector partial derivative of a row vector, $\boldsymbol{y}^{\mathsf{T}}$ is defined as the same as with the row vector, $\boldsymbol{y}$.

For $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x}$ the product rule is applied and a similar principle as above is used to compute the partial derivative.

$$\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} = \frac{\partial}{\partial \boldsymbol{x}_a}\left(\boldsymbol{x}_a^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x}\right) + \frac{\partial}{\partial \boldsymbol{x}_b}\left(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x}_b\right) = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{A} = \boldsymbol{x}^{\mathsf{T}}\left(\boldsymbol{A}^{\mathsf{T}}+\boldsymbol{A}\right) \tag{G.13}$$

and if $\boldsymbol{A}$ is symmetric this reduces to

$$\frac{\partial}{\partial x}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A} \tag{G.14}$$

G.5 Multivariate derivative

Using the Jacobian

$$\boldsymbol{J} = \begin{bmatrix}\frac{\partial \boldsymbol{f}}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}}{\partial x_n}\end{bmatrix} = \begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n}\end{bmatrix} \tag{G.15}$$

H Quaternion Theory

To describe the orientation of the ballbot it has been decided to use quaternions. This appendix contains the relevant theory of quaternions, thus how to describe rotations with quaternions and how to deal with them mathematically. The theory is mostly based on [79] and [80] mixed with own experience. If the reader is unfamiliar with quaternions it is recommended to take a look at the following sources, [81] [82] [83] [84], before continuing. Furthermore, a comprehensive summary of the different rotation descriptions and how to convert between them, is found in [85].

Quaternions are more than just a singularity free representation of rotations, namely a complex number with a scalar part, q_0, and three orthogonal imaginary parts, q_1, q_2, q_3, often represented in vector form rather than with the complex number notation.

$$\boldsymbol{q} = q_0 + q_1\hat{\mathbf{i}} + q_2\hat{\mathbf{j}} + q_3\hat{\mathbf{k}} = \begin{bmatrix} q_0 \\ \vec{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{H.1}$$

where $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are the fundamental unit quaternions, not to be confused with the Cartesian unit vectors, e.g., $\vec{\mathbf{i}}$, since the quaternion unit vectors are four dimensional. Other representations, e.g., in ROS, defines the quaternion elements as

$$\boldsymbol{q} = \begin{bmatrix} w \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \tag{H.2}$$

However, in this thesis the definition in (H.1) is used. Note the use of the vector operator above the quaternion, $\vec{\boldsymbol{q}}$, to denote the vector part. Two operators are introduced: the devectorize operator, $\vee$, to extract the vector part of a quaternion and the vectorize operator, $\wedge$, to construct a quaternion with zero scalar value from the vector part.

$$\vec{\boldsymbol{q}} = \vee\boldsymbol{q} \tag{H.3}$$

$$\wedge\vec{\boldsymbol{q}} = \begin{bmatrix} 0 \\ \vec{\boldsymbol{q}} \end{bmatrix} \tag{H.4}$$

These operators are also defined and used as matrices:

$$\vee = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{I}_3 \end{bmatrix} \tag{H.5}$$

$$\wedge = \begin{bmatrix} \mathbf{0}_{1\times3} \\ \mathbf{I}_3 \end{bmatrix} \tag{H.6}$$

H.1 Quaternion mathematics

Similar to complex numbers general mathematical operators such as addition, subtraction, multiplication and division are defined for quaternions. It will be shown how quaternions and quaternion operations can be dealt with through matrix-vector multiplications. Dealing with the quaternion as a vector allows all the operators to be defined as either vector operations or matrix-vector operations of some kind, some of which are linear operations and others, such as the quaternion multiplication, are non-linear.

Addition and subtraction of quaternions are the same as for real numbers and real vectors.

$$\boldsymbol{d} = \boldsymbol{q} + \boldsymbol{p} = \begin{bmatrix} q_0 + p_0 \\ q_1 + p_1 \\ q_2 + p_2 \\ q_3 + p_3 \end{bmatrix} \tag{H.7}$$

$$\boldsymbol{d} = \boldsymbol{q} - \boldsymbol{p} = \begin{bmatrix} q_0 - p_0 \\ q_1 - p_1 \\ q_2 - p_2 \\ q_3 - p_3 \end{bmatrix} \tag{H.8}$$

(H.9)

Quaternion multiplication, denoted with the multiplication operator, $\circ$, is, however, very different and more complex. Similar to matrix multiplication, quaternion multiplication is commutative so that the multiplication order matters. A quaternion multiplication is defined from a set of multiplication rules for the imaginary parts, similar to complex number multiplications where $i^2 = -1$. This definition of quaternion multiplication is useful when dealing with the quaternions in their complex number notation. However, the multiplication can also be written in vector notation using the dot product and cross product.

$$\boldsymbol{q} \circ \boldsymbol{p} = \begin{bmatrix} q_0 \\ \vec{\boldsymbol{q}} \end{bmatrix} \circ \begin{bmatrix} p_0 \\ \vec{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} q_0 p_0 - \vec{\boldsymbol{q}} \cdot \vec{\boldsymbol{p}} \\ q_0 \vec{\boldsymbol{p}} + p_0 \vec{\boldsymbol{q}} + \vec{\boldsymbol{q}} \times \vec{\boldsymbol{p}} \end{bmatrix} \tag{H.10}$$

Written out this quaternion multiplication is computed as shown below, organized in two different ways.

$$\boldsymbol{q} \circ \boldsymbol{p} = \underbrace{\begin{bmatrix} q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3 \\ q_1 p_0 + q_0 p_1 - q_3 p_2 + q_2 p_3 \\ q_2 p_0 + q_3 p_1 + q_0 p_2 - q_1 p_3 \\ q_3 p_0 - q_2 p_1 + q_1 p_2 + q_0 p_3 \end{bmatrix}}_{\boldsymbol{\Phi}(\boldsymbol{q})\boldsymbol{p}} = \underbrace{\begin{bmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_1 q_0 + p_0 q_1 + p_3 q_2 - p_2 q_3 \\ p_2 q_0 - p_3 q_1 + p_0 q_2 + p_1 q_3 \\ p_3 q_0 + p_2 q_1 - p_1 q_2 + p_0 q_3 \end{bmatrix}}_{\boldsymbol{\Gamma}(\boldsymbol{p})\boldsymbol{q}} \tag{H.11}$$

From the organized way it is clear how the quaternion multiplication can also be represented through matrix-vector multiplication, using a matrix with a specific structure constructed from the quaternion.

$$\boldsymbol{q} \circ \boldsymbol{p} = \boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{p} = \boldsymbol{\Gamma}(\boldsymbol{p})\,\boldsymbol{q} \tag{H.12}$$

Here $\Phi(\boldsymbol{q})$ denotes the matrix operator for left quaternion multiplication and $\Gamma(\boldsymbol{p})$ denotes the matrix operator for right quaternion multiplication.

Noted differently with a slight abuse of notation:

$$\boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{q} \circ \tag{H.13}$$

$$\boldsymbol{\Gamma}(\boldsymbol{p}) = \circ \boldsymbol{p} \tag{H.14}$$

These matrix operators are defined as follows

$$\boldsymbol{\Phi}(\boldsymbol{q}) = \begin{bmatrix} q_0 & -\vec{\boldsymbol{q}}^\mathsf{T} \\ \vec{\boldsymbol{q}}^\mathsf{T} & \mathbf{I}_3 q_0 + \Omega(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \tag{H.15}$$

$$\boldsymbol{\Gamma}(\boldsymbol{p}) = \begin{bmatrix} p_0 & -\vec{\boldsymbol{p}}^\mathsf{T} \\ \vec{\boldsymbol{p}}^\mathsf{T} & \mathbf{I}_3 q_0 - \Omega(\boldsymbol{p}) \end{bmatrix} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \tag{H.16}$$

where $\Omega(\boldsymbol{v})$ is the skew-symmetric matrix representation of a vector, see Appendix F.2. Hence the best way to think of quaternion multiplications is as a linear matrix operator $\boldsymbol{\Phi}(\boldsymbol{q})$ such that $\boldsymbol{q} \circ \boldsymbol{p} \to \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{p}$ and conversely for right multiplication, $\boldsymbol{q} \circ \boldsymbol{p} \to \boldsymbol{\Gamma}(\boldsymbol{p})\, \boldsymbol{q}$.

Similar to matrices the inverse quaternion exists from definition of an identity property. This requires the introduction of a unit quaternion which holds the multiplicative property of unity. This means that any quaternion multiplied with the unit quaternion, no matter if it is a left of right multiplication, results in the same quaternion.

$$\boldsymbol{q}_{\text{unit}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{H.17}$$

$$\boldsymbol{q}_{\text{unit}} \circ \boldsymbol{p} = \boldsymbol{p} \tag{H.18}$$

$$\boldsymbol{p} \circ \boldsymbol{q}_{\text{unit}} = \boldsymbol{p} \tag{H.19}$$

The inverse quaternion should thus fulfil the identity property:

$$\boldsymbol{q} \circ \boldsymbol{q}^{-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{H.20}$$

Quaternions have the property that the inverse is very easy to find and always exist.

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|^2} = \frac{\boldsymbol{q}^*}{\boldsymbol{q}^\mathsf{T}\boldsymbol{q}} \tag{H.21}$$

The conjugated quaternion, $\boldsymbol{q}^*$, is defined as the quaternion with negated vector part.

$$\boldsymbol{q}^* = \begin{bmatrix} q_0 \\ -\vec{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix} \tag{H.22}$$

This results in the following property of multiplication with the conjugated quaternion.

$$\boldsymbol{q}^* \circ \boldsymbol{q} = \left(\boldsymbol{q}^\mathsf{T} \boldsymbol{q} \boldsymbol{q}^{-1}\right) \circ \boldsymbol{q} = \boldsymbol{q}^\mathsf{T} \boldsymbol{q} \left(\boldsymbol{q}^{-1} \circ \boldsymbol{q}\right) = \boldsymbol{q}^\mathsf{T} \boldsymbol{q} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{H.23}$$

Due to the construction of the matrix operators in (H.15) and (H.16), multiplying with a conjugated quaternion is the same as multiplying with the transposed matrix operator.

$$\boldsymbol{\Phi}\left(\boldsymbol{q}^*\right) = \boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T} = \boldsymbol{q}^* \circ \tag{H.24}$$

$$\boldsymbol{\Gamma}\left(\boldsymbol{p}^*\right) = \boldsymbol{\Gamma}(\boldsymbol{p})^\mathsf{T} = \circ \boldsymbol{p}^* \tag{H.25}$$

Similarly to the above result, combining the matrix operator of a conjugated quaternion with the non-conjugated operator, results in a scaled identity matrix.

$$\boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T} \boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{q}^\mathsf{T} \boldsymbol{q} \, \mathbf{I}_4 \tag{H.26}$$

$$\boldsymbol{\Gamma}(\boldsymbol{p})^\mathsf{T} \boldsymbol{\Gamma}(\boldsymbol{p}) = \boldsymbol{p}^\mathsf{T} \boldsymbol{p} \, \mathbf{I}_4 \tag{H.27}$$

To enable arbitrary computations with quaternions using matrix-vector multiplications the conjugate operator, $\mathbf{I}^*$, is introduced.

$$\boldsymbol{q}^* = \mathbf{I}^* \boldsymbol{q} \tag{H.28}$$

where

$$\mathbf{I}^* = \begin{bmatrix} 1 & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{3\times 1} & -\mathbf{I}_3 \end{bmatrix} \tag{H.29}$$

Please note that the following relationships

$$\boldsymbol{\Phi}\left(\mathbf{I}^* \boldsymbol{q}\right) = \boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T} \tag{H.30}$$

$$\boldsymbol{\Phi}\left(\mathbf{I}^* \boldsymbol{q}\right) \neq \mathbf{I}^* \boldsymbol{\Phi}(\boldsymbol{q}) \tag{H.31}$$

and conversely for $\boldsymbol{\Gamma}(\boldsymbol{p})$.

H.2 Quaternions for rotation

Since a rotation in 3-dimensional space only spans three degrees of freedom, using a quaternion with four elements will be overdetermined. When using quaternions to describe rotations a unit norm constraint is thus required, which puts all rotation quaternions on a 4-dimensional unit sphere.

$$\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q}^\mathsf{T} \boldsymbol{q}} = 1 \tag{H.32}$$

$$\Rightarrow \boldsymbol{q}^\mathsf{T} \boldsymbol{q} = 1 \tag{H.33}$$

Therefore, rotation quaternions are not closed under addition and subtraction and is thus only closed under quaternion multiplication.

Note that for quaternions with unit length the previous definition of the inverse quaternion simplifies to just the conjugate. A property that is comparable to rotation matrices where the inverse corresponds to the transpose.

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^* \Big|_{\|\boldsymbol{q}\|=1} \tag{H.34}$$

Therefore, we also have

$$\boldsymbol{q}^* \circ \boldsymbol{q} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{H.35}$$

Rotational quaternions, in the latter generalized and just referred to as quaternions, have the property of being able to describe a rotation by a rotation axis, $\boldsymbol{r}$ and a rotation amount (angle), θ.

$$\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \boldsymbol{r}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{H.36}$$

Note that the inverse quaternion, thus rotating in the opposite direction, is given by the conjugate corresponding to inverting the rotation axis.
The definition in (H.36) gives the rotation quaternions their special singularity free ability to represent any 3-dimensional rotation, but it comes at a cost. Stating that the quaternions are singularity free does not mean that a single rotation has a unique quaternion representation. Due to the construction two quaternions exist for each unique rotation, one on each side of the four dimensional unit sphere. Negating all elements of a quaternion thus represents the same rotation, but it represents it by stating that the rotation is carried out by rotating with a certain amount plus 360°. Rotating a vector with the negative quaternion yields the same result. However, when dealing with controllers using the quaternion error, as presented in Section 8.1, it is crucial to consider the sign of the scalar part since this indicates whether the error is a short or long rotation, see Section 8.1.1.

In this thesis quaternions are used to describe the rotation between two frames, namely the body frame and the ball frame (note that the orientation of the ball frame is coincident with the inertial frame).

Since the rotation axis, around which the rotation is defined, is given in a certain frame, the notation of a rotation quaternion is updated to reflect these frames. The superscript defines in which frame the rotation axis is defined and also to which frame the rotation is carried out, in this case the ball frame. The subscript defines the frame from which the quaternion rotates from, in this case the body frame.

$${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ {}^{\mathrm{K}}\boldsymbol{r}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{H.37}$$

In this case ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$ defines the rotation from frame $\{B\}$ to frame $\{K\}$. Note that similar to rotation matrices, the rotation is represented as how to rotate the final frame, namely frame $\{K\}$, to the initial frame, in this case the body frame $\{B\}$. The quaternion should thus be interpreted as a rotation axis defined in the ball frame, around which a rotation amount transforms the ball frame into the body frame.

As a side note, even though the rotation axis is said to be defined in a single frame, the rotation axis is identical in both frames, ${}^{\mathrm{K}}\boldsymbol{r} = {}^{\mathrm{B}}\boldsymbol{r}$. Note that this is a consequence of the axis-angle definition.

H.3 Vector rotation

Rotating a vector with a quaternion is carried out using a sequence of quaternion multiplications. To rotate a 3-dimensional vector, $\boldsymbol{p}$, the vector has to be be converted to a 4-dimensional quaternion representation. A 3-dimensional vector is represented in a quaternion by setting the scalar, q_0, to 0 and set the vector part, $\vec{\boldsymbol{q}}$, to the vector, $\boldsymbol{p}$. The quaternion representation of a vector, $\boldsymbol{p}$, will be denoted $\tilde{\boldsymbol{p}}$.

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix} = \wedge \boldsymbol{p} \tag{H.38}$$

Note the use of the defined vectorize operator, $\wedge$. Rotating a vector defined in the body frame, ${}^{\mathrm{B}}\boldsymbol{p}$, with a rotation defined by the quaternion, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, results in the corresponding vector defined in the ball frame, ${}^{\mathrm{K}}\boldsymbol{p}$. This rotation is shown in (H.39).

$${}^{\mathrm{K}}\tilde{\boldsymbol{p}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \tag{H.39}$$

Note the use of the quaternion representation for both vectors, ${}^{\mathrm{B}}\tilde{\boldsymbol{p}}$ and ${}^{\mathrm{K}}\tilde{\boldsymbol{p}}$. The reason that we have to both pre- and post-multiply with the quaternion is due to the way that a quaternion only carries half of the rotation amount, $\frac{\theta}{2}$, see (H.37). Conversely the inverse rotation, thus rotating a vector defined in the ball frame to its corresponding vector in the body frame, is carried out by reversing the order.

$${}^{\mathrm{B}}\tilde{\boldsymbol{p}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{H.40}$$

The matrix operators from (H.13) and (H.14) can be applied to convert the quaternion multiplications into matrix multiplications. To split the equation into matrix operations it has to be partitioned first.

$$\begin{aligned} {}^{\mathrm{B}}\tilde{\boldsymbol{p}} &= \underbrace{{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ}_{\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}} \left({}^{\mathrm{K}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) && \text{(H.41)} \\ &= \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \underbrace{\left({}^{\mathrm{K}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)}_{\boldsymbol{\Phi}\left({}^{\mathrm{K}}\tilde{\boldsymbol{p}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}} = \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \underbrace{\left({}^{\mathrm{K}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)}_{\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{K}}\tilde{\boldsymbol{p}}} && \text{(H.42)} \\ &= \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{K}}\tilde{\boldsymbol{p}} && \text{(H.43)} \end{aligned}$$

Similarly for the rotation in the opposite direction

$${}^{\mathrm{K}}\tilde{\boldsymbol{p}} = \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \tag{H.44}$$

Since the rotated vector result is represented as a quaternion, it might be desirable to extract the vector part by using the devectorize operator. Combining the vectorize and devectorize operators and the matrix operators, yields a single equation to rotate a given 3-dimensional vector with a quaternion.

$${}^{\mathrm{B}}\boldsymbol{p} = \underbrace{\vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \wedge}_{{}^{\mathrm{B}}_{\mathrm{K}}\boldsymbol{R}} {}^{\mathrm{K}}\boldsymbol{p} \tag{H.45}$$

$${}^{\mathrm{K}}\boldsymbol{p} = \underbrace{\vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \wedge}_{{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{R}} {}^{\mathrm{B}}\boldsymbol{p} \tag{H.46}$$

One thing is to be able to rotate a vector with a quaternion, another thing is to compute the rotation, in terms of a quaternion, between two vectors, $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$. As described in Appendix F.2 the cross product can be used to extract the normal vector to the plane spanned by the two vectors. This vector corresponds to the rotation axis. The dot product is then used to compute the rotation amount. The final quaternion is constructed by combining these elements.

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{p}_A \times \boldsymbol{p}_B \\ \theta &= \cos^{-1}\left(\frac{\boldsymbol{p}_A \cdot \boldsymbol{p}_B}{\|\boldsymbol{p}_A\| \, \|\boldsymbol{p}_B\|}\right) \\ {}^{\mathrm{A}}_{\mathrm{B}}\boldsymbol{q} &= \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \boldsymbol{r} \sin\left(\frac{\theta}{2}\right) \end{bmatrix} \end{aligned} \tag{H.47}$$

H.4 Quaternion dynamics

The derivative of a quaternion is indeed the elementwise derivatives.

$${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{H.48}$$

However, since the quaternion lives on a four-dimensional sphere a kinematic relationship governs the evolution of the quaternion and thus governs the derivative of the quaternion [86]. This relationship is based on the 3-dimensional relative angular velocity between the two frames that can be defined in either of the two frames, ${}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}$ or ${}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}$, see Appendix E. Note that the angular velocity vector is also represented in a quaternion by vectorization, $\tilde{\boldsymbol{\omega}}$, see (H.38). The kinematic relationships are shown in (H.49) and (H.50).

$${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = \frac{1}{2} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \tag{H.49}$$

$$= \frac{1}{2} {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{H.50}$$

Conversely the angular velocity of a quaternion can be computed with the quaternion derivative. By pre-multiplying (H.49) with $2\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^*$ we get

$${}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2 \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{H.51}$$

which can then be transformed with (H.39) to yield

$${}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* = 2 \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \tag{H.52}$$

Finding derivatives of equations involving quaternions and quaternion multiplications is as simple as finding derivatives of time varying vector functions. By keeping in mind that quaternion multiplications corresponds to matrix multiplications with matrix functions defined from the quaternion, the rules for taking derivatives of matrix and matrix-vector products can be used. Finding the time derivative of the vector rotation, (H.39), simply requires the use of the product rule, see Appendix G.2.

$$\begin{aligned} {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}} = \frac{\mathrm{d}}{\mathrm{d}t}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^*\right) &= {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \\ &+ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\dot{\tilde{\boldsymbol{p}}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \\ &+ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^* \end{aligned} \tag{H.53}$$

For a constant vector within a rotating frame, ${}^{\mathrm{B}}\dot{\tilde{\boldsymbol{p}}} = \mathbf{0}$, the above equation simplifies to a sum of two terms.

$$\begin{aligned}
{}^{\mathrm{K}}\dot{\boldsymbol{p}} &= \vee \left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} + {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{*} \right) \\
&= \vee \left(\underbrace{\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \boldsymbol{\Phi}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right) \mathbf{I}^{*}}_{\boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)} + \underbrace{\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right)}_{\boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)} \right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\
&= \vee \left(\underbrace{\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \boldsymbol{\Phi}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right) \mathbf{I}^{*}}_{\boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)} + \underbrace{\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right)}_{\boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)} \right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}
\end{aligned} \tag{H.54}$$

Two new operators, $\boldsymbol{\Pi}_{\mathrm{p}}(\boldsymbol{q})$ and $\boldsymbol{\Lambda}_{\mathrm{p}}(\boldsymbol{q})$, are defined to simplify equations involving vector rotations. The operators are tied to a certain vector denoted in the subscript, here shown with p.

$$\boldsymbol{\Pi}_{\mathrm{p}}(\boldsymbol{q}) = \boldsymbol{\Phi}(\boldsymbol{q}) \boldsymbol{\Phi}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right) \mathbf{I}^{*} \tag{H.55}$$

$$\boldsymbol{\Lambda}_{\mathrm{p}}(\boldsymbol{q}) = \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right) \tag{H.56}$$

Due to the construction of these operators there is furthermore a key relationship between their vector parts.

$$\vee \boldsymbol{\Pi}_{\mathrm{p}}(\boldsymbol{q}) = \vee \boldsymbol{\Lambda}_{\mathrm{p}}(\boldsymbol{q}) \tag{H.57}$$

The angular velocity from (H.53) thus simplifies to

$$\begin{aligned}
{}^{\mathrm{K}}\dot{\boldsymbol{p}} = \left(\vee \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) + \vee \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} &= 2 \vee \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\
&= 2 \vee \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\
= \left(\vee \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) + \vee \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} &= 2 \vee \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\
&= 2 \vee \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}
\end{aligned} \tag{H.58}$$

Which thereby also simplifies the vectorized version

$$\begin{aligned}
{}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}} &= 2\boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2\boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\
&= 2\boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2\boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}
\end{aligned} \tag{H.59}$$

Another relationship with the operator is how it simplifies when multiplied with its transpose.

$$\begin{aligned}
\boldsymbol{\Pi}_{\mathrm{p}}(\dot{\boldsymbol{q}})^{\mathsf{T}} \boldsymbol{\Pi}_{\mathrm{p}}(\dot{\boldsymbol{q}}) &= \boldsymbol{\Lambda}_{\mathrm{p}}(\dot{\boldsymbol{q}})^{\mathsf{T}} \boldsymbol{\Lambda}_{\mathrm{p}}(\dot{\boldsymbol{q}}) \\
&= \boldsymbol{\Gamma}(\tilde{\boldsymbol{p}})^{\mathsf{T}} \boldsymbol{\Gamma}(\dot{\boldsymbol{q}}) \boldsymbol{\Gamma}(\dot{\boldsymbol{q}})^{\mathsf{T}} \boldsymbol{\Gamma}(\tilde{\boldsymbol{p}}) \\
&= \boldsymbol{\Gamma}(\tilde{\boldsymbol{p}})^{\mathsf{T}} \dot{\boldsymbol{q}}^{\mathsf{T}} \dot{\boldsymbol{q}} \mathbf{I}_4 \boldsymbol{\Gamma}(\tilde{\boldsymbol{p}}) \\
&= \dot{\boldsymbol{q}}^{\mathsf{T}} \dot{\boldsymbol{q}} \boldsymbol{\Gamma}(\tilde{\boldsymbol{p}})^{\mathsf{T}} \boldsymbol{\Gamma}(\tilde{\boldsymbol{p}}) \\
&= \dot{\boldsymbol{q}}^{\mathsf{T}} \dot{\boldsymbol{q}} \tilde{\boldsymbol{p}}^{\mathsf{T}} \tilde{\boldsymbol{p}} \mathbf{I}_4 \\
&= \dot{\boldsymbol{q}}^{\mathsf{T}} \dot{\boldsymbol{q}} \boldsymbol{p}^{\mathsf{T}} \boldsymbol{p} \mathbf{I}_4
\end{aligned} \tag{H.60}$$

These properties are useful when computing the kinetic energy in Section 5.2.2, where the squared norm of the velocity, ${}^{\mathrm{K}}\dot{\boldsymbol{p}}^{\mathsf{T}}\,{}^{\mathrm{K}}\dot{\boldsymbol{p}}$, is used.

Conversely the well known equation for the derivative of a rotating point using the scew-symmetric angular velocity matrix appears when inserting the definition for the quaternion derivative, (H.50), with the angular velocity defined in the ball frame.

$$\begin{aligned} {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}} &= \frac{1}{2}\,{}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* + \frac{1}{2}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ \left({}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^* \\ &= \frac{1}{2}\,{}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ \underbrace{{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^*}_{{}^{\mathrm{K}}\tilde{\boldsymbol{p}}} + \frac{1}{2}\underbrace{{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^*}_{{}^{\mathrm{K}}\tilde{\boldsymbol{p}}} \circ {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}^*_{\mathrm{B}\leftarrow\mathrm{K}} \end{aligned} \tag{H.61}$$

Noting the transformation relationship from (H.39) the equation collapses to

$${}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}} = \frac{1}{2}\,{}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}\tilde{\boldsymbol{p}} + \frac{1}{2}\,{}^{\mathrm{K}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}^*_{\mathrm{B}\leftarrow\mathrm{K}} \tag{H.62}$$

Since both ${}^{\mathrm{K}}\tilde{\boldsymbol{p}}$ and ${}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}$ have zero scalar part, the above equation simplifies to just the cross-product from (H.10), which can be computed using the scew-symmetric matrix as described in Appendix F.2.

$${}^{\mathrm{K}}\dot{\boldsymbol{p}} = {}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}} \times {}^{\mathrm{K}}\boldsymbol{p} = \boldsymbol{\Omega}({}^{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}})\,{}^{\mathrm{K}}\boldsymbol{p} \tag{H.63}$$

H.5 Numerical integration

Performing a numerical discrete integration from one time step to another is useful when simulating dynamic systems and when dealing with discrete controller and estimator implementations. Computing the discrete integration step for a dynamic system is a matter of finding the solution to the continuous differential equation from t_0 to t_1. For Linear Time-Invariant (LTI) systems described by $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$, the solution is given by the matrix exponential given an integration period of $\Delta t = t_1 - t_0$

$$\begin{aligned} \boldsymbol{x}(t) &= \mathrm{e}^{\boldsymbol{A}t}\,\boldsymbol{x}(0) \\ x[k] &= \mathrm{e}^{\boldsymbol{A}\Delta t}\,x[k-1] \end{aligned} \tag{H.64}$$

A similar exponential, called the quaternion exponential, exists for integration of quaternions. Through a Taylor series approximation of the exponential an approximate closed form solution to the exponential has been derived.

$$\mathrm{e}^{\boldsymbol{q}} = \mathrm{e}^{q_0} \begin{bmatrix} \cos(\|\tilde{\boldsymbol{q}}\|) \\ \frac{\tilde{\boldsymbol{q}}}{\|\tilde{\boldsymbol{q}}\|}\sin(\|\tilde{\boldsymbol{q}}\|) \end{bmatrix} \tag{H.65}$$

Using the quaternion derivative definition from (H.49) with the body angular velocity, the closed form solution to quaternion integration is given as [87] [88]

$$\begin{aligned} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}(t) &= \mathrm{e}^{\frac{1}{2}t\,{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}(0) \\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] &= {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \circ \begin{bmatrix} \cos\left(\frac{\Delta t}{2}\,\left\|\,{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right\|\right) \\ \frac{{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]}{\left\|{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right\|}\sin\left(\frac{\Delta t}{2}\,\left\|\,{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right\|\right) \end{bmatrix} \end{aligned} \tag{H.66}$$

This is the most exact approximation to integrate angular velocity and update the quaternion, but it is numerically unstable when the norm of the angular velocity approaches zero, due to the denominator.

Another way to approximate the quaternion integration is as a Linear Time-Invariant (LTI) system given the angular velocity of the rotation. Applying the matrix operators to (H.49) and assuming constant angular velocity over the integration interval, yields a four-dimensional LTI system:

$$ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \Big) \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{H.67} $$

The solution to this ODE is the matrix exponential as shown in (H.64). The integrated quaternion is thus given as

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = \mathrm{e}^{\frac{1}{2}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1] \right)\Delta t} \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \tag{H.68} $$

Depending on the application and necessary accuracy the matrix exponential might be too computationally expensive. The simplest type of approximate integration is obviously the Forward Euler method [89] [90]. As always with Forward Euler the accuracy degrades with the duration of Δt. The smaller step size the more accurate.

$$ \begin{aligned} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] &= {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] + \Delta t \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}[k-1] \\ &= {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] + \frac{1}{2}\Delta t \boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1] \Big) \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \\ &= \left(\mathbf{I}_4 + \frac{1}{2}\Delta t \boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1] \Big) \right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \end{aligned} \tag{H.69} $$

However, the above definition voids one of the most essential arithmetic constraints of the quaternions, namely that they are not closed under addition and subtraction. Therefore, to ensure that the above approximation yields a valid quaternion a normalization step is usually performed after each integration step to keep the quaternion on the unit sphere [91].

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = \frac{{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k]}{\left\| {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] \right\|} \tag{H.70} $$

Instead of adding the computed derivative directly to the quaternion, a delta quaternion can be computed through small angle approximation and added to the quaternion through proper quaternion multiplication. With a small Δt the amount of change in rotation will be small allowing a small angle approximation of (H.36) where $\cos\left(\frac{\theta}{2}\right) \approx 1$.

$$ \Delta \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] = \begin{bmatrix} 1 \\ \frac{1}{2}\Delta t \, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1] \end{bmatrix} \tag{H.71} $$

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \circ \Delta \, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \tag{H.72} $$

Note that due to the small angle approximations, it is still recommended to normalize the resulting quaternion to ensure that the unit constraint is fulfilled.

H.6 Euler angles

In Chapter 1 Euler angles were introduced as the most common way to represent 3-dimensional rotations since they are intuitive and fairly simple to interpret. With Euler angles any rotation is decomposed into an ordered sequence of three rotations along individual axes. Most commonly, especially within aerospace, the Euler 3-2-1 sequence, also known as Euler ZYX sequence, is used, where a sequence of yaw, pitch and roll angles describe the rotation [27].

Roll (ϕ), pitch (θ) and yaw (ψ) corresponds to rotations around the X, Y and Z axes respectively.

$$\begin{aligned}
\mathbf{R}_X(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \\
\mathbf{R}_Y(\theta) &= \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\
\mathbf{R}_Z(\psi) &= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{H.73}$$

where $\mathbf{R}_X(\phi)$ is the Roll SO(3) rotation matrix around the X-axis of ϕ amount, $\mathbf{R}_Y(\theta)$ is the Pitch SO(3) rotation matrix around the Y-axis of θ amount and $\mathbf{R}_Z(\psi)$ is the Yaw SO(3) rotation matrix around the Z-axis of ψ amount.

The Euler ZYX sequence for a rotation from body frame to ball frame, is defined as shown in (H.74).

$${}^{\mathrm{K}}_{\mathrm{B}}\mathbf{R}(\phi,\theta,\psi) = \mathbf{R}_Z(\psi)\mathbf{R}_Y(\theta)\mathbf{R}_X(\phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{H.74}$$

where ${}^{\mathrm{K}}_{\mathrm{B}}\mathbf{R}(\phi,\theta,\psi)$ is the combined SO(3) rotation matrix of Roll, Pitch and Yaw from frame {B} to frame {K}. A rotation is constructed by:

1. Starting in frame {K}, then rotating this with ψ around its z-axis
2. Rotating the new intermediate frame by θ around its y-axis (intermediate y-axis)
3. Finally, rotating the new intermediate frame by ϕ around its intermediate x-axis

The resulting rotation can be identified by reading the sequence from *left to right* with a moving frame, which means that each rotation would be described according to the principal axes of the rotating frame, starting with the final frame {K}. Thus the direction of each axis might change after each rotation.

The reason for choosing this sequence lies in readability of the roll, pitch and yaw angles. The yaw angle describes the heading of the robot by defining the direction of the body x-axis in the xy-plane of the ball frame. The roll and pitch angles defines the inclination of the robot in this constructed heading frame, thus the amount that the robot is pitched forward/backward followed by the amount that the robot is rolled around the resulting x-axis.

Quaternions are far more difficult to interpret and visualize. When a quaternion is close to the unit quaternion, the vector elements of the x and y axes will be close to half of the roll and pitch angles, thus a small angle approximation.

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\Big|_{\phi,\theta,\psi \ll 10^\circ} \approx \begin{bmatrix} 1 \\ \frac{\phi}{2} \\ \frac{\theta}{2} \\ \frac{\psi}{2} \end{bmatrix} \tag{H.75} $$

However, to enable visualization of the robot orientation with the readable Euler angles, a conversion from quaternion to Euler angles is desired. Since Euler angles are not unique such a conversion will not be unique either. We will thus focus on just the Euler ZYX sequence.

If we just wanted to extract the heading from the quaternion, thus the direction in which the body x-axis is pointing in the ball xy-plane, the x-axis base vector of the body frame, ${}^{\mathrm{K}}\mathbf{i}_{\mathrm{B}}$, could just be projected onto the xy-plane of the ball frame. Using (H.39) the projected x-axis vector and heading angle is computed as follows

$$ \begin{aligned} {}^{\mathrm{K}}\mathbf{i}_{\mathrm{B}} &= \vee\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ \hat{\mathbf{i}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^*\right) \\ {}^{\mathrm{K}}\mathbf{i}_{\mathrm{B},0} &= q_0^2 + q_1^2 - q_2^2 - q_3^2 \\ {}^{\mathrm{K}}\mathbf{i}_{\mathrm{B},1} &= 2(q_0 q_3 + q_1 q_2) \\ \psi &= \operatorname{atan2}\left({}^{\mathrm{K}}\mathbf{i}_{\mathrm{B},1},\ {}^{\mathrm{K}}\mathbf{i}_{\mathrm{B},0}\right) \\ &= \operatorname{atan2}\left(2(q_0 q_3 + q_1 q_2),\ q_0^2 + q_1^2 - q_2^2 - q_3^2\right) \end{aligned} \tag{H.76} $$

The above equation illustrates how one of the Euler angles, namely the heading or yaw angle, ψ, is extracted from the quaternion. The conversion of the remaining angles are shown below [85].

$$ \begin{aligned} \psi &= \operatorname{atan2}\left(2(q_0 q_3 + q_1 q_2),\ q_0^2 + q_1^2 - q_2^2 - q_3^2\right) \\ \theta &= \sin^{-1}(2(q_0 q_2 - q_1 q_3)) \\ \phi &= \operatorname{atan2}\left(2(q_0 q_1 + q_2 q_3),\ q_0^2 - q_1^2 - q_2^2 + q_3^2\right) \end{aligned} \tag{H.77} $$

To construct a quaternion from an Euler sequence, the same rotation sequence are constructed with right-multiplied quaternions. The way to read and interpret a sequence of quaternion multiplications are from left to right. Successive rotations performed by quaternion multiplication are therefore ordered from left to right in the applied order. Note how the delta quaternion in (H.72), was right multiplied since the incremental rotation was based on the current orientation. Similarly the Euler ZYX sequence from (H.73) can be constructed as a quaternion sequence as shown in (H.78) [28] [86].

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \boldsymbol{q}_{\mathrm{Z}}(\psi) \circ \boldsymbol{q}_{\mathrm{Y}}(\theta) \circ \boldsymbol{q}_{\mathrm{X}}(\phi) = \begin{bmatrix} \cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \\ \cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) - \sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) \\ \cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \\ \sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) - \cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \end{bmatrix} \tag{H.78} $$

With the quaternions for individual axis rotations defined as

$$\boldsymbol{q}_{\mathrm{X}}(\phi) = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) & \sin\left(\frac{\phi}{2}\right) & 0 & 0 \end{bmatrix}^{\mathsf{T}} \tag{H.79}$$

$$\boldsymbol{q}_{\mathrm{Y}}(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & \sin\left(\frac{\theta}{2}\right) & 0 \end{bmatrix}^{\mathsf{T}} \tag{H.80}$$

$$\boldsymbol{q}_{\mathrm{Z}}(\psi) = \begin{bmatrix} \cos\left(\frac{\psi}{2}\right) & 0 & 0 & \sin\left(\frac{\psi}{2}\right) \end{bmatrix}^{\mathsf{T}} \tag{H.81}$$

H.7 Mathematical tricks (equation manipulation)

As a last part a few mathematical tricks useful for simplifying equations with quaternions are presented. These tricks are used throughout the thesis when dealing with equations involving quaternions.

Remember the relationship of the conjugate quaternion corresponding to transposing the matrix operator. Conjugating a quaternion product is therefore the same as transposing a matrix product:

$$(\boldsymbol{q} \circ \boldsymbol{p})^* = \boldsymbol{p}^* \circ \boldsymbol{q}^* \tag{H.82}$$

The inverse of the matrix operator can be obtained by mapping the inverse of the quaternion to a matrix or by normalizing the transposed of the matrix.

$$\boldsymbol{\Phi}(\boldsymbol{q})^{-1} = \boldsymbol{\Phi}\left(\boldsymbol{q}^{-1}\right) = \boldsymbol{\Phi}\left(\frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|}\right) = \frac{1}{\|\boldsymbol{q}\|}\boldsymbol{\Phi}\left(\boldsymbol{q}^*\right) = \frac{1}{\|\boldsymbol{q}\|}\boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}} \tag{H.83}$$

Quaternion multiplication can also be used to compute cross products. In Appendix F.2.1 it is shown how a cross product can be computed through multiplication with a scew-symmetric matrix. The quaternion multiplication holds similar properties for the vector part, see (H.10), (H.15) and (H.16).

$$\boldsymbol{a} \times \boldsymbol{b} = \tilde{\boldsymbol{a}} \circ \tilde{\boldsymbol{b}} = \vee\, \boldsymbol{\Phi}(\wedge \boldsymbol{a}) \wedge \boldsymbol{b} \tag{H.84}$$

Several other rules apply to the matrix operators, $\boldsymbol{\Phi}(\boldsymbol{q})$ and $\boldsymbol{\Gamma}(\boldsymbol{p})$. The matrix of a product of two quaternions is for example equal to the product of the matrices of each quaternion individually.

$$\boldsymbol{\Phi}(\boldsymbol{q} \circ \boldsymbol{p}) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p}) \tag{H.85}$$

which is related to the following inner operator relationships:

$$\boldsymbol{\Phi}(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{p}) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p}) \tag{H.86}$$

$$\boldsymbol{\Phi}\left(\boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{p}\right) = \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Phi}(\boldsymbol{p}) \tag{H.87}$$

$$\boldsymbol{\Phi}(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{p})^{\mathsf{T}} = (\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p}))^{\mathsf{T}} = \boldsymbol{\Phi}(\boldsymbol{p})^{\mathsf{T}}\, \boldsymbol{\Phi}(\boldsymbol{q}) \tag{H.88}$$

Note that all of the above is similarly applicable to the right multiplication operator, $\boldsymbol{\Gamma}(\boldsymbol{p})$.

Furthermore, the matrix operators hold the property of reversed order equivalence:

$$\boldsymbol{\Gamma}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}} = \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Gamma}(\boldsymbol{q}) \tag{H.89}$$

Actually the order of multiplications are only tied to the sequence of $\boldsymbol{\Phi}(\boldsymbol{q})$ parts and $\boldsymbol{\Gamma}(\boldsymbol{p})$ parts respectively.

$$\begin{aligned} \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Phi}(\dot{\boldsymbol{q}})\, \boldsymbol{\Gamma}(\dot{\boldsymbol{q}})\, \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}} &= \boldsymbol{\Gamma}(\dot{\boldsymbol{q}})\, \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Phi}(\dot{\boldsymbol{q}})\, \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}} \\ &= \boldsymbol{\Gamma}(\dot{\boldsymbol{q}})\, \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Phi}(\dot{\boldsymbol{q}}) \\ &= \boldsymbol{\Gamma}(\dot{\boldsymbol{q}})\, \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Phi}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Phi}(\dot{\boldsymbol{q}}) \end{aligned} \tag{H.90}$$

Note that the inner operator relationships also applies to a combination of the two matrix operators.

$$\boldsymbol{\Gamma}\Big(\boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{\Phi}(\boldsymbol{b})\,\boldsymbol{\Phi}(\boldsymbol{p})^{\mathsf{T}}\Big) = \boldsymbol{\Gamma}(\boldsymbol{p})^{\mathsf{T}}\,\boldsymbol{\Gamma}(\boldsymbol{b})\,\boldsymbol{\Gamma}(\boldsymbol{q}) \tag{H.91}$$

An exaggerated example of equation manipulation is shown below

$$\begin{aligned}\boldsymbol{q}\circ\boldsymbol{p}^*\circ\dot{\boldsymbol{q}}\circ\boldsymbol{q}^* &= \boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{\Phi}(\boldsymbol{p})^{\mathsf{T}}\,\boldsymbol{\Phi}(\dot{\boldsymbol{q}})\,\mathbf{I}^*\,\boldsymbol{q}\\ &= \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\,\boldsymbol{\Gamma}(\dot{\boldsymbol{q}})\,\boldsymbol{\Gamma}(\boldsymbol{p})^{\mathsf{T}}\,\boldsymbol{q}\\ &= \boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{\Phi}(\boldsymbol{p})^{\mathsf{T}}\,\boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\,\dot{\boldsymbol{q}}\\ &= \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\,\boldsymbol{\Gamma}(\dot{\boldsymbol{q}})\,\boldsymbol{\Phi}(\boldsymbol{q})\,\mathbf{I}^*\,\boldsymbol{p}\end{aligned} \tag{H.92}$$

As shown in Appendix H.4 the product rule for taking derivatives applies to quaternion multiplication as well. The result can be generalized as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{q}\circ\boldsymbol{p}) = \dot{\boldsymbol{q}}\circ\boldsymbol{p} + \boldsymbol{q}\circ\dot{\boldsymbol{p}} \tag{H.93}$$

Taking partial derivatives of equations involving quaternion multiplication is fairly simple, since the two matrix operators can be used for reorganization.

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{q}}\boldsymbol{q}\circ\boldsymbol{p} &= \frac{\partial}{\partial\boldsymbol{q}}\boldsymbol{\Gamma}(\boldsymbol{p})\,\boldsymbol{q} = \boldsymbol{\Gamma}(\boldsymbol{p})\\ \frac{\partial}{\partial\boldsymbol{p}}\boldsymbol{q}\circ\boldsymbol{p} &= \frac{\partial}{\partial\boldsymbol{p}}\boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{p} = \boldsymbol{\Phi}(\boldsymbol{q})\end{aligned} \tag{H.94}$$

For the derivation of the quaternion model of the ballbot it is useful to know the partial derivative of the vector transformation, e.g., (H.39), with respect to the quaternion itself. We apply the product rule as shown above while noting that the quaternion conjugate can be replaced with the matrix-vector product $\boldsymbol{q}^* = \mathbf{I}^*\,{}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}$.

$$\begin{aligned}\frac{\partial}{\partial\,{}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\circ{}^{\mathrm{B}}\tilde{\boldsymbol{p}}\circ{}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}^*\right) &= \frac{\partial}{\partial\boldsymbol{q}}\left(\boldsymbol{q}\circ{}^{\mathrm{B}}\tilde{\boldsymbol{p}}\circ{}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}^*\right) + \frac{\partial}{\partial\boldsymbol{q}}\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\circ{}^{\mathrm{B}}\tilde{\boldsymbol{p}}\circ\boldsymbol{q}^*\right)\\ &= \frac{\partial}{\partial\boldsymbol{q}}\left(\boldsymbol{\Gamma}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)^{\mathsf{T}}\,\boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\Big)\,\boldsymbol{q}\right) + \frac{\partial}{\partial\boldsymbol{q}}\left(\boldsymbol{\Phi}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)\,\boldsymbol{\Phi}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\Big)\,\mathbf{I}^*\,\boldsymbol{q}\right)\\ &= \boldsymbol{\Gamma}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)^{\mathsf{T}}\,\boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\Big) + \boldsymbol{\Phi}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)\,\boldsymbol{\Phi}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\Big)\,\mathbf{I}^*\\ &= \boldsymbol{\Lambda}_{\mathrm{p}}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big) + \boldsymbol{\Pi}_{\mathrm{p}}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)\end{aligned} \tag{H.95}$$

Note that the vector part of this partial derivative can be simplified through the relationship in (H.57).

$$\begin{aligned}\frac{\partial}{\partial\,{}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}}\vee\left(\left({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\circ{}^{\mathrm{B}}\tilde{\boldsymbol{p}}\circ{}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}^*\right)\right) &= \vee\boldsymbol{\Lambda}_{\mathrm{p}}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big) + \vee\boldsymbol{\Pi}_{\mathrm{p}}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)\\ &= 2\vee\boldsymbol{\Lambda}_{\mathrm{p}}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)\\ &= 2\vee\boldsymbol{\Pi}_{\mathrm{p}}\Big({}_{\mathrm{B}}^{\mathrm{K}}\boldsymbol{q}\Big)\end{aligned} \tag{H.96}$$

H.8 Summary

Description	Definition
Quaternion definition	$\boldsymbol{q} = \begin{bmatrix} q_0 & \vec{\boldsymbol{q}}^\mathsf{T} \end{bmatrix} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^\mathsf{T}$
Quaternion axis-angle construction	${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ {}^{\mathrm{K}}\boldsymbol{r} \sin\left(\frac{\theta}{2}\right) \end{bmatrix}$
Quaternion representation of a vector	$\tilde{\boldsymbol{p}} = \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix}$
Left multiplication operator	$\boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{q} \circ$
Right multiplication operator	$\boldsymbol{\Gamma}(\boldsymbol{p}) = \circ\, \boldsymbol{p}$
Inner multiplication simplifications	$\boldsymbol{\Phi}\big(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{p}\big) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p})$ $\boldsymbol{\Phi}\Big(\boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T}\, \boldsymbol{p}\Big) = \boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T}\, \boldsymbol{\Phi}(\boldsymbol{p})$
Conjugated quaternion product	$(\boldsymbol{q} \circ \boldsymbol{p})^* = \boldsymbol{p}^* \circ \boldsymbol{q}^*$
Cross product as quaternion product	$\boldsymbol{a} \times \boldsymbol{b} = \tilde{\boldsymbol{a}} \circ \tilde{\boldsymbol{b}}$
Rotation of body vector to inertial frame	${}^{\mathrm{K}}\tilde{\boldsymbol{p}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* = \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)\, \boldsymbol{\Gamma}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^\mathsf{T}\, {}^{\mathrm{B}}\tilde{\boldsymbol{p}}$
Rotation of inertial vector to body frame	${}^{\mathrm{B}}\tilde{\boldsymbol{p}} = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^\mathsf{T}\, \boldsymbol{\Gamma}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)\, {}^{\mathrm{K}}\tilde{\boldsymbol{p}}$
Angular velocity in body frame	${}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = 2\, \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^\mathsf{T}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$
Angular velocity in inertial frame	${}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* = 2\, \boldsymbol{\Gamma}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^\mathsf{T}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$
Quaternion derivative from angular velocity in body frame	${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = \frac{1}{2}\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = \frac{1}{2}\boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}$
Quaternion derivative from angular velocity in inertial frame	${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = \frac{1}{2}\, {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = \frac{1}{2}\boldsymbol{\Gamma}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)\, {}^{\mathrm{K}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}$
Matrix operator rules	$\boldsymbol{\Phi}(\boldsymbol{q}^*) = \boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T}$ $\boldsymbol{\Phi}(\boldsymbol{q} \circ \boldsymbol{p}) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p})$ $\boldsymbol{\Phi}\big(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{p}\big) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p})$ $\boldsymbol{\Phi}\Big(\boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T}\, \boldsymbol{p}\Big) = \boldsymbol{\Phi}(\boldsymbol{q})^\mathsf{T}\, \boldsymbol{\Phi}(\boldsymbol{p})$ $\boldsymbol{\Phi}\big(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{p}\big)^\mathsf{T} = \big(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{p})\big)^\mathsf{T} = \boldsymbol{\Phi}(\boldsymbol{p})^\mathsf{T}\, \boldsymbol{\Phi}(\boldsymbol{q})$ $\boldsymbol{\Gamma}\Big(\boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}(\boldsymbol{b})\, \boldsymbol{\Phi}(\boldsymbol{p})^\mathsf{T}\Big) = \boldsymbol{\Gamma}(\boldsymbol{p})^\mathsf{T}\, \boldsymbol{\Gamma}(\boldsymbol{b})\, \boldsymbol{\Gamma}(\boldsymbol{q})$
Euler ZYX angles from quaternion	$\psi = \operatorname{atan2}\Big(2(q_0 q_3 + q_1 q_2),\, q_0^2 + q_1^2 - q_2^2 - q_3^2\Big)$ $\theta = \sin^{-1}\big(2(q_0 q_2 - q_1 q_3)\big)$ $\phi = \operatorname{atan2}\Big(2(q_0 q_1 + q_2 q_3),\, q_0^2 - q_1^2 - q_2^2 + q_3^2\Big)$

Table H.1: Summary of useful quaternion equations (note that the orientation of the inertial frame and ball frame are equivalent)

Description	Definition
Rotated vector derivative operators	$\boldsymbol{\Pi}_{\mathrm{p}}(\boldsymbol{q}) = \boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{\Phi}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right)\mathbf{I}^{*}$ $\boldsymbol{\Lambda}_{\mathrm{p}}(\boldsymbol{q}) = \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\,\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right)$
Rotated vector derivative operator relationships	$\vee\boldsymbol{\Pi}_{\mathrm{p}}(\boldsymbol{q}) = \vee\boldsymbol{\Lambda}_{\mathrm{p}}(\boldsymbol{q})$ $\boldsymbol{\Pi}_{\mathrm{p}}(\dot{q})\,\boldsymbol{q} = \boldsymbol{\Pi}_{\mathrm{p}}(\boldsymbol{q})\,\dot{\boldsymbol{q}}$
Velocity of constant rotated body vector	${}^{\mathrm{K}}\dot{\boldsymbol{p}} = 2 \vee \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2 \vee \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$ $= 2 \vee \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right){}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} = 2 \vee \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$
Quaternion exponential integration	${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] =$ ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \circ \begin{bmatrix} \cos\left(\frac{\Delta t}{2}\left\|{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right\|\right) \\ \frac{{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]}{\left\|{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right\|}\sin\left(\frac{\Delta t}{2}\left\|{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right\|\right) \end{bmatrix}$
Integration with Matrix exponential	${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = \mathrm{e}^{\frac{1}{2}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\right)\Delta t}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1]$
Integration with Forward Euler	${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] + \Delta t\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}[k-1]$
Integration with Delta quaternion	${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] \circ \begin{bmatrix} 1 \\ \frac{1}{2}\Delta t\,{}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1] \end{bmatrix}$
Time derivative of product	$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{q} \circ \boldsymbol{p}) = \dot{\boldsymbol{q}} \circ \boldsymbol{p} + \boldsymbol{q} \circ \dot{\boldsymbol{p}}$
Partial derivatives of product	$\frac{\partial}{\partial \boldsymbol{q}}\boldsymbol{q} \circ \boldsymbol{p} = \frac{\partial}{\partial \boldsymbol{q}}\boldsymbol{\Gamma}(\boldsymbol{p})\,\boldsymbol{q} = \boldsymbol{\Gamma}(\boldsymbol{p})$ $\frac{\partial}{\partial \boldsymbol{p}}\boldsymbol{q} \circ \boldsymbol{p} = \frac{\partial}{\partial \boldsymbol{p}}\boldsymbol{\Phi}(\boldsymbol{q})\,\boldsymbol{p} = \boldsymbol{\Phi}(\boldsymbol{q})$
Partial derivative of vector rotation	$\frac{\partial}{\partial {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*}\right) =$ $\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right) + \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\boldsymbol{\Phi}\left({}^{\mathrm{B}}\tilde{\boldsymbol{p}}\right)\mathbf{I}^{*} = \boldsymbol{\Lambda}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) + \boldsymbol{\Pi}_{\mathrm{p}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)$

Table H.2: Summary of useful quaternion equations contd.

