CHALMERS
UNIVERSITY OF TECHNOLOGY

Modeling and Control of a Three-Dimensional Inverted Pendulum

A project based on Cubli featuring Kane's method and Model Predictive Control

Master's thesis in Systems, Control and Mechatronics

MARTIN PÄR-LOVE PALM
DAVID WALL

Department of Signal and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:06

Modeling and Control of a Three-Dimensional Inverted Pendulum

A project based on Cubli featuring Kane's method and Model Predictive Control

MARTIN PÄR-LOVE PALM
DAVID WALL

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Division of System, Control and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2017

Modeling and Control of a
Three-Dimensional Inverted Pendulum
A project based on Cubli featuring Kane's method and
Model Predictive Control
Martin Pär-Love Palm
David Wall

© Martin Pär-Love Palm
David Wall , 2017.

Supervisor: Thomas Hedberg, Combine Control System AB
Examiner: Sébastien Gros, Signals and Systems

Master's Thesis 2017:06
Department of Signals and Systems
Division of System, Control and Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The inverted three-dimensional pendulum named Cubex balancing on a corner. Image credits to Jessica Karlsson.

Printed by [Chalmers Library, Reproservice]
Gothenburg, Sweden 2017

Modeling and Control of a
Three-Dimensional Inverted Pendulum
A project based on Cubli featuring Kane's method and
Model Predictive Control
Martin Pär-Love Palm
David Wall
Department of Some Subject or Technology
Chalmers University of Technology

Abstract

The two dimensional inverted pendulum has long served as a typical classroom and textbook example when teaching classical mechanics or control theory. Then variants of the spinning top and gyroscopes have fascinated the general public, engineers and scientists for centuries. The Cubli platform developed by ETH in Switzerland takes the latter two a step further by combining them into a *three-dimensional inverted pendulum*. Three reaction wheels mounted inside a cube shaped frame enables the platform to among other things rise up to and balance on a corner. As an advanced mechatronics system, it serves well for evaluating various aspects related to control methods and sensors. Here, a platform called *Cubex*, being similar to Cubli, was used. A recently presented novel approach to Model Predictive control was evaluated and deployed. Furthermore a new approach for finding an offset of the mass center of this system and a complete general procedure for correcting misaligned Inertial Measurement Units (IMU's) have been developed and successfully implemented. Also, while ETH has released short form conference papers regarding the analytical modeling of the Cubli platform, this work aims to explain the modeling process in more detail using Kane's method. Most relevant artifacts from the project such as source code and modeling files are released in open source repositories.

Keywords: Cubli, Kane's method, Model Predictive Control, Sensor fusion, Rigid body dynamics.

Acknowledgements

We would like to thank our supervisor Thomas Hedberg and examiner Sébastien Gros for your support and interest in the project. A lot of credits has to be given to everybody freely sharing their scientific results in this world, from Euclides to Newton to present researchers. We would also like to praise the Swedish educational system that is free from tuition fees. Without it neither of us would likely have had the huge privilege to attend university for five years and share this little piece of writing with the rest of the world.

Martin Pär-Love Palm

David Wall , Gothenburg, June 2017

Contents

Nomenclature

v	Regular script indicates v is either a scalar or a matrix.
$\boldsymbol{v}$	Bold script indicates $\boldsymbol{v}$ is a vector.
$\boldsymbol{v}_1 \times \boldsymbol{v}_2$	Cross or vector product between two geometric vectors.
$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2$	Dot or scalar product between two vectors.
${}^{B}\boldsymbol{v}$	Vector $\boldsymbol{v}$ expressed in reference frame B . This is a full notation that is frequently omitted in favor of the shortform $\boldsymbol{v}$.
$(\boldsymbol{v}_1; \boldsymbol{v}_2)$	Concatenation of two column vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ by a `Matlab` inspired syntax.
$\dot{v}, \ddot{v}$	First and second derivative with respect to time.
$\|\boldsymbol{v}\|_n$	Is the n-norm of the vector $\boldsymbol{v}$. An omitted subscript n means it is the default 2-norm.