A C++ quaternion library has been developed to implement the above operators and to allow easy computation with quaternions. The library code can be found here: Kugle-Embedded/KugleFirmware/Libraries/Misc/Quaternion/Quaternion.cpp

I Quaternion Model Derivation

This appendix includes some of the math steps involved in the derivation of the quaternion model. It thus serves as an extension to some of the sections in the quaternion model chapter to elaborate on how some of the equations are derived and simplified.
The Euler-Lagrange equation with the kinetic and potential energies is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} - \left(\frac{\partial T_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial V_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} = \mathbf{0}_{6\times 1} \quad \text{(I.1)}$$

In the sections to follow these partial derivatives are derived. Note that the transposed partial derivatives are derived in the following format:

$$\left(\frac{\partial T}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} = \begin{bmatrix} \frac{\partial T}{\partial {}^{\mathrm{I}}x} \\ \frac{\partial T}{\partial {}^{\mathrm{I}}y} \\ \left(\frac{\partial T}{\partial {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}\right)^{\mathsf{T}} \end{bmatrix} \quad \text{(I.2)}$$

I.1 Ball kinetic energy wrt. velocity

$$T_k = \frac{1}{2}\left(M_k + \frac{J_k}{r_k^2}\right) {}^{\mathrm{I}}\dot{x}^2 + \frac{1}{2}\left(M_k + \frac{J_k}{r_k^2}\right) {}^{\mathrm{I}}\dot{y}^2 \quad \text{(I.3)}$$

Partial derivative of ball kinetic energy with respect to $\dot{\boldsymbol{\chi}}$.

$$\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} = \left(M_k + \frac{J_k}{r_k^2}\right) \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ \mathbf{0}_{4\times 1} \end{bmatrix} \quad \text{(I.4)}$$

I.2 Time derivative of ball kinetic energy

The corresponding time derivative of the ball kinetic energy with respect to $\dot{\boldsymbol{\chi}}$.

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} = \begin{bmatrix} M_k + \frac{J_k}{r_k^2} \\ 0 \\ \mathbf{0}_{4\times 1} \end{bmatrix} {}^{\mathrm{I}}\ddot{x} + \begin{bmatrix} 0 \\ M_k + \frac{J_k}{r_k^2} \\ \mathbf{0}_{4\times 1} \end{bmatrix} {}^{\mathrm{I}}\ddot{y} \quad \text{(I.5)}$$

I.3 Body kinetic energy wrt. velocity

$$T_b = \frac{1}{2} {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}^{\mathsf{T}}_{\mathrm{B}\leftarrow\mathrm{I}} {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}} + \frac{1}{2} M_b {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}^{\mathsf{T}}_{\mathrm{COM}\leftarrow\mathrm{I}} {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}} \quad \text{(I.6)}$$

The partial derivative of the body kinetic energy with respect to $\dot{\boldsymbol{\chi}}$ is derived using the properties from Appendix G.3 and Appendix G.4.

$$\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}} = {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}^{\mathsf{T}}_{\mathrm{B}\leftarrow\mathrm{I}} {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} + M_b {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}^{\mathsf{T}}_{\mathrm{COM}\leftarrow\mathrm{I}} \frac{\partial\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} \quad \text{(I.7)}$$

From (H.51) we have the angular velocity.

$$ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}} = {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = 2\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = 2\, \boldsymbol{\Phi}\Big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\Big)^{\mathsf{T}}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{I.8} $$

Leading to a partial derivative

$$ \frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\, \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}} \end{bmatrix} \tag{I.9} $$

The velocity of the center of mass, defined in the inertial frame, is a combination of the translational velocity of the ball and the velocity of the center of mass defined in the ball frame.

$$ {}^{\mathrm{I}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{I}} = {}^{\mathrm{I}}\boldsymbol{v}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{K}}\dot{\boldsymbol{p}}_{\mathrm{COM}\leftarrow\mathrm{K}} \tag{I.10} $$

The partial derivative is thus a sum of partial derivatives

$$ \frac{\partial\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} = \frac{\partial\, {}^{\mathrm{I}}\tilde{\boldsymbol{v}}_{\mathrm{K}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} + \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \dot{\boldsymbol{\chi}}} \tag{I.11} $$

where the partial derivative of the translational velocity is a constant matrix.

$$ \frac{\partial\, {}^{\mathrm{I}}\tilde{\boldsymbol{v}}_{\mathrm{K}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \\ \mathbf{I}_{2} & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \end{bmatrix} \tag{I.12} $$

The velocity of the rotated center of mass was defined in (5.29).

$$ \begin{aligned} {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}} &= {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} + {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{*} \\ &= 2\boldsymbol{\Pi}_{\mathrm{COM}}\Big({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\Big)\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \quad = 2\boldsymbol{\Pi}_{\mathrm{COM}}\Big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\Big)\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\ &= 2\boldsymbol{\Lambda}_{\mathrm{COM}}\Big({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\Big)\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \quad = 2\boldsymbol{\Lambda}_{\mathrm{COM}}\Big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\Big)\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{aligned} \tag{I.13} $$

where the rotated vector derivative operators, $\boldsymbol{\Pi}_{\mathrm{COM}}(\boldsymbol{q})$ and $\boldsymbol{\Lambda}_{\mathrm{COM}}(\boldsymbol{q})$, are defined in Appendix H.4.

$$ \boldsymbol{\Pi}_{\mathrm{COM}}(\boldsymbol{q}) = \boldsymbol{\Phi}(\boldsymbol{q})\, \boldsymbol{\Phi}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}}\Big)\, \mathbf{I}^{*} \tag{I.14} $$

$$ \boldsymbol{\Lambda}_{\mathrm{COM}}(\boldsymbol{q}) = \boldsymbol{\Gamma}(\boldsymbol{q})^{\mathsf{T}}\, \boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}}\Big) \tag{I.15} $$

The partial derivative of the velocity of the center of mass with respect to the state velocity is thus

$$ \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\boldsymbol{\Pi}_{\mathrm{COM}}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big) \end{bmatrix} \tag{I.16} $$

Note that $\boldsymbol{\Pi}_{\mathrm{COM}}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)$ can be exchanged with $\boldsymbol{\Lambda}_{\mathrm{COM}}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)$ and be equivalent.
After inserting the velocity definition from (I.10) and the partial derivative from (I.11) into the definition of the body kinetic energy, (I.7), the equation is expanded by applying the properties of the dot product, see Appendix F.1.

$$ \begin{aligned} \frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}} = {}& {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}^{\mathsf{T}}_{\mathrm{B}\leftarrow\mathrm{I}}\, {}^{\mathrm{B}}\tilde{\boldsymbol{J}}_b \frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} \\ & + M_b \left({}^{\mathrm{I}}\tilde{\boldsymbol{v}}^{\mathsf{T}}_{\mathrm{K}\leftarrow\mathrm{I}} \frac{\partial\, {}^{\mathrm{I}}\tilde{\boldsymbol{v}}_{\mathrm{K}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} + {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}^{\mathsf{T}}_{\mathrm{COM}\leftarrow\mathrm{K}} \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \dot{\boldsymbol{\chi}}} + {}^{\mathrm{I}}\tilde{\boldsymbol{v}}^{\mathsf{T}}_{\mathrm{K}\leftarrow\mathrm{I}} \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \dot{\boldsymbol{\chi}}} + {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}^{\mathsf{T}}_{\mathrm{COM}\leftarrow\mathrm{K}} \frac{\partial\, {}^{\mathrm{I}}\tilde{\boldsymbol{v}}_{\mathrm{K}\leftarrow\mathrm{I}}}{\partial \dot{\boldsymbol{\chi}}} \right) \end{aligned} \tag{I.17} $$

The complete partial derivative of the body kinetic energy is assembled by inserting the individual velocity components and their partial derivatives.

$$\begin{aligned}\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}} &= 2\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \\ &\quad + M_b \begin{bmatrix} 0 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \\ \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \end{bmatrix} \\ &\quad + M_b \left(2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}} \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \end{bmatrix} \\ &\quad + M_b \begin{bmatrix} 0 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \end{bmatrix} \\ &\quad + M_b \left(2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}} \begin{bmatrix} \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \\ \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \end{bmatrix}\end{aligned} \tag{I.18}$$

This can be simplified further by combining some of the matrices and applying the properties of the rotated vector derivative operators, see Appendix H.4.

$$\begin{aligned}\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}} &= 4 \begin{bmatrix} \mathbf{0}_{1\times 2} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \\ &\quad + M_b \begin{bmatrix} {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & \mathbf{0}_{1\times 4} \end{bmatrix} \\ &\quad + 4M_b\ {}^{\mathrm{B}}\boldsymbol{p}^{\mathsf{T}}_{\mathrm{COM}}\ {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} \begin{bmatrix} \mathbf{0}_{1\times 2} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \end{bmatrix} \\ &\quad + 2M_b \begin{bmatrix} 0 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{4\times 2} & \boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \end{bmatrix} \\ &\quad + 2M_b \left(\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}} \begin{bmatrix} \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \\ \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{1\times 2} & \mathbf{0}_{1\times 4} \end{bmatrix}\end{aligned} \tag{I.19}$$

Finally, the partial derivative is transposed to prepare it for usage in the Euler-Lagrange equation.

$$\begin{aligned}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} &= \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 4\,\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{bmatrix} + M_b \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ \mathbf{0}_{4\times 1} \end{bmatrix} \\ &\quad + 4M_b\ {}^{\mathrm{B}}\boldsymbol{p}^{\mathsf{T}}_{\mathrm{COM}}\ {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} \begin{bmatrix} \mathbf{0}_{2\times 1} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{bmatrix} \\ &\quad + 2M_b \begin{bmatrix} \mathbf{0}_{2\times 4} \\ \boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} 0 \\ {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ 0 \end{bmatrix} \\ &\quad + 2M_b \begin{bmatrix} \mathbf{0}_{1\times 2} & \mathbf{I}_2 & \mathbf{0}_{1\times 2} \\ \mathbf{0}_{1\times 4} & \mathbf{0}_{2\times 4} & \mathbf{0}_{1\times 4} \end{bmatrix} \boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\end{aligned} \tag{I.20}$$

I.4 Time derivative of body kinetic energy

The corresponding time derivative of the body kinetic energy with respect to $\dot{\boldsymbol{\chi}}$.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} = &\begin{bmatrix} \mathbf{0}_{2\times 4} \\ 4\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} + \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}} \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} + \begin{bmatrix} \mathbf{0}_{2\times 4} \\ 4\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} \\
&+ \begin{bmatrix} M_b \\ 0 \\ \mathbf{0}_{4\times 1} \end{bmatrix} {}^{\mathrm{I}}\ddot{x} + \begin{bmatrix} 0 \\ M_b \\ \mathbf{0}_{4\times 1} \end{bmatrix} {}^{\mathrm{I}}\ddot{y} \\
&+ \begin{bmatrix} \mathbf{0}_{2\times 4} \\ 4M_b\, {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}^{\mathsf{T}}\, {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} \\
&+ \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 2M_b\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}}\hat{\mathbf{i}} \end{bmatrix} {}^{\mathrm{I}}\dot{x} + \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 2M_b\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}}\hat{\mathbf{j}} \end{bmatrix} {}^{\mathrm{I}}\dot{y} \\
&+ \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 2M_b\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\hat{\mathbf{i}} \end{bmatrix} {}^{\mathrm{I}}\ddot{x} + \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 2M_b\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\hat{\mathbf{j}} \end{bmatrix} {}^{\mathrm{I}}\ddot{y} \\
&+ \begin{bmatrix} 2M_b\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \\ 2M_b\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \\ \mathbf{0}_{4\times 4} \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\
&+ \begin{bmatrix} 2M_b\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \\ 2M_b\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \\ \mathbf{0}_{4\times 4} \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}
\end{aligned} \tag{I.21}$$

I.5 Body kinetic energy wrt. state

$$T_b = \frac{1}{2}\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}^{\mathsf{T}}\, {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}} + \frac{1}{2} M_b\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}}^{\mathsf{T}}\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}} \tag{I.22}$$

We use a similar technique as from the body kinetic energy wrt. velocity. First we define the partial derivative of body kinetic energy with respect to $\boldsymbol{\chi}$, again using the properties from Appendix G.4 and Appendix G.3.

$$\frac{\partial T_b}{\partial \boldsymbol{\chi}} = {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}^{\mathsf{T}}\, {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}}{\partial \boldsymbol{\chi}} + M_b\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}}^{\mathsf{T}} \frac{\partial\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}}}{\partial \boldsymbol{\chi}} \tag{I.23}$$

Based on the relationship for the angular velocity, the partial derivative becomes

$$\frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}}{\partial \boldsymbol{\chi}} = \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\,\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)\mathbf{I}^{*} \end{bmatrix} \tag{I.24}$$

Similarly to before, the partial derivative of the velocity of the center of mass defined in inertial frame is given by a sum of partial derivatives.

$$\frac{\partial\, {}^{\mathrm{I}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{I}}}{\partial \boldsymbol{\chi}} = \underbrace{\frac{\partial\, {}^{\mathrm{I}}\tilde{\boldsymbol{v}}_{\mathrm{K}\leftarrow\mathrm{I}}}{\partial \boldsymbol{\chi}}}_{0} + \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \boldsymbol{\chi}} \tag{I.25}$$

As shown in (I.13) the velocity of the center of mass, defined in the ball frame, can be computed with

$${}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}} = 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{I.26}$$

The partial derivative of the translational velocity with respect to the position and shape vector, $\boldsymbol{\chi}$, is zero. The partial derivative of the velocity of the center of mass defined in the ball frame, (I.13), is derived similarly using the rotated vector derivative operators, $\boldsymbol{\Pi}_{\mathrm{COM}}(\boldsymbol{q})$ and $\boldsymbol{\Lambda}_{\mathrm{COM}}(\boldsymbol{q})$, as defined in Appendix H.4.

$$\frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \boldsymbol{\chi}} = \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \end{bmatrix} \tag{I.27}$$

The partial derivative definitions are substituted back into (I.23) and the properties of the dot product is applied.

$$\begin{aligned} \frac{\partial T_b}{\partial \boldsymbol{\chi}} = {} & {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}^{\mathsf{T}}_{\mathrm{B}\leftarrow\mathrm{I}}\, {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{I}}}{\partial \boldsymbol{\chi}} \\ & + M_b \left({}^{\mathrm{I}}\tilde{\boldsymbol{v}}^{\mathsf{T}}_{\mathrm{K}\leftarrow\mathrm{I}} \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \boldsymbol{\chi}} + {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}^{\mathsf{T}}_{\mathrm{COM}\leftarrow\mathrm{K}} \frac{\partial\, {}^{\mathrm{K}}\dot{\tilde{\boldsymbol{p}}}_{\mathrm{COM}\leftarrow\mathrm{K}}}{\partial \boldsymbol{\chi}} \right) \end{aligned} \tag{I.28}$$

The complete partial derivative of the body kinetic energy is assembled by inserting the individual velocity components and their partial derivatives.

$$\begin{aligned} \frac{\partial T_b}{\partial \boldsymbol{\chi}} = {} & 2\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\,\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \mathbf{I}^* \end{bmatrix} \\ & + M_b \begin{bmatrix} 0 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \end{bmatrix} \\ & + M_b \left(2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} \begin{bmatrix} \mathbf{0}_{4\times 2} & 2\boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \end{bmatrix} \end{aligned} \tag{I.29}$$

This is similarly simplified by combining the matrices and applying the properties of the rotated vector derivative operators, see Appendix H.4.

$$\begin{aligned} \frac{\partial T_b}{\partial \boldsymbol{\chi}} = {} & 4 \begin{bmatrix} \mathbf{0}_{1\times 2} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) {}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \mathbf{I}^* \end{bmatrix} \\ & + 2M_b \begin{bmatrix} 0 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{4\times 2} & \boldsymbol{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \end{bmatrix} \\ & + 4M_b\, {}^{\mathrm{B}}\boldsymbol{p}^{\mathsf{T}}_{\mathrm{COM}}\, {}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \begin{bmatrix} \mathbf{0}_{1\times 2} & {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \end{bmatrix} \end{aligned} \tag{I.30}$$

Finally, the partial derivative is transposed and organized for easier partioning into the different matrices of the Euler-Lagrange equation.

$$\begin{aligned}\left(\frac{\partial T_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} &= \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 4\mathbf{\Pi}^{*}\,\mathbf{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}}\,{}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\mathbf{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\ &+ \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 2M_b\mathbf{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)\hat{\mathbf{i}} \end{bmatrix} {}^{\mathrm{I}}\dot{x} + \begin{bmatrix} \mathbf{0}_{2\times 1} \\ 2M_b\mathbf{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)\hat{\mathbf{j}} \end{bmatrix} {}^{\mathrm{I}}\dot{y} \\ &+ \begin{bmatrix} \mathbf{0}_{2\times 4} \\ 4M_b\,{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}^{\mathsf{T}}\,{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\,\mathbf{I}_4 \end{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\end{aligned} \tag{I.31}$$

I.6 Body potential energy wrt. state

$$V_b = M_b g\,\hat{\mathbf{k}}^{\mathsf{T}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\circ{}^{\mathrm{B}}\tilde{\boldsymbol{p}}_{\mathrm{COM}}\circ{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*}\right) \tag{I.32}$$

Partial derivative of body potential energy with respect to $\boldsymbol{\chi}$. Using the properties from (H.95) for taking the partial derivative of the quaternion product and the properties for the rotated vector derivative operator, Appendix H.4.

$$\begin{aligned}\frac{\partial V_b}{\partial \boldsymbol{\chi}} &= M_b\,g\begin{bmatrix}\mathbf{0}_{1\times 2} & \hat{\mathbf{k}}^{\mathsf{T}}\left(\mathbf{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)+\mathbf{\Lambda}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\right)\end{bmatrix} \\ &= M_b\,g\begin{bmatrix}\mathbf{0}_{1\times 2} & 2\hat{\mathbf{k}}^{\mathsf{T}}\mathbf{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\end{bmatrix}\end{aligned} \tag{I.33}$$

$$\left(\frac{\partial V_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} = \begin{bmatrix}\mathbf{0}_{2\times 1} \\ 2\mathbf{\Pi}_{\mathrm{COM}}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\hat{\mathbf{k}}\end{bmatrix} \tag{I.34}$$

I.7 Wheel kinetic energy wrt. velocity

$$T_w = \frac{1}{2}\dot{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{J}_w\dot{\boldsymbol{\theta}} \tag{I.35}$$

The partial derivative of the wheel kinetic energy with respect to $\dot{\boldsymbol{\chi}}$ is derived using the properties from Appendix G.4 and Appendix G.3.

$$\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}} = \dot{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{J}_w\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\chi}}} \tag{I.36}$$

From the inverse kinematics, (4.35), the definition of the wheel angular velocity is

$$\begin{aligned}\dot{\boldsymbol{\theta}} &= \widetilde{\boldsymbol{W}}\mathbf{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\left(\mathbf{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \\ &= \widetilde{\boldsymbol{W}}\mathbf{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\mathbf{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix} - 2\widetilde{\boldsymbol{W}}\mathbf{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\end{aligned} \tag{I.37}$$

Leading to a partial derivative with respect to the derivative generalized coordinates, $\dot{\boldsymbol{\chi}}$.

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} \frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{I}}\dot{x}} & \frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{I}}\dot{y}} & \frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}} \end{bmatrix} \tag{I.38}$$

The individual partial derivatives are derived from (I.37) using the properties described in Appendix H.7.

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{I}}\dot{x}} = \frac{1}{r_k} \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{j}} \tag{I.39}$$

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{I}}\dot{y}} = -\frac{1}{r_k} \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{i}} \tag{I.40}$$

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}} = -2 \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \tag{I.41}$$

Assembling these partial derivatives back into (I.38) yields

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \dot{\boldsymbol{\chi}}} = \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \begin{bmatrix} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{j}} & -\frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{i}} & -2 \end{bmatrix} \tag{I.42}$$

This can now be combined with the partial derivative definition of the wheel kinetic energy, (I.36).

$$\begin{aligned} \frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}} &= \left(\widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) \right)^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \begin{bmatrix} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{j}} & -\frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{i}} & -2 \end{bmatrix} \\ &= \left(\frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix}^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} - 2\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}} \right) \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \begin{bmatrix} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{j}} & -\frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \hat{\mathbf{i}} & -2 \end{bmatrix} \end{aligned} \tag{I.43}$$

Finally, the partial derivative is transposed to prepare it for usage in the Euler-Lagrange equation.

$$\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}} \right)^{\mathsf{T}} = \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \left(\frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) \tag{I.44}$$

I.8 Time derivative of wheel kinetic energy wrt. velocity

The corresponding time derivative of the wheel kinetic energy with respect to $\dot{\boldsymbol{\chi}}$.

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}} &= \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) + \\
&= \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) + \\
&= \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)^{\mathsf{T}} \left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \right) + \\
&= \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{I}}\dot{x} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \frac{1}{r_k} \left(\hat{\mathbf{j}}\,{}^{\mathrm{I}}\ddot{x} - \hat{\mathbf{i}}\,{}^{\mathrm{I}}\ddot{y} \right) \\
&= -2 \begin{bmatrix} \hat{\mathbf{j}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\hat{\mathbf{i}}^{\mathsf{T}} \frac{1}{r_k} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \boldsymbol{J}_w \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}
\end{aligned}
\tag{I.45}
$$

I.9 Wheel kinetic energy wrt. state

$$
T_w = \frac{1}{2} \dot{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{J}_w \dot{\boldsymbol{\theta}} \tag{I.46}
$$

A similar technique to the partial derivative of the wheel kinetic energy wrt. velocity is used to derive the partial derivative with respect to the generalized coordinates, $\boldsymbol{\chi}$.

$$
\frac{\partial T_w}{\partial \boldsymbol{\chi}} = \dot{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{J}_w \frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{\chi}} \tag{I.47}
$$

The partial derivative of the angular velocity with respect to the generalized coordinates is similarly expanded.

$$
\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{\chi}} = \begin{bmatrix} \frac{\partial \dot{\boldsymbol{\theta}}}{\partial\, {}^{\mathrm{I}}x} & \frac{\partial \dot{\boldsymbol{\theta}}}{\partial\, {}^{\mathrm{I}}y} & \frac{\partial \dot{\boldsymbol{\theta}}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}} \end{bmatrix} \tag{I.48}
$$

Note that the partial derivative with respect to the two position states, $\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{I}}_{\mathrm{I}}x}$ and $\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{I}}_{\mathrm{I}}y}$, is zero since there is no spring energy. The partial derivative with respect to the quaternion is derivative from the inverse kinematics definition, (4.35).

$$\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}} = \frac{1}{r_k}\left(\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right) + \widetilde{\boldsymbol{W}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\boldsymbol{\Gamma}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)\mathbf{I}^{*}\right) \tag{I.49}$$

Similarly to the relation with the rotated vector derivative operator, Appendix H.4, a similar relationship exists here.

$$\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right) = \widetilde{\boldsymbol{W}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\boldsymbol{\Gamma}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)\mathbf{I}^{*} \tag{I.50}$$

The partial derivative thus simplifies to

$$\begin{aligned}\frac{\partial \dot{\boldsymbol{\theta}}}{\partial {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}} &= 2\frac{1}{r_k}\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)\\ &= 2\frac{1}{r_k}\widetilde{\boldsymbol{W}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\boldsymbol{\Gamma}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)\mathbf{I}^{*}\end{aligned} \tag{I.51}$$

This partial derivative is inserted back into (I.47) together with the inverse kinematics definition, (4.35).

$$\begin{aligned}\frac{\partial T_w}{\partial \boldsymbol{\chi}} &= \frac{2}{r_k}\left(\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right)\right)^{\mathsf{T}}\boldsymbol{J}_w\begin{bmatrix}\mathbf{0}_{3\times 2} & \widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)\end{bmatrix}\\ &= \frac{2}{r_k}\left(\frac{1}{r_k}\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} - 2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}^{\mathsf{T}}\right)\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\begin{bmatrix}\mathbf{0}_{4\times 2} & \boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)\end{bmatrix}\end{aligned} \tag{I.52}$$

Finally, the partial derivative is transposed to prepare it for usage in the Euler-Lagrange equation.

$$\left(\frac{\partial T_w}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} = \frac{2}{r_k}\left[\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\mathrm{I}}\dot{y}\\ {}^{\mathrm{I}}\dot{x}\\ 0\end{bmatrix}\right)^{\mathsf{T}}\right]\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\left(\frac{1}{r_k}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\left(\hat{\mathbf{j}}\,{}^{\mathrm{I}}\dot{x}-\hat{\mathbf{i}}\,{}^{\mathrm{I}}\dot{y}\right)-2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}\right) \tag{I.53}$$

I.10 Euler-Lagrange equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}+\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}+\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}-\left(\frac{\partial T_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}}-\left(\frac{\partial T_w}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}}+\left(\frac{\partial V_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}}=\mathbf{0}_{6\times 1} \tag{I.54}$$

The six differential equations from the Euler-Lagrange equation above is partitioned in a matrix differential equation on the following form:

$$\boldsymbol{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}}+\boldsymbol{C}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})\dot{\boldsymbol{\chi}}+\boldsymbol{G}(\boldsymbol{\chi})=\mathbf{0}_{6\times 1} \tag{I.55}$$

where the matrices are extracted from the previous partial derivatives.
The elements for the mass matrix all comes from the time derivative parts, and thus only from equations related to the kinetic energy. Furthermore, the mass matrix is only dependent on the shape variable, in this case the quaternion, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, since there is no spring energies included that couples to the position state. Therefore, the mass matrix will be denoted $\boldsymbol{M}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})$.

$$\boldsymbol{M}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})=\begin{bmatrix}\left(\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}+\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}\right)^{\mathsf{T}}\\ \left(\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}+\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}\right)^{\mathsf{T}}\\ \left(\frac{\partial}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}+\frac{\partial}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}\frac{\partial}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial \dot{\boldsymbol{\chi}}}\right)^{\mathsf{T}}\right)^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}} \tag{I.56}$$

$$=\begin{bmatrix}\boldsymbol{M}_{xx}(\boldsymbol{\chi}) & \boldsymbol{M}_{xy}(\boldsymbol{\chi}) & \boldsymbol{M}_{x\boldsymbol{q}}(\boldsymbol{\chi})\\ \boldsymbol{M}_{yx}(\boldsymbol{\chi}) & \boldsymbol{M}_{yy}(\boldsymbol{\chi}) & \boldsymbol{M}_{y\boldsymbol{q}}(\boldsymbol{\chi})\\ \boldsymbol{M}_{\boldsymbol{q}x}(\boldsymbol{\chi}) & \boldsymbol{M}_{\boldsymbol{q}y}(\boldsymbol{\chi}) & \boldsymbol{M}_{\boldsymbol{q}\boldsymbol{q}}(\boldsymbol{\chi})\end{bmatrix} \tag{I.57}$$

Note that the mass matrix is symmetric, thus $\boldsymbol{M}_{xy}(\boldsymbol{\chi})=\boldsymbol{M}_{yx}(\boldsymbol{\chi})^{\mathsf{T}}$ etc.

In the following section some of the partitioned matrices are extracted and listed for illustrational purposes of the complexity of the model.

$$\begin{aligned}\boldsymbol{M}_{xx}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) &= \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial\,{}^{\mathrm{I}}\dot{x}}\right)^{\mathsf{T}}+\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial\,{}^{\mathrm{I}}\dot{x}}\right)^{\mathsf{T}}+\frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial\,{}^{\mathrm{I}}\dot{x}}\right)^{\mathsf{T}}\\ &= M_k+\frac{J_k}{r_k^2}+M_b+\hat{\mathbf{j}}^{\mathsf{T}}\frac{1}{r_k}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\hat{\mathbf{j}}\end{aligned} \tag{I.58}$$

$$\begin{aligned}\boldsymbol{M}_{yx}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) &= \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial\,{}^{\mathrm{I}}\dot{y}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial\,{}^{\mathrm{I}}\dot{y}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{x}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial\,{}^{\mathrm{I}}\dot{y}}\right)^{\mathsf{T}} \\ &= -\hat{\mathbf{i}}^{\mathsf{T}}\frac{1}{r_k}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\hat{\mathbf{j}}\end{aligned} \tag{I.59}$$

$$\begin{aligned}\boldsymbol{M}_{xy}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) &= \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial\,{}^{\mathrm{I}}\dot{x}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial\,{}^{\mathrm{I}}\dot{x}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial\,{}^{\mathrm{I}}\dot{x}}\right)^{\mathsf{T}} \\ &= -\hat{\mathbf{j}}^{\mathsf{T}}\frac{1}{r_k}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\hat{\mathbf{i}} \\ &= \boldsymbol{M}_{yx}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})^{\mathsf{T}}\end{aligned} \tag{I.60}$$

$$\begin{aligned}\boldsymbol{M}_{yy}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) &= \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial\,{}^{\mathrm{I}}\dot{y}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial\,{}^{\mathrm{I}}\dot{y}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{I}}\ddot{y}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial\,{}^{\mathrm{I}}\dot{y}}\right)^{\mathsf{T}} \\ &= M_k + \frac{J_k}{r_k^2} + M_b + \hat{\mathbf{i}}^{\mathsf{T}}\frac{1}{r_k}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\frac{1}{r_k}\hat{\mathbf{i}}\end{aligned} \tag{I.61}$$

Note how the wheel inertia makes the matrices very complex and coupled due to the coupling between rotational and translational movement, coupled through the motor shaft.
Note also that $\boldsymbol{M}_{xx}(\boldsymbol{\chi}) \neq \boldsymbol{M}_{yy}(\boldsymbol{\chi})$, which exactly shows the benefit of the non-linear coupled model, since the mass in the different directions is different, due to the motor/omniwheel coupling.
The mass matrix elements related to the quaternion are listed below.

$$\begin{aligned}\boldsymbol{M}_{\boldsymbol{q}\boldsymbol{q}}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) &= \frac{\partial}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_k}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_b}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}\right)^{\mathsf{T}} + \frac{\partial}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T_w}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}\right)^{\mathsf{T}} \\ &= 4\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right){}^{\mathrm{B}}\widetilde{\boldsymbol{J}}_b\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} + 4M_b\,{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}}^{\mathsf{T}}\,{}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} + 4\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)\widetilde{\boldsymbol{W}}^{\mathsf{T}}\boldsymbol{J}_w\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}}\end{aligned} \tag{I.62}$$

The gravity matrix captures the parts related to gravity and thus potential. Since only the orientation affects the potential of the system, the gravity matrix only depends on the quaternion, $\boldsymbol{G}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})$.

$$\begin{aligned}\boldsymbol{G}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}) &= \left(\frac{\partial V_b}{\partial \boldsymbol{\chi}}\right)^{\mathsf{T}} \\ &= \begin{bmatrix}\mathbf{0}_{2\times 1} \\ 2\boldsymbol{\Pi}_{\mathrm{COM}}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q})^{\mathsf{T}}\hat{\mathbf{k}}\end{bmatrix}\end{aligned} \tag{I.63}$$

All remaining parts are due to couplings between shape variables and velocity variables. These parts are gathered in the Coriolis matrix, $\boldsymbol{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$. This matrix will also not depend on the position state, and thus only depends on the orientation, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, its derivative, ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$ and the translational velocity, ${}^{\mathrm{I}}\dot{x}$ and ${}^{\mathrm{I}}\dot{y}$.

The resulting Coriolis matrix has the following structure. The variable dependency of each element is listed as the arguments.

$$\boldsymbol{C}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) = \begin{bmatrix}\boldsymbol{C}_{xx}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}) & \boldsymbol{C}_{xy}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}) & \boldsymbol{C}_{x\boldsymbol{q}}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}) \\ \boldsymbol{C}_{yx}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}) & \boldsymbol{C}_{yy}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}) & \boldsymbol{C}_{y\boldsymbol{q}}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}) \\ \boldsymbol{C}_{\boldsymbol{q}x}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) & \boldsymbol{C}_{\boldsymbol{q}y}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{y}) & \boldsymbol{C}_{\boldsymbol{q}\boldsymbol{q}}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}})\end{bmatrix} \tag{I.64}$$

As a final remark, please note that the above matrices are not the final matrices used for controller design, since the unit-norm quaternion constraint has to be enforced, resulting in the matrices $\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})$, $\widetilde{\boldsymbol{C}}(\boldsymbol{\chi},\dot{\boldsymbol{\chi}})$, $\widetilde{\boldsymbol{G}}(\boldsymbol{\chi})$, $\widetilde{\boldsymbol{D}}(\boldsymbol{\chi})$ and $\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi})$.

The final input matrix, $\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi})$, is for example derived as:

$$\widetilde{\boldsymbol{Q}}(\boldsymbol{\chi}) = \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{Q}(\boldsymbol{\chi}) \\ \mathbf{0}_{1\times 3} \end{bmatrix} \tag{I.65}$$

where

$$\widetilde{\boldsymbol{H}}\boldsymbol{Q}(\boldsymbol{\chi}) = \widetilde{\boldsymbol{H}} \begin{bmatrix} \frac{1}{r_k}\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\frac{1}{r_k}\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \tag{I.66}$$

By inserting the constraint matrix, $\widetilde{\boldsymbol{H}}$, from (5.79), the input matrix reduces to

$$\begin{aligned} \widetilde{\boldsymbol{H}}\boldsymbol{Q}(\boldsymbol{\chi}) &= \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{3\times 2} & \begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{I}_3 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \frac{1}{r_k}\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\frac{1}{r_k}\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \\ &= \begin{bmatrix} \frac{1}{r_k}\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -\frac{1}{r_k}\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \\ -2\begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{I}_3 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \\ &= \begin{bmatrix} \frac{1}{r_k}\hat{\mathbf{j}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \\ -\frac{1}{r_k}\hat{\mathbf{i}}^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right) \widetilde{\boldsymbol{W}}^{\mathsf{T}} \\ -2\boldsymbol{W}^{\mathsf{T}} \end{bmatrix} \end{aligned} \tag{I.67}$$

The final mass matrix, $\widetilde{\boldsymbol{M}}(\boldsymbol{\chi})$, is derived as:

$$\widetilde{\boldsymbol{M}}(\boldsymbol{\chi}) = \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi}) \\ \boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi}) \\ \begin{bmatrix} \mathbf{0}_{1\times 2} & {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \end{bmatrix} \end{bmatrix} \tag{I.68}$$

where

$$\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{bmatrix} \tag{I.69}$$

and

$$\widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi}) = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{3\times 2} & \begin{bmatrix} \mathbf{0}_{3\times 1} & \mathbf{I}_3 \end{bmatrix} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\right)^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_{xx}(\boldsymbol{\chi}) & \boldsymbol{M}_{xy}(\boldsymbol{\chi}) & \boldsymbol{M}_{x\boldsymbol{q}}(\boldsymbol{\chi}) \\ \boldsymbol{M}_{yx}(\boldsymbol{\chi}) & \boldsymbol{M}_{yy}(\boldsymbol{\chi}) & \boldsymbol{M}_{y\boldsymbol{q}}(\boldsymbol{\chi}) \\ \boldsymbol{M}_{\boldsymbol{q}x}(\boldsymbol{\chi}) & \boldsymbol{M}_{\boldsymbol{q}y}(\boldsymbol{\chi}) & \boldsymbol{M}_{\boldsymbol{q}\boldsymbol{q}}(\boldsymbol{\chi}) \end{bmatrix} \tag{I.70}$$

This leads to

$$\widetilde{\boldsymbol{H}}\,\boldsymbol{M}(\boldsymbol{\chi}) = \begin{bmatrix} \boldsymbol{M}_{xx}(\boldsymbol{\chi}) & \boldsymbol{M}_{xy}(\boldsymbol{\chi}) & \boldsymbol{M}_{x\boldsymbol{q}}(\boldsymbol{\chi}) \\ \boldsymbol{M}_{yx}(\boldsymbol{\chi}) & \boldsymbol{M}_{yy}(\boldsymbol{\chi}) & \boldsymbol{M}_{y\boldsymbol{q}}(\boldsymbol{\chi}) \\ \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{I}_3\end{bmatrix} \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}}\, \boldsymbol{M}_{\boldsymbol{q}x}(\boldsymbol{\chi}) & \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{I}_3\end{bmatrix} \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}}\, \boldsymbol{M}_{\boldsymbol{q}y}(\boldsymbol{\chi}) & \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{I}_3\end{bmatrix} \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}}\, \boldsymbol{M}_{\boldsymbol{q}\boldsymbol{q}}(\boldsymbol{\chi}) \end{bmatrix} \tag{I.71}$$

so that the mass matrix become

$$\widetilde{\boldsymbol{M}}(\boldsymbol{\chi}) = \begin{bmatrix} \boldsymbol{M}_{xx}(\boldsymbol{\chi}) & \boldsymbol{M}_{xy}(\boldsymbol{\chi}) & \boldsymbol{M}_{x\boldsymbol{q}}(\boldsymbol{\chi}) \\ \boldsymbol{M}_{yx}(\boldsymbol{\chi}) & \boldsymbol{M}_{yy}(\boldsymbol{\chi}) & \boldsymbol{M}_{y\boldsymbol{q}}(\boldsymbol{\chi}) \\ \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{I}_3\end{bmatrix} \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}}\, \boldsymbol{M}_{\boldsymbol{q}x}(\boldsymbol{\chi}) & \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{I}_3\end{bmatrix} \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}}\, \boldsymbol{M}_{\boldsymbol{q}y}(\boldsymbol{\chi}) & \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{I}_3\end{bmatrix} \boldsymbol{\Phi}\big({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}\big)^{\mathsf{T}}\, \boldsymbol{M}_{\boldsymbol{q}\boldsymbol{q}}(\boldsymbol{\chi}) \\ 0 & 0 & {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{\mathsf{T}} \end{bmatrix} \tag{I.72}$$

J Model Linearization

It has not been possible to derive a closed-form symbolic expression of the complete non-linear ODE, mainly due to the difficulty in finding the inverse of the symbolic mass matrix. So to give the reader some insight in the model, the non-linear ODE is linearized around its unstable equilibrium point:

$$\begin{aligned}\tilde{\boldsymbol{x}} &= \begin{bmatrix} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & {}^{\mathrm{K}}_{\mathrm{B}}q_0 & {}^{\mathrm{K}}_{\mathrm{B}}q_1 & {}^{\mathrm{K}}_{\mathrm{B}}q_2 & {}^{\mathrm{K}}_{\mathrm{B}}q_3 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T} \\ \tilde{\boldsymbol{\tau}} &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\mathsf{T}\end{aligned} \tag{J.1}$$

The usual linearization method is to carry out a first-order Taylor expansion.

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{\tau} \\ &\approx \boldsymbol{f}(\tilde{\boldsymbol{x}}) + \boldsymbol{g}(\tilde{\boldsymbol{x}})\tilde{\boldsymbol{\tau}} + \underbrace{\left.\frac{\partial\left(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{\tau}\right)}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\tilde{\boldsymbol{x}} \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}}}}}_{\boldsymbol{A}} \delta\boldsymbol{x} + \underbrace{\left.\frac{\partial\left(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{\tau}\right)}{\partial \boldsymbol{\tau}}\right|_{\substack{\boldsymbol{x}=\tilde{\boldsymbol{x}} \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}}}}}_{\boldsymbol{B}} \delta\boldsymbol{\tau}\end{aligned} \tag{J.2}$$

where $\delta\boldsymbol{x} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$, $\delta\boldsymbol{\tau} = \boldsymbol{\tau} - \tilde{\boldsymbol{\tau}}$, $\boldsymbol{A}$ constitutes the linearized model matrix and $\boldsymbol{B}$ constitutes the linearized input matrix. These linearized matrices can be partitioned as shown below:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{vp}} & \boldsymbol{A}_{\boldsymbol{pq}} & \boldsymbol{A}_{\boldsymbol{pv}} & \boldsymbol{A}_{\boldsymbol{p\dot{q}}} \\ \boldsymbol{A}_{\boldsymbol{\dot{q}p}} & \boldsymbol{A}_{\boldsymbol{\dot{q}q}} & \boldsymbol{A}_{\boldsymbol{\dot{q}v}} & \boldsymbol{A}_{\boldsymbol{\dot{q}\dot{q}}} \\ \boldsymbol{A}_{\boldsymbol{\dot{v}p}} & \boldsymbol{A}_{\boldsymbol{\dot{v}q}} & \boldsymbol{A}_{\boldsymbol{\dot{v}v}} & \boldsymbol{A}_{\boldsymbol{\dot{v}\dot{q}}} \\ \boldsymbol{A}_{\boldsymbol{\ddot{q}p}} & \boldsymbol{A}_{\boldsymbol{\ddot{q}q}} & \boldsymbol{A}_{\boldsymbol{\ddot{q}v}} & \boldsymbol{A}_{\boldsymbol{\ddot{q}\dot{q}}} \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_{\boldsymbol{v\tau}} \\ \boldsymbol{B}_{\boldsymbol{\dot{q}\tau}} \\ \boldsymbol{B}_{\boldsymbol{\dot{v}\tau}} \\ \boldsymbol{B}_{\boldsymbol{\ddot{q}\tau}} \end{bmatrix} = \boldsymbol{g}(\tilde{\boldsymbol{x}}) \tag{J.3}$$

Due to the choice of linearization point in (J.1), the steady state second derivatives, $\boldsymbol{f}(\tilde{\boldsymbol{x}}) = \boldsymbol{0}$ and $\boldsymbol{g}(\tilde{\boldsymbol{x}})\tilde{\boldsymbol{\tau}} = \boldsymbol{0}$, are both zero, assuming an aligned center of mass. Furthermore, since the model is control affine, the input matrix, $\boldsymbol{B}$, corresponds to the input matrix function evaluated at the linearization point, $\boldsymbol{g}(\tilde{\boldsymbol{x}})$.

Since the computation and evaluation of the partial derivatives are cumbersome due to the symbolic expressions, a numerical approximation of the first-order Taylor expansion is computed.

The partial derivative for $\boldsymbol{A}$ is computed as:

$$\begin{aligned}\boldsymbol{A} &= \left.\frac{\partial\left(\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{g}(\boldsymbol{x})\boldsymbol{\tau}\right)}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\tilde{\boldsymbol{x}}\\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}}}} \\ &\approx \begin{bmatrix} \frac{\boldsymbol{f}(\tilde{\boldsymbol{x}}+\boldsymbol{\delta}_x)+\boldsymbol{g}(\tilde{\boldsymbol{x}}+\boldsymbol{\delta}_x)\tilde{\boldsymbol{\tau}}-\boldsymbol{f}(\tilde{\boldsymbol{x}})-\boldsymbol{g}(\tilde{\boldsymbol{x}})\tilde{\boldsymbol{\tau}}}{\|\boldsymbol{\delta}_x\|} & \cdots & \frac{\boldsymbol{f}(\tilde{\boldsymbol{y}}+\boldsymbol{\delta}_{q3})+\boldsymbol{g}(\tilde{\boldsymbol{x}}+\boldsymbol{\delta}_{q3})\tilde{\boldsymbol{\tau}}-\boldsymbol{f}(\tilde{\boldsymbol{y}})-\boldsymbol{g}(\tilde{\boldsymbol{x}})\tilde{\boldsymbol{\tau}}}{\|\boldsymbol{\delta}_{q3}\|} \end{bmatrix}\end{aligned} \tag{J.4}$$

where the pertubation terms are small elementwise pertubations:

$$\begin{aligned}\boldsymbol{\delta}_x &= \begin{bmatrix}\delta & \dots & 0\end{bmatrix}^\mathsf{T} \\ \boldsymbol{\delta}_y &= \begin{bmatrix}0 & \delta & \dots & 0\end{bmatrix}^\mathsf{T} \\ \boldsymbol{\delta}_{q3} &= \begin{bmatrix}0 & \dots & \delta\end{bmatrix}^\mathsf{T}\end{aligned} \tag{J.5}$$

with $\delta \ll 1$.

The linearization is carried out with the MATLAB script found in Kugle-MATLAB/Linearization/ModelLinearization.m which leads to a linearized model of the ballbot on the following form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\delta\boldsymbol{x} + \boldsymbol{B}\delta\boldsymbol{\tau} \tag{J.6}$$

where

$$\boldsymbol{A} = \begin{array}{c} \begin{array}{cccccccccccc} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & {}^{\mathrm{K}}_{\mathrm{B}}q_0 & {}^{\mathrm{K}}_{\mathrm{B}}q_1 & {}^{\mathrm{K}}_{\mathrm{B}}q_2 & {}^{\mathrm{K}}_{\mathrm{B}}q_3 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \end{array} \\ \left[\begin{array}{cccccc|cc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & -9.60 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9.62 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 14.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 14.02 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{c} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \\ {}^{\mathrm{I}}\ddot{x} \\ {}^{\mathrm{I}}\ddot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_3 \end{array} \end{array} \tag{J.7}$$

$$\boldsymbol{B} = \begin{array}{c} \begin{matrix} \tau_0 & \tau_1 & \tau_2 \end{matrix} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & -1.04 & 1.04 \\ 1.20 & -0.60 & -0.60 \\ \hline 0 & 0 & 0 \\ 0.70 & -0.35 & -0.35 \\ 0 & 0.61 & -0.61 \\ -6.90 & -6.90 & -6.90 \end{array}\right] \end{array} \begin{matrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \\ \\ {}^{\mathrm{I}}\ddot{x} \\ {}^{\mathrm{I}}\ddot{y} \\ \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_3 \end{matrix} \tag{J.8}$$

The values are derived using the model parameters from Table 2.1, thus with no friction components, but with the center of mass aligned as in the simulation, see Section 10.2.

$${}^{\mathrm{B}}\boldsymbol{p}_{\mathrm{COM}} = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^{\mathsf{T}} \tag{J.9}$$

The relationship between applied torque, τ_0, τ_1, τ_2, and acceleration, ${}^{\mathrm{I}}\ddot{x}$ and ${}^{\mathrm{I}}\ddot{y}$, apparent in the $\boldsymbol{B}$ matrix, is coherent with the kinematics derived in Chapter 4 and the wheel alignment, Figure 4.1.

Note how the model is completely decoupled when it is linearized around the upright equilibrium. If the model is instead linearized around an arbitrary linearization point, the model matrices are very different, showing a coupled behaviour. We chose an inclination of 5 degree around the x-axis, corresponding to 5 degree roll, and a velocity of 0.1 m/s in the x-direction and 0.2 m/s in the y-direction.

$$\begin{aligned} \tilde{\boldsymbol{x}} &= \begin{bmatrix} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \\ 0 & 0 & \boldsymbol{q}_{\mathrm{X}}(5^\circ)^{\mathsf{T}} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \\ \tilde{\boldsymbol{\tau}} &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \end{aligned} \tag{J.10}$$

In this case the steady state accelerations are no longer zero. Therefore, the linearized matrices now define the derivative of the change in state around the linearization point.

$$\delta\dot{\boldsymbol{x}} = \boldsymbol{A}\delta\boldsymbol{x} + \boldsymbol{B}\delta\boldsymbol{\tau} \tag{J.11}$$

where

$$\boldsymbol{A} = \begin{array}{c} \begin{array}{cccccccccccc} {}^{\mathrm{I}}x & {}^{\mathrm{I}}y & {}^{\mathrm{K}}_{\mathrm{B}}q_0 & {}^{\mathrm{K}}_{\mathrm{B}}q_1 & {}^{\mathrm{K}}_{\mathrm{B}}q_2 & {}^{\mathrm{K}}_{\mathrm{B}}q_3 & {}^{\mathrm{I}}\dot{x} & {}^{\mathrm{I}}\dot{y} & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 & {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \end{array} \\ \left[\begin{array}{cc|cccc|cc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & -9.52 & -0.42 & 0 & 0 & -0.03 & 0 & 0 & -0.01 \\ 0 & 0 & -0.56 & 9.35 & 0 & 0 & 0 & 0 & -0.07 & 0 & 0 & 0 \\ \hline 0 & 0 & 0.08 & -1.21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.29 & 13.76 & 0 & 0 & 0 & 0 & -0.04 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 0 & 13.94 & 0 & 0 & 0 & 0.01 & 0.01 & 0.02 & 0.01 \\ 0 & 0 & 0 & 0 & 0 & -0.03 & 0 & 0 & -0.02 & -0.19 & 0.37 & -0.02 \end{array}\right] \begin{array}{c} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \\ {}^{\mathrm{I}}\ddot{x} \\ {}^{\mathrm{I}}\ddot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_3 \end{array} \end{array} \tag{J.12}$$

$$\boldsymbol{B} = \begin{array}{c} \begin{array}{ccc} \tau_0 & \tau_1 & \tau_2 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline -0.07 & -1.11 & 0.97 \\ 1.20 & -0.60 & -0.60 \\ \hline -0.03 & 0.02 & 0.02 \\ 0.70 & -0.35 & -0.35 \\ 0.33 & 0.93 & -0.27 \\ -6.88 & -6.77 & -6.99 \end{array}\right] \begin{array}{c} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{q}_3 \\ {}^{\mathrm{I}}\ddot{x} \\ {}^{\mathrm{I}}\ddot{y} \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_0 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{q}_3 \end{array} \end{array} \tag{J.13}$$

This coupled behaviour just verifies the choice of a non-linear control strategy which will be able to handle these couplings without performing any linearization steps.

Finally, the poles and zeros are identified in the decoupled linearized model, (J.7) and (J.8). The poles, $\boldsymbol{\lambda}$, are computed as the eigenvalues of the $\boldsymbol{A}$ matrix.

$$\boldsymbol{\lambda} = \text{eig}(\boldsymbol{A}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.7484 & 3.7442 & -3.748 & -3.7442 \end{bmatrix}^{\mathsf{T}} \tag{J.14}$$

The system contain what appears to be eight pure integrators. Six of them are indeed pure integrators, corresponding to the link between velocity and position states. The last two have not been identified, but it could be very slow poles which due to numerical approximations ends up appearing as pure integrators, or it could be a pole related to the scalar value of the quaternion. The system contain two left-half plane poles and two right-half plane poles, all of them related to the balancing dynamics and the link between rotational and translational dynamics. The right-half plane poles are obviously the one making the system unstable, but with a natural frequency of less than 4 rad/s the dynamics of the system are thus relatively slow. For the stable left-half plane poles the time constant of these, assuming them to correspond to first-order dynamics, is $\tau \approx 0.27$ s resulting in a 1% settling time of more than a second.

With complete state output there are no zeros for the linearized system. Thus the right half plane zeros, causing the non-minimum phase behaviour that the ballbot exhibits, first appears when the inner loop has been closed with the balance controller.

The linearized model matrices from (J.7) and (J.8) is used in Appendix P to design an LQR controller for balance control as a comparison to the sliding mode controller designed in the second part of the thesis.

K Planar Model

To get an initial understanding of how the ballbot dynamics works, a simplified planar model in 2D was derived. Instead of including the five degrees of freedom in the full ballbot model, the model includes only two degrees of freedom, namely an inclination angle, ${}^{\mathrm{K}}_{\mathrm{B}}\theta$, and a ball position, ${}^{\mathrm{I}}x$, coupled with the ball angle, ${}^{\mathrm{K}}_{\mathrm{K'}}\phi$. A single actuator is attached as a virtual wheel, denoted with the motor angle ${}^{\mathrm{M}}_{\mathrm{M'}}\psi$. The generalized coordinates and derivatives are:

$$\boldsymbol{q} = \begin{bmatrix} {}^{\mathrm{I}}x \\ {}^{\mathrm{K}}_{\mathrm{B}}\theta \end{bmatrix}, \quad \dot{\boldsymbol{q}} = \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \end{bmatrix} \tag{K.1}$$

The model consists of two rigid bodies: a planar ball in contact with the ground and a massless rod connecting a point-mass to the center of the ball. The same parameters as defined in Section 2.2 is used for the model, except that the inertia of the body around the x-axis is used as the planar models body inertia, $J_b = J_{bx}$. Keep in mind that the inertia of the body includes the inertia of the wheel as well.
A relaxed notation is used for all the angular velocities. The way to interpret the angle derivatives according to the exact notation, presented in Appendix E, is shown below.

$${}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} = \omega_{\mathrm{B}\leftarrow\mathrm{K}} \tag{K.2}$$
$${}^{\mathrm{K}}_{\mathrm{K'}}\dot{\phi} = \omega_{\mathrm{K'}\leftarrow\mathrm{K}} \tag{K.3}$$
$${}^{\mathrm{M}}_{\mathrm{M'}}\dot{\psi} = \omega_{\mathrm{M'}\leftarrow\mathrm{M}} \tag{K.4}$$

Note how the frame identifier has been left out, since the angular velocities are scalar values defining an angular velocity between two 2-dimensional frames.

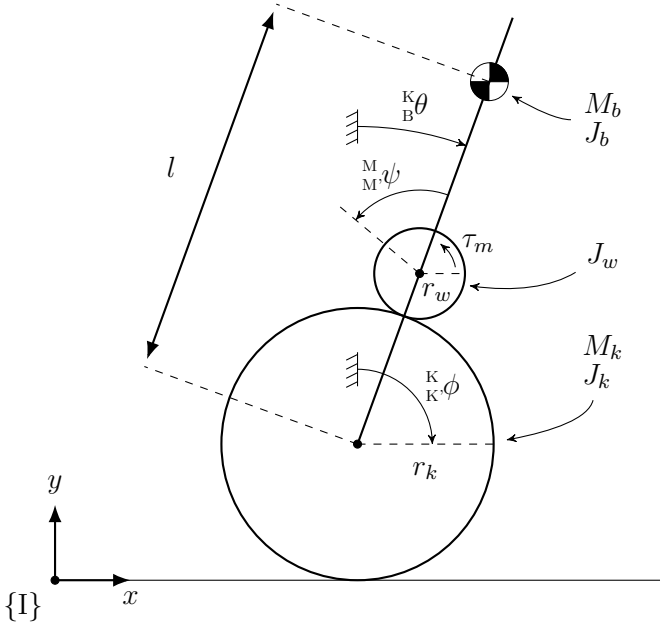


Figure K.1: Planar model including the rigid bodies and direction definitions

This appendix presents the derivation of this planar model using Lagrangian mechanics. Note, however, that since the generalized coordinates of this planar model is equal to the degrees of freedom in the system, constraints are not necessary.

K.1 Frames

The following frames are used in the planar model:

- $\{I\}$: the inertial frame is fixed to the world
- $\{K\}$: the ball frame is attached to the center of the ball but aligned with the inertial frame axes. It is thus only translated in the x-axis with respect to the inertial frame
- $\{K'\}$: the rotating ball frame is rigidly attached to the ball, rotated with the ball angle, ${}^{\mathrm{K}}_{\mathrm{K'}}\phi$, and has its origin in the center, and thus rotates with the ball as the ballbot moves
- $\{B\}$: the body frame is rigidly attached to the rod, tilted with an angle ${}^{\mathrm{K}}_{\mathrm{B}}\theta$ with respect to the ball frame, $\{K\}$, and has its origin in the center of the ball
- $\{M\}$: the motor frame (center of virtual wheel) is rigidly attached to the rod and is thus translated from the body frame in the direction of the rod with a distance, l
- $\{M'\}$: the rotating motor frame (center of virtual wheel) is rotating with the virtual wheel angle, ${}^{\mathrm{M}}_{\mathrm{M'}}\psi$, with respect to the motor frame, $\{M\}$, and has its origin in the center of the motor frame

Note that both the inclination angle and ball angle are clockwise positive whereof the motor angle is counter-clockwise positive.

K.2 Assumptions

The no-slip condition is also assumed for the planar model resulting in a rolling constraint. This hereby kinematically links the position of the ball with the ball angle, ${}^{\mathrm{K}}_{\mathrm{K'}}\phi$, and correspondingly for the angular velocity and translational velocity.

$$ {}^{\mathrm{I}}x = r_k \, {}^{\mathrm{K}}_{\mathrm{K'}}\phi \;\rightarrow\; {}^{\mathrm{I}}\dot{x} = r_k \, {}^{\mathrm{K}}_{\mathrm{K'}}\dot{\phi} \tag{K.5} $$

Furthermore, the contact point between the virtual wheel and the ball is assumed to always be at

$$ {}^{\mathrm{B}}\boldsymbol{p}_c = \begin{bmatrix} 0 \\ r_k \end{bmatrix} \tag{K.6} $$

The instantaneous velocity at contact the contact point is always zero, thus the tangential velocity of the virtual wheel is equal to the tangential velocity of the ball. Due to the choice of directions this assumption requires

$$ r_w \omega_{\mathrm{M'}\leftarrow\mathrm{B}} = r_k \omega_{\mathrm{K'}\leftarrow\mathrm{B}} \tag{K.7} $$

There is assumed to be no friction forces in the system.

Finally, the point-mass, corresponding to the body mass, is centered at

$$ {}^{\mathrm{B}}\boldsymbol{p}_m = \begin{bmatrix} 0 \\ l \end{bmatrix} \tag{K.8} $$

with an inertia, J_b, defined with the same origin as the body frame.

K.3 Kinematics

Before deriving the energy equations we need to derive the kinematics. The kinematics is the relationship between the angular velocity of the wheel, ${}^{\mathrm{M}}_{\mathrm{M'}}\dot{\psi} = \omega_{\mathrm{M'}\leftarrow\mathrm{M}}$, and the derivative of the generalized coordinates, in this case the translational velocity of the ball and angular velocity of the body. The angular velocity of the wheel, $\omega_{\mathrm{M'}\leftarrow\mathrm{M}}$, is with respect to the fixed motor frame, $\{M\}$, which is rigidly attached to the body, $\{B\}$. This velocity will thus correspond to the measurable velocity by an encoder installed on the motor.

The assumption of zero instantaneous velocity at the contact point is used. This relationship states that the angular velocity of the wheel seen from the body frame, $\omega_{\mathrm{M'}\leftarrow\mathrm{B}}$, is equal to the angular velocity of the ball seen from the body frame, $\omega_{\mathrm{K'}\leftarrow\mathrm{B}}$. The angular velocity of the ball is derived using the relations of moving frames described in Appendix E.

$$\begin{aligned}\omega_{\mathrm{K'}\leftarrow\mathrm{B}} &= \omega_{\mathrm{K'}\leftarrow\mathrm{K}} + \omega_{\mathrm{K}\leftarrow\mathrm{B}} \\ &= \omega_{\mathrm{K'}\leftarrow\mathrm{K}} - \omega_{\mathrm{B}\leftarrow\mathrm{K}}\end{aligned} \tag{K.9}$$

Now by reordering (K.7) the angular velocity of the wheel is computed by

$$\begin{aligned}\omega_{\mathrm{M'}\leftarrow\mathrm{B}} &= \frac{r_k}{r_w}\omega_{\mathrm{K'}\leftarrow\mathrm{B}} \\ &= \frac{r_k}{r_w}\left(\omega_{\mathrm{K'}\leftarrow\mathrm{K}} - \omega_{\mathrm{B}\leftarrow\mathrm{K}}\right)\end{aligned} \tag{K.10}$$

Since the rotation of the motor frame, $\{M\}$, and body frame $\{B\}$, coincides, the angular velocity is also defined with respect to the motor frame as desired.

$$\begin{aligned}{}^{\mathrm{M}}_{\mathrm{M'}}\dot{\psi} &= \frac{r_k}{r_w}\left({}^{\mathrm{K}}_{\mathrm{K'}}\dot{\phi} - {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}\right) \\ &= \frac{1}{r_w}{}^{\mathrm{I}}\dot{x} - \frac{r_k}{r_w}{}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}\end{aligned} \tag{K.11}$$

K.4 Energy equations

Similar to Section 5.2 the first step when modelling a system using Lagrangian mechanics is to derive the energy equations. The kinetic and potential energy are derived for the ball, body and wheel.

K.4.1 Ball energy

$$T_k = \frac{1}{2}J_k\,{}^{\mathrm{K}}_{\mathrm{K'}}\dot{\phi}^2 + \frac{1}{2}M_k\,{}^{\mathrm{I}}\dot{x}^2 \tag{K.12}$$

$$V_k = 0 \tag{K.13}$$

Note that the ball has no potential energy since the ball frame is assumed only to move in the x-axis of the inertial frame.

K.4.2 Body energy

$$T_b = \frac{1}{2}J_b\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 + \frac{1}{2}M_b\,{}^{\mathrm{I}}\dot{\boldsymbol{p}}_m^{\mathsf{T}}\,{}^{\mathrm{I}}\dot{\boldsymbol{p}}_m \tag{K.14}$$

$$V_b = M_b\,g\begin{bmatrix}0 & 1\end{bmatrix}{}^{\mathrm{I}}\boldsymbol{p}_m \tag{K.15}$$

$$= M_b\,gl\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \tag{K.16}$$

Where the position of the body center of mass, ${}^{\mathrm{I}}\boldsymbol{p}_m$, is computed by rotation.

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{R} = \begin{bmatrix} \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) & \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \\ -\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) & \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \end{bmatrix} \tag{K.17} $$

$$ {}^{\mathrm{K}}\boldsymbol{p}_m = {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\boldsymbol{p}_m \tag{K.18} $$

$$ \begin{aligned} {}^{\mathrm{I}}\boldsymbol{p}_m &= {}^{\mathrm{I}}\boldsymbol{O}_{\mathrm{K}} + {}^{\mathrm{K}}\boldsymbol{p}_m \\ &= \begin{bmatrix} {}^{\mathrm{I}}x + l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \\ l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \end{bmatrix} \end{aligned} \tag{K.19} $$

The velocity of the center of mass, ${}^{\mathrm{I}}\dot{\boldsymbol{p}}_m$, is now computed by differentiating (K.20).

$$ {}^{\mathrm{I}}\dot{\boldsymbol{p}}_m = {}^{\mathrm{I}}\dot{\boldsymbol{O}}_{\mathrm{K}\leftarrow\mathrm{I}} + {}^{\mathrm{K}}\dot{\boldsymbol{p}}_m \tag{K.20} $$

$$ \begin{aligned} {}^{\mathrm{K}}\dot{\boldsymbol{p}}_m &= {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{R}}\,{}^{\mathrm{B}}\boldsymbol{p}_m + {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{R}\,{}^{\mathrm{B}}\dot{\boldsymbol{p}}_m \\ &= {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{R}}\,{}^{\mathrm{B}}\boldsymbol{p}_m \end{aligned} \tag{K.21} $$

$$ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{R}} = \frac{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{R}}{\partial\,{}^{\mathrm{K}}_{\mathrm{B}}\theta}\frac{\mathrm{d}\,{}^{\mathrm{K}}_{\mathrm{B}}\theta}{\mathrm{d}t} \tag{K.22} $$

$$ \begin{aligned} &= \begin{bmatrix} -\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) & \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \\ -\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) & -\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \end{bmatrix} {}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \\ {}^{\mathrm{I}}\dot{\boldsymbol{p}}_m &= \begin{bmatrix} {}^{\mathrm{I}}\dot{x} + l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \\ -l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \end{bmatrix} \end{aligned} \tag{K.23} $$

The kinetic energy equation includes the squared norm of this velocity vector, which is thus derived.

$$ \begin{aligned} {}^{\mathrm{I}}\dot{\boldsymbol{p}}_m^{\mathsf{T}}\,{}^{\mathrm{I}}\dot{\boldsymbol{p}}_m &= \begin{bmatrix} {}^{\mathrm{I}}\dot{x} + l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \\ -l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} {}^{\mathrm{I}}\dot{x} + l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \\ -l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta \end{bmatrix} \\ &= \left(\left({}^{\mathrm{I}}\dot{x} + l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta\right)^2 + \left(-l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta\right)^2\right) \\ &= {}^{\mathrm{I}}\dot{x}^2 + 2\,{}^{\mathrm{I}}\dot{x}l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta + \left(l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta\right)^2 + \left(-l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta\right)^2 \\ &= {}^{\mathrm{I}}\dot{x}^2 + 2\,{}^{\mathrm{I}}\dot{x}l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta + l^2\,{}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta^2\left(\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)^2 + \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)^2\right) \end{aligned} \tag{K.24} $$

Applying the Pythagorean identity, $\cos(\theta)^2 + \sin(\theta)^2 = 1$, yields:

$$ {}^{\mathrm{I}}\dot{\boldsymbol{p}}_m^{\mathsf{T}}\,{}^{\mathrm{I}}\dot{\boldsymbol{p}}_m = {}^{\mathrm{I}}\dot{x}^2 + 2\,{}^{\mathrm{I}}\dot{x}l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\dot{\mathrm{K}}}_{\mathrm{B}}\theta + l^2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 \tag{K.25} $$

The squared norm is substituted back into the body kinetic energy equation.

$$ T_b = \frac{1}{2}J_b\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 + \frac{1}{2}M_b\,{}^{\mathrm{I}}\dot{x}^2 + M_b\,{}^{\mathrm{I}}\dot{x}l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} + \frac{1}{2}M_b l^2\,{}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 \tag{K.26} $$

K.4.3 Wheel energy

$$ T_w = \frac{1}{2}J_w\,{}^{\mathrm{M}}_{\mathrm{M}}\dot{\psi}^2 \tag{K.27} $$

$$ V_w = 0 \tag{K.28} $$

Note that the wheel is assumed to have no potential energy as the mass of the wheel is included in the mass of the body, affecting the center of mass as well.
Since the wheel inertia has already been included in the body inertia, it is sufficient to just compute the wheel energy based on the angular velocity of the wheel within the motor frame, fixed to the body. The angular velocity of the wheel is computed using the kinematics, (K.11).

$$T_w = \frac{1}{2} J_w \left(\frac{r_k}{r_w} \left({}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi} - {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \right) \right)^2 \tag{K.29}$$

$$= \frac{1}{2} J_w \frac{r_k^2}{r_w^2} \left({}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi}^2 - 2\, {}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} + {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 \right) \tag{K.30}$$

K.5 Equations of motion

The equations of motion is derived with the Euler-Lagrange equation, (5.4):

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\chi}}} \right)^{\mathsf{T}} - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\chi}} \right)^{\mathsf{T}} = \boldsymbol{Q} \tag{K.31}$$

which requires the computation of the Lagrangian.

$$\begin{aligned} \mathcal{L} &= T_k + T_b + T_w - V_b \\ &= \frac{1}{2} J_k \, {}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi}^2 + \frac{1}{2} M_k \, {}^{\mathrm{I}}\dot{x}^2 + \frac{1}{2} J_b \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 + \frac{1}{2} M_b \, {}^{\mathrm{I}}\dot{x}^2 + M_b \, {}^{\mathrm{I}}\dot{x} l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta \right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \\ &+ \frac{1}{2} M_b l^2 \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 + \frac{1}{2} J_w \frac{r_k^2}{r_w^2} \left({}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi}^2 - 2\, {}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} + {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 \right) - M_b \, gl \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta \right) \end{aligned} \tag{K.32}$$

The rolling constraint, ${}^{\mathrm{K}}_{\mathrm{K}}\dot{\phi} = \frac{1}{r_k} \, {}^{\mathrm{I}}\dot{x}$, is applied to get rid of the angular velocity of the ball and the terms are collected.

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \left(J_k \frac{1}{r_k^2} + J_w \frac{1}{r_w^2} + M_k + M_b \right) {}^{\mathrm{I}}\dot{x}^2 \\ &+ \frac{1}{2} \left(J_b + M_b l^2 + J_w \frac{r_k^2}{r_w^2} \right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 \\ &+ \left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta \right) - J_w \frac{r_k}{r_w^2} \right) {}^{\mathrm{I}}\dot{x} \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \\ &- M_b \, gl \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta \right) \end{aligned} \tag{K.33}$$

Next we compute the partial derivatives of the Lagrangian.

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\chi}}} \right)^{\mathsf{T}} = \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathcal{L}}{\partial \, {}^{\mathrm{I}}\dot{x}} \\ \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathcal{L}}{\partial \, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}} \end{bmatrix} \tag{K.34}$$

$$\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\chi}} \right)^{\mathsf{T}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \, {}^{\mathrm{I}}x} \\ \frac{\partial \mathcal{L}}{\partial \, {}^{\mathrm{K}}_{\mathrm{B}}\theta} \end{bmatrix} \tag{K.35}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \, {}^{\mathrm{I}}\dot{x}} &= \left(J_k \frac{1}{r_k^2} + J_w \frac{1}{r_w^2} + M_k + M_b \right) {}^{\mathrm{I}}\dot{x} \\ &+ \left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta \right) - J_w \frac{r_k}{r_w^2} \right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \end{aligned} \tag{K.36}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}} &= \left(J_b + M_b l^2 + J_w \frac{r_k^2}{r_w^2}\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \\ &\quad + \left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) - J_w \frac{r_k}{r_w^2}\right) {}^{\mathrm{I}}\dot{x}\end{aligned} \tag{K.37}$$

$$\frac{\partial \mathcal{L}}{\partial\, {}^{\mathrm{I}}x} = 0 \tag{K.38}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\theta} &= -M_b l \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) {}^{\mathrm{I}}\dot{x}\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \\ &\quad + M_b\, g l \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)\end{aligned} \tag{K.39}$$

Next the time derivative of the partial derivative with respect to the derivative states, is taken.

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial\, {}^{\mathrm{I}}\dot{x}} &= \left(J_k \frac{1}{r_k^2} + J_w \frac{1}{r_w^2} + M_k + M_b\right) {}^{\mathrm{I}}\ddot{x} \\ &\quad - M_b l \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 \\ &\quad + \left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) - J_w \frac{r_k}{r_w^2}\right) {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\theta}\end{aligned} \tag{K.40}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}} &= \left(J_b + M_b l^2 + J_w \frac{r_k^2}{r_w^2}\right) {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\theta} \\ &\quad - M_b l \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}\, {}^{\mathrm{I}}\dot{x} \\ &\quad + \left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) - J_w \frac{r_k}{r_w^2}\right) {}^{\mathrm{I}}\ddot{x}\end{aligned} \tag{K.41}$$

Combining (K.38), (K.39), (K.40) and (K.41) yields the two equations of motion describing the planar model of the ballbot, though without torque input.

$$\begin{aligned} {}^{\mathrm{I}}\ddot{x}\left(J_k \frac{1}{r_k^2} + J_w \frac{1}{r_w^2} + M_k + M_b\right) + {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\theta}\left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) - J_w \frac{r_k}{r_w^2}\right) - {}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 M_b l \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) &= 0 \\ {}^{\mathrm{I}}\ddot{x}\left(M_b l \cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) - J_w \frac{r_k}{r_w^2}\right) + {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\theta}\left(J_b + M_b l^2 + J_w \frac{r_k^2}{r_w^2}\right) - M_b\, g l \sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) &= 0 \end{aligned} \tag{K.42}$$

K.5.1 Including non-conservative forces

The right hand side of the Euler-Lagrange equation are the non-conservative forces. In this case we would like to include the motor torque input, as applied to the wheel. As described in Section 5.4 input forces are included through generalized forces which has to be computed from the Jacobian of the mapping between the coordinates affected by the force and the generalized coordinates.

$$Q_j = \sum_i F_{\mathrm{ext},i} \frac{\partial r_i}{\partial \chi_j} \tag{K.43}$$

In this case, since we would like to include an input torque, the kinematics between the wheel angular velocity, (K.11), and the generalized coordinates is used.

$$\boldsymbol{J}_\tau = \frac{\partial\, {}^{\mathrm{M}}_{\mathrm{M}'}\dot{\psi}}{\partial \dot{\boldsymbol{\chi}}} = \begin{bmatrix} \frac{1}{r_w} \\ -\frac{r_k}{r_w} \end{bmatrix} \tag{K.44}$$

The input force matrix is then defined as

$$\boldsymbol{Q} = \boldsymbol{J}_\tau \, \tau = \begin{bmatrix} \frac{1}{r_w} \\ -\frac{r_k}{r_w} \end{bmatrix} \tau \tag{K.45}$$

K.5.2 Final equations of motion

Including this input force matrix in the two equations of motion, (K.42), leads to the final equations of motion for the planar model, (K.46).

$$\begin{aligned} {}^{\mathrm{I}}\ddot{x}\left(J_k\frac{1}{r_k^2}+J_w\frac{1}{r_w^2}+M_k+M_b\right)+{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\theta}\left(M_b l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)-J_w\frac{r_k}{r_w^2}\right)-M_b l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta}^2 &= \frac{1}{r_w}\tau \\ {}^{\mathrm{I}}\ddot{x}\left(M_b l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)-J_w\frac{r_k}{r_w^2}\right)+{}^{\mathrm{K}}_{\mathrm{B}}\ddot{\theta}\left(J_b+M_b l^2+J_w\frac{r_k^2}{r_w^2}\right)-M_b\, gl\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) &= -\frac{r_k}{r_w}\tau \end{aligned} \tag{K.46}$$

This can be written on the general form as in (5.10).

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{Q}\,\tau \tag{K.47}$$

where the mass matrix, Coriolis matrix, gravity matrix and input matrix are defined as:

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} J_k\frac{1}{r_k^2}+J_w\frac{1}{r_w^2}+M_k+M_b & M_b l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)-J_w\frac{r_k}{r_w^2} \\ M_b l\cos\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right)-J_w\frac{r_k}{r_w^2} & J_b+M_b l^2+J_w\frac{r_k^2}{r_w^2} \end{bmatrix} \tag{K.48}$$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{\chi}}) = \begin{bmatrix} 0 & -M_b l\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right){}^{\mathrm{K}}_{\mathrm{B}}\dot{\theta} \\ 0 & 0 \end{bmatrix} \tag{K.49}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} 0 \\ -M_b\, gl\sin\left({}^{\mathrm{K}}_{\mathrm{B}}\theta\right) \end{bmatrix} \tag{K.50}$$

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{r_w} \\ -\frac{r_k}{r_w} \end{bmatrix} \tag{K.51}$$

Note the similarities with (1.1), which can be constructed by adding the first equation of motion to the second.

K.6 Future work

Instead of using a scalar angle (Euler) for the inclination, a 2D abstraction of the quaternion could have been used. Note that the 2D quaternion, $\boldsymbol{q}$, is not to be confused with neither the generalized coordinates presented above nor the real quaternion used in the rest of this thesis.

$$\boldsymbol{q} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{K.52}$$

$$\dot{\boldsymbol{q}} = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \dot{\theta} \tag{K.53}$$

In this case the generalized coordinates for the planar model would have become overdetermined, similar to the quaternion model in Chapter 5, and the norm constraint would be required.

$$\boldsymbol{\chi} = \begin{bmatrix} x \\ \boldsymbol{q} \end{bmatrix} \qquad \text{where} \quad \boldsymbol{q}^\mathsf{T}\boldsymbol{q} = 1 \tag{K.54}$$

It would hereby be possible to go through all the necessary steps in the Lagrangian mechanics modelling process, including adding the constraint to reduce the Euler-Lagrange equations and generalized coordinate space. This would give some useful insight in the procedure and pitfalls and could help to build intuition before diving into the advanced full model.

L Sliding Mode Controller Theory

This appendix introduces the theory behind the sliding mode controller including the necessary terminologies. The appendix is based on the literature [60] [61] [92]. If the reader is unfamiliar with sliding mode control or is in need of a brushup, a quick but complete walkthrough of sliding mode control and integral sliding mode can be found in [93] and [94].

A Sliding Mode Controller, in the latter abbreviated SMC, is a robust non-linear controller. The aim of a sliding mode controller and the reason for applying a sliding mode controller to control the balance of the ballbot, is to design a control law to effectively account for:

- parameter uncertainties, e.g., imprecision on the mass properties or loads, inaccuracies on the torque constants of the actuators, friction, and so on.
- the presence of unmodeled dynamics, such as structural resonant modes, neglected time-delays (in the actuators, for instance), or finite sampling rate.

The typical structure of a robust controller is composed of a nominal part, similar to feedback linearization or inverse dynamics, and additional terms aimed at dealing with model uncertainty.

In its original form, described and used in this thesis, the sliding mode controller is robust against matched disturbances. Uncertain terms entering the state equation at the point of control input, $\boldsymbol{\delta}(t, \boldsymbol{x}, \boldsymbol{u})$ in (L.1), are categorized as matched disturbances, e.g., parametric uncertainties within the range space of the input matrix, $\boldsymbol{g}(\boldsymbol{x})$.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x}) \left(\boldsymbol{u} + \boldsymbol{\delta}(t, \boldsymbol{x}, \boldsymbol{u})\right) \tag{L.1}$$

The matching condition can be relaxed with e.g., backstepping control or other enhanced versions of the sliding mode controller. Integral Sliding Mode Controller, ISMC, is for example able to handle unmatched disturbances and has a less aggressive output [95] [92] [96] [94] [69].

The functionality of a sliding mode controller is governed by two phases illustrated in Figure L.1.

1. Reaching phase : force the system states onto a predefined, desirable manifold, $s = 0$, in finite time.
2. Sliding phase : stay on the manifold for all future time.

The sliding- and reaching phase are both defined by a scalar variable, s, where $s = 0$ defines the desired sliding manifold also known as the sliding surface. Commonly the sliding manifold defines a first order relationship between certain states and their derivatives, as shown in (L.2), where $\boldsymbol{T}$ is a mapping matrix to extract the given states.

$$s = \boldsymbol{T}\dot{\boldsymbol{\chi}} + K\boldsymbol{T}\boldsymbol{\chi} \tag{L.2}$$

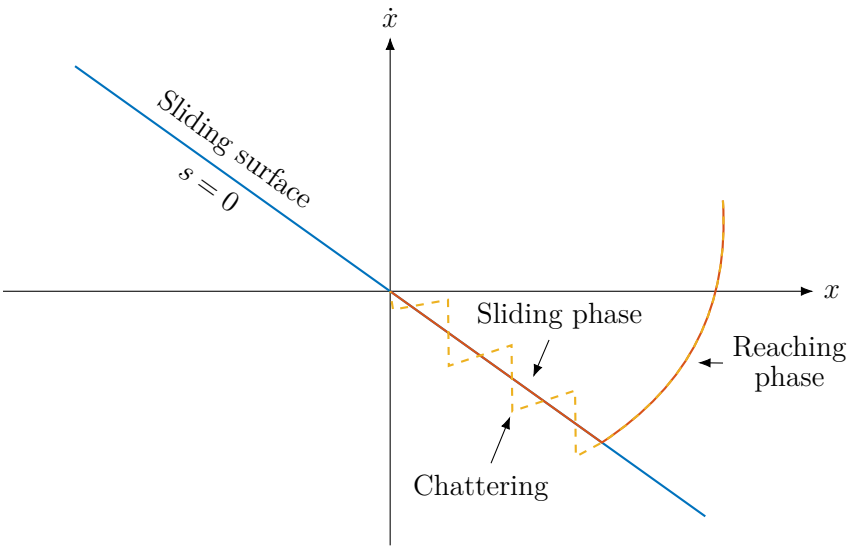


Figure L.1: Phase plot of sliding mode controller on one dimensional system

It is crucial that the derivative of the sliding manifold, $\dot{s}$, contains the controllable inputs, $\boldsymbol{u}$. This is achieved by ensuring that the relative degree of the sliding manifold is $r-1$, with r being the relative degree of the system. The sliding surface thus describes both dynamics and place to be. Note that higher order sliding mode controllers exists, such as the super-twisting controller [67] [92] [97] [98], but this is outside the scope of this thesis.

There is no general method for choosing the sliding surface, but a linear combination of the state variables is an option. If one of the state variables is a system output to be controlled, this variable must be used. Note that the state variables used in the sliding surface must have an equilibrium at zero, which can also be achieved through a coordinate transform. It is also possible to add integral and derivative components to the sliding surface.

The surface is designed to ensure convergence of the system states towards an equilibrium. Moreover the sliding surface should reflect the design specifications and control objectives. When the system is off the manifold, $s \neq 0$, it is said to be in the **reaching phase**. When the system is on the manifold, $s = 0 \wedge \dot{s} = 0$, the system is said to be in the **sliding phase**.

In the reaching phase the controller pushes the system towards the sliding manifold with a predefined rate, assuming a sufficiently large switching gain, η, so that disturbances are overcome. For a sliding mode controller with constant switching gain the reaching time is upper bounded by

$$t_r \leq \frac{|s(0)|}{\eta} \tag{L.3}$$

where $s(0)$ is the initial value of the sliding variable. When the system finally hits the sliding surface it remains there for all future time, $\dot{s} = 0$.

In the sliding phase the order of the system dynamics is reduced by one degree, since a degree of freedom gets tied by the manifold relationship. The sliding surface thereby dictates the system dynamics in the sliding phase, and is thus usually designed to achieve the control objective, e.g., stabilizing the origin of the reduced-order system defined by the surface. The motion on the sliding surface can be given an interesting geometric interpretation as an "average" of the system dynamics on both sides of the surface, see Figure L.2.

The benefit of the sliding mode controller is that motion on the manifold, thus the closed loop dynamics when being on the manifold, known as the sliding mode, is independent of matched uncertainties. The sliding mode controller can handle fairly large uncertainties, limited only by the switching gain but in practice also limited by physical constraints on the amplitude of the control signals. This enables controller design of a reduced-order controller where non-linearities and uncertainties can be neglected, since these are handled by the sliding mode controller. The downside is unfortunately that to give this robustness the basic sliding mode controller works as a bang-bang controller.

The sliding mode control law consists of two parts:

1. Nominal control law : control output to keep the nominal system on the sliding manifold, also known as equivalent control and is commonly derived through inverse dynamics or computed torque.
2. Switching control law : discontinuous (or continuous with a boundary layer) control output which pushes the system towards the sliding manifold in the perpendicular direction, hence the direct path towards the surface.

For a nominal system, hence a system with no uncertainties nor disturbances, no switching would occur and the system would be able to slide on the sliding surface with just the nominal control law, (L.4).

$$\boldsymbol{u}_{\text{equiv}} = \boldsymbol{c}(\boldsymbol{x}) \tag{L.4}$$

The nominal control law is designed so that the nominal dynamics is cancelled to ensure $\dot{s} = 0$. However, with uncertainties or disturbances the actual system will diverge from the sliding surface which activates the discontinuous switching control law, (L.5), defined with a signum function.

$$\boldsymbol{u}_{\text{switch}} = -\boldsymbol{d}(\boldsymbol{x})\,\eta(\boldsymbol{x})\,\text{sgn}(s) \tag{L.5}$$

The switching control law is designed to push the sliding variable, s, towards the sliding manifold, $s = 0$. The direction matrix, $\boldsymbol{d}(\boldsymbol{x})$, maps the perpendicular direction of the sliding surface into actuation space, while the switching gain, $\eta(\boldsymbol{x})$, defines an upper bound of the tolerable uncertainties. For infinitely fast discontinuous switching, the closed loop dynamics of the system would "average" into the nominal system dynamics as illustrated by Filippov in Figure L.2.

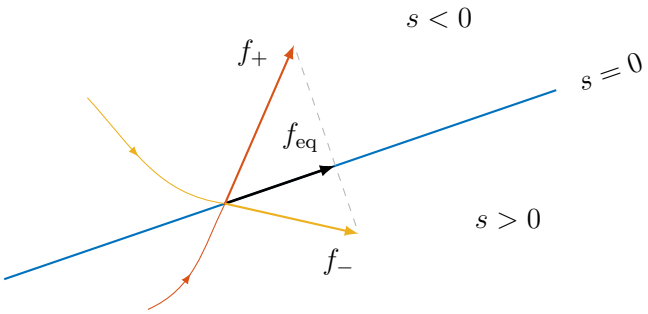


Figure L.2: Illustration of Filippov's method of "averaging" the dynamics

The high frequency switching is, however, what results in a bang-bang control strategy when the system is sliding, which can lead to chattering in the system states. Chattering occurs due to imperfections in the switching device, e.g., actuator, or due to unmodeled dynamics or delays. In specific applications where control chattering is acceptable, the pure switching control laws can yield extremely high performance. Sliding mode is therefore especially efficient for systems where the control output is digital (ON/OFF, relay-based etc.) or for systems where an analog output is approximated with PWM.

The discontinuous control law results in a signal comparable to a pulse-width modulated signal and the combined control output thus looks like a PWM signal with a varying offset due to the equivalent control law.

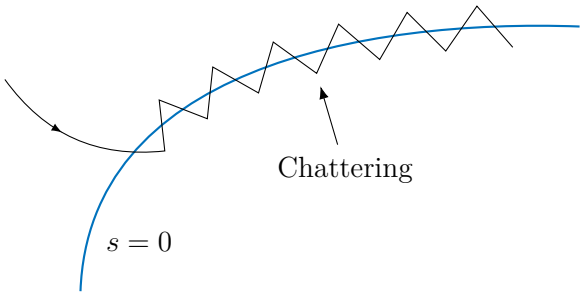


Figure L.3: Chattering phenomena

For systems where chattering would cause tremendous wear or is not be feasible, continuous switching strategies have been proposed to reduce the discontinuity when the system is close to the manifold. The aim of the continuous strategies is to replace the discontinuous control law with an approximation which is smooth in a boundary layer close to the sliding manifold, $|s| < \bar{s}_\epsilon$, a so-called ϵ-tube, in where the switching happens smoothly.

The boundary layer is designed such that it is invariant, that is all trajectories starting inside the boundary layer should stay inside. Instead of having a control law which is discontinuous at $s = 0$, the continuous switching laws smoothens the discontinuous transition from $u = -\eta$ to $u = \eta$. Outside the boundary layer the control law is chosen as before, hence the switched output being either positive or negative, which thereby ensures invariance. Usually the signum function is replaced by the saturation, sigmoid or hysteresis function, offering a continuous or smooth control signal (L.6) [60] [61]. Using the saturation function leads to a linear transition in the boundary layer.

$$\boldsymbol{u}_{\text{switch}} = -\boldsymbol{d}(\boldsymbol{x})\,\eta(\boldsymbol{x})\,\text{sat}\left(\frac{s}{\epsilon}\right) \tag{L.6}$$

where

$$\text{sat}(x) = \begin{cases} x & |x| \leq 1 \\ \text{sgn}(x) & |x| > 1 \end{cases} \tag{L.7}$$

Others [98] suggest to use the tangent hyperbolic function to reduce chatter, yielding a smooth but non-linear transition.
The continuous sliding mode controller achieves ultimate boundedness with an ultimate bound that can be controlled by the design parameter ϵ. The system will reach the set $|s| \leq \bar{s}_\epsilon$ in finite time and remain within the set thereafter, where $\bar{s}_\epsilon$ depends on the ϵ and the upper bound of the matched disturbance, $\bar{\boldsymbol{\Delta}}$ [99] [60].

The boundary layer can be made time- and state-dependent, $\epsilon(t, \boldsymbol{x})$, to achieve an optimal trade-off between tracking precision and robustness at all times, e.g., starting with a wide ϵ-tube and then narrowing it down. However, it is important that the boundary layer is chosen such that the Lyapunov stability criteria are still fulfilled when the trajectories are outside the boundary layer, that is the sliding manifold must be attractive [60].

In other applications it might be relevant to investigate smoothed first- or second-order switching laws as a measure to reduce chatter, but this is again outside the scope of this thesis [68] [69].

L.1 Common practice

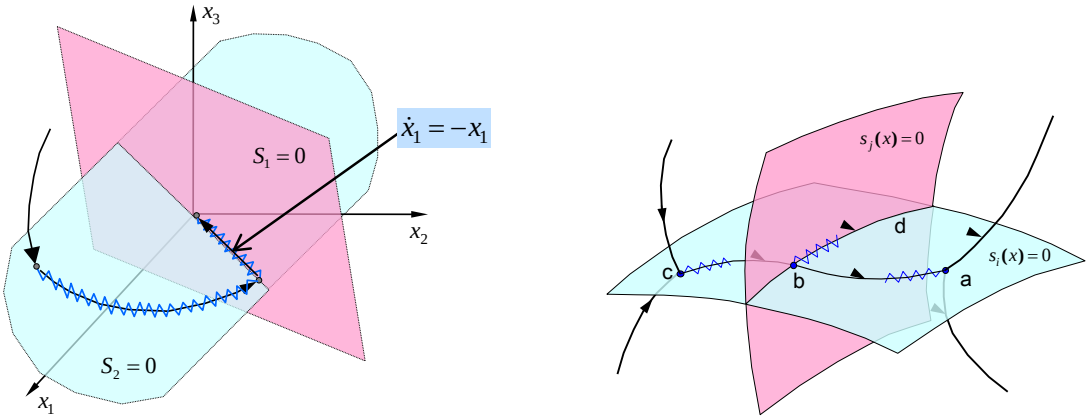
There seems to be two common approaches to the design of a sliding mode controller and the choice of sliding manifold. Slotine [60] presents a design strategy where the designer chooses a sliding manifold which reflects the tracking error, such that $s \to 0$ when $x_e \to 0$. Note that the tracking error is defined as a scalar. For a second order system, that is a system where the control input goes into an integrator chain of two integrators, Slotine proposes a linear sliding manifold on the tracking error:

$$s = \dot{x}_e + K x_e \tag{L.8}$$

When the system is in sliding phase, $s = 0$, the closed loop dynamics will hereby be governed by the first order equation:

$$\dot{x}_e = -K x_e \tag{L.9}$$

As long as $K > 0$ the above system is asymptotically stable with the equilibrium at $x_e = 0$.



(a) Two dimensional sliding mode

(b) Multivariable sliding mode in discontinuity surfaces and their intersection

Figure L.4: Illustrations of multi-dimensional sliding surfaces [100]

The above example can be extended to a tracking error vector by having multiple sliding manifolds as illustrated in Figure L.4.

$$\boldsymbol{s} = \dot{\boldsymbol{x}}_e + \boldsymbol{K}\boldsymbol{x}_e \to \dot{\boldsymbol{x}}_e = -\boldsymbol{K}\boldsymbol{x}_e \tag{L.10}$$

As long as the manifold gain is positive definite, $\boldsymbol{K} \succ 0$, the system is asymptotically stable with the equilibrium at $\boldsymbol{x}_e = \boldsymbol{0}$. Due to the choice of a linear sliding manifold, the controller gain or manifold gain, $\boldsymbol{K}$, can be designed using linear MIMO controller design techniques.

Another approach is presented by Khalil [61] who assumes the system to be in normal form, see (L.11), and the control objective is to drive the system to its equilibrium.

$$\begin{aligned} \dot{\eta} &= f_a(\eta, \xi) \\ \dot{\xi} &= f_b(\eta, \xi) + g(\eta, \xi)u + \delta(t, x, u) \end{aligned} \tag{L.11}$$

Where Slotine uses a tracking error, Khalil defines a linear sliding manifold based on a control law, $\phi(\eta)$, designed to stabilize the internal dynamics, $\dot{\eta} = f_a(\eta, \phi(\eta))$.

$$s = \xi - \phi(\eta) \tag{L.12}$$

The design of $\phi(\eta)$ corresponds to solving a stabilization problem for the nominal internal system of reduced order, $\dot{\eta} = f_a(\eta, \xi)$, which is why any controller design methods can be used for this. The method by Khalil can be extended to reference tracking by a simple change of coordinates.

The striking feature of sliding mode control is its robustness with respect to f_b and g and we only need to know the upper bound of the disturbance, $\delta(t, x, u)$. During the sliding phase the motion is completely independent of f_b and g.

From the design guidelines by Khalil and Slotine, the design of a sliding mode controller can be summarized in five steps:

1. Transform the system into normal/regular form.
2. Construct sliding manifold / sliding surface to stabilize the system e.g., based on desired error dynamics.
3. Derive equivalent control law from the defined sliding surface.
4. Define switching law.
5. Define switching gain based on disturbance rejection requirements.

M Quaternion EKF

In this appendix the Extended Kalman filter used for Quaternion state estimation is designed and tested. A summary of the algorithm is found in Appendix M.8. The MATLAB code for the QEKF can be found at: Kugle-MATLAB/Estimators/QEKF. The C++ implementation of the QEKF, generated by using MATLAB Embedded Coder®, can be found at: Kugle-Embedded/KugleFirmware/Libraries/Modules/Estimators/QEKF.

M.1 Objective

The objective of the Quaternion Extended Kalman Filter, in the latter denoted QEKF, is to estimate the states related to the quaternion. From the state vector, (5.97), the quaternion related states are the quaternion itself, ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$, and its derivative, ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$. The QEKF should thus estimate the orientation and angular velocity of the robot in terms of ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$ and ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$. Thereby the QEKF falls in the category of an attitude estimator.

The QEKF should use the accelerometer and gyroscope measurements from one of the onboard IMUs, see Table C.1. The magnetometer could also have been used, [101] [102] [103], but due to large expected magnetic distortions in the environment in which the ballbot would be used, it would cause more trouble than dead reckoning the heading angle based on integration of the gyroscope angular velocity. However, since the ballbot includes two LiDARs which will be used for localization, a heading estimate can be extracted from the localized SLAM estimate. This heading estimate can be fed back into the QEKF at the available rate, hence lower than the rate at which the QEKF will be running, to improve the heading of the quaternion estimate.

The QEKF should run at the same rate as the balance controller being 200 Hz as defined in Chapter 7 and have a bandwidth at least 6 times faster than the controller bandwidth to limit the influence of the estimator dynamics in the controller [59].

The computed estimates, both ${}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}$ and ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$, should be filtered and smooth. Due to the quaternion derivative propagating directly through the equivalent control law, see Section 9.1, the ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$ estimate should at least be low-pass filtered to reduce high-frequency noise. It should thus be possible to adjust the smoothing on the ${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}$ estimate.

M.2 Prior work

A great overview of common IMU sensor fusion methods is given in [104]. Since we want to estimate the complete orientation and not just roll and/or pitch, the general complimentary filter [105], used by most hobbyists to fuse accelerometer and gyroscope measurements, is to no use.

One of the most commonly used 3D attitude/orientation estimators for IMU is the Madgwick filter by Sebastian Madgwick [101] [106]. Being a quaternion estimator it serves the purpose of estimating a singularity free orientation. The filter exists in two variants, one using only accelerometer and gyroscope (IMU) and another using the magnetometer as well (MARG). The orientation is estimated through a gradient descent approach where the step size is varied by the angular velocity from the gyroscope and the minimization problem is defined from the alignment between the estimated and measured gravity vector from the accelerometer.

Madgwick is an efficient filter with a very few lines of code executed in each iteration. This makes the Madgwick filter optimal for smaller embedded systems where the large floating point matrix inversions involved in an Extended Kalman filter, might be unfeasible. Some have even combined the Madgwick filter with a Kalman filter, using the resulting quaternion estimate from the gradient descent as an input to a Kalman filter, which fuses this estimate with the angular velocity measured by the gyroscope [102] [107]. Initially the Madgwick filter was used on Kugle V0, but one of the main deficiencies of the filter is the lack of a properly filtered quaternion derivative estimate.

Even though an Extended Kalman filter is computationally more expensive than a gradient descent method, it allows the designer to model and control the dynamics of the estimates through model-based optimal filtering. Since the propagation of and general arithmetic with quaternions involves non-linear operations, the non-linear versions of the Kalman filter, e.g., the Extended Kalman filter or Unscented Kalman filter [108], has to be used [109].

When it comes to Kalman filtering two types of approaches exist: model-driven and sensor-driven. The model-driven approach is the classical where the dynamic model of the system is used to propagate the states in the Kalman filter utilizing the actuated input to the system. In the sensor-driven approach the dynamic model is replaced by a kinematic model, utilizing one or multiple sensor inputs to propagate the states. In both cases one or multiple sensors are also used to correct the estimate. In the case of attitude estimation with an accelerometer and gyroscope, common methods take the model-free approach by using the gyroscope to propagate the orientation estimate which is then corrected by the accelerometer.

On the Rezero ballbot from ETH, the students have developed a single sensor driven Extended Kalman Filter to estimate all necessary states for the system, being the position, velocity and orientation, based on their kinematic model [110]. By combining everything into one Kalman filter they can utilize the encoder readings to correct orientation and visa versa use the accelerometer measurements to propagate the velocity estimate. The orientation is estimated as a rotation matrix which is corrected using an error state model. Unfortunately they do not estimate the angular velocity of the robot and neither do they talk about computation time and implementation.

Extended Kalman filters for quaternion estimation is a greatly researched topic with generally two common ways of doing it, both taking the sensor-driven approach: additive filtering and multiplicative filtering [111] [112].

For the general Kalman filter the estimation error is assumed to be the linear difference between the estimate and the true variable:

$$\delta \boldsymbol{x} = \boldsymbol{x} - \hat{\boldsymbol{x}} \tag{M.1}$$

where the estimate is defined as the expectation of a stochastic variable or a vector of stochastic variables.

$$\hat{\boldsymbol{x}} = \mathbb{E}[\boldsymbol{x}] \tag{M.2}$$

In the general Kalman filter the update equation is linearly applied.

$$\boldsymbol{x}^{k|k} = \boldsymbol{x}^{k|k-1} + \boldsymbol{K}^k \left(\boldsymbol{z}^k - \boldsymbol{h}(\boldsymbol{x}^{k|k-1}) \right) \tag{M.3}$$

However, when dealing with quaternions, which are not closed under addition and subtraction, this way of computing the estimation error and the way of correcting the estimate, would lead to non-unit error quaternions. Thus to keep the quaternion estimate on the 4-dimensional unit sphere, a multiplicative error definition can be used.

$$\delta \boldsymbol{q} = \boldsymbol{q} \circ \hat{\boldsymbol{q}}^* \tag{M.4}$$

In a multiplicative Kalman filter an error state formulation is used where the internal states of the Kalman filter is actually error states. This requires the process model and measurement models, from which the Jacobians are derived for covariance propagation, to be derived as error models using the error state formulation. The previous state estimate is propagated based on the normal process model. The apriori error states, being the prediction, are, however, always set to zero, since the assumption is that the predicted states corresponds to the true states, thus with zero error. The aposteriori error states are thereby solely based on the correction step. After computing the correction the aposteriori error states are applied to the predicted state estimates in a multiplicative way.
Thus the difference between an additive and multiplicative Kalman filter is that the multiplicative filter estimates the error between the predicted state and the true state, while the additive filter estimates the true state. Examples of multiplicative Extended Kalman filters for quaternion estimation includes [113] [90][114][115].