1
Introduction

1.1 Background

Gyroscopes, spinning tops and alike have baffled and amused the general public, engineers and scientists for centuries. Cubli [1], developed by ETH (Eidgenössische Technische Hochschule) in Zürich Cubli combines the former two concepts . It could be described as spinning top with no less than three gyroscopic wheels within it. Then last year students at *Chalmers University of Technology* decided to build a similar platform for their master thesis [2] that we now chose to call Cubex.
While there is no practical use for Cubex itself today, many aspects covered during the Cubex and Cubli projects have direct applications in projects regarding mechatronics and control system. The main purpose of this thesis is evaluating a solver for Model Predictive Control (MPC) that was described in a recent paper [3]. Also, Cubex can be a great general platform for trying out approaches to for instance control- and sensor fusion algorithms. Within this thesis we present our own approach to correct misaligned Inertial Measurement Units using a system identification technique.
In the Cubli reports [1] the approach for modeling presented is useful for many other systems. This approach is evaluated and explained in more detail in this report. A literature study was carried out in this context and a compact formulation of Kane's method on matrix form is presented. We are confident to say that this formulation is very practical for many modeling applications, but not easy to find summarized.

1.2 Purpose

The main purpose is developing and analyzing an inverted 3D pendulum called Cubex. This platform is then intended to be used for experiments and research in advanced mechatronics. To demonstrate how Cubex platform can be used in research, a novel approach to Model Predictive Control is tested on it. A few other algorithms useful for either Cubex or other mechatronic systems have been developed.

1.3 Ethical aspects

In our first days at the university the lecturer in mathematics described linear algebra as "extremely powerful" with a large amount of seriousness in the voice. Some people

started laughing, as they thought it sounded funny that for the time being rather incomprehensible symbols within brackets could be "extremely powerful". Today we know better. Mathematics helps us look into the mysteries of the universe, unraveling everything from abstractions as atoms to how the suns warm our skin during bright day. Like a powerful telescope, it helps us zoom into the mysteries of physics and see fantastic phenomenons as conservation of angular momentum clear before our eyes. Being able to better understand the universe, we gain further possibilities to use it as a tool manipulate it. As any tool, it can be used for the good or the bad of the humanity. While these tools are used for for instance medical advancements, they are also what enabled elaborated plans for pulverizing the earth and even the moon multiple times. With these tools there comes power and responsibility. Having had the privilege to learn to handle some of these tools we owe something to the world. We owe the world using these tools with responsibility, in a way that benefits humanity and and the planet. Do you handle your powers and privileges responsibly?

1.4 Open source repositories

An open source git repository with all software developed during the project can be found at `https://MarzCoder@bitbucket.org/MarzCoder/cubex.git`. This contains the files for generating code to the Cubex platform using `Simulink`. It also contains `Mathematica` notebooks related to the anlaytical modeling of the system. It is released under a GNU license meaning you have to share any changes you do might to the repository if you fork it. Furthermore a Google Drive repository was used during the project that contains further information. Contact the authors to request access to this.

2
Overview

The main aim of this project was evaluating control strategies using the Cubli based Cubex platform. Therefore it is natural to gain overview of this project beginning with the overview of the control loop and hardware shown in Figure 2.1. The control system of choice indirectly determines a torque vector $\boldsymbol{T}$ that acts on the cube casing and motors. Then the behavior of the system is governed by rigid body dynamics. A user that is familiar with Euler's laws of rotation of will probably finding it being useful to first inspect the final equation of motion in Section 4.5.5. As will be evident, this rather complex MIMO is hard to control without a model based approach. Therefore a large part of the report is dedicated to describing the modeling in Section 4. It seems that modeling of this system using Kane's method has not been explained with such detail in literature before. A reader who wants to model this or a similar system, will probably find our results from a literature study regarding Kane's method on matrix form to be useful. This is presented in Section 3.6.3

The model is linearized and used for control system synthesis in Section 6. Here a novel approach to fast approximate gradient MPC (Model Predictive Control) is explored and compared to LQR (Linear Quadratic Regulator) operation.