Even though the results in [109] shows that lower estimation errors can be achieved with multiplicative filters, it is shown in [91] how a crude brute force normalization of the quaternion after additive correction, see (M.5), will ensure convergence.

$$\boldsymbol{q}_{+}^{k|k} = \frac{\boldsymbol{q}_{-}^{k|k}}{\left\| \boldsymbol{q}_{-}^{k|k} \right\|} \tag{M.5}$$

One of the issues with the Extended Kalman filter is the need of the Jacobians of both the model used for propagation and the sensor model. The Jacobians are needed to propagate the covariance, but deriving these Jacobians symbolically can sometimes be cumbersome, especially for the model-based approaches. In this case the Unscented Kalman filter can be used which propagates the covariance through a set of sigma-points [116] [117] [118].

Secondly if computational complexity is an issue, e.g., dealing with large matrix inversions, other alternatives such as the square-root Kalman filter or information filter [119], can be used.

For simplicity, the QEKF described in this appendix is designed as a augmented sensor-driven Extended Kalman filter using the brute force normalization approach described in [91]. In the sensor-driven implementations only the quaternion is estimated. The quaternion derivative can be computed from the angular velocity reading from the gyroscope but is not filtered in any way. Since smoothing of the quaternion derivative is required, the angular velocity is included as a state to estimate. This results in the filter becoming model-based rather than sensor-driven, but as we will see in Appendix M.6 the process model is far simpler than using the full dynamic model since it relies on the kinetic relationship of the quaternion derivative.

In the future it is worth to explore the accuracy and improvement of using a multiplicative implementation and also if other approaches can be used to improve the angular velocity estimate.

M.3 Sensor models

As the first step in the design of the QEKF, models of the three type of sensors are derived. The accelerometer and gyroscope readings come from the MEMS IMU and their model is inspired from [103] who does not account for additional error sources, such as cross-axis sensitivity, gyro g-sensitivity, nonlinearity and hysteresis as otherwise described in [120]. The heading input from the SLAM algorithm is assumed to be the heading angle, following the definition from (H.76).

Since the IMU is not located in the origin of the body and the orientation of the IMU, due to installation tolerances, might not be exactly aligned with the orientation of the body, the IMU frame, $\{IMU\}$, is introduced.

M.3.1 Accelerometer

An accelerometer measures accelerations in terms of the difference in forces acting on the casing of the IMU minus the forces acting on the internals of the IMU. Since the internals are only affected by the downwards pointing gravity vector, the accelerometer will measure a positive acceleration upwards when being at rest.

$$ {}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} = \boldsymbol{K}_{\text{acc}} \, {}^{\text{IMU}}_{\text{I}}\boldsymbol{R} \left({}^{\text{I}}\ddot{\boldsymbol{O}}_{\text{IMU}\leftarrow\text{I}} - {}^{\text{I}}\boldsymbol{g} \right) + {}^{\text{IMU}}\boldsymbol{b}_{\text{acc}} + {}^{\text{IMU}}\boldsymbol{v}_{\text{acc}} \tag{M.6} $$

where $\boldsymbol{K}_{\text{acc}}$ is a sensor-specific constant scale factor, ${}^{\text{IMU}}_{\text{I}}\boldsymbol{R}$ is the rotation matrix defining the orientation of the IMU in the inertial frame, ${}^{\text{I}}\ddot{\boldsymbol{O}}_{\text{IMU}\leftarrow\text{I}}$ is the acceleration of the origin of the IMU and thus also the casing, ${}^{\text{IMU}}\boldsymbol{b}_{\text{acc}}$ is a sensor-specific constant bias and ${}^{\text{IMU}}\boldsymbol{v}_{\text{acc}} \sim \mathcal{N}(\boldsymbol{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{acc}})$ is a Gaussian white noise variable, thus uncorrelated zero-mean white noise. The gravity vector, ${}^{\text{I}}\boldsymbol{g}$, is defined as a downward-pointing vector in the inertial frame according to

$$ {}^{\text{I}}\boldsymbol{g} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{M.7} $$

The IMU is rigidly attached to the body with a static transformation, ${}^{\text{IMU}}_{\text{B}}\boldsymbol{T}$, known as the extrinsincs. To use the above sensor model this extrinsics has to be known, e.g., from a calibration. The sensor model is simplified by assuming the accelerometer scale and bias to be pre-calibrated constants, see Appendix M.4, providing calibration compensated measurements. The static alignment rotation, ${}^{\text{IMU}}_{\text{B}}\boldsymbol{R}$, is also assumed to be pre-calibrated and compensated by applying the inverse rotation to all sampled measurements. Furthermore, the acceleration of the body is neglected since it would overly complicate the filter by needing the acceleration and thus the translational dynamics of the model.

$$ {}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} = -{}^{\text{B}}_{\text{I}}\boldsymbol{R} \, {}^{\text{I}}\boldsymbol{g} + {}^{\text{IMU}}\boldsymbol{v}_{\text{acc}} \tag{M.8} $$

M.3.2 Gyroscope

A gyroscope measures angular velocity but is similarly exposed to sensor scale and bias.

$$ {}^{\text{IMU}}\boldsymbol{z}_{\text{gyro}} = \boldsymbol{K}_{\text{gyro}} \, {}^{\text{IMU}}\boldsymbol{\omega}_{\text{IMU}\leftarrow\text{I}} + {}^{\text{IMU}}\boldsymbol{b}_{\text{gyro}} + {}^{\text{IMU}}\boldsymbol{v}_{\text{gyro}} \tag{M.9} $$

where $\boldsymbol{K}_{\text{gyro}}$ is a sensor-specific constant scale factor, ${}^{\text{IMU}}\boldsymbol{b}_{\text{gyro}}$ is a sensor-specific time-varying bias and ${}^{\text{IMU}}\boldsymbol{v}_{\text{gyro}} \sim \mathcal{N}(\boldsymbol{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{gyro}})$ is a Gaussian white noise variable, thus uncorrelated zero-mean white noise.

Note that in the context of MEMS sensors, the component of the angular velocity from the rotation of earth can be neglected as it will be in the same order as the sensor errors. This bias is therefore also neglected from the model.

The gyroscope scale is assumed to be pre-calibrated, Appendix M.4, but the gyroscope bias might develop over time and should thus not be assumed constant. The gyroscope bias is usually modelled as a random-walk vector random process with statistically independent components.

For a random-walk process the derivative of the variable is stochastic.

$$\dot{}^{\text{IMU}}\boldsymbol{b}_{\text{gyro}} = {}^{\text{IMU}}\boldsymbol{w}_{\text{bias}} \tag{M.10}$$

where ${}^{\text{IMU}}\boldsymbol{w}_{\text{bias}} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{bias}})$ is white Gaussian noise whose covariance, $\boldsymbol{\Sigma}_{\text{bias}}$, defines the rate of the random-walk process.

The gyroscope measurement model, including the static transform between the body frame and the IMU, thereby simplifies to

$${}^{\text{IMU}}\boldsymbol{z}_{\text{gyro}} = {}^{\text{IMU}}_{\text{B}}\boldsymbol{R}\,{}^{\text{B}}\boldsymbol{\omega}_{\text{B}\leftarrow\text{I}} + {}^{\text{IMU}}\boldsymbol{b}_{\text{gyro}} + {}^{\text{IMU}}\boldsymbol{v}_{\text{gyro}} \tag{M.11}$$

Similarly to the accelerometer the static alignment rotation, ${}^{\text{IMU}}_{\text{B}}\boldsymbol{R}$, is assumed to be pre-calibrated and compensated by applying the inverse rotation to all sampled measurements. The resulting simplified sensor model of the gyroscope is thus

$${}^{\text{IMU}}\boldsymbol{z}_{\text{gyro}} = {}^{\text{B}}\boldsymbol{\omega}_{\text{B}\leftarrow\text{I}} + {}^{\text{IMU}}\boldsymbol{b}_{\text{gyro}} + {}^{\text{IMU}}\boldsymbol{v}_{\text{gyro}} \tag{M.12}$$

M.3.3 Heading angle input

The SLAM algorithm to be used for localization will be able to provide a heading estimate. Heading is defined as the direction of the projection of the body x-axis onto the inertial xy-plane. With this definition the heading angle is equal to the yaw angle of an Euler ZYX sequence. Assuming the LiDARs used for SLAM to be calibrated, so that the SLAM estimate is in the same frame as the body, and that an estimated yaw angle, ψ_{LiDAR}, can be extracted, the measurement model is then given as

$$z_\psi = \psi_{\text{LiDAR}} + {}^{\text{IMU}}\boldsymbol{v}_\psi \tag{M.13}$$

where ${}^{\text{IMU}}\boldsymbol{v}_\psi \sim \mathcal{N}(0, \sigma_\psi)$ defines the white Gaussian noise on the heading estimate which should be related to the expected estimation error of the SLAM algorithm.

M.4 Sensor calibration

Calibrating the accelerometer and gyroscope scale and bias is a well researched topic which can be done either offline or online [120]. In this project sensor calibration is performed offline by calibrating the gyroscope bias and the orientation alignment between the body frame and the IMU frame, defining ${}^{\text{IMU}}_{\text{B}}\boldsymbol{R}$.

Initially the QEKF was developed using the MPU-9250 IMU as mentioned in Section 2.1. The MPU-9250 comes factory calibrated to a certain extent, but experience shows that especially the accelerometer scale was incorrect and varied greatly across several tested units. It would therefore be necessary to calibrate the MPU-9250 or implement a different filtering scheme including online calibration [120]. Due to prioritizations and lack of time it was decided to carry out all tests using the Xsens IMU which comes pre-calibrated. Scale and bias calibration are thus outside the scope of this thesis. Furthermore, the noise level of the Xsens IMU sensor outputs are almost 100 times lower on the gyroscope and 3-10 times lower on the accelerometer than the MPU-9250, see Appendix M.4.2.

M.4.1 Sensor alignment

The first part of the calibration involves calibrating the alignment between the body frame and IMU frame. Instead of dealing with estimation and/or calibration of the actual center of mass of the robot, calibration of the center of mass is combined with the IMU alignment calibration.

This will of course change the steady state angle of attack on the ball, but for small offsets the influence is assumed to be negligible. The balance point of the robot is found manually and the alignment calibration is carried out at this point.

The alignment calibration consists in computing the rotation matrix, ${}^{\text{IMU}}_{\text{B}}\boldsymbol{R}$, whose inverse can be used to rotate the current measured acceleration vector, ${}^{\text{IMU}}\boldsymbol{z}_{\text{acc}}$, into the inertial frame. When the robot is held in an orientation corresponding to the unit quaternion where ${}^{\text{B}}_{\text{I}}\boldsymbol{R} = \mathbf{I}_3$, the rotated accelerometer measurement should correspond to the gravity vector, ${}^{\text{I}}\boldsymbol{g}$. By averaging several accelerometer measurements while holding the robot still the Gaussian noise component is greatly reduced, and the alignment is then computed as an axis-angle rotation from the cross product, see (H.47).

$$\begin{aligned} \boldsymbol{a} &= \frac{{}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}}{\left\| {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} \right\|} \\ \boldsymbol{b} &= -\frac{{}^{\text{I}}\boldsymbol{g}}{\| {}^{\text{I}}\boldsymbol{g} \|} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \boldsymbol{v} &= \boldsymbol{a} \times \boldsymbol{b} \end{aligned} \tag{M.14}$$

where ${}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}$ is the averaged accelerometer measurement. Note that the cross product of two unit vectors computes the axis-angle rotation vector to rotate vector $\boldsymbol{a}$ into vector $\boldsymbol{b}$. The rotation axis, $\bar{\boldsymbol{v}}$, and rotation amount, θ, is defined as

$$\begin{aligned} \cos(\theta) &= \boldsymbol{a} \cdot \boldsymbol{b} \\ \sin(\theta) &= \|\boldsymbol{v}\| = \sqrt{\boldsymbol{v}^{\mathsf{T}}\boldsymbol{v}} \\ \theta &= \arcsin \|\boldsymbol{v}\| \\ \bar{\boldsymbol{v}} &= \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} \end{aligned} \tag{M.15}$$

Next the alignment rotation matrix is computed from the axis-angle vector using Rodrigues' rotation formula [26].

$$\begin{aligned} {}^{\text{B}}_{\text{IMU}}\boldsymbol{R} &= \mathbf{I}_3 + \sin(\theta)\Omega(\bar{\boldsymbol{v}}) + (1 - \cos(\theta))\left(\bar{\boldsymbol{v}}\bar{\boldsymbol{v}}^{\mathsf{T}} - \mathbf{I}_3\right) \\ &= \mathbf{I}_3 + \Omega(\boldsymbol{v}) + \Omega(\boldsymbol{v})\Omega(\boldsymbol{v})\frac{1 - \cos(\theta)}{\sin(\theta)^2} \\ &= \mathbf{I}_3 + \Omega(\boldsymbol{v}) + \Omega(\boldsymbol{v})\Omega(\boldsymbol{v})\frac{1}{1 + \cos(\theta)} \end{aligned} \tag{M.16}$$

This leads to a calibrated body frame whose z-axis is parallel to the inertial frame z-axis when the center of mass is right above the origin, hence at the unstable equilibrium. The calibration is implemented at ↗`Kugle-Embedded/KugleFirmware/Libraries/Devices/IMU/IMU.cpp#L172-L245` with the alignment rotation matrix computation at ↗`Kugle-Embedded/KugleFirmware/Libraries/Devices/IMU/IMU.cpp#L334-L411`.

M.4.2 Covariance determination

As a final step of the sensor calibration the noise covariance matrices, $\boldsymbol{\Sigma}_{\text{acc}}$ and $\boldsymbol{\Sigma}_{\text{gyro}}$, are determined while the bias random-walk covariance, $\boldsymbol{\Sigma}_{\text{bias}}$, is left as a tuning parameter. Accelerometer and gyroscope measurements are sampled at 200 Hz while the sensor is completely still, see Figure M.1.

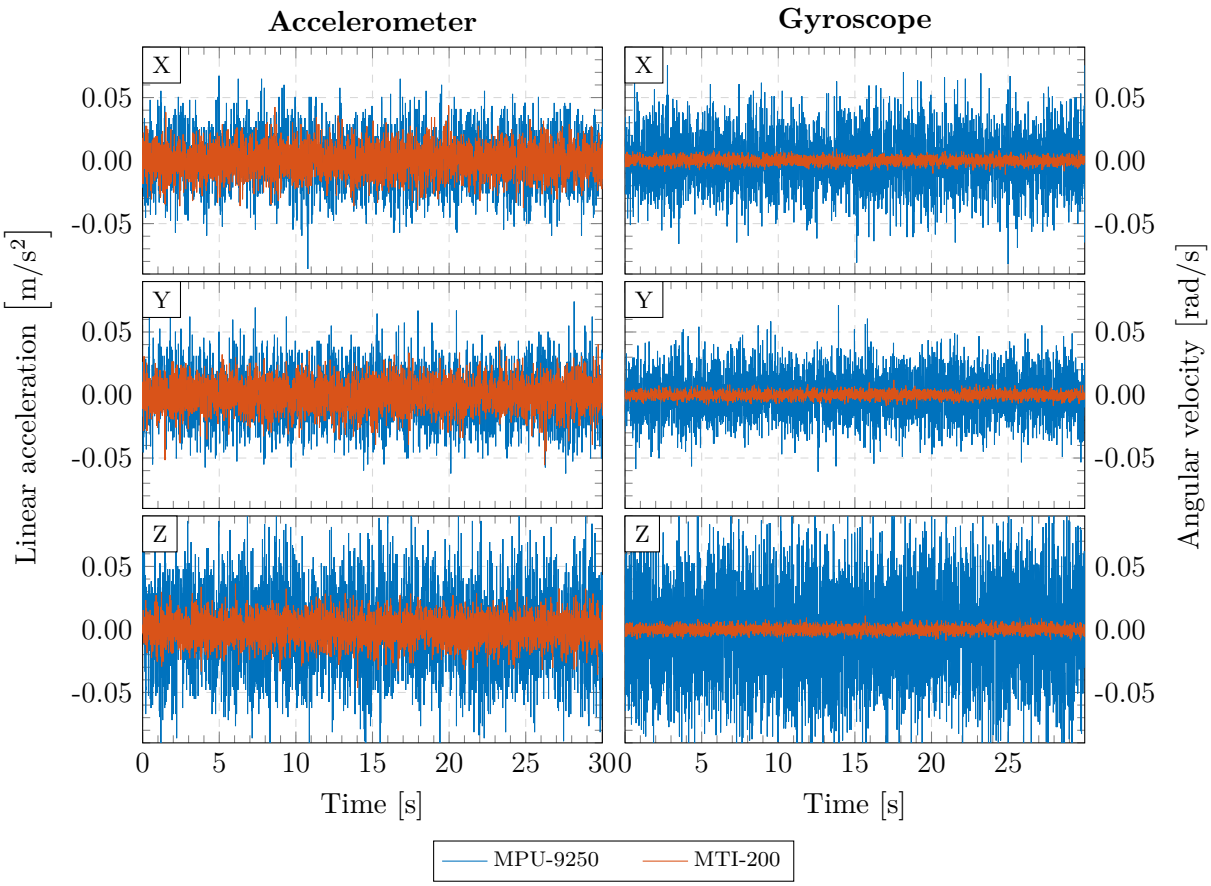


Figure M.1: Accelerometer and gyroscope samples from which the mean has been removed. Sampled at 200 Hz. Used for covariance determination.

The covariance of the MPU-9250 IMU at 200 Hz is computed to:

$$\mathbf{\Sigma}_{\text{acc,MPU}} = \begin{bmatrix} 0.3945 & 0.0006 & -0.0234 \\ 0.0006 & 0.3922 & -0.0036 \\ -0.0234 & -0.0036 & 0.9941 \end{bmatrix} \cdot 10^{-3} \tag{M.17}$$

$$\mathbf{\Sigma}_{\text{gyro,MPU}} = \begin{bmatrix} 0.4969 & 0.0201 & 0.0035 \\ 0.0201 & 0.1749 & -0.0260 \\ 0.0035 & -0.0260 & 1.3970 \end{bmatrix} \cdot 10^{-3} \tag{M.18}$$

The covariance of the MTI-200 IMU at 200 Hz is computed to:

$$\mathbf{\Sigma}_{\text{acc,MTI}} = \begin{bmatrix} 0.1432 & 0.0257 & 0.0135 \\ 0.0257 & 0.1321 & 0.0257 \\ 0.0135 & 0.0257 & 0.1417 \end{bmatrix} \cdot 10^{-3} \tag{M.19}$$

$$\mathbf{\Sigma}_{\text{gyro,MTI}} = \begin{bmatrix} 0.7678 & -0.0016 & 0.0061 \\ -0.0016 & 0.7446 & -0.0035 \\ 0.0061 & -0.0035 & 0.7952 \end{bmatrix} \cdot 10^{-5} \tag{M.20}$$

Note especially the much lower noisefloor of the gyroscope measurements from the MTI-200.

The MPU-9250 is configured in 4g accelerometer mode and 2000 °/s gyroscope mode. The internal low-pass filters are disabled and no sample rate divider is set, resulting in an internal sample rate of 8 kHz. The MTI Xsens IMU is configured to an internal sample rate of 400 Hz.

M.5 Estimator states

The QEKF should estimate both the quaternion and quaternion derivative, but due to the random-walk of the gyroscope bias, which can be observed by the other sensor inputs, it is desired to include an estimate of this bias. To reduce the state space and size of the Jacobians it is decided to estimate the body angular velocity rather than the actual quaternion derivative. The quaternion derivative is then computed from the quaternion and angular velocity estimate according to (H.49). The 10-dimensional state space estimated by the QEKF is thus:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}} \\ {}^{\mathrm{B}}\hat{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \\ {}^{\mathrm{IMU}}\hat{\boldsymbol{b}}_{\mathrm{gyro}} \end{bmatrix} \tag{M.21}$$

M.6 Process model

The Kalman filter is governed by a process model of the states. Since it is decided not to use the dynamic model of the ballbot, the process model is derived based on the kinematic relationships described in Appendix H.4.

M.6.1 Quaternion

The quaternion develops according to the kinematic relationship:

$${}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} = \frac{1}{2}\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = \frac{1}{2}\boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}\Big)\,{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{M.22}$$

Since the relationship is kinematic and thus exact, no process noise is included.

M.6.2 Angular velocity

To get the desired low-pass filtering of the angular velocity estimate, and since no dynamic model is included to provide any better information about the development of the angular velocity, a constant process model is assumed.

$${}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} = \mathbf{0}_{3\times1} + \boldsymbol{w}_{\omega} \tag{M.23}$$

where $\boldsymbol{w}_{\omega} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\omega})$ is a tunable noise term to reflect the fact that the angular velocity does indeed change due to dynamics. The tunable magnitude of the noise, $\boldsymbol{\Sigma}_{\omega}$, defines how hard the angular velocity estimate is filtered.

M.6.3 Gyroscope bias

Finally, the development of the gyroscope bias is governed by the random-walk process, which assumes a constant or slowly varying gyroscope bias.

$${}^{\mathrm{IMU}}\dot{\boldsymbol{b}}_{\mathrm{gyro}} = \mathbf{0}_{3\times1} + \boldsymbol{w}_{\mathrm{bias}} \tag{M.24}$$

where $\boldsymbol{w}_{\mathrm{bias}} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\mathrm{bias}})$ is similarly a tunable noise parameter whose magnitude, $\boldsymbol{\Sigma}_{\mathrm{bias}}$, defines the rate of the random-walk process, thus how fast the gyroscope bias might change.

M.6.4 Discretization

The QEKF is implemented as a discrete Extended Kalman filter and the process model thus has to be discretized. As described in Appendix H.5 there are several ways to perform numerical integration of the kinematic equations governing the quaternion. The three mentioned methods in Appendix H.5 have all been tested and yields similar performance, likely due to the fast sample rate of 200 Hz. For simplicity and since the estimated quaternion is normalized after correction, it is chosen to use the Forward Euler approach from (H.69).

$$ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] = \left(\mathbf{I}_4 + \frac{1}{2}\Delta t\, \boldsymbol{\Gamma}\Big({}^{\mathrm{B}}\widetilde{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]\Big)\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1] + \boldsymbol{w}_q[k-1] \tag{M.25} $$

where Δt denotes the sampling time and a tunable noise term, $\boldsymbol{w}_q \sim \mathcal{N}(\mathbf{0}_{4\times1}, \boldsymbol{\Sigma}_q)$, is included to reflect discretization errors.
The process model of the angular velocity and gyroscope bias is discretized such that equivalence of the noise strength is preserved.

$$ {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}[k] = {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1] + \boldsymbol{w}_{\omega_d}[k-1] \tag{M.26} $$

$$ {}^{\mathrm{IMU}}\boldsymbol{b}_{\mathrm{gyro}}[k] = {}^{\mathrm{IMU}}\boldsymbol{b}_{\mathrm{gyro}}[k-1] + \boldsymbol{w}_{\mathrm{bias}_d}[k-1] \tag{M.27} $$

The equivalence requires the covariance of the discretized noise terms to be scaled by the sampling rate such that $\boldsymbol{w}_{\omega_d} \sim \mathcal{N}(\mathbf{0}_{4\times1}, \Delta t\, \boldsymbol{\Sigma}_\omega)$ and $\boldsymbol{w}_{\mathrm{bias}_d} \sim \mathcal{N}(\mathbf{0}_{4\times1}, \Delta t\, \boldsymbol{\Sigma}_{\mathrm{bias}})$ [90].

The complete process model is defined by stacking each of the models:

$$ \boldsymbol{x}[k] = \begin{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k] \\ {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}[k] \\ {}^{\mathrm{IMU}}\boldsymbol{b}_{\mathrm{gyro}}[k] \end{bmatrix} \tag{M.28} $$

M.6.5 Jacobian

To propagate the estimation error covariance the Extended Kalman filter linearizes the process model by forming a Jacobian, which is then applied to propagate the covariance. Since the process noise are all additive, there is no need for computing the Jacobian with respect to the noise. The Jacobian of the process model with respect to the states is defined as:

$$ \boldsymbol{F} = \left.\frac{\partial \boldsymbol{x}[k]}{\partial \boldsymbol{x}[k-1]}\right|_{\substack{\boldsymbol{x}[k-1]=\hat{\boldsymbol{x}}^{k-1|k-1} \\ \boldsymbol{w}[k-1]=\mathbf{0}}} = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{qq}} & \boldsymbol{F}_{\boldsymbol{q\omega}} & \boldsymbol{F}_{\boldsymbol{qb}} \\ \boldsymbol{F}_{\boldsymbol{\omega q}} & \boldsymbol{F}_{\boldsymbol{\omega\omega}} & \boldsymbol{F}_{\boldsymbol{\omega b}} \\ \boldsymbol{F}_{\boldsymbol{bq}} & \boldsymbol{F}_{\boldsymbol{b\omega}} & \boldsymbol{F}_{\boldsymbol{bb}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{qq}} & \boldsymbol{F}_{\boldsymbol{q\omega}} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \mathbf{I}_3 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \end{bmatrix} \tag{M.29} $$

where the two sub-Jacobians are derived below.

$$ \boldsymbol{F}_{\boldsymbol{qq}} = \left.\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k-1]}\right|_{\substack{\boldsymbol{x}[k-1]=\hat{\boldsymbol{x}}^{k-1|k-1} \\ \boldsymbol{w}[k-1]=\mathbf{0}}} = \mathbf{I}_4 + \frac{1}{2}\Delta t\, \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{k-1|k-1}_{\mathrm{B}\leftarrow\mathrm{K}}\right) \tag{M.30} $$

The second sub-Jacobian is derived by swapping the quaternion operator as described in Appendix H.1.

$$ \boldsymbol{F}_{\boldsymbol{q\omega}} = \left.\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}[k]}{\partial\, {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}}[k-1]}\right|_{\substack{\boldsymbol{x}[k-1]=\hat{\boldsymbol{x}}^{k-1|k-1} \\ \boldsymbol{w}[k-1]=\mathbf{0}}} = \frac{1}{2}\Delta t\, \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k-1|k-1}\right)^{\wedge} \tag{M.31} $$

Note how all Jacobians are evaluated at the previous aposteriori state estimate and with no noise.

M.7 Measurement prediction

The correction step of a Kalman filter requires a measurement model to compute the predicted measurement. A measurement model describes the relationship between a predicted state estimate and the corresponding predicted measurement without noise. For the QEKF three measurement models are derived, one for each sensor, based on the sensor models defined in Appendix M.3.

M.7.1 Accelerometer

The accelerometer is assumed just to measure the gravity vector according to (M.8). Since it is decided to correct the quaternion additively and normalize afterwards, the norm of the innovation, being the difference between measurement and estimated measurement, will affect how much the quaternion estimate is pushed outside of the unit sphere before normalization. To reduce this to a known bound, the accelerometer measurements are normalized.

$$ {}^{\text{IMU}}\bar{\boldsymbol{z}}_{\text{acc}} = \frac{{}^{\text{IMU}}\boldsymbol{z}_{\text{acc}}}{\| {}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} \|} \tag{M.32} $$

Doing the same for the estimated measurement, ${}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}$, will bound the norm of the innovation to:

$$ \left\| {}^{\text{IMU}}\bar{\boldsymbol{z}}_{\text{acc}} - {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} \right\| \leq \left\| {}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} \right\| + \left\| {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} \right\| \leq 2 \tag{M.33} $$

Hopefully this will make the QEKF less susceptible to the unmodelled accelerations when the body is moving.

Normalizing (M.8) and replacing the gravity vector rotation with a quaternion vector rotation, see Appendix H.3, the measurement model of the normalized measurement is then given by

$$ \begin{aligned} {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} &= - \vee \left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{*} \circ \left(\wedge \frac{{}^{\text{I}}\boldsymbol{g}}{\| {}^{\text{I}}\boldsymbol{g} \|} \right) \circ {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} \right) \\ &= - \left(\vee \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} \right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} \right) \wedge \right) \frac{{}^{\text{I}}\boldsymbol{g}}{\| {}^{\text{I}}\boldsymbol{g} \|} \\ &= \left(\vee \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} \right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} \right) \wedge \right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned} \tag{M.34} $$

M.7.2 Gyroscope

The gyroscope measures the angular velocity of the IMU with respect to the inertial frame according to (M.12). The IMU is rigidly attached to the body and the alignment calibration ensures that the measurements are given in the body frame. Since the orientation of the inertial frame and ball frame, $\{K\}$, are coinciding, the predicted gyroscope measurement will be the estimated angular velocity plus the time varying bias.

$$ {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{gyro}} = {}^{\text{B}}\hat{\boldsymbol{\omega}}_{\text{B}\leftarrow\text{K}} + {}^{\text{IMU}}\hat{\boldsymbol{b}}_{\text{gyro}} \tag{M.35} $$

M.7.3 Heading angle

The heading angle was extracted from the quaternion in (H.76). The predicted heading angle measurement is hereby extracted using

$$ \hat{z}_{\psi} = \operatorname{atan2}\left({}^{\text{K}}\mathbf{i}_{\text{B},1},\ {}^{\text{K}}\mathbf{i}_{\text{B},0} \right) \tag{M.36} $$

where the x-axis vector is computed by

$$\begin{aligned} {}^{\mathrm{K}}\mathbf{i}_{\mathrm{B},0} &= {}^{\mathrm{K}}_{\mathrm{B}}q_0^2 + {}^{\mathrm{K}}_{\mathrm{B}}q_1^2 - {}^{\mathrm{K}}_{\mathrm{B}}q_2^2 - {}^{\mathrm{K}}_{\mathrm{B}}q_3^2 \\ {}^{\mathrm{K}}\mathbf{i}_{\mathrm{B},1} &= 2({}^{\mathrm{K}}_{\mathrm{B}}q_0\,{}^{\mathrm{K}}_{\mathrm{B}}q_3 + {}^{\mathrm{K}}_{\mathrm{B}}q_1\,{}^{\mathrm{K}}_{\mathrm{B}}q_2) \end{aligned} \tag{M.37}$$

M.7.4 Measurement vector

The heading angle will not be available at all sample instances, which is why the measurement model changes depending on whether the heading input from the SLAM algorithm is available or not.

When the heading angle is not available the predicted measurement vector includes only the accelerometer and gyroscope measurements.

$$\hat{\boldsymbol{z}}[k] = \begin{bmatrix} {}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{acc}}[k] \\ {}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{gyro}}[k] \end{bmatrix} \tag{M.38}$$

When the heading angle is available the predicted measurement vector is extended so that the heading can be incorporated and corrected.

$$\hat{\boldsymbol{z}}[k] = \begin{bmatrix} {}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{acc}}[k] \\ {}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{gyro}}[k] \\ \hat{z}_{\psi}[k] \end{bmatrix} \tag{M.39}$$

The same stacking is used when constructing the measurement vector, $\boldsymbol{z}$, from the sampled measurements.

M.7.5 Jacobian

To compute the measurement prediction covariance due to uncertainty in the state estimates, the Extended Kalman filter linearizes the measurement model by forming a Jacobian which is then applied to the estimation error covariance. Since the measurement model noise are all additive, there is no need for computing the Jacobian with respect to the noise. The Jacobian of the measurement model without heading angle is defined as

$$\boldsymbol{H} = \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial \hat{\boldsymbol{x}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{aq}} & \boldsymbol{H}_{\boldsymbol{a\omega}} & \boldsymbol{H}_{\boldsymbol{ab}} \\ \boldsymbol{H}_{\boldsymbol{gq}} & \boldsymbol{H}_{\boldsymbol{g\omega}} & \boldsymbol{H}_{\boldsymbol{gb}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{aq}} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{I}_3 \end{bmatrix} \tag{M.40}$$

When the heading angle is available the extended Jacobian become

$$\boldsymbol{H} = \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial \hat{\boldsymbol{x}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{aq}} & \boldsymbol{H}_{\boldsymbol{a\omega}} & \boldsymbol{H}_{\boldsymbol{ab}} \\ \boldsymbol{H}_{\boldsymbol{gq}} & \boldsymbol{H}_{\boldsymbol{g\omega}} & \boldsymbol{H}_{\boldsymbol{gb}} \\ \boldsymbol{H}_{\psi\boldsymbol{q}} & \boldsymbol{H}_{\psi\boldsymbol{\omega}} & \boldsymbol{H}_{\psi\boldsymbol{b}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{aq}} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{I}_3 \\ \boldsymbol{H}_{\psi\boldsymbol{q}} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{M.41}$$

The sub-Jacobians are derived below. The partial derivative of the accelerometer measurement with respect to the quaternion estimate is simplified using the relationship from (H.96).

$$\begin{aligned}\boldsymbol{H}_{\boldsymbol{aq}} = \left.\frac{\partial\, {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}[k]}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\boldsymbol{0}}} &= \vee\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right) + \vee\boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)\boldsymbol{\Gamma}\left(\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right)\boldsymbol{\Gamma}^{*}\\ &= 2\vee\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right)\end{aligned} \tag{M.42}$$

The second sub-Jacobian is computed by applying the chain rule. Note that the "evaluated at" has been left out on the right hand side for visual purpose only.

$$\begin{aligned}\boldsymbol{H}_{\psi\boldsymbol{q}} = \left.\frac{\partial \hat{z}_{\psi}[k]}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\boldsymbol{0}}} &= \frac{\partial \operatorname{atan2}\left({}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1},\ {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}\right)}{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}}\frac{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}} + \frac{\partial \operatorname{atan2}\left({}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1},\ {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}\right)}{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1}}\frac{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1}}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}}\\ &= \frac{\partial \operatorname{atan2}\left({}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1},\ {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}\right)}{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B,xy}}}\frac{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B,xy}}}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}}\end{aligned} \tag{M.43}$$

where the partial derivatives of atan2 is defined as

$$\begin{aligned}\frac{\partial \operatorname{atan2}(y,\, x)}{\partial x} &= -\frac{y}{x^2+y^2}\\ \frac{\partial \operatorname{atan2}(y,\, x)}{\partial y} &= \frac{x}{x^2+y^2}\end{aligned} \tag{M.44}$$

which are combined into a joint partial derivative vector for the atan2 part.

$$\left.\frac{\partial \operatorname{atan2}\left({}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1},\ {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}\right)}{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B,xy}}}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \frac{1}{\left({}^{\text{K}\hat{}}\mathbf{i}^{k|k-1}_{\text{B},0}\right)^2 + \left({}^{\text{K}\hat{}}\mathbf{i}^{k|k-1}_{\text{B},1}\right)^2}\begin{bmatrix}-{}^{\text{K}\hat{}}\mathbf{i}^{k|k-1}_{\text{B},1} & {}^{\text{K}\hat{}}\mathbf{i}^{k|k-1}_{\text{B},0}\end{bmatrix} \tag{M.45}$$

The partial derivative with respect to the quaternion of the x-axis vector defined in (M.37), is computed as

$$\left.\frac{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B,xy}}}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \left.\begin{bmatrix}\frac{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},0}}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}}\\ \frac{\partial\, {}^{\text{K}\hat{}}\mathbf{i}_{\text{B},1}}{\partial\, {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}}\end{bmatrix}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix}2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{0} & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{1} & -2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{2} & -2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{3}\\ 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{3} & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{2} & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{1} & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_{0}\end{bmatrix} \tag{M.46}$$

M.7.6 Observability

One way to analyze observability of an Extended Kalman filter is by using the observability matrix with the linearized Jacobians.

$$\mathbf{O} = \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{HF} \\ \boldsymbol{HFF} \\ \vdots \\ \boldsymbol{HF}^{n-1} \end{bmatrix} \tag{M.47}$$

Hereby observability can be analyzed at different operating points. When the heading input is not included the gyroscope bias in the inertial z-axis direction becomes unobservable. The consequence is that the heading part of the quaternion estimate (yaw angle) might drift over time when the ballbot is upright. However, thanks to the gyroscope, both the quaternion and angular velocity estimate is always observable.

M.8 Algorithm

The QEKF follows the general algorithm of an Extended Kalman filter summarized below, except for the last step where the quaternion estimate is normalized to keep it on the unit-sphere. The QEKF estimates the following state vector:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}} \\ {}^{\mathrm{B}}\hat{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}} \\ {}^{\mathrm{IMU}}\hat{\boldsymbol{b}}_{\mathrm{gyro}} \end{bmatrix} \tag{M.48}$$

The filter is initialized with an upright unit quaternion, zero angular velocity and zero bias. The estimation error covariance is initialized as a diagonal matrix with more trust put into the heading to reduce an initial jump in heading until the first heading angle input arrives.

$$\hat{\boldsymbol{x}}^{0|0} = \Big[\underbrace{1 \quad 0 \quad 0 \quad 0}_{{}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{\mathsf{T}}} \Big| \underbrace{0 \quad 0 \quad 0}_{{}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{\mathsf{T}}_{\mathrm{B}\leftarrow\mathrm{K}}} \Big| \underbrace{0 \quad 0 \quad 0}_{{}^{\mathrm{IMU}}\hat{\boldsymbol{b}}^{\mathsf{T}}_{\mathrm{gyro}}} \Big]^{\mathsf{T}} \tag{M.49}$$

$$\boldsymbol{P}^{0|0} = \mathrm{diag}(1\mathrm{e}-7 \cdot \mathbf{I}_3, \quad 1\mathrm{e}-9, \quad 1\mathrm{e}-7 \cdot \mathbf{I}_3, \quad 1\mathrm{e}-8 \cdot \mathbf{I}_3) \tag{M.50}$$

For each time step at a periodicity of $\Delta t = \frac{1}{200\ \mathrm{Hz}} = 5$ ms the following algorithm is executed.

1. **Prediction step**

   (a) Compute apriori estimates

$$ {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} = \left(\mathbf{I}_4 + \frac{1}{2}\Delta t\, \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{k-1|k-1}_{\mathrm{B}\leftarrow\mathrm{K}}\right)\right) {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{k-1|k-1} \tag{M.51}$$

$$ {}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{k|k-1}_{\mathrm{B}\leftarrow\mathrm{K}} = {}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{k-1|k-1}_{\mathrm{B}\leftarrow\mathrm{K}} \tag{M.52}$$

$$ {}^{\mathrm{IMU}}\boldsymbol{b}^{k|k-1}_{\mathrm{gyro}} = {}^{\mathrm{IMU}}\boldsymbol{b}^{k-1|k-1}_{\mathrm{gyro}} \tag{M.53}$$

(b) Compute process model partial derivatives

$$\boldsymbol{F}_{\boldsymbol{qq}} = \mathbf{I}_4 + \frac{1}{2}\Delta t\, \boldsymbol{\Gamma}\left({}^{\mathrm{B}}\hat{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}^{k-1|k-1} \right) \tag{M.54}$$

$$\boldsymbol{F}_{\boldsymbol{q\omega}} = \frac{1}{2}\Delta t\, \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k-1|k-1} \right) \wedge \tag{M.55}$$

(c) Construct process model Jacobian

$$\boldsymbol{F} = \left. \frac{\partial \boldsymbol{x}[k]}{\partial \boldsymbol{x}[k-1]} \right|_{\substack{\boldsymbol{x}[k-1]=\hat{\boldsymbol{x}}^{k-1|k-1} \\ \boldsymbol{w}[k-1]=\mathbf{0}}} = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{qq}} & \boldsymbol{F}_{\boldsymbol{q\omega}} & \mathbf{0}_{4\times 3} \\ \mathbf{0}_{3\times 4} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 4} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{bmatrix} \tag{M.56}$$

(d) Compute apriori estimation error covariance

$$\boldsymbol{P}^{k|k-1} = \boldsymbol{F}\boldsymbol{P}^{k-1|k-1}\boldsymbol{F}^{\mathsf{T}} + \boldsymbol{Q} \tag{M.57}$$

2. **Sample measurements** to time k

3. **Correction step**

(a) Normalize accelerometer measurement

$${}^{\mathrm{IMU}}\bar{\boldsymbol{z}}_{\mathrm{acc}} = \frac{{}^{\mathrm{IMU}}\boldsymbol{z}_{\mathrm{acc}}}{\| {}^{\mathrm{IMU}}\boldsymbol{z}_{\mathrm{acc}} \|} \tag{M.58}$$

(b) Construct measurement vector based on available measurements

- If heading from SLAM is not available:

$$\boldsymbol{z} = \begin{bmatrix} {}^{\mathrm{IMU}}\bar{\boldsymbol{z}}_{\mathrm{acc}} \\ {}^{\mathrm{IMU}}\boldsymbol{z}_{\mathrm{gyro}} \end{bmatrix} \tag{M.59}$$

- If heading from SLAM is available:

$$\boldsymbol{z} = \begin{bmatrix} {}^{\mathrm{IMU}}\bar{\boldsymbol{z}}_{\mathrm{acc}} \\ {}^{\mathrm{IMU}}\boldsymbol{z}_{\mathrm{gyro}} \\ z_\psi \end{bmatrix} \tag{M.60}$$

(c) Compute predicted measurements

$${}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{acc}} = \left(\vee\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right) \wedge \right) \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}} \tag{M.61}$$

$${}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{gyro}} = {}^{\mathrm{B}}\hat{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}^{k|k-1} + {}^{\mathrm{IMU}}\hat{\boldsymbol{b}}_{\mathrm{gyro}}^{k|k-1} \tag{M.62}$$

$$\hat{z}_\psi = \operatorname{atan2}\left(2\left({}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_0^{k|k-1}\, {}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_3^{k|k-1} + {}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_1^{k|k-1}\, {}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_2^{k|k-1} \right), \right. \tag{M.63}$$

$$\left. \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_0^{k|k-1} \right)^2 + \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_1^{k|k-1} \right)^2 - \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_2^{k|k-1} \right)^2 - \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{q}_3^{k|k-1} \right)^2 \right)$$

(d) Construct predicted measurement vector

- If heading from SLAM is not available:

$$\hat{\boldsymbol{z}} = \begin{bmatrix} {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} \\ {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{gyro}} \end{bmatrix} \tag{M.64}$$

- If heading from SLAM is available:

$$\hat{\boldsymbol{z}} = \begin{bmatrix} {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} \\ {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{gyro}} \\ \hat{z}_{\psi} \end{bmatrix} \tag{M.65}$$

(e) Compute measurement model partial derivatives

$$\boldsymbol{H}_{\boldsymbol{aq}} = 2 \vee \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}} \boldsymbol{\Phi} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \tag{M.66}$$

$$\boldsymbol{H}_{\psi\boldsymbol{q}} = \frac{\begin{bmatrix} -{}^{\text{K}}\hat{\mathbf{i}}^{k|k-1}_{\text{B},1} & {}^{\text{K}}\hat{\mathbf{i}}^{k|k-1}_{\text{B},0} \end{bmatrix}}{\left({}^{\text{K}}\hat{\mathbf{i}}^{k|k-1}_{\text{B},0}\right)^2 + \left({}^{\text{K}}\hat{\mathbf{i}}^{k|k-1}_{\text{B},1}\right)^2} \begin{bmatrix} 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_0 & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_1 & -2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_2 & -2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_3 \\ 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_3 & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_2 & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_1 & 2\,{}^{\text{K}}_{\text{B}}\hat{q}^{k|k-1}_0 \end{bmatrix} \tag{M.67}$$

(f) Construct measurement model Jacobian

- If heading from SLAM is not available:

$$\begin{aligned} \boldsymbol{H} &= \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial \hat{\boldsymbol{x}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} \\ &= \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{aq}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{I}_3 \end{bmatrix} \end{aligned} \tag{M.68}$$

- If heading from SLAM is available:

$$\begin{aligned} \boldsymbol{H} &= \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial \hat{\boldsymbol{x}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} \\ &= \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{aq}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{I}_3 \\ \boldsymbol{H}_{\psi\boldsymbol{q}} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix} \end{aligned} \tag{M.69}$$

(g) Compute predicted measurement covariance

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}^{k|k-1}\boldsymbol{H}^{\mathsf{T}} + \boldsymbol{R} \tag{M.70}$$

(h) Compute Kalman gain

$$\boldsymbol{K} = \boldsymbol{P}^{k|k-1}\boldsymbol{H}^{\mathsf{T}}\boldsymbol{S}^{-1} \tag{M.71}$$

(i) Compute aposteriori estimate

$$\hat{\boldsymbol{x}}^{k|k} = \hat{\boldsymbol{x}}^{k|k-1} + \boldsymbol{K}(\boldsymbol{z} - \hat{\boldsymbol{z}}) \tag{M.72}$$

(j) Compute aposteriori estimation error covariance

$$\boldsymbol{P}^{k|k} = (\mathbf{I}_{10} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{k|k-1} \tag{M.73}$$

(k) Quaternion normalization

$${}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k} = \frac{{}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k}}{\left\| {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k} \right\|} \tag{M.74}$$

4. **Estimate extraction**

   (a) Compute quaternion derivative estimate

$$ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}^{k|k} = \frac{1}{2}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{k|k}_{\mathrm{B}\leftarrow\mathrm{K}} \right) {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k} \tag{M.75} $$

The QEKF is implemented in the MATLAB file: Kugle-MATLAB/Estimators/QEKF/QEKF.m
The QEKF is implemented in C++ using MATLAB Embedded Coder® with a wrapper library written here: Kugle-Embedded/KugleFirmware/Libraries/Modules/Estimators/QEKF/QEKF.cpp

M.8.1 Covariances

The discrete covariance matrices are defined as

$$ \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{q}} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \boldsymbol{\Sigma}_{\omega_d} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{4\times3} & \boldsymbol{\Sigma}_{\mathrm{bias}_d} \end{bmatrix} \tag{M.76} $$

$$ \boldsymbol{R}_{\mathrm{IMU}} = \begin{bmatrix} \alpha_{\mathrm{trust}}\boldsymbol{\Sigma}_{\mathrm{acc}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{\Sigma}_{\mathrm{gyro}} \end{bmatrix} \qquad \boldsymbol{R}_{\mathrm{full}} = \begin{bmatrix} \alpha_{\mathrm{trust}}\boldsymbol{\Sigma}_{\mathrm{acc}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \boldsymbol{\Sigma}_{\mathrm{gyro}} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \sigma_{\psi} \end{bmatrix} \tag{M.77} $$

The process covariance of the kinematic relationship is set to zero, even though the model is discretized.

$$ \boldsymbol{\Sigma}_{\boldsymbol{q}} = \mathbf{0}_{4\times4} \tag{M.78} $$

The discrete angular velocity process covariance is set to a diagonal matrix with a tunable smoothing factor, σ_{ω}^2.

$$ \boldsymbol{\Sigma}_{\omega_d} = \sigma_{\omega}^2\,\mathbf{I}_3 \tag{M.79} $$

Similarly the gyroscope bias process covariance is set to a diagonal matrix with a tunable rate factor, σ_{bias}^2.

$$ \boldsymbol{\Sigma}_{\mathrm{bias}_d} = \sigma_{\mathrm{bias}}^2\,\mathbf{I}_3 \tag{M.80} $$

The accelerometer and gyroscope measurement covariances are both set to the estimated covariances from Appendix M.4.2.
Note that the discrete process covariances can be related to their continuous counterparts in order to preserve equivalence of the noise strength. This leads to the following relationships

$$ \boldsymbol{\Sigma}_{\omega_d} = \Delta t\,\boldsymbol{\Sigma}_{\omega} \qquad \boldsymbol{\Sigma}_{\mathrm{bias}_d} = \Delta t\,\boldsymbol{\Sigma}_{\mathrm{bias}} \tag{M.81} $$

Similarly the discrete measurement covariance can be related through

$$ \boldsymbol{\Sigma}_{\mathrm{acc}} = \frac{\boldsymbol{\Sigma}_{\mathrm{acc}_c}}{\Delta t} \qquad \boldsymbol{\Sigma}_{\mathrm{gyro}} = \frac{\boldsymbol{\Sigma}_{\mathrm{gyro}_c}}{\Delta t} \tag{M.82} $$

These relationships will have to be taken into account if the Kalman filter is tuned for one sample rate and one wants to reuse the parameters with another sample rate. Further explanation for conversion between continuous and discrete noise variance is provided in [108] on page 230-233.
The resulting aposteriori estimation error covariance is partitioned with the following identifiers:

$$ \boldsymbol{P}^{k|k} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{q}} & \boldsymbol{\Sigma}_{\boldsymbol{q\omega}} & \boldsymbol{\Sigma}_{\boldsymbol{qb}} \\ \boldsymbol{\Sigma}_{\boldsymbol{q\omega}} & \boldsymbol{\Sigma}_{\boldsymbol{\omega}} & \boldsymbol{\Sigma}_{\boldsymbol{\omega b}} \\ \boldsymbol{\Sigma}_{\boldsymbol{qb}} & \boldsymbol{\Sigma}_{\boldsymbol{\omega b}} & \boldsymbol{\Sigma}_{\boldsymbol{b}} \end{bmatrix} \tag{M.83} $$

Furthermore, the estimation error covariance of the extracted quaternion derivative estimate can be computed through linearization. The Jacobian of the quaternion derivative with respect to the estimated states is computed from (M.75).

$$\boldsymbol{T} = \left.\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}[k]}{\partial \hat{\boldsymbol{x}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{T_q} & \boldsymbol{T_\omega} & \boldsymbol{0}_{4\times 3} \end{bmatrix} \tag{M.84}$$

where the sub-Jacobians are computed as

$$\boldsymbol{T_q} = \left.\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \frac{1}{2}\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\hat{\boldsymbol{\omega}}^{k|k}_{\mathrm{B}\leftarrow\mathrm{K}}\right) \tag{M.85}$$

$$\boldsymbol{T_\omega} = \left.\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}[k]}{\partial\, {}^{\mathrm{B}}\hat{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \frac{1}{2}\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k}\right) \tag{M.86}$$

The estimation error covariance of the quaternion derivative is then computed by applying the Jacobian.

$$\boldsymbol{\Sigma}_{\dot{\boldsymbol{q}}} = \boldsymbol{T}\,\boldsymbol{P}^{k|k}\,\boldsymbol{T}^{\mathsf{T}} \tag{M.87}$$

M.9 Test & results

As mentioned in Appendix M.4 the MPU-9250 IMU was originally used during the development of the QEKF. However, due to time constraints and lack of proper calibration, all tests and results in this thesis, including the following QEKF tests, are based on accelerometer and gyroscope measurements from the MTI-200. Note however that both MPU-9250 and MTI-200 measurements are logged simultaneously during all tests.