For fast and efficient software development more graphical programming using `Simulink` with the `Waijung` Blockset was used rather than writing code. The main benefits with this in our opinion is that the developer gains better overview and that the software can be simulated more easily. This, together with some details regarding the electrical system and the hardware in general are found in Section5.

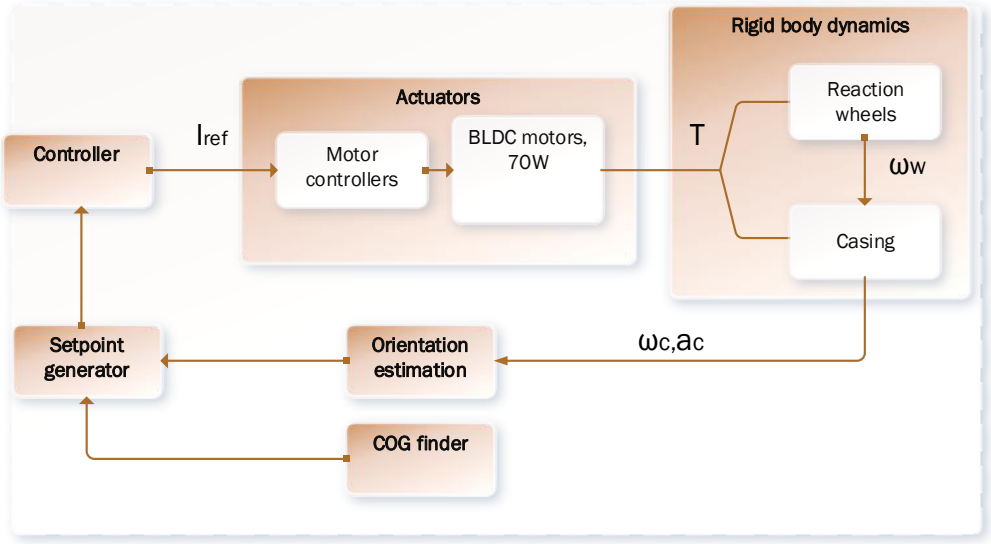


Figure 2.1: An overview of the control is used for orientation in this report

In order to balance the cube on a corner and such, accurate and fast estimates of orientation are required. This will be a very important aspect for anybody wanting to replicate this project. We used Inertial Measurement Units for the purpose. These need proper alignment, or at least knowledge about how they are aligned. An algorithm for finding a rotation matrix related to this is found in Section 8.3 together with some suggestions on how to reduce Gaussian noise . Although an accurate orientation estimate is at hand, balancing action might be flawed by the center of gravity being slightly offset. Algorithms for finding the offset of the center of gravity and/or mitigating the effects of it are found in Section 7. Cubex cannot balance properly without such an algorithm.

3

Theoretical background

A brief introduction to some theoretical concepts used in various parts of the report.

3.1 Rotation matrices

A rotation matrix R_B^A is a matrix that will make some vector ${}^B\boldsymbol{r}$ expressed in some Cartesian coordinate frame B be expressed in another Cartesian coordinate frame A. Beginning with a the two dimensional case as depicted in Figure 3.1.