The tunable covariance parameters have been tuned to

$$\sigma_\omega^2 = 3.16228e-07 \tag{M.88}$$

$$\sigma_{\text{bias}}^2 = 10^{-10} \tag{M.89}$$

$$\sigma_\psi^2 = 3\,3846\text{e}-05 \tag{M.90}$$

The heading noise, σ_ψ^2, is set to this value since it corresponds to a 3σ standard deviation of 1°.

M.9.1 Performance evaluation

The QEKF is initially tested in the expected operating region by manually tilting the robot in the following sequence:

1. Tilt roll to anywhere between $-10°$ to $-30°$
2. Tilt roll to anywhere between $10°$ to $30°$
3. Tilt pitch to anywhere between $-10°$ to $-30°$
4. Tilt pitch to anywhere between $10°$ to $30°$
5. Tilt forward right, thus with roll and pitch positive
6. Tilt forward left, thus with roll negative and pitch positive
7. Turn yaw $90°$ (counter-clockwise) while being upright

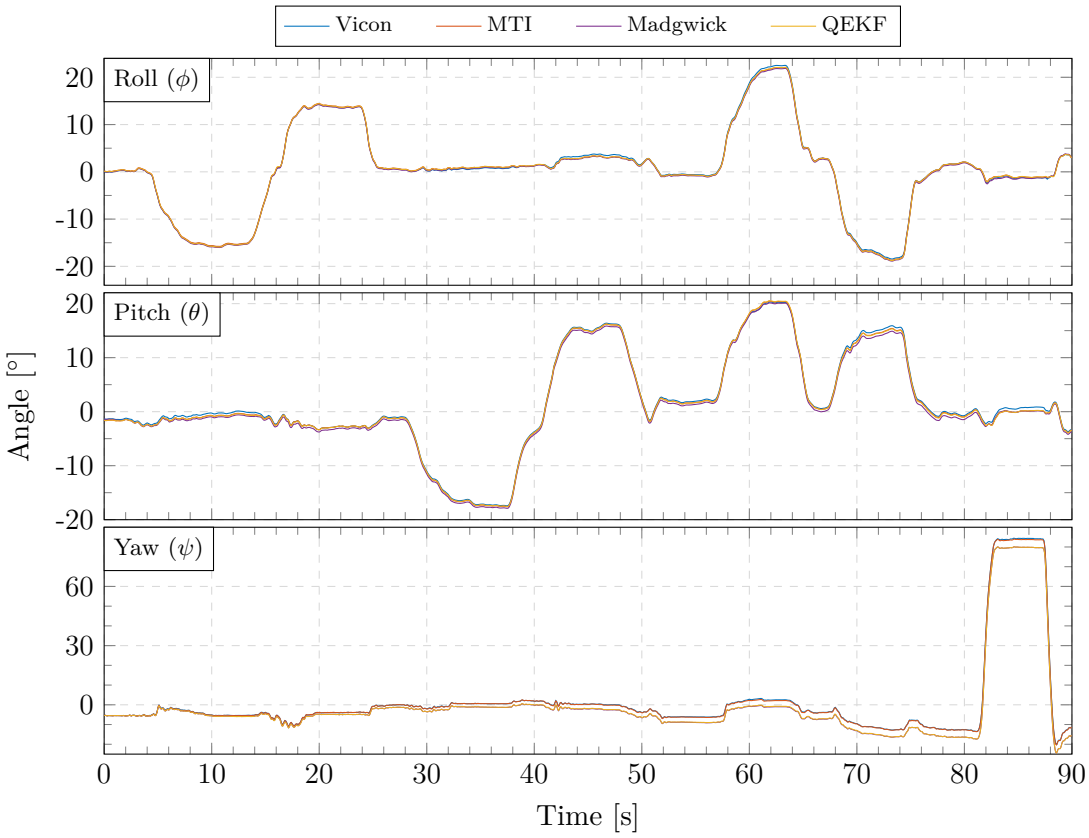


Figure M.2: Comparison of Euler angles during tilt test using the QEKF and Madgwick filter to estimate orientation

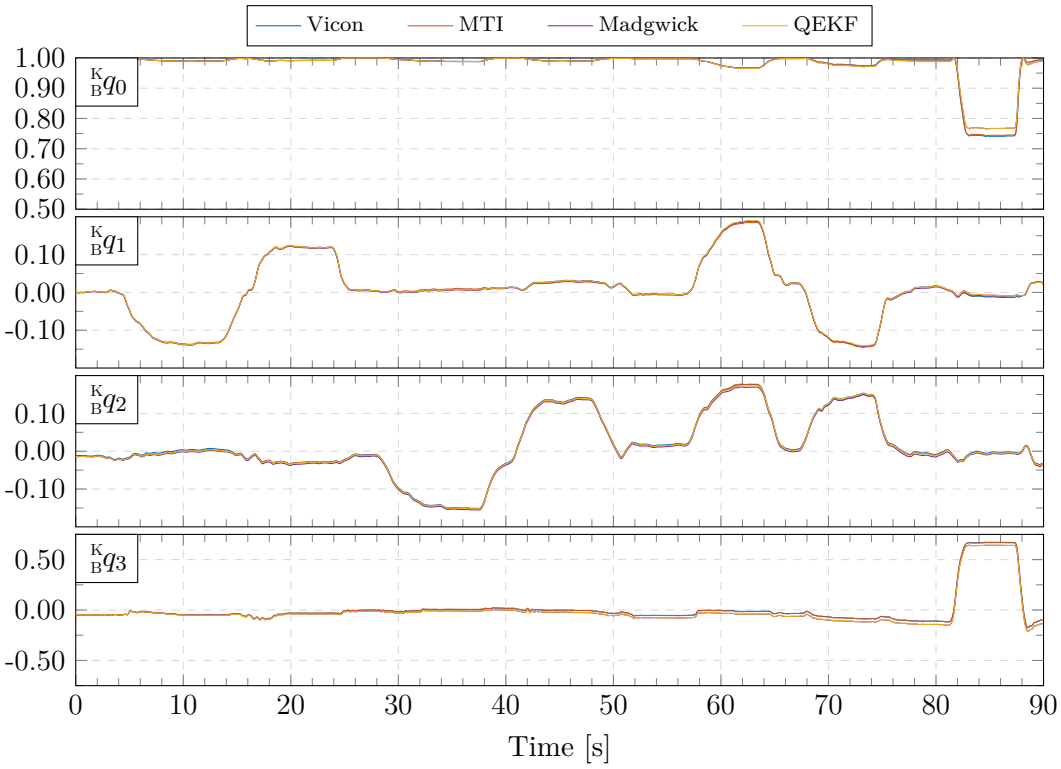


Figure M.3: Comparison of quaternion during tilt test using the QEKF and Madgwick filter to estimate orientation.

The measurements have been post-processed with the Madgwick filter [106] with the tuning factor set to $\beta = 0.001$. To give an intuition of the relationships between the quaternion, quaternion derivative, Euler angles and angular velocities, the results are shown in Figure M.3, Figure M.7, Figure M.2 and Figure M.6 respectively. The MTI output is the state estimates computed internally in the MTI IMU using their proprietary algorithm. Besides the Vicon measurements this estimate is deemed close to ground-truth.

Note how the quaternion elements in Figure M.3 seem directly related to roll, pitch and yaw in Figure M.2, but this is only because the angles are kept small. Both the Madgwick and QEKF yaw estimate drifts approximately 4° over the time span of the test of 90 seconds. Since no heading input is provided in this test, yaw drift is expected. The MTI-200 includes an internal algorithm, the VRU, which both estimates the orientation and also reduces heading drift by using the internal magnetometer.

Zooming in on the section where the robot is tilted forward right, see Figure M.4, shows how small the estimation error is.

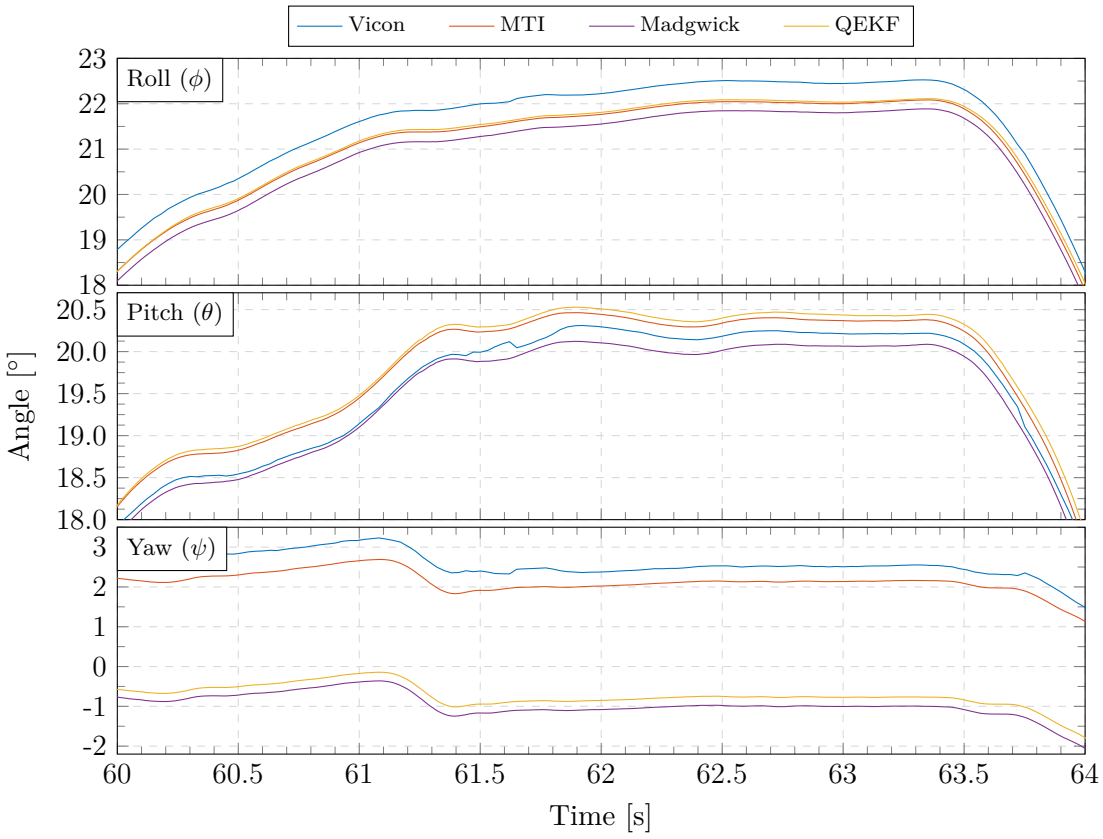


Figure M.4: Zoomed plot of Euler angle comparison during tilt test

The Euler angle estimation errors when compared to either the Vicon measurements or the MTI estimates, are shown in Figure M.5. Note how the estimation error on yaw drifts since the heading input is not used.

The quaternion derivative estimates and the body angular velocity estimates computed from this, are shown in Figure M.6 and Figure M.7. Note that the angular velocity estimate of the MTI-200 IMU is the same as the provided calibrated gyroscope readings.

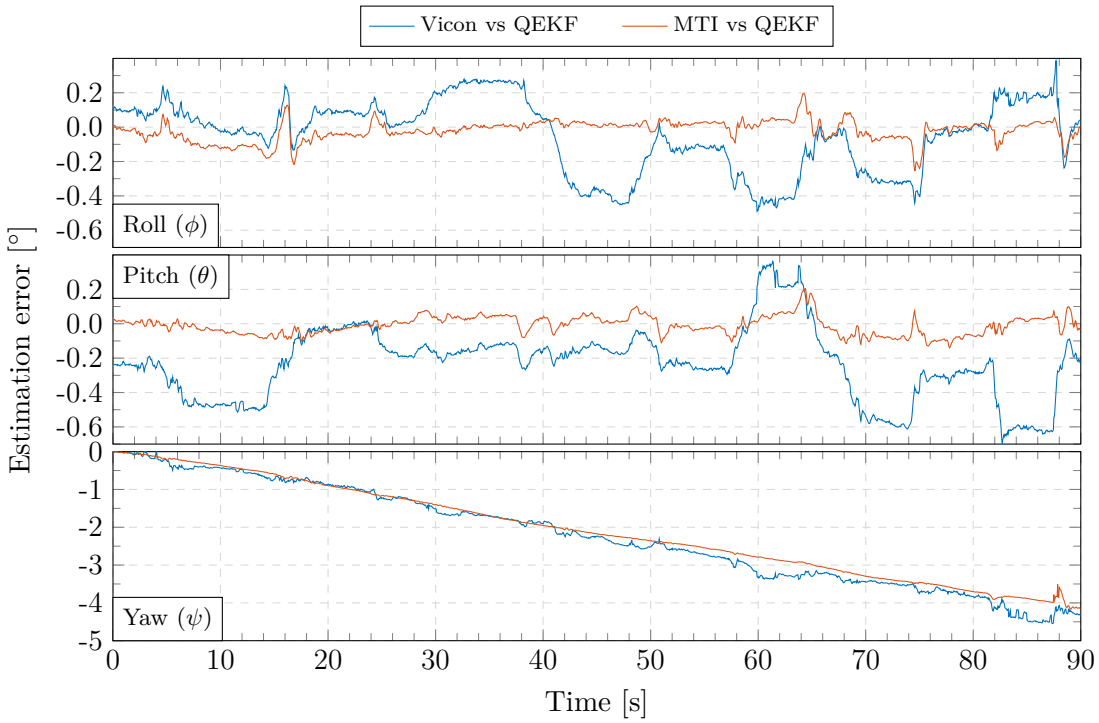


Figure M.5: Estimation error during tilt test

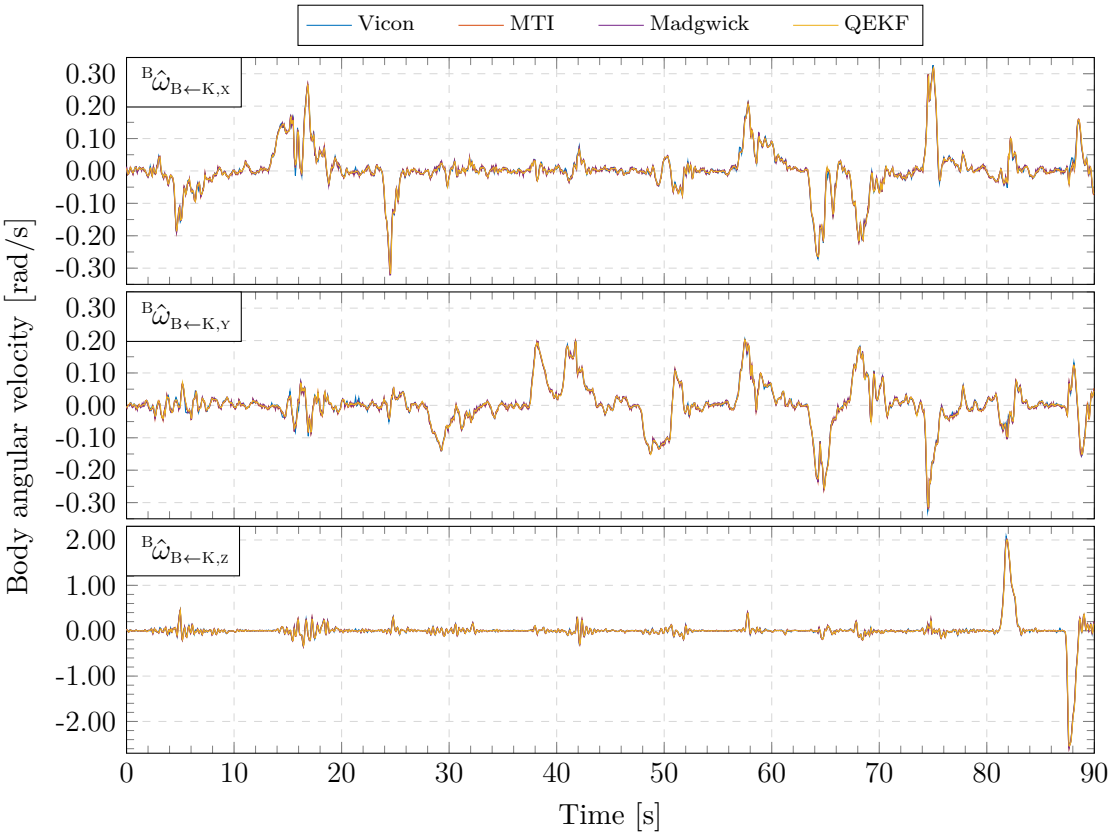


Figure M.6: Comparison of angular velocity during tilt test using the QEKF and Madgwick filter. Note that the QEKF estimate is more smooth than the Madgwick estimate.

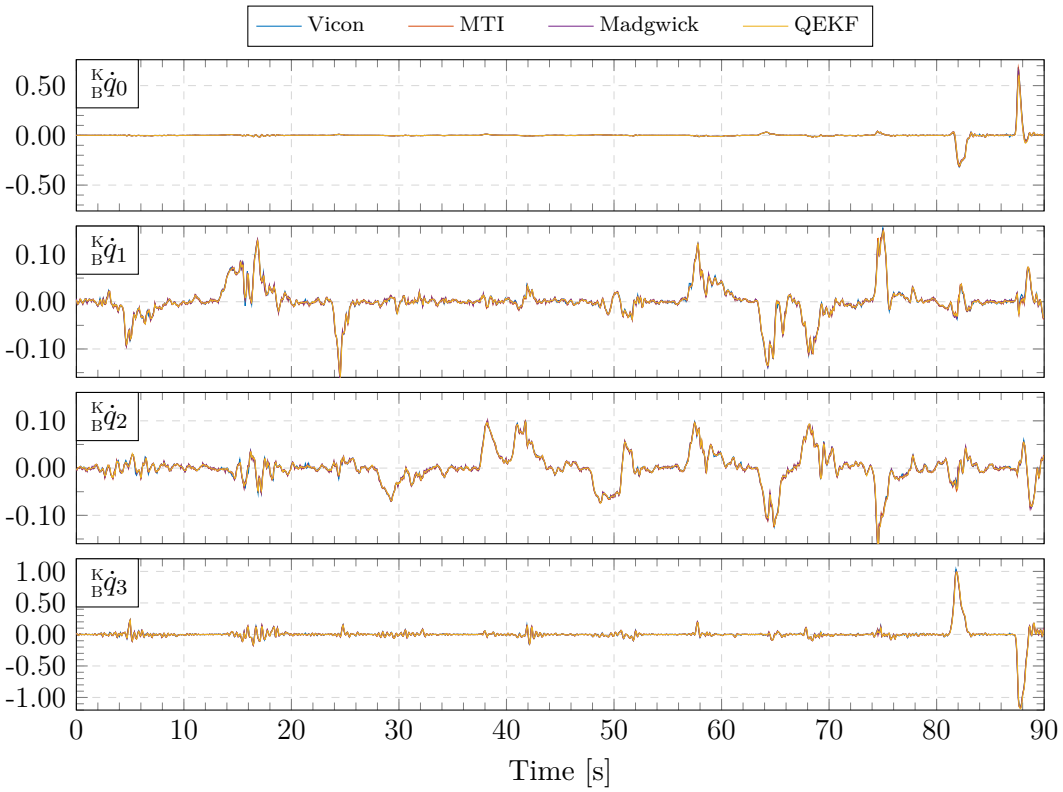


Figure M.7: Comparison of quaternion derivative during tilt test using the QEKF and Madgwick filter. Note that the QEKF estimate is more smooth than the Madgwick estimate.

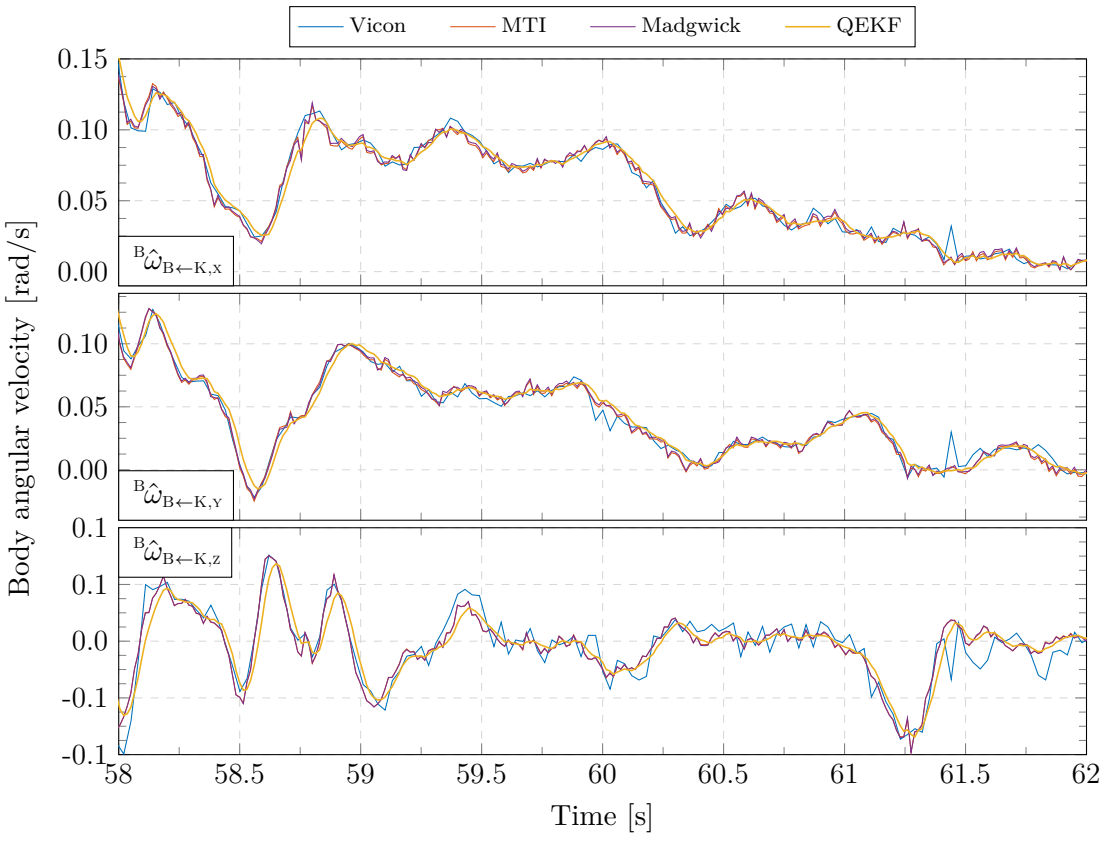


Figure M.8: Zoomed plot of angular velocity comparison during tilt test. Note how the QEKF estimate lags slightly behind but is smooth.

Zooming in on the angular velocity for the other forward tilt, see Figure M.8, reveals how the QEKF estimate lags slightly behind the measurements, but is far smoother. Furthermore, the difference to the Madgwick filter is especially noticeable on the quaternion derivative and angular velocity since the noise on the gyroscope measurements propagates straight through the Madgwick filter

M.9.2 Bias estimation

Bias estimation is especially valuable when using the MPU-9250. As the only test, the bias estimation test is carried out with the MPU-9250 since the in-run bias stability of the MTI-200 IMU is so high that the bias estimate is almost zero at all time. As an example the evolution of the estimated y-axis gyroscope bias during the previous tilt test is shown in Figure M.9.

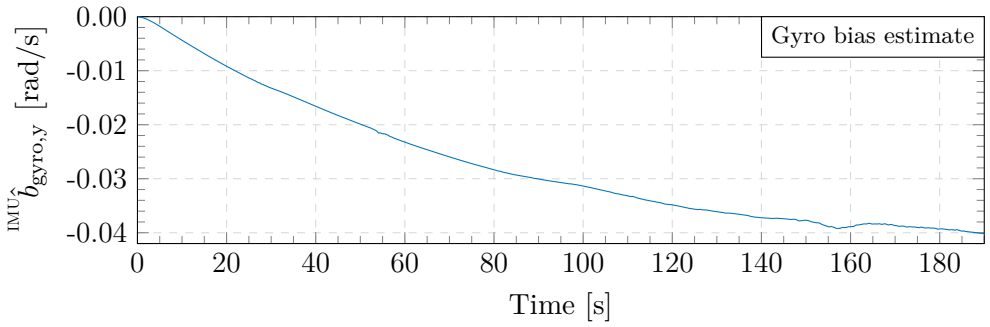


Figure M.9: Example of y-axis gyroscope bias estimation during a test with mixed movements

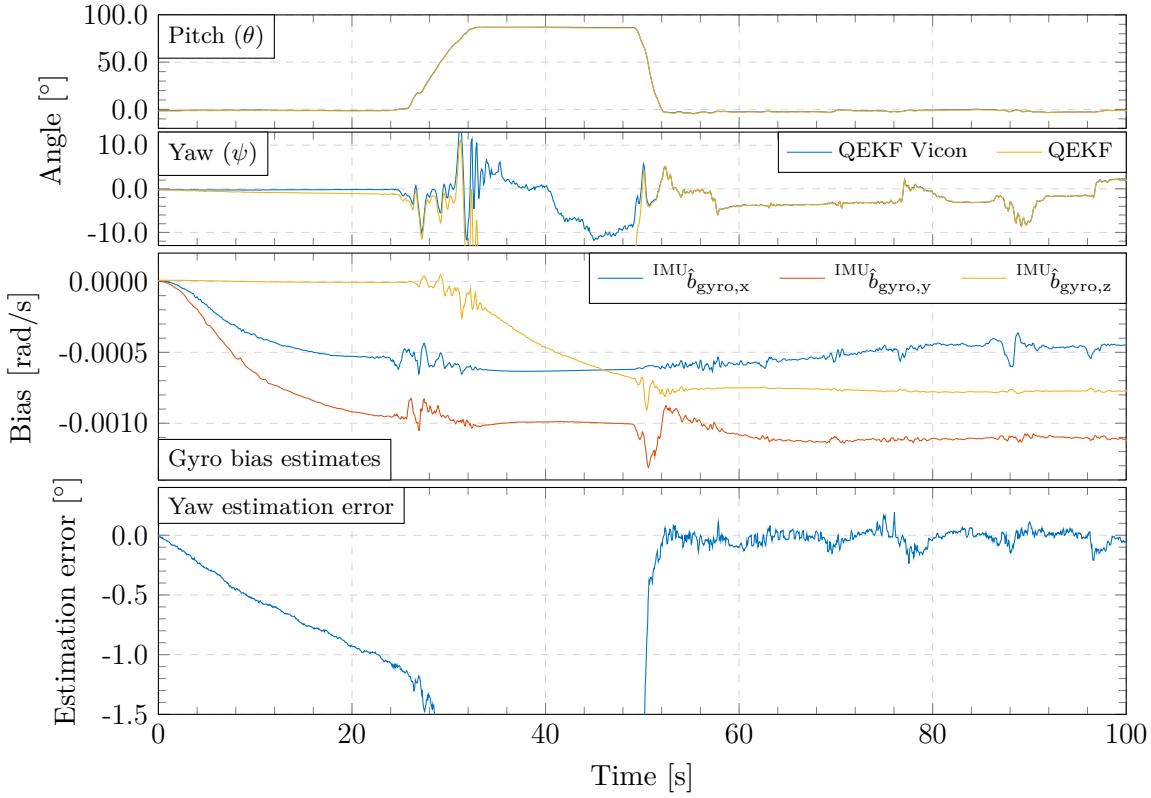


Figure M.10: Z-axis gyroscope bias estimation. Tilting the sensor makes the z-axis bias observable. In this case the robot is tilted 90° on pitch.

The drift of the yaw angle is mainly due to the unobservable z-axis gyroscope bias. However, if the ballbot is laid down, as shown in Figure M.10, the z-axis is made observable for a short while, allowing the QEKF bias estimate for the z-axis to settle.

M.9.3 Heading input

Alternatively a heading input can be used to stabilize the heading and the z-axis gyroscope bias. To test the heading input functionality the Vicon yaw angle is fed into the QEKF at a rate of 20 Hz by rerunning the QEKF in a post-processing loop using the recorded measurements from the previous test. The Vicon yaw angle is only provided during the first 40 seconds, whereafter the QEKF is only using the IMU measurements.

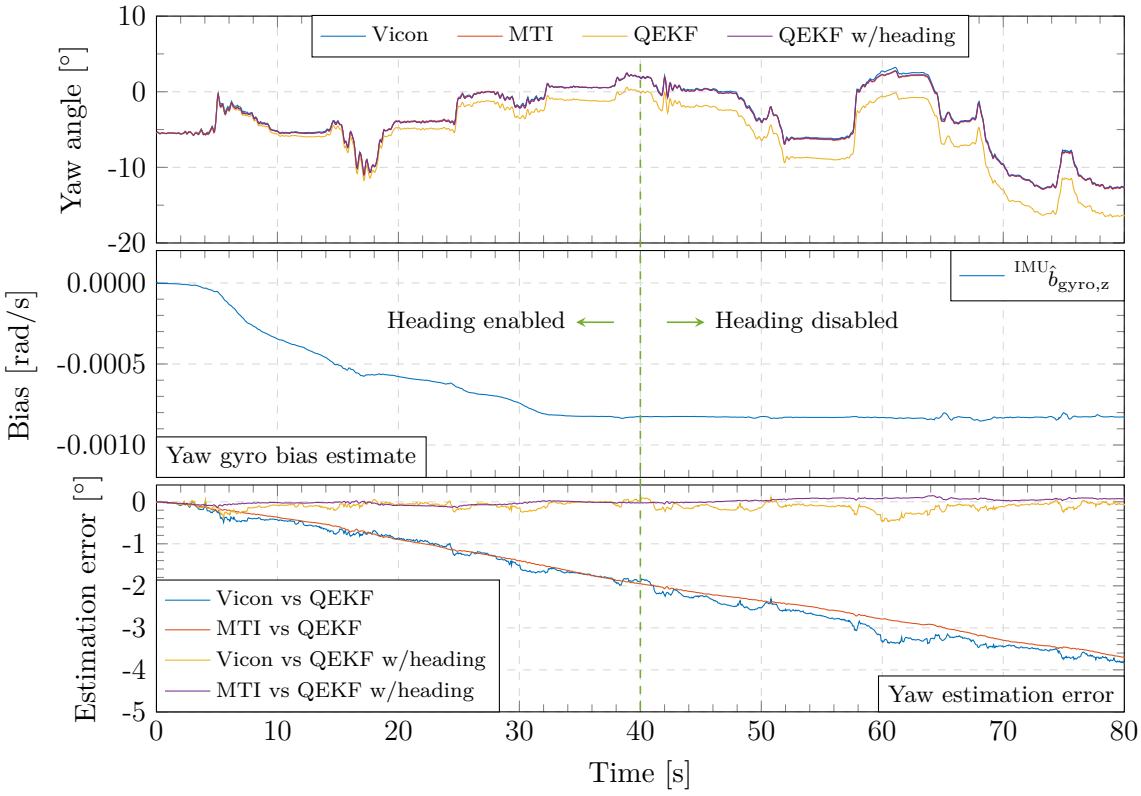


Figure M.11: Comparison of yaw estimate when using the heading input. Heading input is included for the first 40 s. Yaw gyro bias is only estimated when heading input is included.

Not only do the heading input help to correct the yaw angle estimate it also enables estimation of the yaw gyroscope bias. The improved gyroscope bias estimate results in smaller drift when the heading input is disabled after 40 s, keeping the estimation error close to zero.

M.9.4 Full orientation test

A full orientation test is carried out to ensure that the quaternion estimator works across the Euler angle singularities, being the reason for using and estimating quaternions. The ballbot is lifted up in the air and rotated 360° around an arbitrary axis, resulting in the robot pointing upside down about halfway.

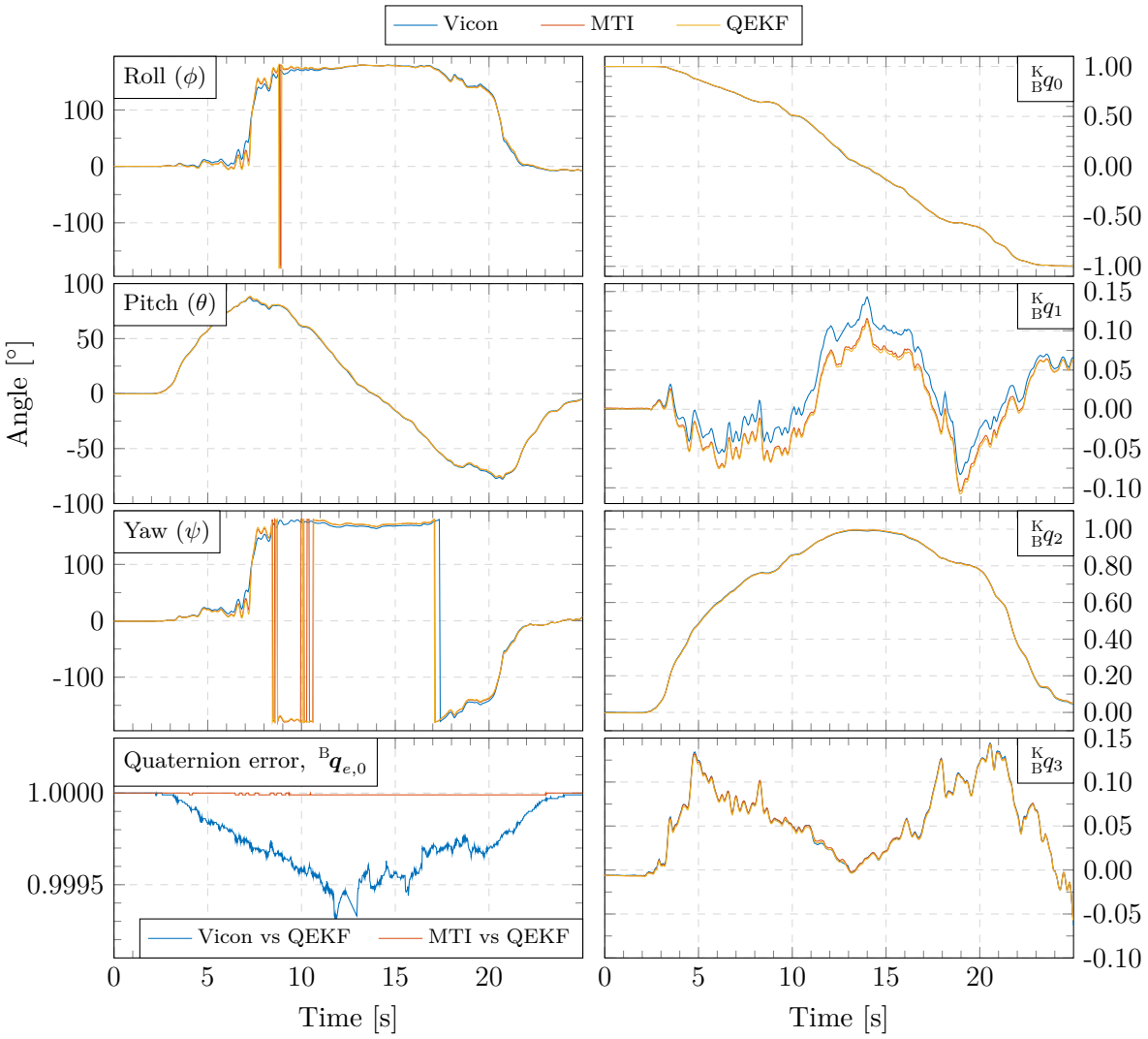


Figure M.12: Full orientation test where the ballbot is lifted in the air and rotated 360° to get upside down as well. The estimated quaternion is shown on the right with the corresponding Euler angles show on the left.

Note the singularity being apparent on both the roll and yaw Euler angles on the left, while the quaternion estimate on the right is smooth and thus singularity free and matches the Vicon measurements fairly well.

M.9.5 Performance with controller

Finally, the QEKF is tested with the sliding mode balance controller running, so that the performance can be evaluated including motor vibrations affecting the accelerometer measurements.

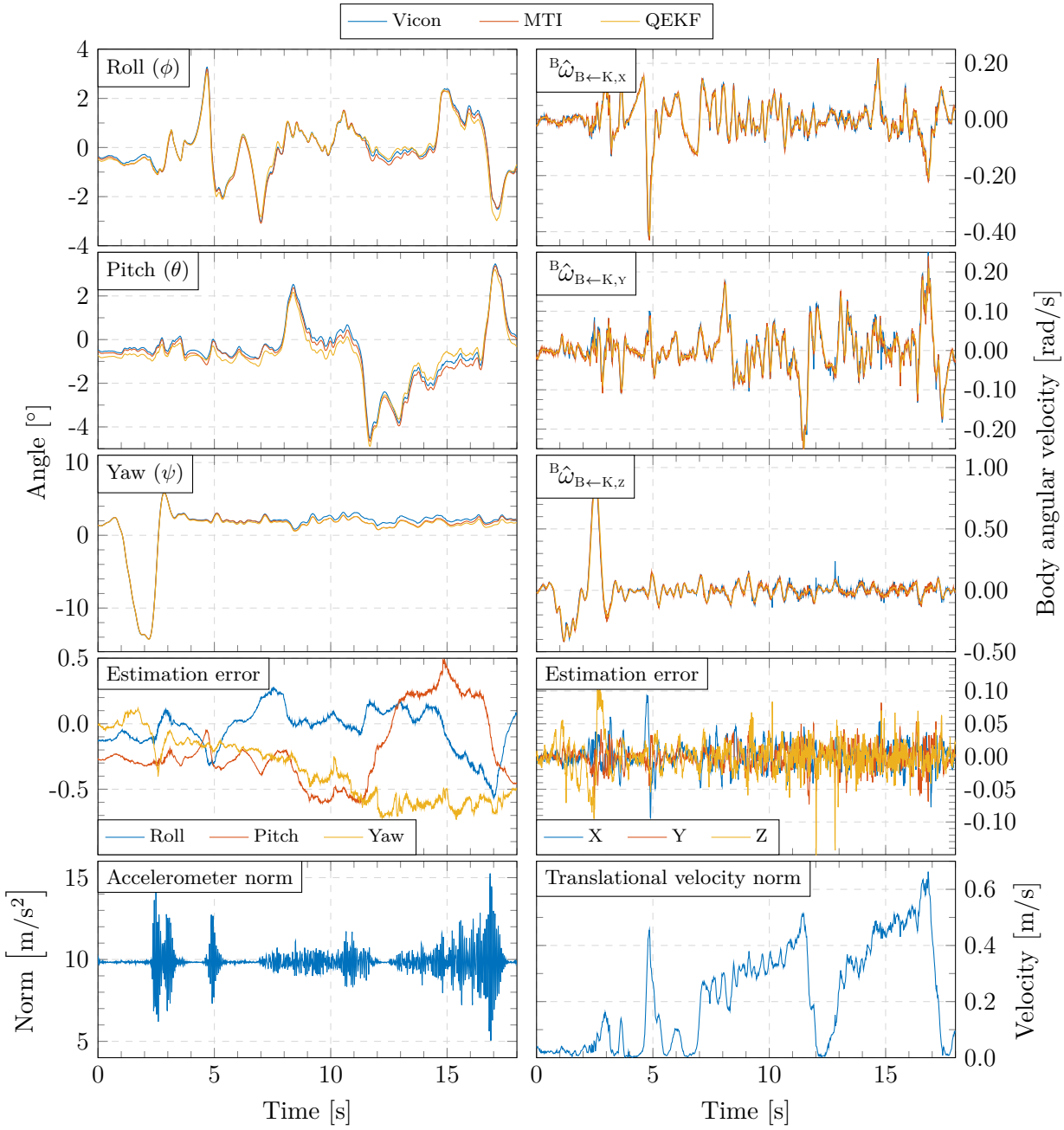


Figure M.13: Quaternion estimation test with the balance controller running causing small vibrations due to the switching torque. Estimation errors are based on the QEKF estimates and the Vicon measurements.

Even with the controller and motors running the estimation errors are kept below ±0.5°.

This concludes the design and test of the QEKF.

N Velocity EKF

In this appendix the Extended Kalman filter used for velocity state estimation is designed and tested. A summary of the algorithm is found in Appendix N.7. The MATLAB code for the VEKF can be found at: Kugle-MATLAB/Estimators/VEKF. The C++ implementation of the VEKF, generated by using MATLAB Embedded Coder®, can be found at: Kugle-Embedded/KugleFirmware/Libraries/Modules/Estimators/VelocityEKF.

N.1 Objective

The objective of the Velocity Extended Kalman Filter, in the latter denoted VEKF, is to estimate the inertial frame velocity, ${}^{\text{I}}\dot{x}$ and ${}^{\text{I}}\dot{y}$, of the ballbot given the estimated orientation from the QEKF, see Appendix M.

With the kinematics from Chapter 4 it is possible compute the inertial frame velocity from the motor angular velocities. Using the encoders, incremental changes in the angle of the motor shafts can be detected and the motor angular velocities can thus be computed numerically. This will, however, result in noisy velocity estimates as shown in Figure 6.1 and Figure 6.2.

The VEKF should estimate the inertial frame velocity, ${}^{\text{I}}\dot{x}$ and ${}^{\text{I}}\dot{y}$, by fusing the encoder angle measurements with acceleration readings from one of the onboard IMUs, see Table C.1. The velocity estimate should be smooth with a tunable low-pass filtering parameter.

The VEKF should run at the same rate as the balance controller being 200 Hz as defined in Chapter 7 and have a bandwidth at least 6 times faster than the controller bandwidth to limit the influence of the estimator dynamics in the controller [59].

N.2 Design considerations

A smooth velocity estimate could be computed by just low-pass filtering the velocity output of the kinematic model using the motor angular velocities approximated numerically by differentiating the encoder measurements. Low-pass filtering the velocity will, however, introduce lag in the estimate which is undesirable for control applications. Since the accelerometer provides measurements of the current acceleration of the IMU, although noisy, they could be used to reduce the lag in the low-pass filtered velocity estimates. This is where the Kalman filter comes into play.

A sensor-driven Kalman filter, similar to the one described in [110], would use the accelerometer measurements to propagate the velocity estimate and then correct the estimate with the encoder measurements. However, due to the amount of expected noise in both the accelerometer and encoder measurements it will be a difficult balance to get a smooth velocity estimate.

The VEKF is inspired from [110] but is instead designed as a model-based Kalman filter, see Appendix M.2, using a kinematic model to avoid the use of the complicated non-linear dynamic model. The approach is similar to the one used in Appendix M where all sensors are used to correct the estimate and the process model assumes the estimate to stay constant to give the desired tunable low-pass effect.

The VEKF could have been extended into a position estimator by including the localized position estimates from the SLAM algorithm which uses the LiDAR sensors. However, since the only purpose of the VEKF is to estimate the velocity, such an extension is deemed outside the scope of this project.

N.3 Sensor models

The velocity estimator relies on two sensor sources: the encoders and the accelerometer within one of the MEMS IMUs.

N.3.1 Encoders

On each of the motors a quadrature encoder is installed, generating pulses for each incremental change in the angle of the motor shaft, θ_i, by a mechanically predefined amount. These pulses are accumulated by a hardware quadrature decoder inside the microprocessor, allowing the absolute encoder count, $\boldsymbol{i}_{\text{ticks}}$, to be read. Since the encoders are installed before the gearing the resolution of the reading is increased. The total number of measurable edges generated by the encoder per wheel revolution is $n_{\text{ticks}} = 70997.33$ as defined in Section 2.2. The accumulated reading is thus a quantized reading of the continuous angle of the motor shaft according to (N.1).

$$\boldsymbol{i}_{\text{ticks}} = \text{round}\left(\frac{n_{\text{ticks}}}{2\pi}\,\boldsymbol{\theta}\right) \tag{N.1}$$

where $\boldsymbol{\theta}$ is the vector of motor angles defined in (N.2).

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \tag{N.2}$$

The quantization is modelled by rounding half away from zero, see (N.3), so that the rounding is symmetric and independent of the sign. Note that the rounding is computed elementwise.

$$\text{round}(x) = \text{sgn}(x)\,\lfloor\,|x| + 0.5\,\rfloor \tag{N.3}$$

As described in Chapter 4 the kinematics of the ballbot is non-holonomic, which is why there is no unique mapping between the system states and the absolute angles of the motors shafts. A change in the encoder ticks over a time window, Δt, can, however, be used as a numerical approximation of the angular velocity of the motor shafts, which is related to the system states through the inverse kinematics in (4.35).

$$\begin{aligned} \Delta\boldsymbol{i}_{\text{ticks}} &= \text{round}\left(\frac{n_{\text{ticks}}}{2\pi}\,\boldsymbol{\theta}(t+\Delta t)\right) - \text{round}\left(\frac{n_{\text{ticks}}}{2\pi}\,\boldsymbol{\theta}(t)\right) \\ &\approx \frac{n_{\text{ticks}}}{2\pi}\left(\boldsymbol{\theta}(t+\Delta t) - \boldsymbol{\theta}(t)\right) = \frac{n_{\text{ticks}}}{2\pi}\Delta t\,\dot{\boldsymbol{\theta}} \end{aligned} \tag{N.4}$$

For high encoder resolutions, fast angular velocities or long time windows, the rounding will be negligible. However, since the time window is relatively short and the angular velocities are small due to the limited translational velocity, the encoder model is approximated by adding a noise term.

$$\boldsymbol{z}_{\text{enc}} = \Delta\boldsymbol{i}_{\text{ticks}} = \frac{n_{\text{ticks}}}{2\pi}\Delta t\,\dot{\boldsymbol{\theta}} + \boldsymbol{v}_{\text{enc}} \tag{N.5}$$

where the noise term is assumed to be Gaussian white noise, $\boldsymbol{v}_{\text{enc}} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{enc}})$ given the noise covariance, $\boldsymbol{\Sigma}_{\text{enc}}$, as a diagonal matrix with equal entries since the noise of each encoder is uncorrelated with the others.

$$\left| \left(\text{round}\left(\frac{n_{\text{ticks}}}{2\pi} \theta_i(t + \Delta t) \right) - \text{round}\left(\frac{n_{\text{ticks}}}{2\pi} \theta_i(t) \right) \right) - \frac{n_{\text{ticks}}}{2\pi} \Delta t \, \dot{\theta}_i \right| \leq 1 \tag{N.6}$$

The quantization error of (N.4) is bounded by 1, as shown in (N.6), since the largest quantization error occurs when the ticks are e.g., rounded up by 0.5 at time t and rounded down by 0.5 at time $t + \Delta t$. For Gaussian distributions 99.7% of all samples will be within $\pm 3\sigma$. The noise is thus approximated by using the quantization bound of 1 as the 3σ.

$$\begin{aligned} 3\,\sigma_{\text{enc}} = 1 \rightarrow \sigma_{\text{enc}} &= \frac{1}{3} \\ \boldsymbol{\Sigma}_{\text{enc}} = \sigma_{\text{enc}}^2 \, \mathbf{I}_3 &= \frac{1}{9} \mathbf{I}_3 \end{aligned} \tag{N.7}$$

N.3.2 Accelerometer

In Appendix M.3 a model of the accelerometer, based on inspiration from [103] and [120], was shown to be:

$${}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} = \boldsymbol{K}_{\text{acc}} \, {}^{\text{IMU}}_{\text{I}}\boldsymbol{R} \left({}^{\text{I}}\ddot{\boldsymbol{O}}_{\text{IMU}\leftarrow\text{I}} - {}^{\text{I}}\boldsymbol{g} \right) + {}^{\text{IMU}}\boldsymbol{b}_{\text{acc}} + {}^{\text{IMU}}\boldsymbol{v}_{\text{acc}} \tag{N.8}$$

The accelerometer model used in the QEKF, Appendix M.3.1, is greatly simplified and the accelerations due to motion of the IMU, ${}^{\text{I}}\ddot{\boldsymbol{O}}_{\text{IMU}\leftarrow\text{I}}$, are neglected. In the VEKF these accelerations can, however, be used to correct the velocity estimates. The sensor specific gain, $\boldsymbol{K}_{\text{acc}}$ and the alignment of the accelerometer in the body frame is assumed to be pre-calibrated and compensated as described in Appendix M.3.1. It is decided to keep the accelerometer bias since this can be estimated and corrected as part of the VEKF. Furthermore, the inclination of the robot is assumed to be small, so that the acceleration of the IMU in inertial frame will have a negligible z-component. By assuming the angular accelerations of the ballbot to be small, the measurable accelerations will be dominated by the linear accelerations of the ballbot, ${}^{\text{I}}\ddot{x}$ and ${}^{\text{I}}\ddot{x}$.

$${}^{\text{I}}\ddot{\boldsymbol{O}}_{\text{IMU}\leftarrow\text{I}} = \begin{bmatrix} {}^{\text{I}}\ddot{x} \\ {}^{\text{I}}\ddot{y} \\ 0 \end{bmatrix} \tag{N.9}$$

The accelerometer model thus reduces to

$${}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} = {}^{\text{B}}_{\text{I}}\boldsymbol{R} \begin{bmatrix} {}^{\text{I}}\ddot{x} \\ {}^{\text{I}}\ddot{y} \\ g \end{bmatrix} + {}^{\text{IMU}}\boldsymbol{b}_{\text{acc}} + {}^{\text{IMU}}\boldsymbol{v}_{\text{acc}} \tag{N.10}$$

where ${}^{\text{IMU}}\boldsymbol{v}_{\text{acc}} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_{\text{acc}})$ is a Gaussian white noise variable, modelling the uncorrelated sensor noise.

N.4 Estimator states

The VEKF should estimate the inertial frame velocity of the ballbot, but since a low-pass filtering effect is desired and since corrections from the accelerometer should be used to reduce lag in the velocity estimates, acceleration estimates are added to the state vector.

Furthermore, the accelerometer bias should be estimated and is therefore added to the state vector as well. The 7-dimensional state space estimated by the VEKF is thus

$$\hat{\boldsymbol{x}} = \begin{bmatrix} {}^{\text{I}}\dot{\hat{x}} \\ {}^{\text{I}}\dot{\hat{y}} \\ {}^{\text{I}}\ddot{\hat{x}} \\ {}^{\text{I}}\ddot{\hat{y}} \\ {}^{\text{IMU}}\hat{\boldsymbol{b}}_{\text{acc}} \end{bmatrix} \tag{N.11}$$

N.5 Process model

The Kalman filter is governed by a process model of the states. Since it is decided not to use the dynamic model of the ballbot, the process model is derived based on the first-order relationship between the states.

N.5.1 Velocity

The velocity develops according to the exact first-order relationship with no process noise included using the derivatives present in the state vector.

N.5.2 Acceleration

Since no dynamic model is included to provide any better information about the development of the acceleration, a constant process model is assumed.

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\text{I}}\ddot{x} &= 0 + w_{ax} \\ \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\text{I}}\ddot{y} &= 0 + w_{ay} \end{aligned} \tag{N.12}$$

where $w_{ax} \sim \mathcal{N}(0, \sigma_a)$ and $w_{ay} \sim \mathcal{N}(0, \sigma_a)$ are a tunable noise terms to reflect the fact that the acceleration does indeed change due to the dynamics. The tunable magnitude of the noise, σ_a, defines how hard the acceleration estimates are filtered.

N.5.3 Accelerometer bias

Finally, the development of the accelerometer bias is governed by the random-walk process, which assumes a constant accelerometer bias.

$${}^{\text{IMU}}\dot{\boldsymbol{b}}_{\text{acc}} = \boldsymbol{0}_{3\times 1} + \boldsymbol{w}_{\text{bias}} \tag{N.13}$$

where $\boldsymbol{w}_{\text{bias}} \sim \mathcal{N}(\boldsymbol{0}_{3\times 1}, \boldsymbol{\Sigma}_{\text{bias}})$ is similarly a tunable noise parameter whose magnitude, $\boldsymbol{\Sigma}_{\text{bias}}$, defines how fast the accelerometer bias might change and thus how much variation to allow in the accelerometer bias.

N.5.4 Discretization

Similar to the QEKF, see Appendix M.6.4, the VEKF is implemented as a discrete Extended Kalman filter and thus has to be discretized.

The velocity states are propagated using the Forward Euler approximation.

$$
\begin{aligned}
{}^{\text{I}}\dot{x}[k] &= {}^{\text{I}}\dot{x}[k-1] + \Delta t\ {}^{\text{I}}\ddot{x}[k-1] + w_{vx}[k-1] && \text{(N.14)}\\
{}^{\text{I}}\dot{y}[k] &= {}^{\text{I}}\dot{y}[k-1] + \Delta t\ {}^{\text{I}}\ddot{y}[k-1] + w_{vy}[k-1] && \text{(N.15)}
\end{aligned}
$$

where Δt denotes the sampling time and the tunable noise terms, $w_{vx} \sim \mathcal{N}(0, \sigma_v)$ and $w_{vy} \sim \mathcal{N}(0, \sigma_v)$, are included to reflect discretization errors.

The process model of the acceleration and accelerometer bias is also discretized using Forward Euler, yielding

$$
\begin{aligned}
{}^{\text{I}}\ddot{x}[k] &= {}^{\text{I}}\ddot{x}[k-1] + \Delta t\, w_{ax}[k-1] && \text{(N.16)}\\
{}^{\text{I}}\ddot{y}[k] &= {}^{\text{I}}\ddot{y}[k-1] + \Delta t\, w_{ay}[k-1] && \text{(N.17)}\\
{}^{\text{IMU}}\boldsymbol{b}_{\text{acc}}[k] &= {}^{\text{IMU}}\boldsymbol{b}_{\text{acc}}[k-1] + \Delta t\, \boldsymbol{w}_{\text{bias}}[k-1] && \text{(N.18)}
\end{aligned}
$$

The complete process model is defined by stacking each of the models.

$$
\boldsymbol{x}[k] = \begin{bmatrix} {}^{\text{I}}\dot{x}[k] \\ {}^{\text{I}}\dot{y}[k] \\ {}^{\text{I}}\ddot{x}[k] \\ {}^{\text{I}}\ddot{y}[k] \\ {}^{\text{IMU}}\boldsymbol{b}_{\text{acc}}[k] \end{bmatrix} \tag{N.19}
$$

N.5.5 Jacobian

In the Extended Kalman filter the estimation error covariance is propagated through the process model by linearization. The Jacobian of the process model with respect to the state estimates is computed. Since the noise terms are all assumed to be additive there is no need for computing the Jacobian with respect to the noise terms.

$$
\boldsymbol{F} = \left.\frac{\partial \boldsymbol{x}[k]}{\partial \boldsymbol{x}[k-1]}\right|_{\substack{\boldsymbol{x}[k-1]=\hat{\boldsymbol{x}}^{k-1|k-1} \\ \boldsymbol{w}[k-1]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{F_{vv}} & \boldsymbol{F_{va}} & \boldsymbol{F_{vb}} \\ \boldsymbol{F_{av}} & \boldsymbol{F_{aa}} & \boldsymbol{F_{ab}} \\ \boldsymbol{F_{bv}} & \boldsymbol{F_{ba}} & \boldsymbol{F_{bb}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_2 & \Delta t\, \mathbf{I}_2 & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 2} & \mathbf{I}_2 & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 2} & \mathbf{I}_3 \end{bmatrix} \tag{N.20}
$$

N.6 Measurement prediction

The predicted estimates are corrected with the encoder and accelerometer measurements. To do so, the Extended Kalman filter requires a measurement model that relates the state estimates to the measurements. The measurement model is both used to compute a predicted measurement vector for innovation calculation and the model is linearized to compute the noise covariance of the predicted measurement. For the VEKF two measurement models are derived, one for each sensor, based on the sensor models from Appendix N.3.

N.6.1 Encoders

The predicted encoder ticks is computed using the sensor model from Appendix N.3.1 combined with the inverse kinematics from (4.35).

$$\begin{aligned}\hat{\boldsymbol{z}}_{\text{enc}} = \Delta\boldsymbol{i}_{\text{ticks}} &= \frac{n_{\text{ticks}}}{2\pi}\Delta t\,\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right)^{\mathsf{T}}\left(\boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right)\frac{1}{r_k}\begin{bmatrix}0\\ -{}^{\text{I}}\dot{\hat{y}}\\ {}^{\text{I}}\dot{\hat{x}}\\ 0\end{bmatrix} - 2\,{}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}}\right)\\ &= \frac{n_{\text{ticks}}}{2\pi}\Delta t\,\widetilde{\boldsymbol{W}}\left(\frac{1}{r_k}\,{}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{*}\circ\begin{bmatrix}0\\ -{}^{\text{I}}\dot{\hat{y}}\\ {}^{\text{I}}\dot{\hat{x}}\\ 0\end{bmatrix}\circ\,{}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} - 2\,{}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{*}\circ\,{}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}}\right)\end{aligned} \tag{N.21}$$

N.6.2 Accelerometer

The measurement model of the accelerometer is based on (N.10) where the rotation matrix is replaced by the quaternion vector rotation, see Appendix H.3, using the quaternion estimated by the QEKF.

$$\begin{aligned}{}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} &= \vee\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{*}\circ\left(\wedge\begin{bmatrix}{}^{\text{I}}\ddot{x}\\ {}^{\text{I}}\ddot{y}\\ g\end{bmatrix}\right)\circ\,{}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right) + {}^{\text{IMU}}\hat{\boldsymbol{b}}_{\text{acc}}\\ &= \left(\vee\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right)\wedge\right)\begin{bmatrix}{}^{\text{I}}\ddot{x}\\ {}^{\text{I}}\ddot{y}\\ g\end{bmatrix} + {}^{\text{IMU}}\hat{\boldsymbol{b}}_{\text{acc}}\end{aligned} \tag{N.22}$$

Note that using the accelerometer model with the quaternion and quaternion derivative estimates from the QEKF, might lead to a slightly biased acceleration estimate, since the QEKF fuses the accelerometer measurements assuming that no acceleration is present. Hopefully this bias will be less present in the velocity estimate due to the correction with the encoder measurements.

N.6.3 Measurement vector

The predicted measurement vector is contructed by stacking the predicted measurement vectors of the two sensors.

$$\hat{\boldsymbol{z}}[k] = \begin{bmatrix}\hat{\boldsymbol{z}}_{\text{enc}}[k]\\ {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}[k]\end{bmatrix} \tag{N.23}$$

The same stacking is used when constructing the measurement vector, $\boldsymbol{z}$, from the sampled measurements. Note that the encoder measurement, $\hat{\boldsymbol{z}}_{\text{enc}}$, is given as a difference in ticks, $\Delta\boldsymbol{i}_{\text{ticks}}$, while the internal quadrature decoder returns the absolute value, $\boldsymbol{i}_{\text{ticks}}$. The difference thus has to be computed at each time step by storing the old absolute value.

N.6.4 Jacobian

To compute the measurement prediction covariance due to uncertainty in the state estimates, the Extended Kalman filter linearizes the measurement model by forming a Jacobian, which is then applied to the estimation error covariance. The Jacobian of the measurement model with respect to the states is defined as

$$\boldsymbol{H} = \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial \hat{\boldsymbol{x}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{H_{ev}} & \boldsymbol{H_{ea}} & \boldsymbol{H_{eb}} \\ \boldsymbol{H_{av}} & \boldsymbol{H_{aa}} & \boldsymbol{H_{ab}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H_{ev}} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{H_{aa}} & \mathbf{I}_3 \end{bmatrix} \tag{N.24}$$

The two sub-Jacobians are derived below.

$$\boldsymbol{H_{ev}} = \left.\begin{bmatrix} \frac{\partial \hat{z}_{\text{enc}}[k]}{\partial {}^{\text{I}}\dot{\hat{x}}[k]} & \frac{\partial \hat{z}_{\text{enc}}[k]}{\partial {}^{\text{I}}\dot{\hat{y}}[k]} \end{bmatrix}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \frac{n_{\text{ticks}}}{2\pi}\Delta t\, \widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)\frac{1}{r_k}\begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{N.25}$$

$$\boldsymbol{H_{aa}} = \left.\begin{bmatrix} \frac{\partial {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}[k]}{\partial {}^{\text{I}}\dot{\hat{x}}[k]} & \frac{\partial {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}}[k]}{\partial {}^{\text{I}}\dot{\hat{y}}[k]} \end{bmatrix}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \left(\vee\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}}\boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)\right)\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{N.26}$$

N.6.5 Measurement covariance

To deal with uncertainty in the quaternion estimate, ${}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}$, and the quaternion derivative estimate, ${}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}}$, used in the computation of both measurement predictions, the noise contributions from these estimates have to be added to the measurement covariance. Prior to this inclusion the measurement covariance matrix is defined solely based on the additive noise terms as

$$\widetilde{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\text{enc}} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{\Sigma}_{\text{acc}} \end{bmatrix} \tag{N.27}$$

The quaternion and quaternion derivative estimates include additive noise defined from the estimation error covariance of the QEKF.

$$\begin{aligned} {}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}} &= {}^{\text{K}}_{\text{B}}\boldsymbol{q} + \boldsymbol{v_q} \\ {}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}} &= {}^{\text{K}}_{\text{B}}\dot{\boldsymbol{q}} + \boldsymbol{v_{\dot{q}}} \end{aligned} \tag{N.28}$$

where $\boldsymbol{v_q} \sim \mathcal{N}(\boldsymbol{0}_{4\times1}, \boldsymbol{\Sigma_q})$ and $\boldsymbol{v_{\dot{q}}} \sim \mathcal{N}(\boldsymbol{0}_{4\times1}, \boldsymbol{\Sigma_{\dot{q}}})$ are the estimation errors of the quaternion and quaternion derivative estimates. The covariances are extracted from the aposteriori estimation error covariance, $\boldsymbol{P}^{k|k}$, of the QEKF, see Appendix M.8.1.

To compute the noise contribution into the measurement predictions, the measurement model is linearized with respect to the quaternion and quaternion derivative.

$$\boldsymbol{T}_{\boldsymbol{q}} = \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{T}_{\boldsymbol{eq}} \\ \boldsymbol{T}_{\boldsymbol{aq}} \end{bmatrix} \tag{N.29}$$

$$\boldsymbol{T}_{\dot{\boldsymbol{q}}} = \left.\frac{\partial \hat{\boldsymbol{z}}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{T}_{\boldsymbol{e}\dot{\boldsymbol{q}}} \\ \boldsymbol{0}_{3\times 4} \end{bmatrix} \tag{N.30}$$

where the sub-Jacobians are derived below.

$$\boldsymbol{T}_{\boldsymbol{eq}} = \left.\frac{\partial \hat{\boldsymbol{z}}_{\mathrm{enc}}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \frac{n_{\mathrm{ticks}}}{\pi}\Delta t\, \widetilde{\boldsymbol{W}} \left(\frac{1}{r_k} \boldsymbol{\Phi} \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right)^{\mathsf{T}} \boldsymbol{\Phi} \left(\begin{bmatrix} 0 \\ -{}^{\mathrm{I}}\dot{\hat{y}}^{k|k-1} \\ {}^{\mathrm{I}}\dot{\hat{x}}^{k|k-1} \\ 0 \end{bmatrix} \right) - \boldsymbol{\Gamma} \left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}^{k|k-1} \right) \boldsymbol{\Gamma}^{*} \right) \tag{N.31}$$

$$\begin{aligned} \boldsymbol{T}_{\boldsymbol{aq}} = \left.\frac{\partial\, {}^{\mathrm{IMU}}\hat{\boldsymbol{z}}_{\mathrm{acc}}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} &= \vee\boldsymbol{\Phi} \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right)^{\mathsf{T}} \boldsymbol{\Phi} \left(\begin{bmatrix} 0 \\ {}^{\mathrm{I}}\ddot{\hat{x}}^{k|k-1} \\ {}^{\mathrm{I}}\ddot{\hat{y}}^{k|k-1} \\ g \end{bmatrix} \right) + \vee\boldsymbol{\Gamma} \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right) \boldsymbol{\Gamma} \left(\begin{bmatrix} 0 \\ {}^{\mathrm{I}}\ddot{\hat{x}}^{k|k-1} \\ {}^{\mathrm{I}}\ddot{\hat{y}}^{k|k-1} \\ g \end{bmatrix} \right) \boldsymbol{\Gamma}^{*} \\ &= 2 \vee \boldsymbol{\Phi} \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right)^{\mathsf{T}} \boldsymbol{\Phi} \left(\begin{bmatrix} 0 \\ {}^{\mathrm{I}}\ddot{\hat{x}}^{k|k-1} \\ {}^{\mathrm{I}}\ddot{\hat{y}}^{k|k-1} \\ g \end{bmatrix} \right) \end{aligned} \tag{N.32}$$

Note that the relationship from (H.96) is used to simplify the above partial derivatives.

$$\boldsymbol{T}_{\boldsymbol{e}\dot{\boldsymbol{q}}} = \left.\frac{\partial \hat{\boldsymbol{z}}_{\mathrm{enc}}[k]}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\hat{\boldsymbol{q}}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = -\frac{n_{\mathrm{ticks}}}{\pi}\Delta t\, \widetilde{\boldsymbol{W}} \boldsymbol{\Phi} \left({}^{\mathrm{K}}_{\mathrm{B}}\hat{\boldsymbol{q}}^{k|k-1} \right)^{\mathsf{T}} \tag{N.33}$$

The noise contribution is finally added to the measurement covariance.

$$\boldsymbol{R} = \widetilde{\boldsymbol{R}} + \boldsymbol{T}_{\boldsymbol{q}} \boldsymbol{\Sigma}_{\boldsymbol{q}} \boldsymbol{T}_{\boldsymbol{q}}^{\mathsf{T}} + \boldsymbol{T}_{\dot{\boldsymbol{q}}} \boldsymbol{\Sigma}_{\dot{\boldsymbol{q}}} \boldsymbol{T}_{\dot{\boldsymbol{q}}}^{\mathsf{T}} \tag{N.34}$$

N.7 Algorithm

The VEKF follows the general algorithm of an Extended Kalman filter summarized below. The VEKF estimates the following state vector:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} {}^{\mathrm{I}}\dot{\hat{x}} \\ {}^{\mathrm{I}}\dot{\hat{y}} \\ {}^{\mathrm{I}}\ddot{\hat{x}} \\ {}^{\mathrm{I}}\ddot{\hat{y}} \\ {}^{\mathrm{IMU}}\hat{\boldsymbol{b}}_{\mathrm{acc}} \end{bmatrix} \tag{N.35}$$

The filter is initialized with zero velocity, zero acceleration and zero bias. The estimation error covariance is initialized as a diagonal matrix with very low uncertainty on the accelerometer bias since this is assumed to be mostly pre-calibrated and compensated.

$$\hat{\boldsymbol{x}}^{0|0} = \begin{bmatrix} {}^{\mathrm{I}}\dot{\hat{x}} & {}^{\mathrm{I}}\dot{\hat{y}} & {}^{\mathrm{I}}\ddot{\hat{x}} & {}^{\mathrm{I}}\ddot{\hat{y}} & {}^{\mathrm{IMU}}\hat{\boldsymbol{b}}_{\mathrm{acc}}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} = \left[\begin{array}{cc|cc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]^{\mathsf{T}} \tag{N.36}$$

$$\boldsymbol{P}^{0|0} = \mathrm{diag}(1\mathrm{e}-1 \cdot \mathbf{I}_2, \quad 1\mathrm{e}-2 \cdot \mathbf{I}_2, \quad 1\mathrm{e}-9 \cdot \mathbf{I}_3) \tag{N.37}$$

For each time step at a periodicity of $\Delta t = \frac{1}{200\ \mathrm{Hz}} = 5$ ms the following algorithm is executed.

1. **Prediction step**

   (a) Compute apriori estimates

$$ {}^{\mathrm{I}}\dot{x}^{k|k-1} = {}^{\mathrm{I}}\dot{x}^{k-1|k-1} + \Delta t\, {}^{\mathrm{I}}\ddot{x}^{k-1|k-1} \tag{N.38}$$

$$ {}^{\mathrm{I}}\dot{y}^{k|k-1} = {}^{\mathrm{I}}\dot{y}^{k-1|k-1} + \Delta t\, {}^{\mathrm{I}}\ddot{y}^{k-1|k-1} \tag{N.39}$$

$$ {}^{\mathrm{I}}\ddot{x}^{k|k-1} = {}^{\mathrm{I}}\ddot{x}^{k-1|k-1} \tag{N.40}$$

$$ {}^{\mathrm{I}}\ddot{y}^{k|k-1} = {}^{\mathrm{I}}\ddot{y}^{k-1|k-1} \tag{N.41}$$

$$ {}^{\mathrm{IMU}}\boldsymbol{b}_{\mathrm{acc}}^{k|k-1} = {}^{\mathrm{IMU}}\boldsymbol{b}_{\mathrm{acc}}^{k-1|k-1} \tag{N.42}$$

   (b) Construct process model Jacobian

$$\boldsymbol{F} = \left.\frac{\partial \boldsymbol{x}[k]}{\partial \boldsymbol{x}[k-1]}\right|_{\substack{\boldsymbol{x}[k-1]=\hat{\boldsymbol{x}}^{k-1|k-1} \\ \boldsymbol{w}[k-1]=\mathbf{0}}} = \begin{bmatrix} \mathbf{I}_2 & \Delta t\, \mathbf{I}_2 & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 2} & \mathbf{I}_2 & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 2} & \mathbf{I}_3 \end{bmatrix} \tag{N.43}$$

   (c) Compute apriori estimation error covariance

$$\boldsymbol{P}^{k|k-1} = \boldsymbol{F}\boldsymbol{P}^{k-1|k-1}\boldsymbol{F}^{\mathsf{T}} + \boldsymbol{Q} \tag{N.44}$$

2. **Sample measurements** to time k

3. **Correction step**

   (a) Compute encoder difference measurement

$$\boldsymbol{z}_{\mathrm{enc}} = \boldsymbol{i}_{\mathrm{ticks}}[k] - \boldsymbol{i}_{\mathrm{ticks}}[k-1] \tag{N.45}$$

(b) Construct measurement vector with measurements

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_{\text{enc}} \\ {}^{\text{IMU}}\boldsymbol{z}_{\text{acc}} \end{bmatrix} \tag{N.46}$$

(c) Compute predicted measurements

$$\hat{\boldsymbol{z}}_{\text{enc}} = \frac{n_{\text{ticks}}}{2\pi} \Delta t \, \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right)^{\mathsf{T}} \left(\boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right) \frac{1}{r_k} \begin{bmatrix} 0 \\ -{}^{\text{I}}\dot{\hat{y}} \\ {}^{\text{I}}\dot{\hat{x}} \\ 0 \end{bmatrix} - 2\,{}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}} \right) \tag{N.47}$$

$${}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} = \left(\vee\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}\right) \wedge \right) \begin{bmatrix} {}^{\text{I}}\ddot{x} \\ {}^{\text{I}}\ddot{y} \\ g \end{bmatrix} + {}^{\text{IMU}}\hat{\boldsymbol{b}}_{\text{acc}} \tag{N.48}$$

(d) Construct predicted measurement vector

$$\hat{\boldsymbol{z}} = \begin{bmatrix} \hat{\boldsymbol{z}}_{\text{enc}} \\ {}^{\text{IMU}}\hat{\boldsymbol{z}}_{\text{acc}} \end{bmatrix} \tag{N.49}$$

(e) Compute measurement model partial derivatives

$$\boldsymbol{H}_{\boldsymbol{ev}} = \frac{n_{\text{ticks}}}{2\pi} \Delta t \, \widetilde{\boldsymbol{W}} \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right) \frac{1}{r_k} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}^{\mathsf{T}} \tag{N.50}$$

$$\boldsymbol{H}_{\boldsymbol{aa}} = \left(\vee\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}} \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right) \right) \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{\mathsf{T}} \tag{N.51}$$

(f) Construct measurement model Jacobian

$$\boldsymbol{H} = \left. \frac{\partial \hat{\boldsymbol{z}}[k]}{\partial \hat{\boldsymbol{x}}[k]} \right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1} \\ \boldsymbol{v}[k]=\boldsymbol{0}}} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{ev}} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{H}_{\boldsymbol{aa}} & \mathbf{I}_3 \end{bmatrix} \tag{N.52}$$

(g) Compute measurement noise partial derivatives

$$\boldsymbol{T}_{\boldsymbol{eq}} = \frac{n_{\text{ticks}}}{\pi}\Delta t\,\widetilde{\boldsymbol{W}}\left(\frac{1}{r_k}\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ -{}^{\text{I}}\dot{\hat{y}}^{k|k-1}\\ {}^{\text{I}}\dot{\hat{x}}^{k|k-1}\\ 0\end{bmatrix}\right) - \boldsymbol{\Gamma}\left({}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}}^{k|k-1}\right)\mathbf{I}^{*}\right) \tag{N.53}$$

$$\boldsymbol{T}_{\boldsymbol{aq}} = 2 \vee \boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}}\boldsymbol{\Phi}\left(\begin{bmatrix}0\\ {}^{\text{I}}\ddot{\hat{x}}^{k|k-1}\\ {}^{\text{I}}\ddot{\hat{y}}^{k|k-1}\\ g\end{bmatrix}\right) \tag{N.54}$$

$$\boldsymbol{T}_{\boldsymbol{e\dot{q}}} = -\frac{n_{\text{ticks}}}{\pi}\Delta t\,\widetilde{\boldsymbol{W}}\boldsymbol{\Phi}\left({}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}^{k|k-1}\right)^{\mathsf{T}} \tag{N.55}$$

(h) Construct noise Jacobians with respect to QEKF estimates

$$\boldsymbol{T}_{\boldsymbol{q}} = \left.\frac{\partial\hat{\boldsymbol{z}}[k]}{\partial\,{}^{\text{K}}_{\text{B}}\hat{\boldsymbol{q}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\mathbf{0}}} = \begin{bmatrix}\boldsymbol{T}_{\boldsymbol{eq}}\\ \boldsymbol{T}_{\boldsymbol{aq}}\end{bmatrix} \tag{N.56}$$

$$\boldsymbol{T}_{\dot{\boldsymbol{q}}} = \left.\frac{\partial\hat{\boldsymbol{z}}[k]}{\partial\,{}^{\text{K}}_{\text{B}}\dot{\hat{\boldsymbol{q}}}[k]}\right|_{\substack{\boldsymbol{x}[k]=\hat{\boldsymbol{x}}^{k|k-1}\\ \boldsymbol{v}[k]=\mathbf{0}}} = \begin{bmatrix}\boldsymbol{T}_{\boldsymbol{e\dot{q}}}\\ \mathbf{0}_{3\times4}\end{bmatrix} \tag{N.57}$$

(i) Compute measurement noise covariance with noise contribution from QEKF estimates

$$\boldsymbol{R} = \widetilde{\boldsymbol{R}} + \boldsymbol{T}_{\boldsymbol{q}}\boldsymbol{\Sigma}_{\boldsymbol{q}}\boldsymbol{T}_{\boldsymbol{q}}^{\mathsf{T}} + \boldsymbol{T}_{\dot{\boldsymbol{q}}}\boldsymbol{\Sigma}_{\dot{\boldsymbol{q}}}\boldsymbol{T}_{\dot{\boldsymbol{q}}}^{\mathsf{T}} \tag{N.58}$$

(j) Compute predicted measurement covariance

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}^{k|k-1}\boldsymbol{H}^{\mathsf{T}} + \boldsymbol{R} \tag{N.59}$$

(k) Compute Kalman gain

$$\boldsymbol{K} = \boldsymbol{P}^{k|k-1}\boldsymbol{H}^{\mathsf{T}}\boldsymbol{S}^{-1} \tag{N.60}$$

(l) Compute aposteriori estimate

$$\hat{\boldsymbol{x}}^{k|k} = \hat{\boldsymbol{x}}^{k|k-1} + \boldsymbol{K}(\boldsymbol{z} - \hat{\boldsymbol{z}}) \tag{N.61}$$

(m) Compute aposteriori estimation error covariance

$$\boldsymbol{P}^{k|k} = (\mathbf{I}_7 - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}^{k|k-1} \tag{N.62}$$

The VEKF is implemented in the MATLAB file: ↗`Kugle-MATLAB/Estimators/VEKF/VelocityEKF.m`. The VEKF is implemented in C++ using MATLAB Embedded Coder® with a wrapper library written here: ↗`Kugle-Embedded/KugleFirmware/Libraries/Modules/Estimators/VelocityEKF/VelocityEKF.cpp`.

N.7.1 Covariances

The process covariance matrice is defined as:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_v \, \mathbf{I}_2 & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times3} \\ \mathbf{0}_{2\times2} & \Delta t^2 \, \sigma_a \, \mathbf{I}_2 & \mathbf{0}_{2\times3} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \Delta t^2 \, \boldsymbol{\Sigma}_{\text{bias}} \end{bmatrix} \tag{N.63}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\Sigma}_{\text{enc}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{\Sigma}_{\text{acc}} \end{bmatrix} + \boldsymbol{T}_{\boldsymbol{q}} \boldsymbol{\Sigma}_{\boldsymbol{q}} \boldsymbol{T}_{\boldsymbol{q}}^{\mathsf{T}} + \boldsymbol{T}_{\dot{\boldsymbol{q}}} \boldsymbol{\Sigma}_{\dot{\boldsymbol{q}}} \boldsymbol{T}_{\dot{\boldsymbol{q}}}^{\mathsf{T}} \tag{N.64}$$

The process covariance of the kinematic relationship is set to zero, even though the model is discretized.

$$\sigma_v = 0 \tag{N.65}$$

The resulting aposteriori estimation error covariance is partitioned with the following identifiers:

$$\boldsymbol{P}^{k|k} = \begin{bmatrix} \boldsymbol{\Sigma_v} & \boldsymbol{\Sigma_{va}} & \boldsymbol{\Sigma_{vb}} \\ \boldsymbol{\Sigma_{va}} & \boldsymbol{\Sigma_a} & \boldsymbol{\Sigma_{ab}} \\ \boldsymbol{\Sigma_{vb}} & \boldsymbol{\Sigma_{ab}} & \boldsymbol{b_b} \end{bmatrix} \tag{N.66}$$

N.8 Test & results

The VEKF is tested and verified through two tests, one without the controller running and another one with the sliding mode balance controller running, where the balance is kept while the ballbot is pushed around.

N.8.1 Without controller

The test without the controller is the same test as in Figure 6.2, where it was used to verify the kinematics. In Figure N.1 the ballbot is moved manually while keeping it upright within $\pm5°$.

The velocity estimation error, shown in Figure N.2, is mostly between ±0.05 m/s when compared to the numerically computed velocity from Vicon, except for a few outliers with estimation errors up to 0.125 m/s.

N.8.2 With balance controller

In the second test, shown in Figure N.3, the sliding mode balance controller is enabled and given a zero quaternion reference, thus just to keep the balance of the ballbot but not stabilize the position nor velocity. The ballbot is then dragged around to excite different translational velocities.

With the controller running the estimation error, shown in Figure N.4, is kept below 0.2 m/s but is mostly within ±0.1 m/s. Having the motors running do seem to affect the performance slightly but the estimates are still far better than the kinematics-based.

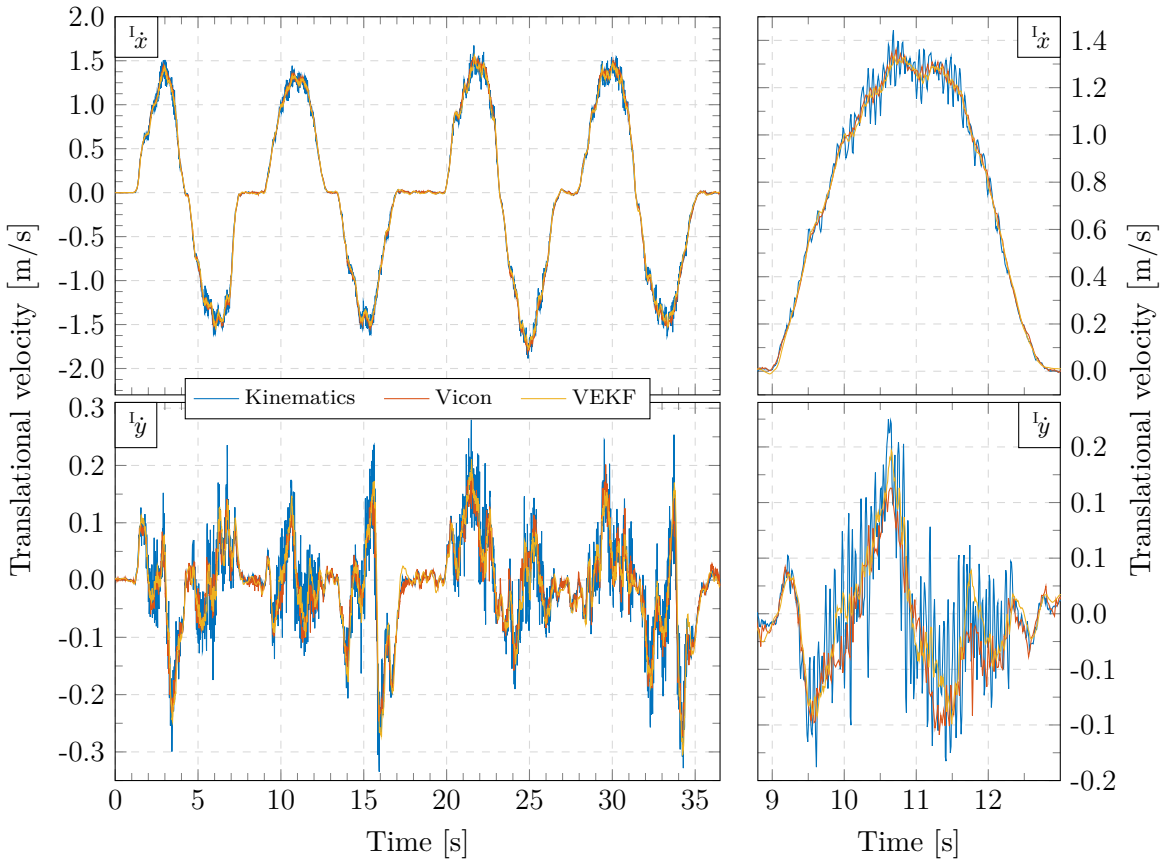


Figure N.1: Comparison of translational velocity during test where the ballbot is moved around manually. Note how the velocity estimate is far less noisy than the kinematics-based velocity.

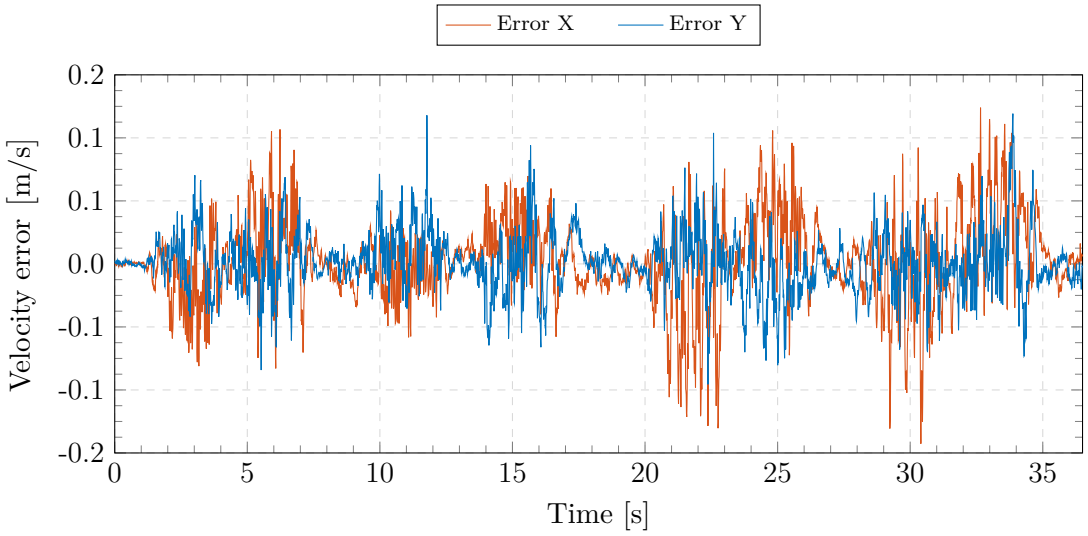


Figure N.2: Velocity estimation error during test with manual movement

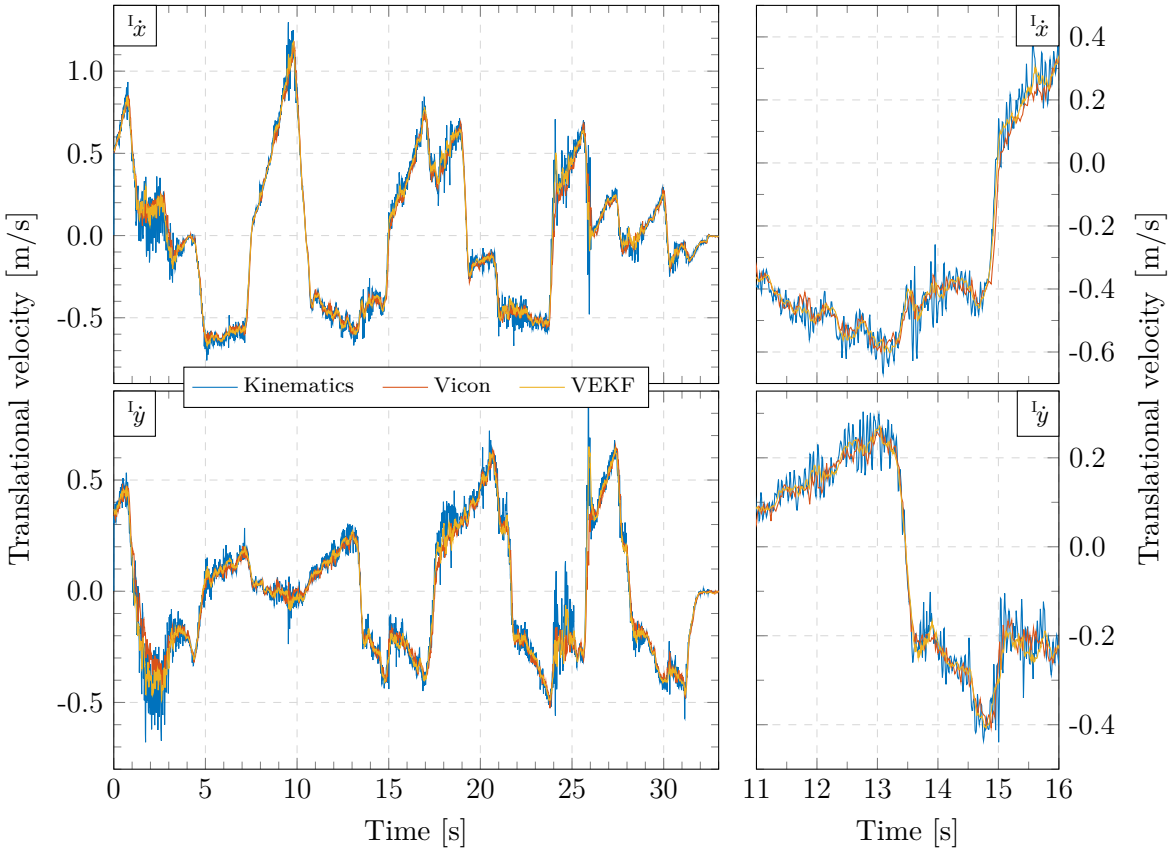


Figure N.3: Comparison of translational velocity during test with the balance controller running and thus actuating the motors. Noise from the switching torque is not apparent in velocity estimate.

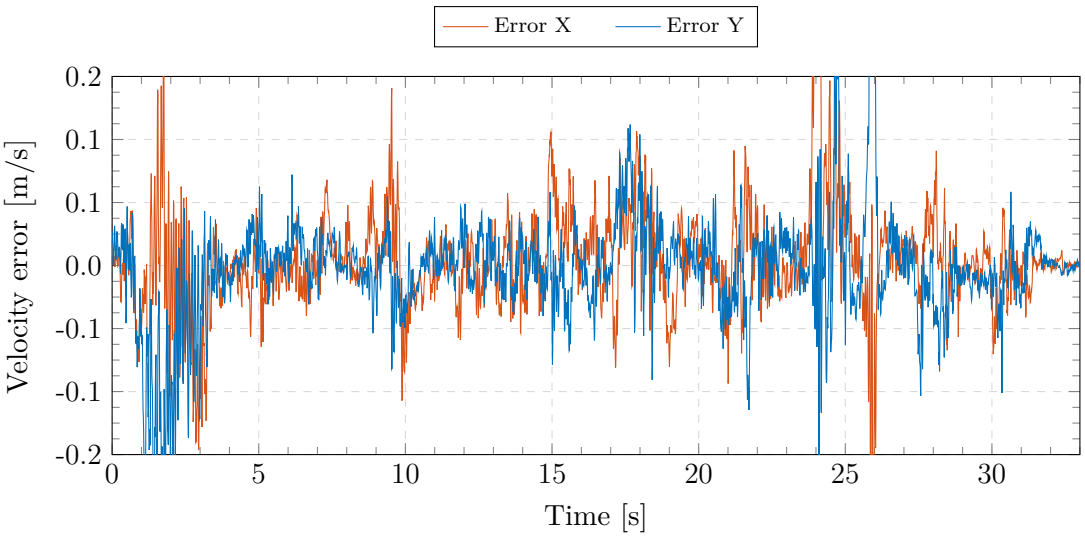


Figure N.4: Velocity estimation error during test with balance controller running

O Implementation Considerations

During implementation and test several observations and considerations have been made related to both tuning and implementation-specific modifications.

O.1 Tuning strategy

Since there are no real rule-of-thumbs for tuning a sliding mode controller the following section presents two common strategies for choosing the sliding surface gain, $\boldsymbol{K}$, and the switching gain, $\boldsymbol{\eta}$.

For easy tuning and testing the ROS driver developed in `Kugle-ROS/kugle_driver` includes several dynamic reconfigurable parameters which can thereby be modified using e.g., the *rqt_reconfigure* GUI [121] as shown in Figure O.1. When a parameter is modified it is automatically sent to the embedded firmware and updated accordingly. This has made testing a breeze since one could quickly evaluate different parameters without having to compile and upload a new firmware.

O.1.1 Sliding surface gain selection

According to Slotine [60] the gain in a first order linear sliding surface ends up defining the control bandwidth. This bandwidth should be chosen so that structural resonant modes, neglected time delays and the sampling rate do not affect the control performance. Hence these criteria infers an upper bound of K.

1. Structural resonant modes
   - K must be smaller than the frequency f_R of the lowest unmodeled structural resonant mode.
   - $K < K_R = \frac{2\pi}{3} f_R$
2. Neglected time delays
   - K must be smaller than the largest/longest unmodeled time delay, T_A.
   - $K < K_A = \frac{1}{3T_A}$
3. Sampling rate
   - With a full-period processing delay from the sampling rate, K should fulfil:
   - $K < K_S = \frac{1}{5} f_s$

O.1.2 Switching gain selection

Although large switching gains (theoretically) lead to extremely strong robustness properties, practical limitations rule out this choice. Large switching gains may lead to extreme wear in gear-boxes, extreme power consumption, and the excitation of unmodeled fast dynamics. Therefore, the switching gain should be chosen as small as possible while still guaranteeing the required robustness. Another possibility is to define an adaptive switching gain, which automatically tunes the switching gain to the actual circumstances.

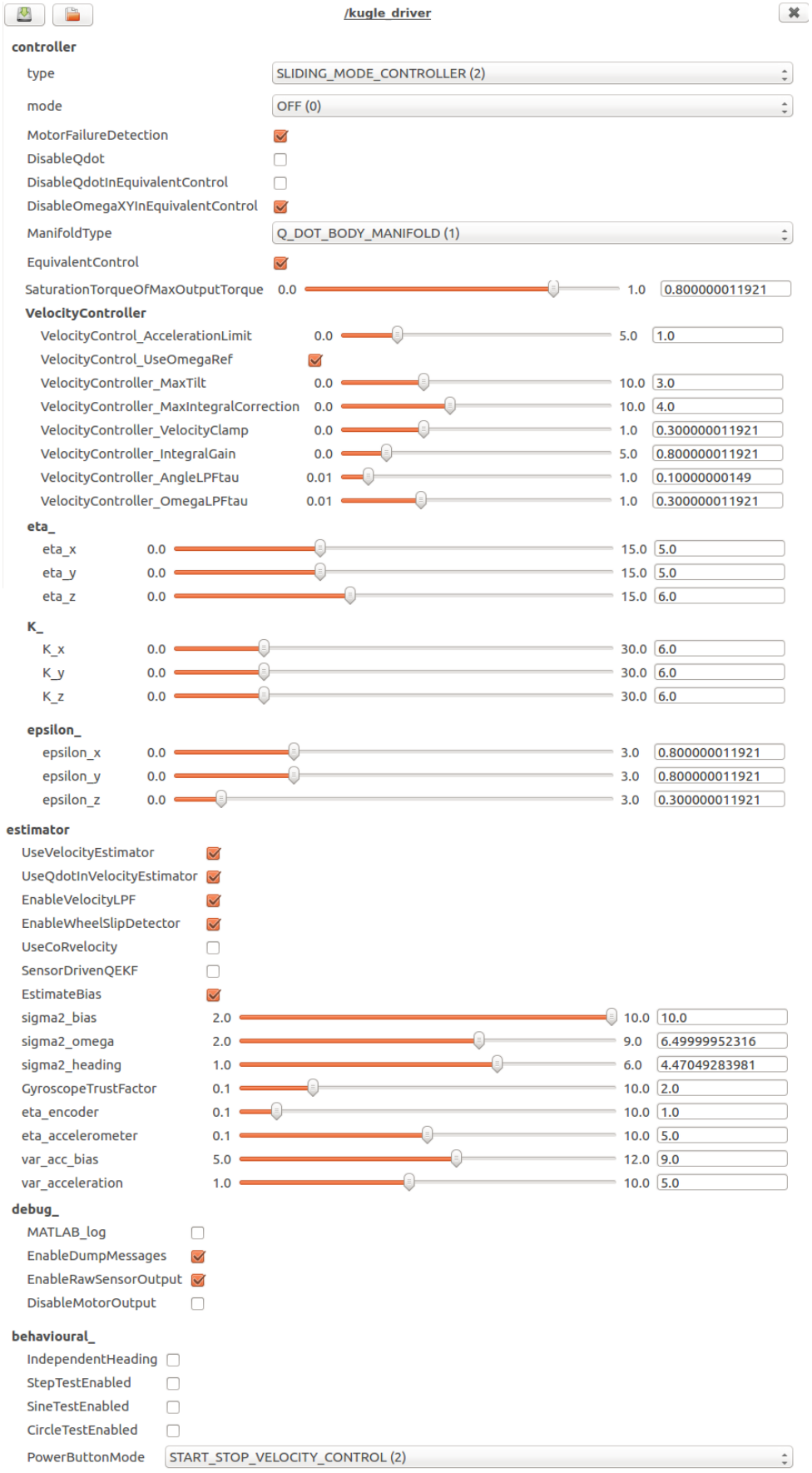


Figure O.1: List of modifiable parameters with the *rqt_reconfigure* GUI

O.1.3 Optimal gains

The best working gains have been found through extensive tuning and practical tests. It is impossible to say what exactly lead to these specific gains and the reader should note that these gains are specific to the physical type and size of ballbot and especially motors and wheels, used in this thesis.

Two set of gains have been found, both intended for use with the quaternion derivative sliding surface. One set is tuned to be aggressive and intended for reference testing and tight tracking. The other set is tuned to be non-aggressive and has been tuned in the cascaded configuration with the velocity controller described in Chapter 14 and Appendix Q.

The aggressive gains are:

$$\begin{aligned} \boldsymbol{K} &= \begin{bmatrix} 15 & 15 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\eta} &= \begin{bmatrix} 6 & 6 & 3 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\epsilon} &= \begin{bmatrix} 0.5 & 0.5 & 0.2 \end{bmatrix}^\mathsf{T} \end{aligned} \tag{O.1}$$

The non-aggressive gains are:

$$\begin{aligned} \boldsymbol{K} &= \begin{bmatrix} 6 & 6 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\eta} &= \begin{bmatrix} 5 & 5 & 6 \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\epsilon} &= \begin{bmatrix} 0.8 & 0.8 & 0.8 \end{bmatrix}^\mathsf{T} \end{aligned} \tag{O.2}$$

O.2 Practical experiences

A few implementation-specific modifications have been made along the way. The list below summarizes some of the modifications which have helped to improve the overall performance and stability of the actual system. The list includes considerations and actions that have been applied to remedy discovered issues on the real system.

- Low-pass filtering of the angular velocity estimate
  - The controller is very sensitive to noise in the angular velocity estimate, ${}^{\mathrm{B}}\hat{\boldsymbol{\omega}}_{\mathrm{B}\leftarrow\mathrm{K}}$
  - In the equivalent control part any noise present in the angular velocity propagates directly on to the computed control torque, causing an erratic behaviour.
  - Since the Madgwick filter does not smooth the quaternion derivative and thus the angular velocity estimate, a low-pass filter was initially put on the angular velocity estimate to remedy this issue.
  - Later the QEKF was modified to include this low-pass filter as part of the process model.

- Removal of ${}^{\mathrm{B}}\omega_x$ and ${}^{\mathrm{B}}\omega_y$ in the equivalent control part
  - Depending on the sliding manifold gain the controller would sometime go into an oscillating high frequency limit cycle.
  - This limit cycle seems to be caused by the angular velocity estimate kicking right through the equivalent control part.
  - The problem might be due to bandwidth, thus the controller gains set too high, causing instability.
  - Lowering the sliding surface gains was not an option since the system then became too slow to stay upright.
  - Instead the problem was solved by removing just the x- and y-axis components of the angular velocity estimate from the equivalent control part.

- Rate limitation and/or low-pass filtering the reference input
  - With the sliding mode controller any step change in the reference will be seen as a sudden disturbance, causing the control torque to spike and saturate. These spikes are likely to cause wheel slip.
  - To avoid sudden spikes on the reference input, rate limitations are applied to all reference inputs and a low-pass filter is put on the angular velocity reference output of the velocity controller.

- Saturate torque to less than what the motors drivers can actually deliver
  - Large torque spikes are more likely to cause wheel slip and thus make the robot jump around on the ball.
  - The key principle of the sliding mode controller is the saturated switching torque, which is exactly what can cause these sudden torque spikes.
  - Any sudden disturbance or reference change would cause wheel slip and make the robot fall due to the torque spikes.
  - To reduce the problem the torque output is saturated at 80% of the max output torque, thus 1.6 N m instead of 2 N m
  - This does not solve the problem with wheel slip but reduces the likelihood.
  - The problem might also be related to an incorrect friction coefficient between the omniwheel and ball.

- Wheel slip detector
  - When wheel slip happens it causes an erratic behaviour of the robot, which it is sometimes able to recover from and sometimes results in several seconds of unrecoverable jumping.
  - Disabling the quaternion derivative completely slows down the system a lot, but thereby also makes it more tolerant and "bouncy".
  - The wheel slip detector uses the encoder measurements to detect a sudden wheel acceleration, greater than the expected acceleration when the wheel is on the ball.
  - If a wheel slip is detected the quaternion derivative and thus the angular velocity, is disabled from both the equivalent control and switching control, until the slip is no longer detected when the wheel velocity is back in a normal operating range.

- Using motor failure detection as lift detection
  - The ESCON motor drivers includes a protection mechanism for the motors if they run too hot, either reducing the output current or even turning them off completely.
  - To reset a motor failure, the commanded output has to be reset by setting it to zero.
  - If the ESCON motor driver shuts down during operation the ballbot falls erratically since full torque will be applied on the two other motors to correct.
  - Motor failure is detected by comparing the commanded control torque with the torque feedback from the ESCON driver.
  - At motor failure the controller is disabled and all motors are turned off
  - Similarly the ESCON drivers includes a maximum angular velocity limitation. Hence if the robot is lifted and a torque is commanded, the wheels will spin up until they hit the angular velocity limitation. In that case the torque feedback will decrease to zero while the commanded torque remains.
  - The angular velocity limitation is used to detect lifts, since the feedback torque quickly diverges from the commanded torque if lifted.

P Balance LQR

As a comparison to the sliding mode controller designed in Chapter 9 for balance control, a linear controller is developed. The controller is similarly developed on just the quaternion dynamics, (9.2), and neglects the translational dynamics, assuming it to be a disturbance.

The MATLAB code for computing the gains and testing the balance LQR can be found at: Kugle-MATLAB/Controllers/BalanceLQR. The C++ implementation of the balance LQR can be found at: Kugle-Embedded/KugleFirmware/Libraries/Modules/Controllers/LQR.