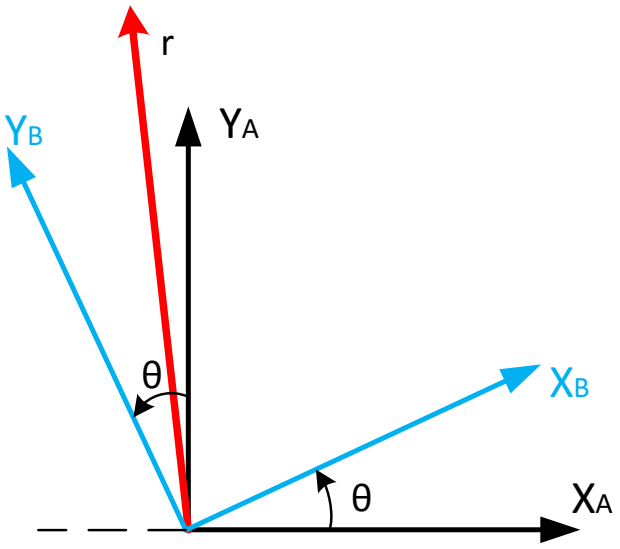


Figure 3.1: The vector $\boldsymbol{r}$ can be expressed in various coordinate systems more easily by using rotation matrices

Here the vector $\boldsymbol{r}$ having the length r can either be expressed in either the coordinate system (X_A, Y_A) or (X_B, Y_B). Say the vector $\boldsymbol{r}$ expressed in (X_B, Y_B) as ${}^B\boldsymbol{r} = \begin{bmatrix} r_{xB} & r_{yB} \end{bmatrix}^T$.
Using basic trigonometry we find that the ${}^B\boldsymbol{r}$ can be expressed in (X_A, Y_A) by

$$r_{xA} = \cos(\theta) r_{xB} - \sin(\theta) r_{yB} \tag{3.1}$$
$$r_{yA} = \sin(\theta) r_{xB} + \cos(\theta) r_{yB} \tag{3.2}$$

or more compactly on matrix form

$$^{A}\boldsymbol{r} = R_B^A \, ^{B}\boldsymbol{r} = R(\theta) \, ^{B}\boldsymbol{r} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} ^{B}\boldsymbol{r} \tag{3.3}$$

This is the basic idea for two dimensions, which has a rather straight forward extension to three dimensions where rotations around the X, Y, Z axises respectively are found as

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

These are so called *elementary rotation matrices* describing rotation around just one axis. A rotation matrix R_B^A contain the orthogonal unit vectors describing the Cartesian coordinate system B expressed in the coordinate system A, so

$$R_B^A = \begin{bmatrix} ^{A}\boldsymbol{e}_x^B & ^{A}\boldsymbol{e}_y^B & ^{A}\boldsymbol{e}_z^B \end{bmatrix}$$

A consequence of this is orthogonality between the column vectors of the matrix according to

$$^{A}\boldsymbol{e}_x^B \times ^{A}\boldsymbol{e}_y^B = ^{A}\boldsymbol{e}_z^B$$

Note how the rotation axises in the elementary rotation matrices (3.4) remain unchanged. Some key properties of rotation matrices are

- Their inverse is their transpose so multiplying a rotation matrix with its transpose yields an identity matrix according to $RR^T = I$
- Multiple rotation matrices can be multiplied together. A rotation matrix from some frame 3 to a frame 1 could then be formed by $R_3^1 = R_2^1 R_3^2$
- A consequence of the previous is that transposing order of rotation so $(R_B^A)^T = R_A^B$

These results are proven in for instance [4]

3.2 Vector derivatives

There are some analytical relations that are useful when differentiating geometric vectors. For this context, we are mostly interested in obtaining various expressions

for the velocity and acceleration for different points of the cube. Beginning by examinating Figure 3.2. A corner of the cube c is designated as a general reference point, as this point is assumed to be a static contact point. This is the corner around which the cube is assumed to be balancing.
Generally, let $\boldsymbol{r}_{oc}$ denote a vector between the origin of the world frame, and $\boldsymbol{r}_{ci}$ a vector from corner c to a point $\boldsymbol{p}_i$ in the cube, as depicted two-dimensionally in Figure 3.2