P.1 Error dynamics

When designing a linear controller a linearization point has to be chosen. One way to design a reference tracking LQR controller for a non-linear system, is to derive and linearize the error dynamics. The balance controller should take in a quaternion reference, ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$, and an angular velocity reference defined in body frame, ${}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}$. In Section 8.1 the quaternion error was defined as

$$ {}^{\mathrm{B}}\boldsymbol{q}_e = {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{P.1} $$

while the angular velocity error is just the Euclidean error:

$$ \begin{aligned} {}^{\mathrm{B}}\boldsymbol{\omega}_e &= {}^{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{B}\leftarrow\mathrm{K}} - {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} \\ &= 2 \vee {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} - {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}} \end{aligned} \tag{P.2} $$

Since the four-dimensional quaternion error contain redundant information due to the norm constraint, the vector part is extracted for the error dynamics. The vector part and the angular velocity error are stacked to form the 6-dimensional error state vector for the balance LQR.

$$ \boldsymbol{x}_e = \begin{bmatrix} {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e \\ {}^{\mathrm{B}}\boldsymbol{\omega}_e \end{bmatrix} \tag{P.3} $$

The benefit of using the vector part of the quaternion error is that the equilibrium point of the error state vector becomes the zero vector, which is also chosen as the linearization point.

$$ \tilde{\boldsymbol{x}}_e = \boldsymbol{0}_{6\times 1} \tag{P.4} $$

The error dynamics is defined as the derivative of the error state vector. Inserting the definitions from (8.25) and (9.7) yields:

$$ \dot{\boldsymbol{x}}_e = \begin{bmatrix} {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e \\ {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \vee {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} + \vee {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^*_{\mathrm{ref}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \\ 2 \vee {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} - {}^{\tilde{\mathrm{B}}}\dot{\boldsymbol{\omega}}_{\mathrm{ref}} \end{bmatrix} \tag{P.5} $$

Since the dynamics have to be defined by the error states, the quaternion and quaternion derivatives are substituted by

$$ \begin{aligned} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} &= {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ {}^{\mathrm{B}}\boldsymbol{q}_e \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} &= \frac{1}{2} {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \circ \left({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e + {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \right) \\ &= \frac{1}{2} {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ {}^{\mathrm{B}}\boldsymbol{q}_e \circ \left({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e + {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \right) \end{aligned} \tag{P.6} $$

Furthermore, the angular velocity reference is assumed to be slowly varying such that ${}^{\tilde{\mathrm{B}}}\dot{\tilde{\boldsymbol{\omega}}}_{\mathrm{ref}} = \mathbf{0}_{3\times1}$. The error dynamics thus reduces to

$$\dot{\boldsymbol{x}}_e = \begin{bmatrix} {}^{\mathrm{B}}\dot{\tilde{\boldsymbol{q}}}_e \\ {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \vee {}^{\mathrm{B}}\boldsymbol{q}_e \circ {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e + \frac{1}{2} \vee {}^{\mathrm{B}}\boldsymbol{q}_e \circ {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} - \frac{1}{2} \vee {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \circ {}^{\mathrm{B}}\boldsymbol{q}_e \\ 2 \vee {}^{\mathrm{B}}\boldsymbol{q}_e^* \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} \end{bmatrix} \tag{P.7}$$

Note how the error dynamics depends on the quaternion dynamics, ${}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}$, which includes the controllable torque that the controller should actuate.

P.2 Linearization

We now want to linearize this system with regards to the error state vector by making a first-order Taylor series expansion.

$$\dot{\boldsymbol{x}}_e = \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{\tau}) \approx \boldsymbol{f}(\tilde{\boldsymbol{x}}_e, \tilde{\boldsymbol{\tau}}) + \underbrace{\left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{\tau})}{\partial \boldsymbol{x}_e}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}}}}}_{\boldsymbol{A}} \delta\boldsymbol{x}_e + \underbrace{\left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{\tau})}{\partial \boldsymbol{\tau}}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}}}}}_{\boldsymbol{B}} \delta\boldsymbol{\tau} \tag{P.8}$$

where $\delta\boldsymbol{x}_e = \boldsymbol{x}_e - \tilde{\boldsymbol{x}}_e$, $\delta\boldsymbol{\tau} = \boldsymbol{\tau} - \tilde{\boldsymbol{\tau}}$, $\boldsymbol{A}$ constitutes the linearized error dynamics matrix and $\boldsymbol{B}$ constitutes the linearized input matrix for the error dynamics. Note that due to the choice of linearization point, $\delta\boldsymbol{x}_e = \tilde{\boldsymbol{x}}_e$.
The steady state torque, $\tilde{\boldsymbol{\tau}}$, should be chosen such that

$$\boldsymbol{f}(\tilde{\boldsymbol{x}}_e, \tilde{\boldsymbol{\tau}}) = \mathbf{0}_{6\times1} \quad \rightarrow \quad 2 \vee {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}^* \circ {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} = \mathbf{0}_{3\times1} \tag{P.9}$$

The steady state torque is found by inserting the dynamics from (9.2) and substituting the quaternion and quaternion derivative with (P.6), using the error states at the equilibrium point.

$$\begin{aligned} \mathbf{0}_{3\times1} &= 2 \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\right)^{\mathsf{T}} \left(\boldsymbol{f}_q({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}, {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\dot{\boldsymbol{q}}_{\mathrm{ref}}, {}^{\mathrm{I}}\dot{x}, {}^{\mathrm{I}}\dot{y}) + \boldsymbol{g}_q({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}})\,\tilde{\boldsymbol{\tau}}\right) \\ \tilde{\boldsymbol{\tau}} &= -\left(\vee\boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\right)^{\mathsf{T}} \boldsymbol{g}_q({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}})\right)^{-1} \vee \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\right)^{\mathsf{T}} \boldsymbol{f}_q({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}, {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\dot{\boldsymbol{q}}_{\mathrm{ref}}, {}^{\mathrm{I}}\dot{x}, {}^{\mathrm{I}}\dot{y}) \end{aligned} \tag{P.10}$$

Note the similarities between this steady state torque and the equivalent control laws derived in Section 9.1.

Since the error dynamics in (P.7) includes the system dynamics where the states has been replaced with (P.6), a linearization point for the quaternion reference and angular velocity reference has to be chosen as well. We chose this to be the upright unit quaternion and standing still, thus no angular velocity.

$${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad {}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{P.11}$$

The two linearized matrices are partioned as follows:

$$\boldsymbol{A}_e = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{\tau})}{\partial \boldsymbol{x}_e}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \begin{bmatrix} \boldsymbol{A}_{\dot{\boldsymbol{q}}_e \boldsymbol{q}_e} & \boldsymbol{A}_{\dot{\boldsymbol{q}}_e \boldsymbol{\omega}_e} \\ \boldsymbol{A}_{\dot{\boldsymbol{\omega}}_e \boldsymbol{q}_e} & \boldsymbol{A}_{\dot{\boldsymbol{\omega}}_e \boldsymbol{\omega}_e} \end{bmatrix} \tag{P.12}$$

$$\boldsymbol{B}_e = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{\tau})}{\partial \boldsymbol{\tau}}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \begin{bmatrix} \boldsymbol{B}_{\dot{\boldsymbol{q}}_e \boldsymbol{\tau}} \\ \boldsymbol{B}_{\dot{\boldsymbol{\omega}}_e \boldsymbol{\tau}} \end{bmatrix} \tag{P.13}$$

with the sub-matrices derived below.

$$\boldsymbol{A}_{\dot{\boldsymbol{q}}_e \boldsymbol{q}_e} = \left.\frac{\partial\, {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \frac{1}{2} \vee \left(\boldsymbol{\Gamma}\left({}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e \right) + \boldsymbol{\Gamma}\left({}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \right) - \boldsymbol{\Phi}\left({}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}} \right) \right) \wedge \left|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \boldsymbol{0}_{3\times 3} \right. \tag{P.14}$$

$$\boldsymbol{A}_{\dot{\boldsymbol{q}}_e \boldsymbol{\omega}_e} = \left.\frac{\partial\, {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e}{\partial\, {}^{\mathrm{B}}\boldsymbol{\omega}_e}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \frac{1}{2} \vee \left(\boldsymbol{\Phi}\left({}^{\mathrm{B}}\boldsymbol{q}_e \right) \right) \wedge \left|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \frac{1}{2} \vee \mathbf{I}_4 \wedge = \frac{1}{2}\mathbf{I}_3 \right. \tag{P.15}$$

Where we have used the partial derivative trick

$$\begin{aligned} \frac{\partial\, {}^{\mathrm{B}}\boldsymbol{q}_e}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e} &= \begin{bmatrix} \boldsymbol{0}_{1\times 3} \\ \mathbf{I}_3 \end{bmatrix} = \wedge \\ \frac{\partial\, {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e} &= \frac{\partial\, {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e}{\partial\, {}^{\mathrm{B}}\boldsymbol{q}_e} \frac{\partial\, {}^{\mathrm{B}}\boldsymbol{q}_e}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e} \end{aligned} \tag{P.16}$$

and similarly for ${}^{\mathrm{B}}\boldsymbol{\omega}_e$.

$$\begin{aligned} \boldsymbol{A}_{\dot{\boldsymbol{\omega}}_e \boldsymbol{q}_e} &= \left.\frac{\partial\, {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} \\ &= 2 \vee \left(\boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} \right) \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} \mathbf{I}^* \wedge + \boldsymbol{\Phi}\left({}^{\mathrm{B}}\boldsymbol{q}_e \right)^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}} \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e} \right) \left|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} \right. \end{aligned} \tag{P.17}$$

$$\boldsymbol{A}_{\dot{\boldsymbol{\omega}}_e \boldsymbol{\omega}_e} = \left.\frac{\partial\, {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e}{\partial\, {}^{\mathrm{B}}\boldsymbol{\omega}_e}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = 2 \vee \boldsymbol{\Phi}\left({}^{\mathrm{B}}\boldsymbol{q}_e \right)^{\mathsf{T}} \boldsymbol{\Phi}\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \right)^{\mathsf{T}} \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}} \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{B}}\boldsymbol{\omega}_e} \left|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} \right. \tag{P.18}$$

Note that at the equilibrium point the quaternion dynamics is also zero due to the choice of steady state torque, ${}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}} = \mathbf{0}_{4\times 1}$. Furthermore, the two right hand side partial derivatives are computed from (P.6).

$$\begin{aligned}\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e} &= \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}{\partial\, {}^{\mathrm{B}}\boldsymbol{q}_e}\frac{\partial\, {}^{\mathrm{B}}\boldsymbol{q}_e}{\partial\, {}^{\mathrm{B}}\vec{\boldsymbol{q}}_e} = \boldsymbol{\Phi}\Big({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\Big)\wedge \\ \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{B}}\boldsymbol{\omega}_e} &= \frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e}\frac{\partial\, {}^{\mathrm{B}}\tilde{\boldsymbol{\omega}}_e}{\partial\, {}^{\mathrm{B}}\boldsymbol{\omega}_e} = \frac{1}{2}\boldsymbol{\Phi}\Big({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\Big)\,\boldsymbol{\Phi}\Big({}^{\mathrm{B}}\boldsymbol{q}_e\Big)\wedge\end{aligned} \tag{P.19}$$

When evaluated at the linearization/equilibrium point the two sub matrices thus reduces to

$$\boldsymbol{A}_{\dot{\boldsymbol{\omega}}_e\boldsymbol{q}_e} = 2\vee\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}}\wedge = 2\vee\boldsymbol{A}_{\ddot{\boldsymbol{q}}\boldsymbol{q}}\wedge \tag{P.20}$$

$$\boldsymbol{A}_{\dot{\boldsymbol{\omega}}_e\boldsymbol{\omega}_e} = \vee\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}}\wedge = \vee\boldsymbol{A}_{\ddot{\boldsymbol{q}}\dot{\boldsymbol{q}}}\wedge \tag{P.21}$$

where $\boldsymbol{A}_{\ddot{\boldsymbol{q}}\boldsymbol{q}}$ and $\boldsymbol{A}_{\ddot{\boldsymbol{q}}\dot{\boldsymbol{q}}}$ come from the linearized model in Appendix J.

Similarly the sub-matrices for the input matrix is derived.

$$\boldsymbol{B}_{\dot{\boldsymbol{q}}_e\boldsymbol{\tau}} = \left.\frac{\partial\, {}^{\mathrm{B}}\dot{\vec{\boldsymbol{q}}}_e}{\partial\boldsymbol{\tau}}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = \mathbf{0}_{3\times 3} \tag{P.22}$$

$$\boldsymbol{B}_{\dot{\boldsymbol{\omega}}_e\boldsymbol{\tau}} = \left.\frac{\partial\, {}^{\mathrm{B}}\dot{\boldsymbol{\omega}}_e}{\partial\boldsymbol{\tau}}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = 2\vee\boldsymbol{\Phi}\Big({}^{\mathrm{B}}\boldsymbol{q}_e\Big)^{\mathsf{T}}\,\boldsymbol{\Phi}\Big({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\Big)^{\mathsf{T}}\left.\frac{\partial\, {}^{\mathrm{K}}_{\mathrm{B}}\ddot{\boldsymbol{q}}}{\partial\boldsymbol{\tau}}\right|_{\substack{\boldsymbol{x}_e=\tilde{\boldsymbol{x}}_e \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} \\ {}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}={}^{\tilde{\mathrm{B}}}\tilde{\boldsymbol{\omega}}_{\mathrm{ref}}}} = 2\vee\boldsymbol{B}_{\ddot{\boldsymbol{q}}\boldsymbol{\tau}} \tag{P.23}$$

where $\boldsymbol{B}_{\ddot{\boldsymbol{q}}\boldsymbol{\tau}}$ come from the linearized model in Appendix J.

The evaluated sub-matrices are reassembled into the linearized error dynamics matrices.

$$\boldsymbol{A}_e = \begin{bmatrix}\mathbf{0}_{3\times 3} & \frac{1}{2}\mathbf{I}_3 \\ 2\vee\boldsymbol{A}_{\ddot{\boldsymbol{q}}\boldsymbol{q}}\wedge & \vee\boldsymbol{A}_{\ddot{\boldsymbol{q}}\dot{\boldsymbol{q}}}\wedge\end{bmatrix} \tag{P.24}$$

$$\boldsymbol{B}_e = \begin{bmatrix}\mathbf{0}_{3\times 3} \\ 2\vee\boldsymbol{B}_{\ddot{\boldsymbol{q}}\boldsymbol{\tau}}\end{bmatrix} \tag{P.25}$$

P.3 Controller gains

With the linearized error dynamics the LQR controller gains can be derived. It is chosen to use an infinite horizon LQR so that a static set of controller gains can be computed.

The gains for an LQR controller are found optimally based on the minimization of a quadratic cost function with the states and control output.

$$J(\boldsymbol{\tau}) = \int_{t=0}^{\infty}\boldsymbol{x}_e^{\mathsf{T}}\,\boldsymbol{Q}\,\boldsymbol{x}_e + \boldsymbol{\tau}^{\mathsf{T}}\,\boldsymbol{R}\,\boldsymbol{\tau} \tag{P.26}$$

$$\begin{aligned}&\underset{\boldsymbol{\tau}}{\text{minimize}} \quad J(\boldsymbol{\tau}) \\ &\text{subject to} \quad \dot{\boldsymbol{x}}_e = \boldsymbol{A}_e\,\boldsymbol{x}_e + \boldsymbol{B}_e\,\boldsymbol{\tau}\end{aligned} \tag{P.27}$$

In discrete time the cost function and minimization problem is

$$J(\mathbf{U}) = \sum_{k=0}^{\infty} \boldsymbol{x}_e[k]^\mathsf{T} \boldsymbol{Q}\, \boldsymbol{x}_e[k] + \boldsymbol{\tau}[k]^\mathsf{T} \boldsymbol{R}\, \boldsymbol{\tau}[k] \tag{P.28}$$

$$\begin{aligned} &\underset{\mathbf{U}}{\text{minimize}} \quad J(\mathbf{U}) \\ &\text{subject to} \quad \boldsymbol{x}_e[k+1] = \boldsymbol{A}_{e,d}\, \boldsymbol{x}_e[k] + \boldsymbol{B}_{e,d}\, \boldsymbol{\tau}[k] \end{aligned} \tag{P.29}$$

where $\mathbf{U} = \{\boldsymbol{\tau}[0] \dots \boldsymbol{\tau}[\infty]\}$ is the set of control inputs from time $k = 0 \dots \infty$. For the infinite horizon problem the optimal solution ends up on a form where the optimal control output, $\hat{\boldsymbol{\tau}}$, can be determined from the current state:

$$\hat{\boldsymbol{\tau}} = \boldsymbol{K} \boldsymbol{x}_e \tag{P.30}$$

The optimal gains are found as the solution to the algebraic Riccati equation. Due to the fast sample rate it is chosen not to discretize the continuous dynamics and instead solve the continuous time optimization problem. The solution is given by the continuous time algebraic Riccati equation [122].

$$\boldsymbol{A}^\mathsf{T} \boldsymbol{X} + \boldsymbol{X} \boldsymbol{A} - \boldsymbol{X} \boldsymbol{B} \boldsymbol{R}^{-1} \boldsymbol{B}^\mathsf{T} \boldsymbol{X} + \boldsymbol{Q} = 0 \tag{P.31}$$

where $\boldsymbol{X}$ is the unknown matrix with which the optimal control gain can be computed.

$$\boldsymbol{K} = \boldsymbol{R}^{-1} \boldsymbol{B}^\mathsf{T} \boldsymbol{X} \tag{P.32}$$

The following weighting matrices are chosen to put most weight on the quaternion error related to roll and pitch, and make it less agressive for yaw motion.

$$\begin{aligned} \boldsymbol{Q} &= \operatorname{diag}\Big(\begin{array}{ccc|ccc} {}^{\mathrm{B}}q_{e,1} & {}^{\mathrm{B}}q_{e,2} & {}^{\mathrm{B}}q_{e,3} & {}^{\mathrm{B}}\omega_{e,x} & {}^{\mathrm{B}}\omega_{e,y} & {}^{\mathrm{B}}\omega_{e,z} \\ 1000 & 1000 & 1 & 0.1 & 0.1 & 0.01 \end{array}\Big) \\ \boldsymbol{R} &= 0.05 \cdot \mathbf{I}_3 \end{aligned} \tag{P.33}$$

This leads to the following gain matrix, computed with MATLAB using the `lqr` function, see Kugle-MATLAB/Controllers/BalanceLQR/GainComputation.m.

$$\boldsymbol{K} = \left[\begin{array}{ccc|ccc} {}^{\mathrm{B}}q_{e,1} & {}^{\mathrm{B}}q_{e,2} & {}^{\mathrm{B}}q_{e,3} & {}^{\mathrm{B}}\omega_{e,x} & {}^{\mathrm{B}}\omega_{e,y} & {}^{\mathrm{B}}\omega_{e,z} & \\ 129.6 & 0 & -2.582 & 7.936 & 0 & -0.359 & \boldsymbol{\tau}_0 \\ -64.80 & 112.2 & -2.582 & -3.968 & 6.881 & -0.359 & \boldsymbol{\tau}_1 \\ -64.80 & -112.2 & -2.582 & -3.968 & -6.881 & -0.359 & \boldsymbol{\tau}_2 \end{array}\right] \tag{P.34}$$

The balance LQR is implemented in C++ in the embedded firmware within Kugle-Embedded/KugleFirmware/Libraries/Modules/Controllers/LQR/LQR.cpp.

Q Velocity LQR

One of the main issues when testing the balance controller, e.g., both the sliding mode controller, Chapter 9, and the balance LQR, Appendix P, is that the translational dynamics are left uncontrolled. If the center of mass is just slightly misaligned compared to the calibrated upright reference quaternion, the ballbot will start to accelerate in a given direction while keeping the balance. To compensate for this and also to test the two balance controllers with other references than just the pre-programmed test references, a velocity LQR controller is developed.

The MATLAB code for computing the gains and testing the velocity LQR can be found at: `Kugle-MATLAB/Controllers/VelocityLQR`. The C++ implementation of the velocity LQR can be found at: `Kugle-Embedded/KugleFirmware/Libraries/Modules/Controllers/VelocityLQR`.

Q.1 Objective

The objective of the velocity LQR is to stabilize the translational dynamics of the ballbot by adjusting the quaternion reference and angular velocity reference to the balance controller. The controller should take in a desired velocity reference in the inertial frame, ${}^{\mathrm{I}}\dot{x}_{\mathrm{ref}}$ and ${}^{\mathrm{I}}\dot{y}_{\mathrm{ref}}$, as shown in Figure 7.2. Since a misaligned center of mass leads to an acceleration bias, the velocity controller should include an integral term to compensate for this. This integral term will also enable station keeping when the velocity reference is set to zero and thus keep the ballbot from drifting away. Thereby the velocity controller becomes a position controller at zero velocity reference.

Q.2 Error dynamics

The velocity LQR should track a given velocity reference.

$$\begin{aligned} {}^{\mathrm{I}}\dot{x}_e &= {}^{\mathrm{I}}\dot{x} - {}^{\mathrm{I}}\dot{x}_{\mathrm{ref}} \\ {}^{\mathrm{I}}\dot{y}_e &= {}^{\mathrm{I}}\dot{y} - {}^{\mathrm{I}}\dot{y}_{\mathrm{ref}} \end{aligned} \tag{Q.1}$$

An integral term on the velocity error is included, but should only be active when the velocity reference is zero since it is just supposed to compensate for miscalibrations of the center of mass and avoid causing oscillatory behaviour.

$$\begin{aligned} {}^{\mathrm{I}}x_e &= \int {}^{\mathrm{I}}\dot{x}_e \triangleq {}^{\mathrm{I}}x - {}^{\mathrm{I}}x_{\mathrm{ref}} \\ {}^{\mathrm{I}}y_e &= \int {}^{\mathrm{I}}\dot{y}_e \triangleq {}^{\mathrm{I}}y - {}^{\mathrm{I}}y_{\mathrm{ref}} \end{aligned} \tag{Q.2}$$

where the position references, ${}^{\mathrm{I}}x_{\mathrm{ref}}$ and ${}^{\mathrm{I}}y_{\mathrm{ref}}$, should not be used for position control but are rather floating references that moves with the ballbot when the velocity reference is non-zero.

One way of designing the velocity LQR would be to define the error state vector as

$$\boldsymbol{x}_e = \begin{bmatrix} {}^{\mathrm{I}}x_e \\ {}^{\mathrm{I}}y_e \\ {}^{\mathrm{I}}\dot{x}_e \\ {}^{\mathrm{I}}\dot{y}_e \end{bmatrix} \tag{Q.3}$$

whose derivative is defined as

$$\dot{\boldsymbol{x}}_e = \begin{bmatrix} {}^{\mathrm{I}}\dot{x}_e \\ {}^{\mathrm{I}}\dot{y}_e \\ {}^{\mathrm{I}}\ddot{x} - {}^{\mathrm{I}}\ddot{x}_{\mathrm{ref}} \\ {}^{\mathrm{I}}\ddot{y} - {}^{\mathrm{I}}\ddot{y}_{\mathrm{ref}} \end{bmatrix} = \begin{bmatrix} {}^{\mathrm{I}}\dot{x}_e \\ {}^{\mathrm{I}}\dot{y}_e \\ {}^{\mathrm{I}}\ddot{x} \\ {}^{\mathrm{I}}\ddot{y} \end{bmatrix} \tag{Q.4}$$

Note how the derivative of the velocity reference is neglected since the reference is assumed to be slowly varying. The translational dynamics, ${}^{\mathrm{I}}\ddot{x}$ and ${}^{\mathrm{I}}\ddot{y}$, would then be replaced by the derived dynamics from Section 5.7 where the torque, $\boldsymbol{\tau}$, is replaced by the steady state torque from either Section 9.1 or Appendix P, and the quaternion and quaternion derivative are replaced by the quaternion and quaternion derivative reference, the latter computed from the angular velocity reference.

$$\begin{bmatrix} {}^{\mathrm{I}}\ddot{x} \\ {}^{\mathrm{I}}\ddot{y} \end{bmatrix} = \boldsymbol{f}_{xy}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y}) + \boldsymbol{g}_{xy}({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q},\, {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}},\, {}^{\mathrm{I}}\dot{x},\, {}^{\mathrm{I}}\dot{y})\, \boldsymbol{\tau}\Big|_{\substack{{}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \\ {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}}={}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\dot{\boldsymbol{q}}_{\mathrm{ref}} \\ \boldsymbol{\tau}=\tilde{\boldsymbol{\tau}}}} \tag{Q.5}$$

This design approach would, however, completely neglect the dynamics of the balance controller and the bandwidth of the velocity controller would thus have to be much lower to allow a cascaded control design.

Q.2.1 Error dynamics including balance controller

A better way is to take the dynamics into account by constructing an augmented error dynamics system with the inclusion of the closed balance controller loop. The augmented state vector for the velocity controller is chosen as

$$\boldsymbol{x}_{ve} = \begin{bmatrix} \boldsymbol{x}_e \\ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \end{bmatrix} \tag{Q.6}$$

The reason for adding the quaternion reference to the augmented state space is due to the kinematic relationship between the quaternion reference ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$ and the angular velocity reference, ${}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}$, which should be controlled by the velocity LQR. The velocity LQR is thus designed to compute a desired angular velocity reference, ${}^{\tilde{\mathrm{B}}}\boldsymbol{\omega}_{\mathrm{ref}}$, which is then integrated at each time step to yield the quaternion reference, ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$. This kinematic behaviour is included in the dynamics by the inclusion of ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$.

Q.2.2 Error dynamics with ignored heading

The augmented state space is modified even further, since the velocity controller should not care about controlling the heading. Instead of dealing with a velocity error in the inertial frame and setting quaternion references and angular velocity references between inertial to body frame, the linear velocity controller can be made globally stable by changing to the heading frame.
As described in Appendix R the reference quaternion can be decomposed into a desired heading quaternion, ${}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}}$, and a desired heading frame inclination quaternion, ${}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$.

$${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} = {}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}}\, {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \tag{Q.7}$$

If we let the velocity controller control the heading frame inclination quaternion, ${}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}$, it will become heading independent.

The inertial frame velocity would just have to be rotated accordingly into the heading frame.

$$\begin{bmatrix} {}^{\mathrm{H}}\dot{x} \\ {}^{\mathrm{H}}\dot{y} \end{bmatrix} = {}^{\mathrm{H}}_{\mathrm{I}}\boldsymbol{R} \begin{bmatrix} {}^{\mathrm{I}}\dot{x} \\ {}^{\mathrm{I}}\dot{y} \end{bmatrix} \tag{Q.8}$$

The final augmented state vector is then composed by:

$$\boldsymbol{x}_{ve} = \begin{bmatrix} {}^{\mathrm{H}}x - {}^{\mathrm{H}}x_{\mathrm{ref}} \\ {}^{\mathrm{H}}y - {}^{\mathrm{H}}y_{\mathrm{ref}} \\ {}^{\mathrm{H}}_{\mathrm{B}}q_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}q_2 \\ {}^{\mathrm{H}}\dot{x} - {}^{\mathrm{H}}\dot{x}_{\mathrm{ref}} \\ {}^{\mathrm{H}}\dot{y} - {}^{\mathrm{H}}\dot{y}_{\mathrm{ref}} \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},1} \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},2} \end{bmatrix} \tag{Q.9}$$

Note that just the x- and y-axis elements from the vector part of the two quaternions are included, since both heading inclination quaternions can be represented approximately by

$${}^{\mathrm{H}}_{\mathrm{B}}\boldsymbol{q} \approx \begin{bmatrix} \cos\left(\frac{\theta_{\mathrm{tilt}}}{2}\right) \\ {}^{\mathrm{H}}\boldsymbol{r}_{\mathrm{tilt}} \sin\left(\frac{\theta_{\mathrm{tilt}}}{2}\right) \\ 0 \end{bmatrix} \tag{Q.10}$$

The resulting error dynamics for the velocity controller ends up on the form

$$\dot{\boldsymbol{x}}_{ve} = \boldsymbol{f}_{ve}(\boldsymbol{x}_{ve}, \boldsymbol{\omega}_{\mathrm{ref}}) = \begin{bmatrix} {}^{\mathrm{H}}\dot{x} - {}^{\mathrm{H}}\dot{x}_{\mathrm{ref}} \\ {}^{\mathrm{H}}\dot{y} - {}^{\mathrm{H}}\dot{y}_{\mathrm{ref}} \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{H}}\ddot{x} \\ {}^{\mathrm{H}}\ddot{y} \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},1} \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},2} \end{bmatrix} = \begin{bmatrix} x_{ve,4} \\ x_{ve,5} \\ x_{ve,6} \\ x_{ve,7} \\ {}^{\mathrm{H}}\ddot{x} \\ {}^{\mathrm{H}}\ddot{y} \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},1} \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},2} \end{bmatrix} \tag{Q.11}$$

where

$$\boldsymbol{\omega}_{\mathrm{ref}} = \begin{bmatrix} {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}} \\ {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}} \end{bmatrix} \tag{Q.12}$$

and the system dynamics, ${}^{\mathrm{H}}\ddot{x}$, ${}^{\mathrm{H}}\ddot{y}$, ${}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1$ and ${}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2$ are replaced by the dynamics from Section 5.7 but with $\boldsymbol{\tau}$ replaced by the equivalent control law from Chapter 9 or the steady state torque from Appendix P. When linearizing the heading dynamics, ${}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1$ and ${}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2$, are extracted by linearizing the general quaternion dynamics at an operating point of zero heading.

Q.3 Linearization

The error dynamics is linearized around the zero equilibrium, $\tilde{\boldsymbol{x}}_{ve} = \mathbf{0}_{10\times1}$ and $\tilde{\boldsymbol{\omega}}_{\text{ref}} = \mathbf{0}_{2\times1}$.

$$\dot{\boldsymbol{x}}_{ve} \approx \boldsymbol{f}_{ve}(\tilde{\boldsymbol{x}}_{ve}, \tilde{\boldsymbol{\omega}}_{\text{ref}}) + \underbrace{\left.\frac{\partial \boldsymbol{f}_{ve}(\boldsymbol{x}_{ve}, \boldsymbol{\omega}_{\text{ref}})}{\partial \boldsymbol{x}_{ve}}\right|_{\substack{\boldsymbol{x}_{ve}=\tilde{\boldsymbol{x}}_{ve} \\ \boldsymbol{\omega}_{\text{ref}}=\tilde{\boldsymbol{\omega}}_{\text{ref}}}}}_{\boldsymbol{A}_{ve}} \delta\boldsymbol{x}_{ve} + \underbrace{\left.\frac{\partial \boldsymbol{f}_{ve}(\boldsymbol{x}_{ve}, \boldsymbol{\omega}_{\text{ref}})}{\partial \boldsymbol{\omega}_{\text{ref}}}\right|_{\substack{\boldsymbol{x}_{ve}=\tilde{\boldsymbol{x}}_{ve} \\ \boldsymbol{\omega}_{\text{ref}}=\tilde{\boldsymbol{\omega}}_{\text{ref}}}}}_{\boldsymbol{B}_{ve}} \delta\boldsymbol{\omega}_{\text{ref}} \tag{Q.13}$$

where $\delta\boldsymbol{x}_{ve} = \boldsymbol{x}_{ve} - \tilde{\boldsymbol{x}}_{ve} = \boldsymbol{x}_{ve}$, $\delta\boldsymbol{\omega}_{\text{ref}} = \boldsymbol{\omega}_{\text{ref}} - \tilde{\boldsymbol{\omega}}_{\text{ref}} = \boldsymbol{\omega}_{\text{ref}}$, $\boldsymbol{A}_{ve}$ constitutes the linearized model matrix and $\boldsymbol{B}_{ve}$ constitutes the linearized input matrix.

The equilibrium point is chosen as the point where the balance controller tracks the zero reference inclination angle, given zero angular velocity reference. For a nominal system with a calibrated center of mass such that it is above the origin, the translational dynamics will at that point be zero as well.

$$\boldsymbol{f}_{ve}(\tilde{\boldsymbol{x}}_{ve}, \tilde{\boldsymbol{\omega}}_{\text{ref}}) = \mathbf{0}_{10\times1} \tag{Q.14}$$

The linearized matrices can be partitioned as shown below:

$$\boldsymbol{A}_{ve} = \begin{bmatrix} \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{I}_2 & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{I}_2 & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}} & \boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{v}} & \boldsymbol{A}_{\dot{\boldsymbol{v}}\dot{\boldsymbol{q}}} & \boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}_r} \\ \mathbf{0}_{2\times2} & \boldsymbol{A}_{\ddot{\boldsymbol{q}}\boldsymbol{q}} & \boldsymbol{A}_{\ddot{\boldsymbol{q}}\boldsymbol{v}} & \boldsymbol{A}_{\ddot{\boldsymbol{q}}\dot{\boldsymbol{q}}} & \boldsymbol{A}_{\ddot{\boldsymbol{q}}\boldsymbol{q}_r} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix} \qquad \boldsymbol{B}_{ve} = \begin{bmatrix} \boldsymbol{B}_{\boldsymbol{v}\boldsymbol{\omega}_r} \\ \boldsymbol{B}_{\dot{\boldsymbol{q}}\boldsymbol{\omega}_r} \\ \boldsymbol{B}_{\dot{\boldsymbol{v}}\boldsymbol{\omega}_r} \\ \boldsymbol{B}_{\ddot{\boldsymbol{q}}\boldsymbol{\omega}_r} \\ \boldsymbol{B}_{\dot{\boldsymbol{q}}_r\boldsymbol{\omega}_r} \end{bmatrix} \tag{Q.15}$$

The closed loop dynamics of the sliding mode balance controller is linearized numerically in ↗ `Kugle-MATLAB/Linearization/ClosedLoopLinearization_SlidingMode_WithOmegaRef.m` by the pertubation method, similar to the method used in Appendix J.

The resulting linearization for the velocity controller using the sliding mode controller gains from Appendix O.1.3 is shown in (Q.19) and (Q.20), computed with ↗ `Kugle-MATLAB/Controllers/VelocityLQR/GainComputation.m`.

Q.3.1 Poles and zeros

With the closed loop dynamics being a part of the error dynamics, it is possible to identify the closed loop poles of the balance controller. Furthermore, the expected right-half plane zeros in the closed loop from angular velocity reference input to velocity and position output can be identified.
The poles, $\boldsymbol{\lambda}$, are computed as the eigenvalues of the $\boldsymbol{A}_{ve}$ matrix.

$$\boldsymbol{\lambda} = \text{eig}(\boldsymbol{A}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -6.00 & -6.00 & -6.25 & -6.25 \end{bmatrix}^\mathsf{T} \tag{Q.16}$$

The zeros from ${}^{\tilde{\text{B}}}\omega_{\text{ref,x}}$ input to ${}^{\text{H}}\dot{y}_e$ velocity output is:

$$\boldsymbol{z}_{{}^{\tilde{\text{B}}}\omega_{\text{ref,x}} \to {}^{\text{H}}\dot{y}_e} = \begin{bmatrix} 2.91 & -2.91 & -3.06 \end{bmatrix}^\mathsf{T} \tag{Q.17}$$

Conversely for ${}^{\tilde{\text{B}}}\omega_{\text{ref,y}}$ input to ${}^{\text{H}}\dot{x}_e$ velocity output:

$$\boldsymbol{z}_{{}^{\tilde{\text{B}}}\omega_{\text{ref,y}} \to {}^{\text{H}}\dot{x}_e} = \begin{bmatrix} 2.91 & -2.91 & -3.06 \end{bmatrix}^\mathsf{T} \tag{Q.18}$$

Notice the right-half plane zeros as expected. These are exactly the zeros causing the non-minimum phase behaviour the ballbot is known to have.

$$\boldsymbol{A}_{ve} = \begin{array}{c} \begin{array}{cccccccccc} {}^{\mathrm{H}}x_e & {}^{\mathrm{H}}y_e & {}^{\mathrm{H}}_{\mathrm{B}}q_1 & {}^{\mathrm{H}}_{\mathrm{B}}q_2 & {}^{\mathrm{H}}\dot{x}_e & {}^{\mathrm{H}}\dot{y}_e & {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 & {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},1} & {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},2} \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 1.00 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 0 \\ \hline 0 & 0 & 0 & 79.08 & 0 & 0 & 0 & 21.09 & 0 & -64.55 \\ 0 & 0 & -78.99 & 0 & 0 & 0 & -21.06 & 0 & 64.46 & 0 \\ \hline 0 & 0 & -37.50 & 0 & 0 & 0 & -12.25 & 0 & 37.50 & 0 \\ 0 & 0 & 0 & -37.50 & 0 & 0 & 0 & -12.25 & 0 & 37.50 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} {}^{\mathrm{H}}\dot{x}_e \\ {}^{\mathrm{H}}\dot{y}_e \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{H}}\ddot{x}_e \\ {}^{\mathrm{H}}\ddot{y}_e \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},1} \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},2} \end{array} \tag{Q.19}$$

$$\boldsymbol{B}_{ve} = \begin{array}{c} \begin{array}{cc} {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}} & {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}} \end{array} \\ \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ \hline 0 & -10.54 \\ 10.53 & 0 \\ \hline 6.13 & 0 \\ 0 & 6.13 \\ \hline 0.50 & 0 \\ 0 & 0.50 \end{array}\right] \end{array} \begin{array}{c} {}^{\mathrm{H}}\dot{x}_e \\ {}^{\mathrm{H}}\dot{y}_e \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 \\ {}^{\mathrm{H}}\ddot{x}_e \\ {}^{\mathrm{H}}\ddot{y}_e \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2 \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},1} \\ {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\dot{q}_{\mathrm{ref},2} \end{array} \tag{Q.20}$$

Q.4 Controller gains

The velocity controller is designed to run at the same sample rate as the balance controller by including the closed loop dynamics. The LQR gains are computed based on the continuous model, similar to how it was done in Appendix P.3, even though the controller is implemented in discrete time at a sample rate of 200 Hz.

The following weighting matrices are chosen to put a lot of weight on the position and velocity states and on the angular velocity reference output to make the controller less agressive.

$$\boldsymbol{Q} = \operatorname{diag}\Big(\begin{array}{cc|cc|cc|cc|cc} {}^{\mathrm{H}}x_e & {}^{\mathrm{H}}y_e & {}^{\mathrm{H}}_{\mathrm{B}}q_1 & {}^{\mathrm{H}}_{\mathrm{B}}q_2 & {}^{\mathrm{H}}\dot{x}_e & {}^{\mathrm{H}}\dot{y}_e & {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 & {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},1} & {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},2} \\ 20 & 20 & 0.01 & 0.01 & 10 & 10 & 0.1 & 0.1 & 0.01 & 0.01 \end{array}\Big) \tag{Q.21}$$

$$\boldsymbol{R} = 20 \cdot \mathbf{I}_2$$

This leads to the following gain matrix, computed with MATLAB using the `lqr` function, see Kugle-MATLAB/Controllers/VelocityLQR/GainComputation.m.

$$\boldsymbol{K} = \begin{array}{c} \begin{array}{cccccccccc} {}^{\mathrm{H}}x_e & {}^{\mathrm{H}}y_e & {}^{\mathrm{H}}_{\mathrm{B}}q_1 & {}^{\mathrm{H}}_{\mathrm{B}}q_2 & {}^{\mathrm{H}}\dot{x}_e & {}^{\mathrm{H}}\dot{y}_e & {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_1 & {}^{\mathrm{H}}_{\mathrm{B}}\dot{q}_2 & {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},1} & {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}q_{\mathrm{ref},2} \end{array} \\ \left[\begin{array}{cc|cc|cc|cc|cc} 0.00 & -1.00 & 9.29 & 0.00 & 0.00 & -1.56 & 3.51 & 0.00 & 4.78 & 0.00 \\ 1.00 & 0.00 & 0.00 & 9.29 & 1.56 & 0.00 & 0.00 & 3.51 & 0.00 & 4.78 \end{array}\right] \end{array} \begin{array}{c} \\ \begin{array}{c} {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}} \\ {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}} \end{array} \end{array} \tag{Q.22}$$

Q.5 Implementation considerations

The velocity LQR is implemented and tested in MATLAB, see Kugle-MATLAB/Controllers/VelocityLQR, and Simulink, see Kugle-MATLAB/Simulation/KugleSim_VelocityLQR.slx.

Other than just the LQR controller and the rotation of velocities into the heading frame, a few other features have been implemented to make a smooth behaviour.

1. Integral is only enabled the first time the system is balancing after the controller is turned on, and it is only enabled until the velocity or velocity reference goes above a certain threshold.
2. Automatic update of position reference which stays constant when the velocity and velocity reference is below the threshold.
    - This allows the robot to be dragged around, such as to move the position reference from one place to another.
3. Saturation of angular velocity reference.
4. First order low-pass filter on the angular velocity reference to avoid sudden jumps.
5. Integration of the angular velocity reference into a quaternion reference.
    - Small-angle approximation allows the quaternion reference integration to be carried out directly as integration of roll and pitch angles.
6. Saturation of inclination angles.

The controller and the above features have been implemented in C++ as part of the embedded firmware for testing on the actual robot, see Kugle-Embedded/KugleFirmware/Libraries/Modules/Controllers/VelocityLQR/VelocityLQR.cpp.

R Heading Independent Control

When pushing the ballbot around the heading control, as part of the balance controller, is indeed effective but very aggressive. With the balance controller enabled the ballbot can be pushed around and the controller ensures that it stays balanced even while driving. The motion when pushed seems almost levitational. Unfortunately the behaviour seems very dependent on whether or not the heading is disturbed. If the heading is disturbed the counteracting control seem to brake the translational motion and thus affects the levitational behaviour.

To test whether the braking is caused by corrections in heading, a heading independent control mode is developed. In the heading independent mode the reference setpoint to the balance controller and the state estimates are augmented to remove any influence from yaw.

R.1 Quaternion reference augmentation

The reference is augmented by extracting the desired inclination in the inertial frame and then recompute the reference quaternion given the current heading. Thereby the desired heading is removed from the reference quaternion and the resulting reference quaternion will have its heading set to the current.

Initially lets define how a reference quaternion can be decomposed. A reference quaternion can either be decomposed into a desired heading quaternion and a desired heading frame inclination quaternion as shown in (R.1).

$$ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} = {}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}} \, {}^{\tilde{\mathrm{H}}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \tag{R.1} $$

The heading quaternion, ${}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}}$, has zero x- and y-axis vector parts since it follows the definition

$$ {}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}} = \begin{bmatrix} \cos\left(\frac{\psi_{\mathrm{ref}}}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\psi_{\mathrm{ref}}}{2}\right) \end{bmatrix} \tag{R.2} $$

Another way is to decompose the reference quaternion into a desired inertial frame inclination quaternion and a desired body z-axis rotation quaternion as shown in (R.3).

$$ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} = {}^{\mathrm{K}}_{\tilde{\mathrm{T}}}\boldsymbol{q}_{\mathrm{ref}} \, {}^{\tilde{\mathrm{T}}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \tag{R.3} $$

The tilt frame, $\{T\}$, is introduced as an intermediate frame of the rotated body after applying the inclination angle in inertial frame but before rotating around the body z-axis. ${}^{\mathrm{K}}_{\tilde{\mathrm{T}}}\boldsymbol{q}_{\mathrm{ref}}$ should thus have a zero z-axis vector part.

If we want to make the reference quaternion heading independent we will have to extract the desired inclination in inertial frame. The inclination is given by the direction in which the body z-axis vector is pointing.

A tilt vector, ${}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{tilt}}$, is constructed from the xy-projection of the z-axis body vector in the inertial frame.

$$\begin{aligned}
{}^{\mathrm{K}}\mathbf{k}_{\mathrm{B}} &= \vee\left({}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}} \circ \hat{\mathbf{k}} \circ {}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\boldsymbol{q}^{*}_{\mathrm{ref}}\right) \\
{}^{\mathrm{K}}\mathbf{k}_{\mathrm{B},0} &= 2(q_0 q_2 + q_1 q_3) \\
{}^{\mathrm{K}}\mathbf{k}_{\mathrm{B},1} &= 2(q_0 q_1 - q_2 q_3) \\
{}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{tilt}} &= \begin{bmatrix} {}^{\mathrm{K}}\mathbf{k}_{\mathrm{B},0} \\ {}^{\mathrm{K}}\mathbf{k}_{\mathrm{B},1} \end{bmatrix}
\end{aligned} \tag{R.4}$$

The length of this vector is equivalent to the sine of the inclination angle.

$$\theta_{\mathrm{tilt}} = \sin^{-1}\left(\left\| {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{tilt}} \right\|\right) \tag{R.5}$$

The direction vector of the inclination, given by the normalization of ${}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{tilt}}$, can be combined with the inclination angle to form the inclination quaternion, ${}^{\mathrm{K}}_{\tilde{\mathrm{T}}}\boldsymbol{q}_{\mathrm{ref}}$.

$${}^{\mathrm{K}}_{\tilde{\mathrm{T}}}\boldsymbol{q}_{\mathrm{ref}} = \begin{bmatrix} \cos\left(\frac{\theta_{\mathrm{tilt}}}{2}\right) \\ -\frac{{}^{\mathrm{K}}\mathbf{k}_{\mathrm{B},1}}{\left\| {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{tilt}} \right\|} \sin\left(\frac{\theta_{\mathrm{tilt}}}{2}\right) \\ \frac{{}^{\mathrm{K}}\mathbf{k}_{\mathrm{B},0}}{\left\| {}^{\mathrm{K}}\boldsymbol{p}_{\mathrm{tilt}} \right\|} \sin\left(\frac{\theta_{\mathrm{tilt}}}{2}\right) \\ 0 \end{bmatrix} \tag{R.6}$$

The above steps are similarly carried out for the current quaternion to extract the current inclination quaternion, ${}^{\mathrm{K}}_{\mathrm{T}}\boldsymbol{q}$, so that the current body z-axis rotation quaternion, ${}^{\mathrm{T}}_{\mathrm{B}}\boldsymbol{q}$, can be extracted.

$${}^{\mathrm{T}}_{\mathrm{B}}\boldsymbol{q} = {}^{\mathrm{K}}_{\mathrm{T}}\boldsymbol{q}^{*} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \tag{R.7}$$

Finally, the augmented reference quaternion, ${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}}$, is constructed by combining the desired inclination quaternion and the current body z-axis rotation quaternion.

$${}^{\mathrm{K}}_{\tilde{\mathrm{B}}}\tilde{\boldsymbol{q}}_{\mathrm{ref}} = {}^{\mathrm{K}}_{\tilde{\mathrm{T}}}\boldsymbol{q}_{\mathrm{ref}}\ {}^{\mathrm{T}}_{\mathrm{B}}\boldsymbol{q} \tag{R.8}$$

The heading reference augmentation is implemented and tested in MATLAB,[Kugle-MATLAB/Misc/HeadingIndependentReference.m](), and has later been implemented in C++ for testing on the actual ballbot, [Kugle-Embedded/KugleFirmware/Libraries/Misc/Quaternion/Quaternion.cpp#L464-L538]().

R.1.1 Small angle approximation

However, for small inclination angles the heading quaternion and body z-axis rotation quaternion will be almost equal.

$${}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}} \approx {}^{\tilde{\mathrm{T}}}_{\tilde{\mathrm{B}}}\boldsymbol{q}_{\mathrm{ref}}\Big|_{\phi,\theta,\psi \ll 10^{\circ}} \tag{R.9}$$

The above augmentation could therefore be done simpler, by assuming these two rotations to be equal. In that case the z-axis rotation quaternions are replaced by the heading quaternions, formed from the heading angle, ψ and ψ_{ref}, extracted according to (H.76).

$${}^{\mathrm{K}}_{\tilde{\mathrm{H}}}\boldsymbol{q}_{\mathrm{ref}} = \begin{bmatrix} \cos\left(\frac{\psi_{\mathrm{ref}}}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\psi_{\mathrm{ref}}}{2}\right) \end{bmatrix} \qquad {}^{\mathrm{K}}_{\mathrm{H}}\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\psi}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\psi}{2}\right) \end{bmatrix} \tag{R.10}$$

The inclination quaternion is then computed similarly and the augmented quaternion is constructed in the same way.

R.2 Quaternion derivative state augmentation

With just the augmentation of the quaternion reference the balance controller will still try to counteract yaw motion, due to the tracking of the angular velocity reference and thus the yaw angular velocity component. To make the heading mode completely independent of heading disturbances, the quaternion derivative has to be augmented by removing the yaw angular velocity component. It is decided that it makes sense to remove the inertial frame yaw angular velocity, thus the z-component of (H.52).

$$\begin{aligned} {}^{\mathrm{K}}_{\mathrm{B}}\tilde{\boldsymbol{q}} &= \left(\wedge \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 0 \end{bmatrix} \vee \left({}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q}^{*} \right) \right) \circ {}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \\ &= \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right) \begin{bmatrix} 0 & \mathbf{0}_{1\times 2} & 0 \\ \mathbf{0}_{2\times 1} & \mathbf{I}_2 & \mathbf{0}_{2\times 1} \\ 0 & \mathbf{0}_{1\times 2} & 0 \end{bmatrix} \boldsymbol{\Gamma}\left({}^{\mathrm{K}}_{\mathrm{B}}\boldsymbol{q} \right)^{\mathsf{T}} {}^{\mathrm{K}}_{\mathrm{B}}\dot{\boldsymbol{q}} \end{aligned} \tag{R.11}$$

The above augmentations enables the user to freely move and spin the ballbot around without the balance controller trying to counteract the yaw motion. The quaternion derivative state augmentation is implemented in C++ at ↗ `Kugle-Embedded/KugleFirmware/Libraries/Misc/Quaternion/Quaternion.cpp#L540-L571`.

S Model Predictive Controller

Due to time constraints the development of the shape-accelerated model predictive controller, feeding quaternion references, ${}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}$, and angular velocity references, ${}^{\tilde{\text{B}}}\boldsymbol{\omega}_{\text{ref}}$, to the balance controller, has not been described within the thesis. A path-following MPC capable of controlling and stabilizing the translational dynamics of the ballbot has been developed in MATLAB and tested and verified with Simulink using the complete closed-loop dynamics, thus the non-linear model and sliding mode balance controller. Model predictive control is generally speaking an iterative control strategy where the control law is computed at each step from the result of solving an optimization problem, usually based on a receding horizon cost function [123] [124].

The MPC in this thesis is developed on a simplified linear steady-state shape-acceleration model of the ballbot but includes non-linear constraints, e.g., for obstacle avoidance, resulting in a non-linear optimization problem. The optimization problem is described and condensed with ACADO [125] [126] [127] which solves the NLP through a series of QP problems, so-called sequential quadratic programming (SQP), using the quadratic programming solver, qpOASES [128] [128], for each individual QP problem. The ACADO toolkit is written in C and C++ but includes wrappers and documentation for MATLAB [129] and Simulink [130]. The ACADO toolkit has the benefit of being able to generate efficient C++ code to develop receding horizon model predictive controllers [131] [132]. To get started with ACADO the following tutorials and slides offers a good introductions [133] [134] [135] [136]. Other mentionable slides that gives an introduction to optimal control and how ACADO can be used to solve optimal control problems include [137] [138] [139]. Finally, the complete reference book for ROS from 2017 published by Springer [140] includes some examples on how to use the ACADO toolkit to export C++ code for the MPC.

S.1 Objective

The objective of the shape-accelerated model predictive controller is to follow a reference trajectory given from a high-level path planner by setting the quaternion reference, ${}^{\text{K}}_{\tilde{\text{B}}}\boldsymbol{q}_{\text{ref}}$, and angular velocity reference, ${}^{\tilde{\text{B}}}\boldsymbol{\omega}_{\text{ref}}$, to the balance controller. The MPC should thus be capable of controlling and stabilizing the underactuated translational dynamics of the ballbot. The reference trajectory is assumed to be given by a sequence of geometric reference points which the system should navigate through.

S.2 Trajectory tracking vs Path-following

When developing a controller to track a given reference trajectory, one would usually develop a trajectory tracking controller given a time-series of reference points to converges to [141].

$$\lim_{t\to\infty} \left\| \boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t) \right\| = 0 \tag{S.1}$$

The objective of a trajectory tracking MPC with a horizon of N steps is thus to minimize the Euclidean distance between the current robot state at step k, $(x_k(\boldsymbol{u}), y_k(\boldsymbol{u}))$ and the corresponding reference at step k, $(x_{\text{ref},k}, y_{\text{ref},k})$, by adjusting the controls, $\boldsymbol{u}$, over the horizon. Note that the control vector, $\boldsymbol{u}$, is a single vector that contains the control inputs over the full horizon.

$$J = \sum_{k=0}^{N-1} \Big(x_k(\boldsymbol{u}) - x_{\text{ref},k} \Big)^2 + \Big(y_k(\boldsymbol{u}) - y_{\text{ref},k} \Big)^2 \tag{S.2}$$

For an underactuated system the reference given to a trajectory tracking controller might easily end up running ahead of the system if the system dynamics is not considered when generating the references. An example is illustrated in Figure S.1 where the robot starts from a stand-still and has to accelerate before proper tracking is achieved. A trajectory tracking controller would be punished unnecessarily and incorrectly since the system dynamics would not allow the reference to be tracked before the system has accelerated. In worst case this would lead to an infeasible optimization problem without any solution, thereby leading to a lack of convergence of the MPC.

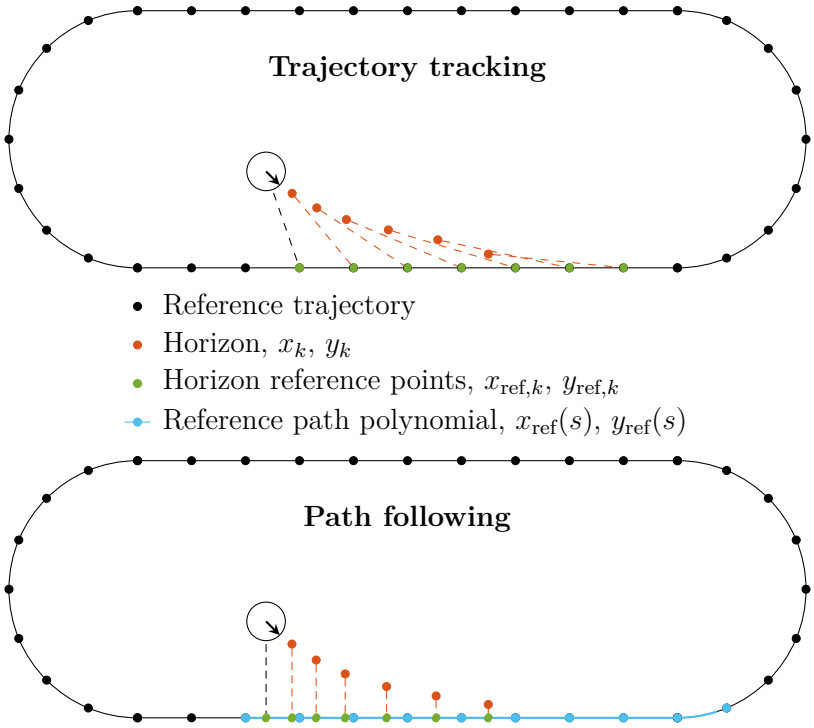


Figure S.1: Comparison of Trajectory tracking and Path-following controller during acceleration of system (starting from stand-still)

One way to solve this problem is to describe the reference trajectory as a geometric path without any preassigned timing law and then add an extra degree of freedom to the optimization problem in terms of a path parameter, $s(t)$, see (S.3) [142]. By using a distance-parametrized reference path, $\boldsymbol{p}_{\mathrm{ref}}(s)$, the path parameter, s, adds a degree of freedom to the timing law by linking the longitudinal velocity of the robot to the derivative of the path parameter. Depending on the dynamics the path parameter can be adjusted by the MPC accordingly such that the references used for computing the cost involves the lateral deviation from the path, see Figure S.1, which should be minimized to achieve tracking. Furthermore, strict forward motion can be ensured by enforcing $\dot{s} > 0$.

$$\lim_{t\to\infty} \left\| \boldsymbol{x}(t) - \boldsymbol{p}_{\mathrm{ref}}\left(s(t)\right) \right\| = 0 \tag{S.3}$$

The objective of a path-following MPC with a horizon of N steps is thus a matter of minimizing the Euclidean distance to N-points along the geometric reference path, $\left(x_{\mathrm{ref}}\left(s_k(\boldsymbol{u})\right), y_{\mathrm{ref}}\left(s_k(\boldsymbol{u})\right)\right)$, whose position along the path, $s_k(\boldsymbol{u})$, can be varied with the extra degree of freedom added to the controls.

$$J = \sum_{k=0}^{N-1} \Big(x_k(\boldsymbol{u}) - x_{\mathrm{ref}}\Big(s_k(\boldsymbol{u})\Big)\Big)^2 + \Big(y_k(\boldsymbol{u}) - y_{\mathrm{ref}}\Big(s_k(\boldsymbol{u})\Big)\Big)^2 \tag{S.4}$$

The MPC in this thesis is developed as a path-following controller rather than a trajectory-following controller. The extra degree of freedom through the space-time relationship with the path parameter, enables the MPC to choose when it is most optimal to accelerate and decelerate based on the cost function which punishes deviations from the path but also deviations from a desired velocity profile [143].

Examples of other path-following MPC problems are given in [144] [145] [146].

S.3 Path parametrization

It is assumed that the desired reference trajectory is given by an arbitrary sequence of geometric reference points in the inertial frame, since it is assumed that most global path planners returns some sort of connected graph with nodes of the references points to go to.
To simplify computation and complexity the MPC should not work on the global reference path but instead on a local path in the heading frame, extracted at each iteration
It is decided to extract a local path as an approximate arc curve length parametrized 9th-order polynomial, ensuring continuous differentiability. Other common parametrizations include Bezier curves and splines [147] [148] which are basically several piecewise polynomials, and the most common parametrization for especially holonomic vehicles is the Dubins path [149].
The first step in every iteration of the MPC is therefore to compute this local distance-parametrized polynomial from the reference points. Initially a sequence of m-number of points are extracted from the current set of reference points with the first point being the nearest point to the robot and the following points are extracted until the accumulated Euclidean distance corresponds to 1.5 of the previous predicted horizon distance, s_{end}.
Unfortunately closed-form expressions for the arc curve length [150] of a polynomial do not exist, making the path-parametrization difficult. The Euclidean distance between the reference points are computed to serve as a starting point for the arc curve length. Thereafter the reference path is parametrized with this Euclidean distance followed by a refinement using a closed-form approximation of the arc curve length [151] [152] [153].

The computation of the reference path polynomial is carried out through 4 steps as illustrated in Figure S.2.
The first step involves fitting two polynomials to the relationship between the accumulated Euclidean distance, $d_0, \ldots, d_m$, used as an initial approximation of the path distance, and the x- and y-position of the extracted m-number of points.
The resulting polynomials, $x_{\text{ref}}(d)$ and $y_{\text{ref}}(d)$, are used in the second step to compute the actual arc curve length [152] at the accumulated Euclidean distance points, $d_0, \ldots, d_m$, such that an inverse arc curve length polynomial, $d(s)$, can be fitted.
This polynomial is used in the third step to compute the corresponding distance values for n-number of evenly spaced arc curve length values, $s_0, \ldots, s_n$, such that the corresponding x- and y- reference positions can be looked up, giving the pairs

$$\begin{aligned} x_{\text{pairs}} &= \left\{ \left(s_0; x_{\text{ref}}(d(s_0))\right), \ldots, \left(s_n; x_{\text{ref}}(d(s_n))\right) \right\} \\ y_{\text{pairs}} &= \left\{ \left(s_0; y_{\text{ref}}(d(s_0))\right), \ldots, \left(s_n; y_{\text{ref}}(d(s_n))\right) \right\} \end{aligned} \tag{S.5}$$

The reference pairs from (S.5) are used in the final step to fit two refined arc curve length approximated reference polynomials, $x_{\text{ref}}(s)$ and $y_{\text{ref}}(s)$.

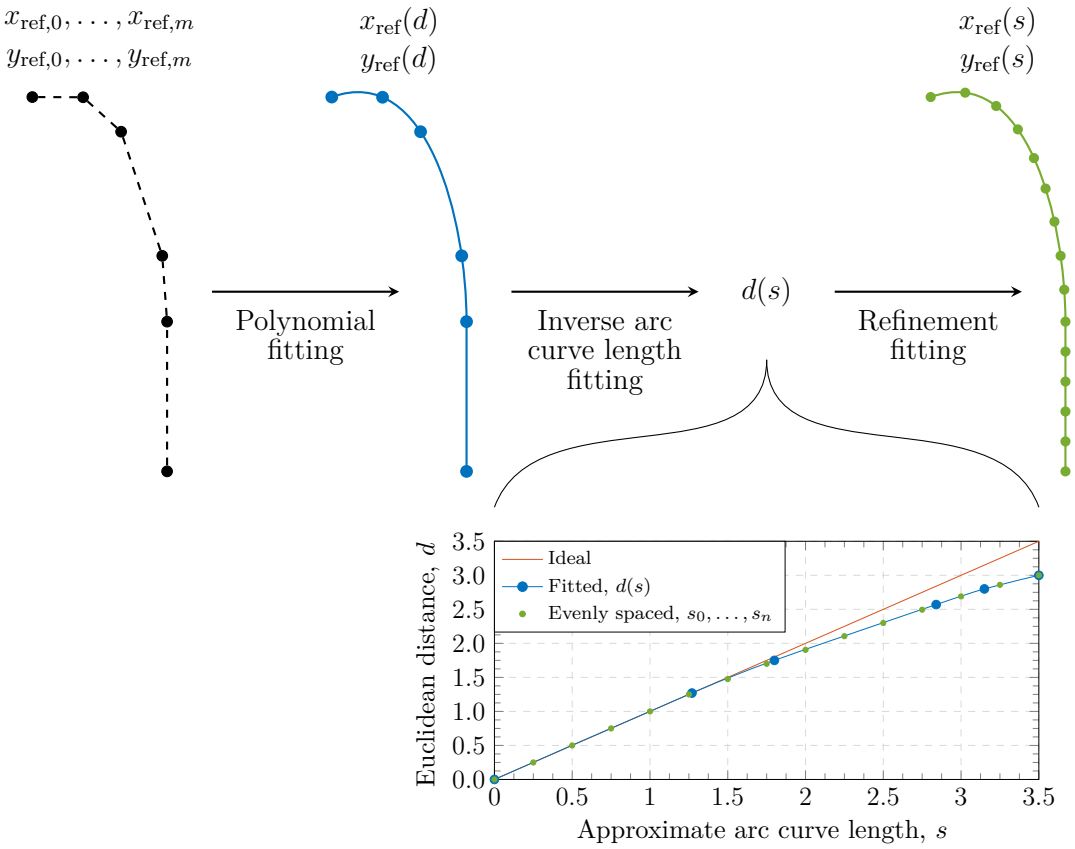


Figure S.2: Arc curve length polynomial fitting procedure for geometric reference path

After parametrization the reference path is given as:

$$\begin{aligned} {}^{\text{H}}x_{\text{ref}}(s) &= \sum_{i=0}^{9} c_{x,i} s^i \\ {}^{\text{H}}y_{\text{ref}}(s) &= \sum_{i=0}^{9} c_{y,i} s^i \end{aligned} \tag{S.6}$$

where $c_{x,i}$ and $c_{y,i}$ for $i = 0, \ldots, 9$ are the computed polynomial coefficients for the reference path.

An easy-to-use C++ library for polynomial fitting and arc curve length parametrization has been developed in `Kugle-Misc/libs/mpc/Path.cpp`.

S.4 Optimization problem

A 10-dimensional state space is used to describe the system within the MPC.

$$\boldsymbol{\chi} = \left[{}^{\text{H}}_{\text{B}}q_1 \quad {}^{\text{H}}_{\text{B}}q_2 \;\middle|\; {}^{\text{H}}x \quad {}^{\text{H}}y \;\middle|\; {}^{\text{H}}\dot{x} \quad {}^{\text{H}}\dot{y} \;\middle|\; s \quad \dot{s} \;\middle|\; {}^{\tilde{\text{B}}}\omega_{\text{ref,x}} \quad {}^{\tilde{\text{B}}}\omega_{\text{ref,y}} \right]^{\mathsf{T}} \tag{S.7}$$

where s is the path parametrization variable and $\dot{s}$ is the velocity along the path.
Note how the position and velocity states are defined in the heading frame since the MPC is designed in a local heading frame to improve feasibility and to become heading independent similar to the velocity LQR described in Appendix Q. Thereby this leaves the heading angle and heading angular velocity as a degree of freedom.

The control variables of the MPC control problem are defined below.

$$\boldsymbol{u} = \left[{}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,x}} \quad {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,y}} \quad \ddot{s} \;\middle|\; \gamma_v \quad \gamma_q \quad \gamma_o \right]^{\mathsf{T}} \tag{S.8}$$

where γ_v, γ_q and γ_o are slack variables used in the constraints to help with feasibility. γ_v is related to the velocity constraint, γ_q is related to the quaternion constraint and γ_o is related to the obstacle constraint. Slack variables are added to the constraints to allow certain constraints to be slightly violated to keep the problem feasible even though the system might be started outside a feasible region, where constraints are voided, or goes out in an infeasible region during the horizon. This helps to ensure feasibility and convergence. Note that violation of the constraints through the slack variables are punished with a very high cost.

The angular velocity reference output is included in the state vector rather than being a part of the control variables, such that the derivative of the angular velocity reference can be included in the control variables instead. By lifting the control variables in this way, constraints and cost can be applied on the derivative of the angular velocity reference, equivalent to angular acceleration which is related to torque. Constraints on ${}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,x}}$ and ${}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,y}}$ are thus related to the torque saturations of the balance controller.

Instead of using the full non-linear closed-loop model with the sliding mode controller, a simplified model, based on the linearization from Appendix Q.3, is used within the MPC. This model describes the relationship between inclination angles and resulting translational acceleration and is equivalent to the steady-state behaviour of the ballbot dynamics given a certain inclination. The optimization problem is thus subject to the following dynamics

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}{}^{\mathrm{H}}_{\mathrm{B}}q_1 &= \frac{1}{2}\,{}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}} & & \\
\frac{\mathrm{d}}{\mathrm{d}t}{}^{\mathrm{H}}_{\mathrm{B}}q_2 &= \frac{1}{2}\,{}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}} & \frac{\mathrm{d}}{\mathrm{d}t}s &= \dot{s} \\
\frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}x &= {}^{\mathrm{H}}\dot{x} & \frac{\mathrm{d}}{\mathrm{d}t}\dot{s} &= \ddot{s} \\
\frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}y &= {}^{\mathrm{H}}\dot{y} & \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}} &= {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,x}} \\
\frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}\dot{x} &= c_q\,{}^{\mathrm{H}}_{\mathrm{B}}q_2 & \frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}} &= {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,y}} \\
\frac{\mathrm{d}}{\mathrm{d}t}\,{}^{\mathrm{H}}\dot{y} &= -c_q\,{}^{\mathrm{H}}_{\mathrm{B}}q_1 & &
\end{aligned} \tag{S.9}$$

The coefficient, c_q, is the steady state acceleration coefficient extracted from the linearized matrix in (Q.19) using the values from $\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}}$ and $\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}_r}$ as shown in (S.10).

$$\begin{aligned}
\begin{bmatrix} {}^{\mathrm{H}}\ddot{x} \\ {}^{\mathrm{H}}\ddot{y} \end{bmatrix} &= \left(\boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}} + \boldsymbol{A}_{\dot{\boldsymbol{v}}\boldsymbol{q}_r} \right) \begin{bmatrix} {}^{\mathrm{H}}_{\mathrm{B}}q_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}q_2 \end{bmatrix} \\
&= \begin{bmatrix} 0 & c_{qx} \\ -c_{qy} & 0 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{H}}_{\mathrm{B}}q_1 \\ {}^{\mathrm{H}}_{\mathrm{B}}q_2 \end{bmatrix} \\
&= c_q \begin{bmatrix} {}^{\mathrm{H}}_{\mathrm{B}}q_2 \\ -{}^{\mathrm{H}}_{\mathrm{B}}q_1 \end{bmatrix}
\end{aligned} \tag{S.10}$$

Note that $c_q = 14.53$ when based on the linearization from Appendix Q.3.

The optimization problem is formulated with a least squares cost function on error terms, states and control inputs during the horizon and a final horizon cost based on only error terms and states. The cost is computed as

$$\begin{aligned} J &= \sum_{k=0}^{N-1} \left(\|\boldsymbol{h}\|_{\boldsymbol{W}} \right) + \|\boldsymbol{h}_N\|_{\boldsymbol{W_N}} \\ &= \sum_{k=0}^{N-1} \left(\boldsymbol{h}^\mathsf{T} \boldsymbol{W} \boldsymbol{h} \right) + \boldsymbol{h}_N^\mathsf{T} \boldsymbol{W_N} \boldsymbol{h}_N \end{aligned} \tag{S.11}$$

where the two cost vectors, $\boldsymbol{h}$ and $\boldsymbol{h}_N$, are defined below.

$$\boldsymbol{h} = \left[e_{\text{lon}} \quad e_{\text{lat}} \quad e_{\text{vel}} \quad e_{\text{end}} \quad e_{\text{obs}} \;\middle|\; {}^{\text{H}}_{\text{B}}q1 \quad {}^{\text{H}}_{\text{B}}q2 \quad {}^{\tilde{\text{B}}}\omega_{\text{ref,x}} \quad {}^{\tilde{\text{B}}}\omega_{\text{ref,y}} \;\middle|\; {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,x}} \quad {}^{\tilde{\text{B}}}\dot{\omega}_{\text{ref,y}} \quad \gamma_v \quad \gamma_q \quad \gamma_o \right]^\mathsf{T}$$

$$\boldsymbol{h}_N = \left[e_{\text{lon}} \quad e_{\text{lat}} \quad e_{\text{vel}} \quad e_{\text{end}} \quad e_{\text{obs}} \;\middle|\; {}^{\text{H}}_{\text{B}}q1 \quad {}^{\text{H}}_{\text{B}}q2 \quad {}^{\tilde{\text{B}}}\omega_{\text{ref,x}} \quad {}^{\tilde{\text{B}}}\omega_{\text{ref,y}} \right]^\mathsf{T} \tag{S.12}$$

The cost is based on three different parts; error cost, state cost and control cost, separated with the vertical lines. Note that the control cost is not included in the final horizon cost, $\boldsymbol{h}_N$, since controls are only computed for each step within the horizon, $k = 0, \ldots, N-1$.

S.4.1 Longitudinal and Lateral error

A global tracking controller would normally be defined from the position tracking error

$$\begin{aligned} e_x &= {}^{\text{H}}x - {}^{\text{H}}x_{\text{ref}}(s) \\ e_y &= {}^{\text{H}}y - {}^{\text{H}}y_{\text{ref}}(s) \end{aligned} \tag{S.13}$$

However, since the MPC in this case should follow a geometric path and a desired velocity reference, the path-following error is defined by a longitudinal and lateral error as illustrated in Figure S.3.

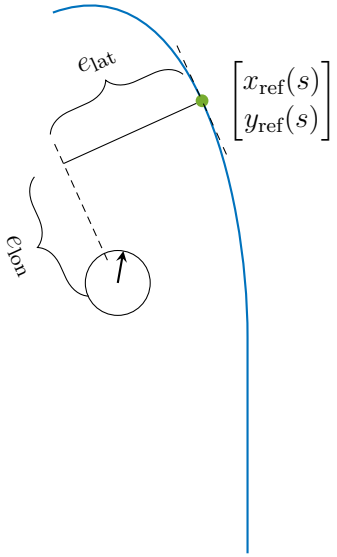


Figure S.3: Definition of longitudinal and lateral error to the geometric reference path

The lateral error is defined as the perpendicular deviation from a given point on the reference path to a projection of the current position of the robot. This can be computed as the dot product between the tracking error and a lateral direction vector defined in the heading frame which is pointing away from the trajectory to the left when looking in the direction of travel.

$$e_{\text{lat}} = \begin{bmatrix} -\sin(\psi_{\text{ref}}) \\ \cos(\psi_{\text{ref}}) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} e_x \\ e_y \end{bmatrix} \tag{S.14}$$

The lateral direction vector is based on the direction angle of the path at the given point, ψ_{ref}. Since the reference path is distance parametrized and it is a polynomial with continuous derivatives, this direction angle angle can be computed as

$$\psi_{\text{ref}} = \text{atan2}\left({}^{\text{H}}\dot{y}_{\text{ref}}(s),\ {}^{\text{H}}\dot{x}_{\text{ref}}(s) \right) \tag{S.15}$$

Constructing the lateral direction vector requires the cosine and sine of this direction angle and the atan2 relationship can thus be replaced by

$$\begin{aligned} \cos(\psi_{\text{ref}}) &= \frac{{}^{\text{H}}\dot{x}_{\text{ref}}(s)}{\sqrt{{}^{\text{H}}\dot{x}_{\text{ref}}(s)^2 + {}^{\text{H}}\dot{y}_{\text{ref}}(s)^2}} \\ \sin(\psi_{\text{ref}}) &= \frac{{}^{\text{H}}\dot{y}_{\text{ref}}(s)}{\sqrt{{}^{\text{H}}\dot{x}_{\text{ref}}(s)^2 + {}^{\text{H}}\dot{y}_{\text{ref}}(s)^2}} \end{aligned} \tag{S.16}$$

Conversely the longitudinal error is defined as the deviation in the same direction as the reference path and is thus related to how much the reference is either lagging behind or is in front of the system. This can therefore be computed as the dot product between the tracking error and a longitudinal direction vector positive forward.

$$e_{\text{lon}} = \begin{bmatrix} \cos(\psi_{\text{ref}}) \\ \sin(\psi_{\text{ref}}) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} e_x \\ e_y \end{bmatrix} \tag{S.17}$$

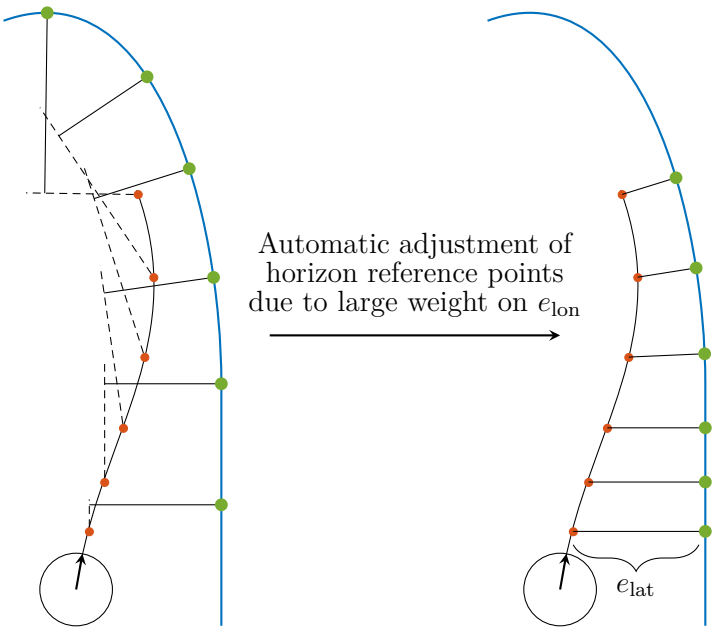


Figure S.4: Longitudinal reference alignment ensures that the lateral error corresponds to the actual lateral deviation from the reference path

Reference trajectory points should align with predicted horizon states and the longitudinal error should therefore be close to zero. Since the path parameter, s, can be almost arbitrarily adjusted over the horizon with the second derivative, $\ddot{s}$, the longitudinal alignment is enforced by applying a large weight on e_{lon}, e.g. $W_{\text{lon}} \gg 100 W_{\text{lat}}$, as illustrated in Figure S.4.

This will ensure that the reference points, marked with the green dots, used for computation of the lateral error, is computed correctly according to the predicted locations of the robot.

S.4.2 Velocity error

A velocity error term is included in the cost to ensure velocity tracking of a desired longitudinal velocity while simultaneously ensuring progress indirectly.
From the inertial frame velocity, the longitudinal velocity component is computed by taking the dot product with the longitudinal direction vector as used previously.

$$v_{\text{lon}} = \begin{bmatrix} \cos(\psi_{\text{ref}}) \\ \sin(\psi_{\text{ref}}) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} {}^{\text{H}}\dot{x} \\ {}^{\text{H}}\dot{y} \end{bmatrix} \tag{S.18}$$

Then the velocity error is defined as the difference to the desired velocity, v_{ref}.

$$e_{\text{vel}} = v_{\text{lon}} - v_{\text{ref}} \tag{S.19}$$

S.4.3 Away-from-end error

An away from end error is added to ensure progress as well, even though it will easily affect the velocity tracking if the weight is within the same magnitude as the weight on the velocity error.

$$e_{\text{end}} = s - s_{\max} \tag{S.20}$$

The error pushes the system towards the end of the fitted trajectory by rewarding the driven distance related to the path parameter, s.

S.4.4 Obstacle avoidance

Obstacles are handled by incorporating them as constraints on the position in terms of a circular keep-out area. The obstacles are assumed to be circular and defined with a given center, $({}^{\text{H}}x_{o,i}, {}^{\text{H}}y_{o,i})$ and radius, r_i, as illustrated in Figure S.5.

This enables a proximity metric, $p_{\text{o,i}}$, to be computed, indicating the current distance to an obstacle.

$$p_{\text{o,i}} = \sqrt{\left({}^{\text{H}}x - {}^{\text{H}}x_{o,i}\right)^2 + \left({}^{\text{H}}y - {}^{\text{H}}y_{o,i}\right)^2} - r_i \tag{S.21}$$

Currently the MPC handles up to five nearby obstacles. The obstacle constraints are defined as:

$$\begin{aligned} p_{\text{o,i}} &\geq -\gamma_o \quad \text{for } i = 0, \ldots, 4 \\ \gamma_o &\geq 0 \end{aligned} \tag{S.22}$$

An obstacle avoidance error term is also added to push the robot away from the obstacle, as the MPC would otherwise make the robot drive on the constraint boundary making the system very vulnerable to disturbances and noise.

$$e_{\text{obs}} = \sum_{i=0}^{4} \mathrm{e}^{k_p(-p_{\text{o,i}} + c_p)} \tag{S.23}$$

where k_p and c_p are the gain and offset that defines the exponential barrier function used to push the robot away.

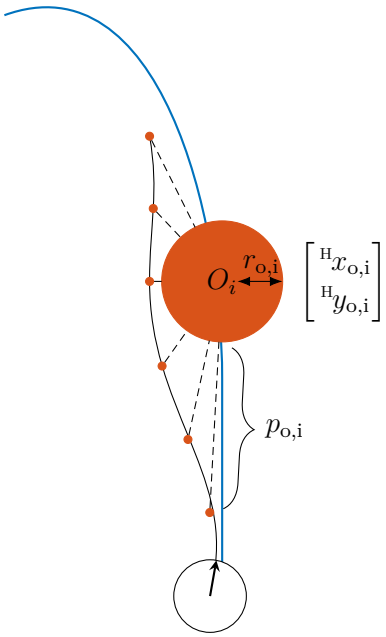


Figure S.5: Predicted horizon when doing obstacle avoidance around an obstacle defined by its origin and radius.

The pre-processing algorithm takes care of extracting nearby obstacles and compressing them into a description of five circular obstacles that can be provided to the MPC. A real-time simulation of the MPC doing obstacle avoidance is made in `Kugle-Misc/src/MPC_Test`.

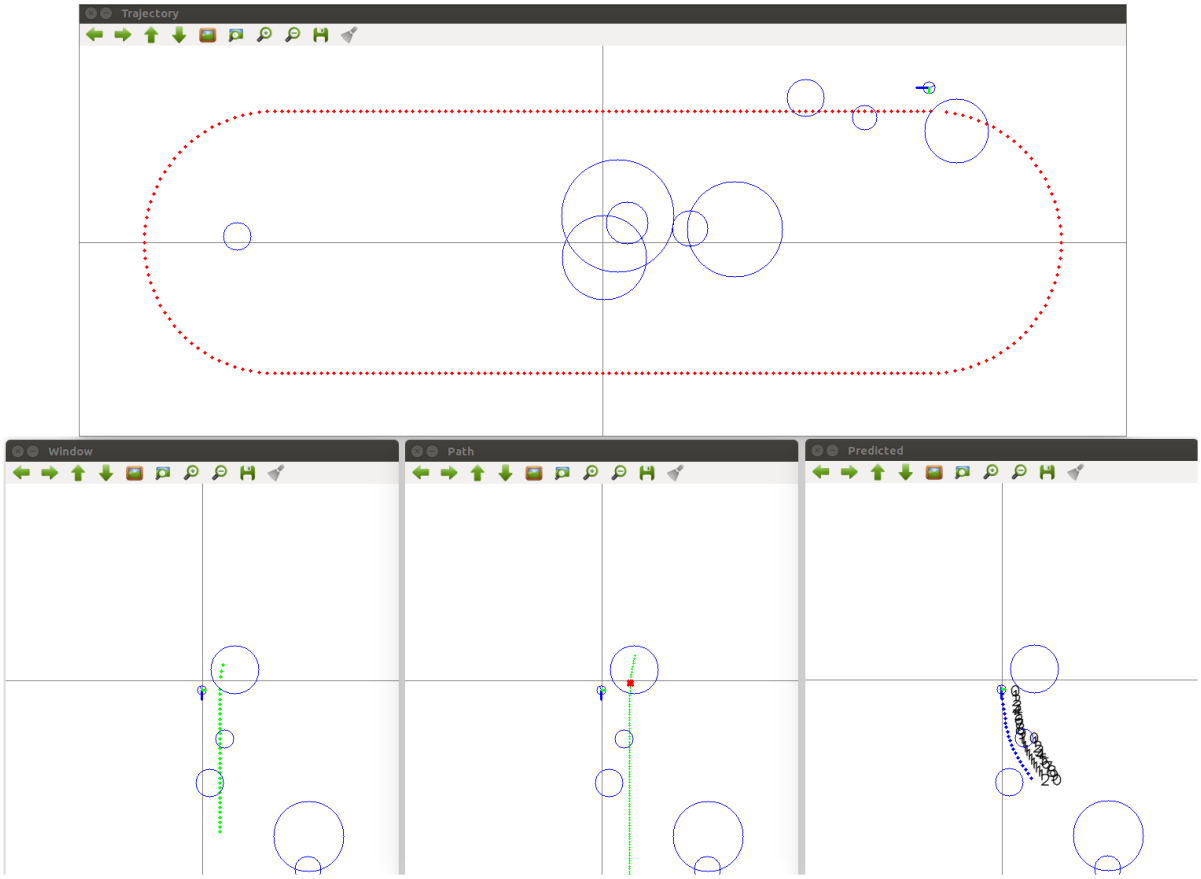


Figure S.6: 2D simulation of shape-accelerated path-following MPC performing obstacle avoidance

The simulation includes a top-down view as shown in Figure S.6 with the ballbot drawn as a circle with a blue line pointing in the velocity direction and a green arrow indicating the inclination direction, and thus also the direction of acceleration.

The red dots marks the discrete points of the reference trajectory. At each iteration the MPC extracts a window of the nearest reference trajectory points, shown as the green points in the bottom left view. These points are then parametrized into a path polynomial, visualized as the green path in the bottom center view. Finally, the MPC predicts a horizon of 20 steps as shown in the bottom right view.

S.4.5 Constraints

Finally, the constraints are defined as

$$\begin{aligned}
-q_{\max} &\leq {}^{\mathrm{H}}_{\mathrm{B}}q_1 &&\leq q_{\max} \\
-q_{\max} &\leq {}^{\mathrm{H}}_{\mathrm{B}}q_2 &&\leq q_{\max} \\
-\omega_{\max} &\leq {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}} &&\leq \omega_{\max} \\
-\omega_{\max} &\leq {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}} &&\leq \omega_{\max} \\
-\dot{\omega}_{\max} &\leq {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,x}} &&\leq \dot{\omega}_{\max} \\
-\dot{\omega}_{\max} &\leq {}^{\tilde{\mathrm{B}}}\dot{\omega}_{\mathrm{ref,y}} &&\leq \dot{\omega}_{\max}
\end{aligned}
\qquad
\begin{aligned}
0 \quad &\leq s && \leq s_{\max} \\
&\dot{s} && \geq 0 \\
&v && \leq v_{\max} + \gamma_v \\
&p_{\mathrm{o,i}} && \geq -\gamma_o \\
&\gamma_v && \geq 0 \\
&\gamma_q && \geq 0 \\
&\gamma_o && \geq 0
\end{aligned}
\tag{S.24}$$

where constraints are applied to both state and control variables. The maximum quaternion constraint helps to limit the maximum inclination to an angle of $\theta_{\max}$, which in return can be related to a maximum acceleration.

$$q_{\max} = \sin\left(\frac{1}{2}\theta_{\max}\right) + \gamma_q \tag{S.25}$$

The velocity constraint is defined to limit the Euclidean velocity, as defined in (S.26), to an upper bound, $v_{\max}$.

$$v = \sqrt{{}^{\mathrm{H}}\dot{x}^2 + {}^{\mathrm{H}}\dot{y}^2} \tag{S.26}$$

The constraints on the angular velocity reference and the derivative smoothens the resulting trajectory. It was especially necessary to add $\dot{\omega}_{\max}$ to limit the maximum angular acceleration which is related to torque saturations in the balance controller.

As mentioned previously strict constraints do not work well with a system with disturbances and model uncertainties, resulting in a infeasible optimization problem. This is why the slack variables, γ_v, γ_q and γ_o, are introduced to ensure feasibility even though the problem is initialized in or goes through an infeasible region.

The maximum path parameter value, $s_{\max}$, is set to bound the optimization problem to the valid range of the fitted polynomial, so that it does not go outside of the fitted region. Overall the fitted polynomials are bad at generalizing outside of their fitted domain, so the path parameter should generally be constrained to avoid the references exploding.

A set of terminal constraints are added to help ensure stability and feasibility. The quaternion elements, ${}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_1$ and ${}^{\mathrm{H}}_{\mathrm{B}}\ddot{q}_2$, should both be zero, corresponding to the ballbot being upright at the end of the horizon. Furthermore the angular velocity references, ${}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}}$ and ${}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}}$, should also be zero such that the upright quaternion remains, since the controls are zero.

Finally, the inequality constraints on the Euclidean velocity and the driven distance, which should be kept within the parametrized polynomial, are also required as terminal constraints.

$$\begin{aligned} {}^{\mathrm{H}}_{\mathrm{B}}q_1[N] &= 0 \\ {}^{\mathrm{H}}_{\mathrm{B}}q_2[N] &= 0 \\ {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,x}}[N] &= 0 \\ {}^{\tilde{\mathrm{B}}}\omega_{\mathrm{ref,y}}[N] &= 0 \\ v[N] &\leq v_{\max} \\ s[N] &\leq s_{\max} \end{aligned} \tag{S.27}$$

S.5 Weights

The individual costs are defined by the entries in the weighting matrices, $\boldsymbol{W}$ and $\boldsymbol{W}_N$. The matrices are chosen to be diagonal matrices whose diagonal elements thus define the weight on each individual cost term from (S.12).

S.5.1 Weight considerations

A few considerations with regards to choosing the optimal weights have been observed during simulation and tuning.

- W_{lat}, W_{obs} and W_v limits how big avoidable obstacles can be.
- $W_{\dot{\omega}}$ and W_ω control aggressiveness.
- W_v, $W_{\dot{\omega}}$, W_ω and W_q defines how fast the velocity tracking converges.
- W_{end} should be much smaller than W_v otherwise velocity tracking fails.
- W_{end} can even be set to zero as long as W_v is sufficiently large to avoid obstacles without stopping (related to the first bullet).

S.5.2 Chosen weights

The following weights, found during simulation with different test trajectories and randomly placed obstacles, are deemed the best.

$$\begin{aligned} W_{\mathrm{lon}} &= 9999 \\ W_{\mathrm{lat}} &= 25 \\ W_{\mathrm{vel}} &= 80 \\ W_{\mathrm{end}} &= 0 \\ W_{\mathrm{obs}} &= 20 \end{aligned} \qquad \begin{aligned} W_q &= 10 \\ W_\omega &= 20 \end{aligned} \qquad \begin{aligned} W_{\dot{\omega}} &= 100 \\ W_{\gamma_v} &= 99999 \\ W_{\gamma_q} &= 9999 \\ W_{\gamma_o} &= 9999 \end{aligned} \tag{S.28}$$

S.6 Implementation

The MPC is developed and generated using ACADO [125] [126] [127], as described in the beginning, and the MPC is run at a sample rate of $f = 10$ Hz and with a horizon of $N = 20$ samples.

The cost function and constraints are described with the ACADO language in MATLAB, see `Kugle-MATLAB/Controllers/MPC/GenerateMPC.m`. This MATLAB script exports the corresponding C++ code for real-time execution of the MPC while simultaneously compiling this into a MEX file for use in MATLAB.
This MEX file is used in both the independent simulation script at `Kugle-MATLAB/Controllers/MPC/SimulateMPC.m` and within the closed-loop Simulink simulation at `Kugle-MATLAB/Simulation/KugleSim_MPC.slx` which includes the balance controller, the estimators and the non-linear model.
The exported C++ code is furthermore tested in `Kugle-Misc/src/MPC_Test` with the corresponding wrapper library in `Kugle-Misc/libs/mpc/MPC.cpp`.

The individual parts of the MPC control loop, including the path-parametrization and how the MPC optimization problem is solved with the ACADO-generated code, is illustrated in Figure S.8.

S.7 ROS simulation

The MPC has been integrated within the ROS navigation stack [154] as a new local planner, see [43]. This allows the operator to set a navigation goal through standard ROS user interfaces, e.g., *rqt*, and the MPC will then follow the trajectory coming from the global planner by fitting local path polynomials to this in real-time. The MPC ROS code is implemented as a C++ library in `Kugle-ROS/kugle_mpc`.

To simulate and test the ROS implementation a Gazebo simulation environment of the high-level shape-accelerated dynamics of the ballbot has been developed, see Figure S.7.

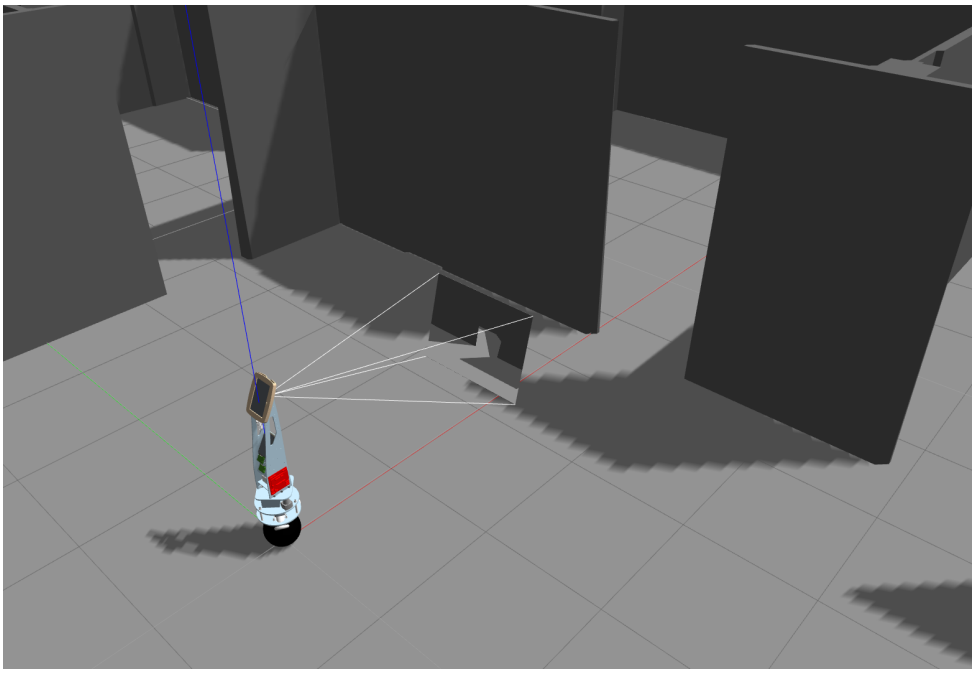


Figure S.7: Gazebo simulation of Kugle V1 within a model of the control lab at Aalborg University

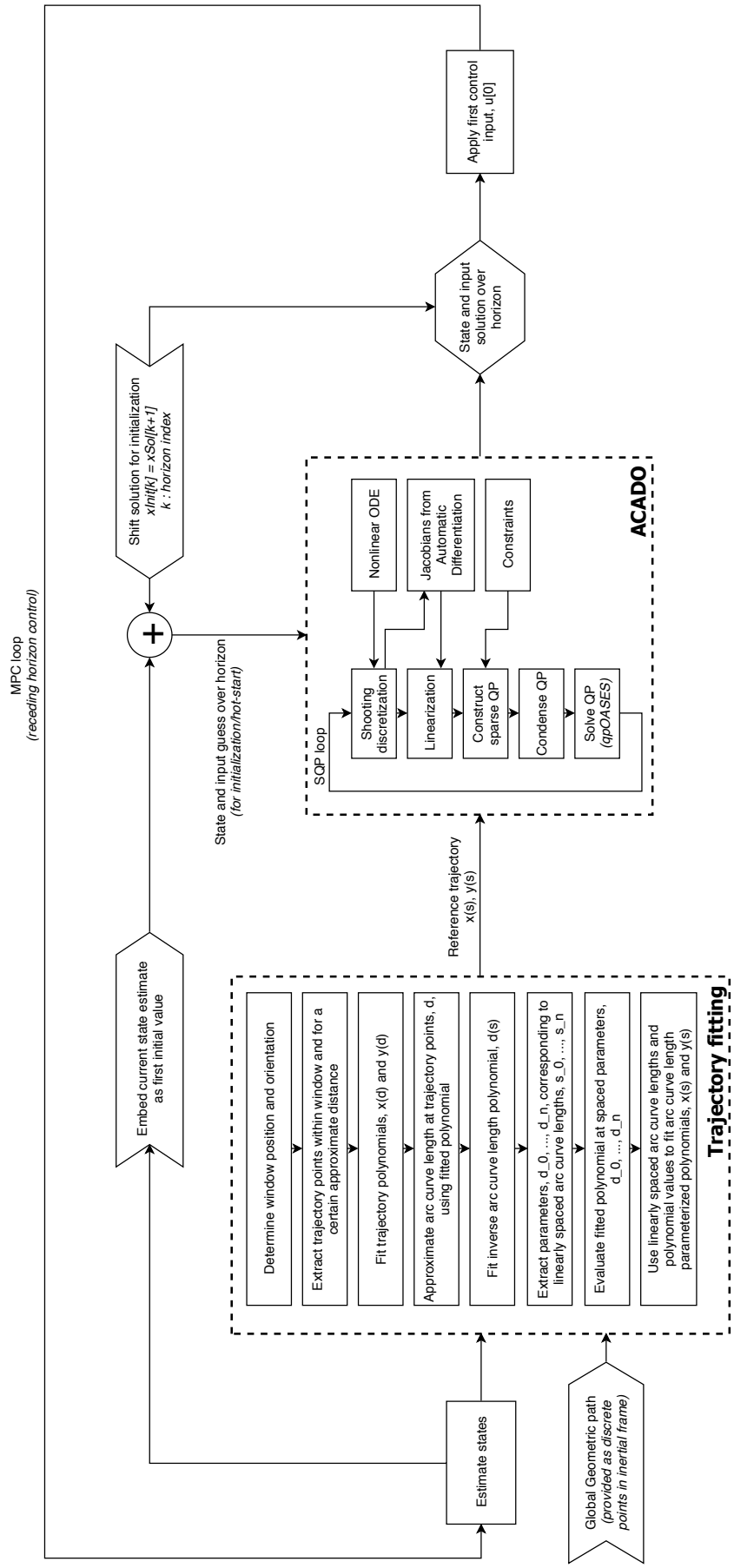
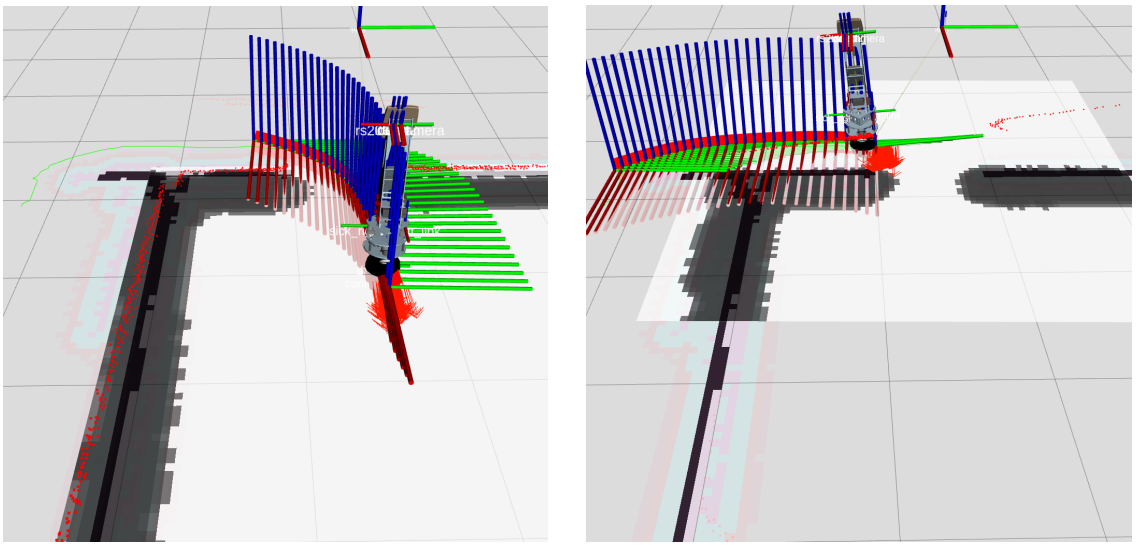


Figure S.8: MPC control loop

Since the MPC has also been tested and verified in Simulink with the closed-loop dynamics, it is deemed more important to use Gazebo to simulate high-level behaviours and sensor measurements.

The Gazebo simulation uses the CAD model of Kugle V1 on which two simulated LiDAR sensors, a camera and IMU are placed. The Gazebo model of the ballbot takes in the same type of ROS messages as the ROS driver, `Kugle-ROS/kugle_driver`, running on the actual ballbot and is thereby developed to be a complete 1-to-1 test environment.

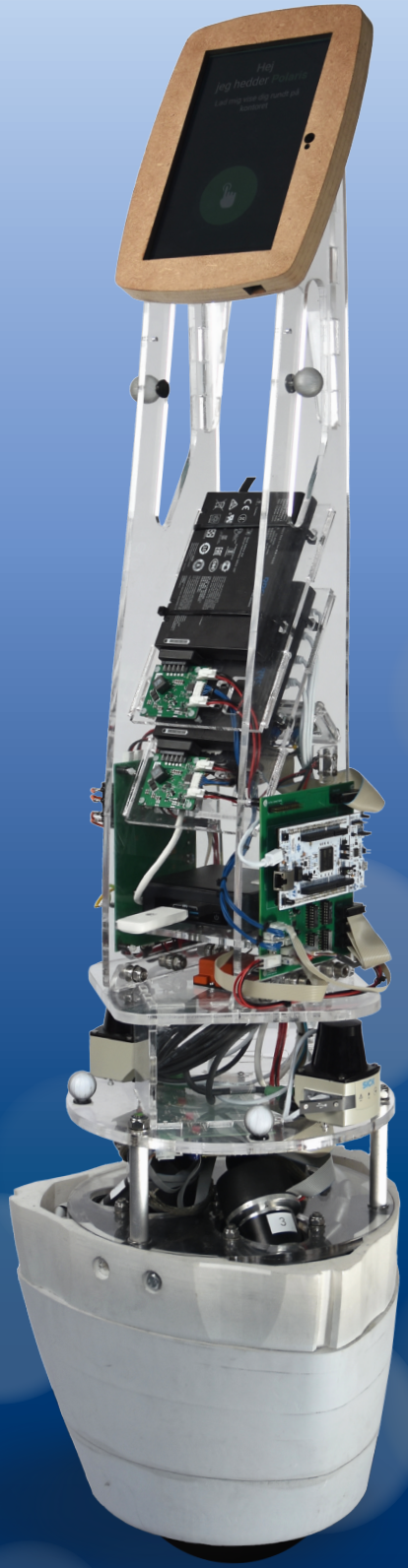
Note, however, that the Gazebo model does not include the rotational dynamics and thus takes in angular velocity references which are integrated kinematically leading to an exact angle of the robot, thereby assuming perfect tracking of the balance controller. The Gazebo simulation can be found in `Kugle-Gazebo`. An rviz view of the simulated ballbot running the MPC in real-time as the local planner, is shown in Figure S.9.



(a) Ballbot moving out through door passage

(b) Notice how the ballbot tilts to accelerate, which is also visualized in the rviz view

Figure S.9: MPC running on the simulated Kugle robot. The visualized transforms shows the predicted horizon, including both position and orientation. Note how the ballbot tilts to accelerate after having passed through the door.



AALBORG UNIVERSITY
STUDENT REPORT