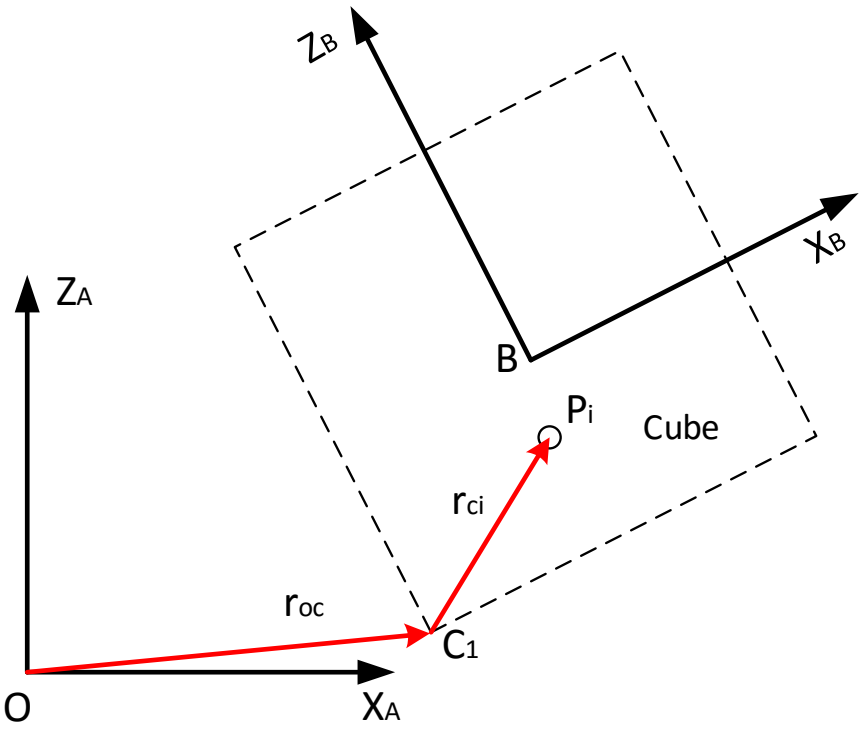


Figure 3.2: Illustrating vectors used to derive expressions for velocity of different point of the body. It is assumed the the corner c_1 is fix and that c_1 coincides with the global origin of the global coordinates (X_w, Y_w, Z_w) (so r_{oc} is a zero vector in reality)

Differentiating this vector yields the velocity of the point $\boldsymbol{p}_i$

$$\boldsymbol{v}_i = \frac{d(\boldsymbol{r}_{oc} + \boldsymbol{r}_{ci})}{dt} = \boldsymbol{v}_c + \frac{d\boldsymbol{r}_{ci}}{dt} = \boldsymbol{v}_c + \boldsymbol{\omega} \times \boldsymbol{r}_{ci} \tag{3.5}$$

The cross product in the last term is from a general result regarding differentiation of rotating geometrical vectors [5]. As the cube is assumed to always have the corner c fixed at one contact point the velocity $\boldsymbol{v}_c = \boldsymbol{0}$. The point $\boldsymbol{p}_i$ is assumed fixed in

relation to the cube (e.g. it represents a rigidly mounted part). The derivate of the vector can therefore be determined by just the cross product

$$\boldsymbol{v}_i = \frac{\mathrm{d}\ \boldsymbol{r}_{ci}}{\mathrm{dt}} = \boldsymbol{\omega}_c \times \boldsymbol{r}_{ci} \tag{3.6}$$

where $\boldsymbol{r}_{ci}$ is a vector going from the corner c to the point $\boldsymbol{p}_i$ and $\boldsymbol{\omega}_c = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$ is the pseudo vector describing the rotation of the cube casing relative to the world frame.
Differentiating (3.6) again yields an expression for the acceleration of point i

$$\boldsymbol{a}_i = \frac{\mathrm{d}\ \boldsymbol{v}_i}{\mathrm{dt}} = \quad \dot{\boldsymbol{\omega}}_c \times \boldsymbol{r}_{ci} + \boldsymbol{\omega}_c \times \dot{\boldsymbol{r}}_{ci} = \quad \dot{\boldsymbol{\omega}}_c \times \boldsymbol{r}_{ci} + \boldsymbol{\omega}_c \times (\boldsymbol{\omega}_c \times \boldsymbol{r}_{ci}) \tag{3.7}$$

where rightmost expression is obtained by re-substituting (3.6) into (3.7).

3.2.1 Velocity and acceleration in different frames

A well known results it that the cross product is invariant under proper rotations using a rotation matrix R so

$$R(\boldsymbol{v}_1 \times \boldsymbol{v}_2) = (R\boldsymbol{v}_1) \times (R\boldsymbol{v}_2)$$

Say it is desirable to express the velocity v_i which is a vector quantity defined in frame A in some other frame. Frame B will be used as an example, so

$${}^B\boldsymbol{v}_i = R_A^B\ \boldsymbol{v}_i = R_A^B(\boldsymbol{\omega} \times \boldsymbol{r}_{ci}) = (R_A^B\ \boldsymbol{\omega}) \times (R_A^B r_{ci}) = {}^B\boldsymbol{\omega} \times {}^B\boldsymbol{r}_{ci} \tag{3.8}$$

Thus, the result (3.8) shows that the velocity expressed in the frame B can be calculated using the same cross product as in (3.6), but with the position vector and angular velocity expressed in the frame B. This can lead to simpler expressions as the position vector defined in the body frame ${}^B\boldsymbol{r}_{ci}$ is constant for a rigid body such as the cube. This result directly extends to acceleration $\boldsymbol{a}_i$ in (3.7)

$${}^B\boldsymbol{a}_i = \frac{\mathrm{d}\ {}^B\boldsymbol{v}_i}{\mathrm{dt}} = {}^B\dot{\boldsymbol{\omega}} \times {}^B\boldsymbol{r}_{ci} + {}^B\boldsymbol{\omega} \times ({}^B\boldsymbol{\omega} \times {}^B\boldsymbol{r}_{ci}) \tag{3.9}$$

It is important to note is that (3.8) and (3.7) are still describing how a point i translate and accelerate relative to frame A. The difference is that these vectorial relations are now expressed in frame B. For a visualization, substitute $\boldsymbol{a}_i$ and $\boldsymbol{v}_i$ for $\boldsymbol{r}$ in Figure 3.1 and imagine how they would appear differently in frame A and B while still representing the same quantity.

3.3 Generalized coordinates

Generally a free body in three dimensions needs three variables for describing its position, one for each Cartesian coordinate (x, y, z). This means a particle system with Nb bodies would need $3N_b$ for describing all positions. Usually, though, there are constraint between each body such that they cannot move independent of one another. If there are M constraints this reduces the number of Cartesian coordinates to

$$n = 3N_b - M \tag{3.10}$$

Under these circumstances one can describe the positions cartesian position (x, y, z) using n so called generalized coordinates $q_1(t) \dots q_n(t)$ [6].

Example 1. *For instance, say the vector in* $\boldsymbol{r} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ *3.3 cannot move in the* Y *direction and has a fix length* r*. This means that it is confined to the constraints*

$$\begin{aligned} y &= 0 \\ z^2 + z^2 - r^2 &= 0 \end{aligned}$$

There are two constraints so $M = 2$ *and* $n = 1$ *. The orientation of* $\boldsymbol{r}$ *can then be described using a single generalized coordinate* $q_1 \triangleq \theta$

The constraint in the example can written in terms of generalized coordinates

$$f(q_1, \dots, q_n, t) = 0 \tag{3.11}$$

This is called a holonomical constraint. If the system on the other hand is subject to constraints that are functions of for instance the derivatives of $\dot{q}_1 \dots \dot{q}_n$ also, it might happen that the system must be described with a larger number of generalized coordinates n then number of degrees of freedom p. This introduces various computational challenges. [7]. The cube is a holonomic system, as it will later be shown that it has 6 degrees of freedom that can be described using 6 variables, so this issue is not further elaborated here.

3.3.1 Generalized forces

A generalized force Q_j is a scalar quantity that virtual does work along a virtual displacement of generalized coordinate q_j according to

$$\delta W_j = Q_j \delta q_j \tag{3.12}$$

This relation is derived from the general principle of virtual work [5]. A consequence of this is that the virtual work δW_j must have the dimension work, while the generalized coordinate and forces can have other dimensions. Typically the dimensions of the generalized coordinates is either a linear displacement or an angle, and the generalized force a linear force or a torque.

3.4 Kinetic energy

Generally, a body in three dimensions has the kinetic energy

$$T = \underbrace{\frac{1}{2} m \boldsymbol{v}^T \boldsymbol{v}}_{T_T} + \underbrace{\frac{1}{2} \boldsymbol{\omega}^T I_w \omega}_{T_R} \tag{3.13}$$

Here T_T is kinetic energy related to translation T_R to rotation, $\boldsymbol{v}$ is the velocity of the mass center of the body and $\boldsymbol{\omega}$ its angular velocity, relative to an inertial world frame subscripted w. Continuing, m is the mass of the body and I_w its inertia tensor. The inertia tensor is defined in the world frame here and generally time varying. Note that if the center of mass is not chosen as a reference point for the velocity, then (3.13) will have more terms. [5].

3.4.1 Using expressions in body frame

The kinetic energy can be calculated having all vector quantities and the inertia tensor (3.13) expressed in the body frame. This can lead to simpler expressions. Denote the inertia tensor in the body frame I_B, which is constant. This can be related to the inertia in the world frame via the rotation matrix between the world and the body frame according to [8]

$$I_w = R_B^w I_B R_B^{wT} \tag{3.14}$$

SimilarlyTry the algorithm out thoruhly and Finnish wriring stuf, the angular velocity ω expressed in the world frame can be related to the angular velocity expressed in the body fixed frame according to

$$\boldsymbol{\omega} = R_B^w \, {}^B\boldsymbol{\omega} \tag{3.15}$$

Combining (3.13), (3.14) and(3.15)

$$\begin{aligned} T_R = & \frac{1}{2} \boldsymbol{\omega}^T I_w \boldsymbol{\omega} = \\ & \frac{1}{2} (R_B^{wB} \boldsymbol{\omega})^T R_B^w I_B R_B^{wT} (R_B^{wB} \boldsymbol{\omega}) = \\ & \frac{1}{2} {}^B\boldsymbol{\omega}^T (R_B^w)^T R_B^w I_B (R_B^w)^T R_B^{wB} \boldsymbol{\omega} = \\ & \frac{1}{2} {}^B\boldsymbol{\omega}^T I_B {}^B\boldsymbol{\omega} \end{aligned} \tag{3.16}$$

In (3.16) it was utilized that multiplying a proper rotation matrix with its transpose yields an identity matrix, $RR^T = I$. The result in (3.16) is particularly useful as the inertia I_B is expressed in the rotating body frame and is therefore constant.

Furthermore, if the body frame is aligned with the principal axis of the body the inertia tensor will be a diagonal matrix [8].Proceeding in a similar way for the translational energy,

$$T_T = \frac{1}{2}m\boldsymbol{v}^T\boldsymbol{v} = \frac{1}{2}m(R_B^{W\ B}\boldsymbol{v})^{TB}R_B^{WB}\boldsymbol{v} = \frac{1}{2}m^B\boldsymbol{v}^{T\ B}\boldsymbol{v} \tag{3.17}$$

These expressions are used for Lagrangian mechanics within the context of this work.

3.5 Lagrangian mechanics

The Lagrangian approach to mechanics provides a different way to derive equations of motion then using Newton's and Euler's laws of motion directly. A key advantage is that only forces that do a net work on the system are considered. Contact forces between rigid bodies and normal forces can then be left out of the analysis, as opposed to when using Newtonian mechanics and free body diagrams. Lagrangian mechanics does not contradict Newtonian mechanics, but is rather another way of viewing it. It can be derived using the principle of virtual work [5, 9].

3.5.1 Generalized coordinates

The work flow when using Lagrangian mechanics typically begins by defining a set of generalized coordinates. These must satisfy a number of criteria. First off, they must be **complete** such that they completely can describe the orientation of the all bodies. Second, they must be **independent**, such that if all coordinates are locked except for one, the last one can still move freely. Third, they must be **holonomic** meaning that the number of degrees of freedom the system has equals the number of coordinates needed to describe it position and orientation [10]
The reason behind this can be sought in the derivation of Lagrangian mechanics. Here the principle of virtual work is used, which in its turn requires coordinates that completely can describe the orientation and displacement of the system [5].

3.5.2 Lagrangian

When the generalized holonomic coordinates are determined, the next step is finding the Lagrangian

$$\mathcal{L} = T - U \tag{3.18}$$

where T is the kinetic energy of the system and its conservative energy U. For a system of $k = 1 \dots N_B$ bodies, the kinetic energy can generally be calculated as

$$\begin{aligned} T_k =& \frac{1}{2}m_k||\boldsymbol{v}_k||^2 + \frac{1}{2}\boldsymbol{\omega}_k^T I_k \boldsymbol{\omega}_k^T \\ T =& \sum T_k \end{aligned} \tag{3.19}$$

by applying (3.13) for each body indexed i and then summing.
Conservative energy is energy that is retained within the system. This is usually potential energy, such as elevation in relation go a gravity field (i.e. altitude) or energy stored in a spring. For instance, conservative energy for a some object with the mass m lifted h meters above some arbitrary reference point is mgh , where g is the gravitational constant.

3.5.3 Equations of motion

Then, the equations of motions are found as (3.19)

$$\underbrace{\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right)}_{P_g} = \underbrace{\overbrace{\frac{\partial \mathcal{L}}{q_i}}^{\text{Conservative}} + \overbrace{Q_i}^{\text{Non-conservative}}}_{\text{Generalized forces}} \tag{3.20}$$

Here P_g is something called *generalized momenta* . Similar to linear momenta, it is conserved if no external generalized force Q_i is applied. The impact of each term in 3.20 is perhaps best illustrated using a simple example.
Example 2. *Let's apply Lagrangian mechanics on the simple system in Figure 3.3. The object is constrained such that it doesn't rotate or translate in the z direction. The general coordinates are defined as* $q_1 \triangleq x$ *and* $q_2 \triangleq y$ *. The Lagrangian is simply*

$$\mathcal{L} = T - U = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) - mgy \tag{3.21}$$

By applying (3.20) on 3.21 we obtain

$$\frac{\partial \mathcal{L}}{\partial \dot{q}_1} = m\dot{q}_1 = m\dot{x} \qquad \frac{\partial \mathcal{L}}{\partial \dot{q}_2} = m\dot{q}_2 = m\dot{y}$$
$$\frac{\partial \mathcal{L}}{\partial q_1} = 0 \qquad \frac{\partial \mathcal{L}}{\partial q_2} = -mg$$

We see that the top two terms, representing generalized momenta, are the same as the linear momenta for the system. If no generalized force is applied, as according to the two lower terms, momenta is preserved. This will happen for the momentum in the x direction as no force is applied in the x direction. Continuing by taking the time derivative, as assuming non-conservative forces to be zero we obtain

$$m\ddot{x} = 0$$
$$m\ddot{y} = -mg$$

Here non-conservative forces Q_i such as fluid resistance of somebody pushing the object could directly be added to the right hand side. This is exactly the same result as we would get more easily using Newtonian mechanics. Generally though, Lagrangian mechanics makes modeling of more complex multi body systems, such as the cube, significantly easier.

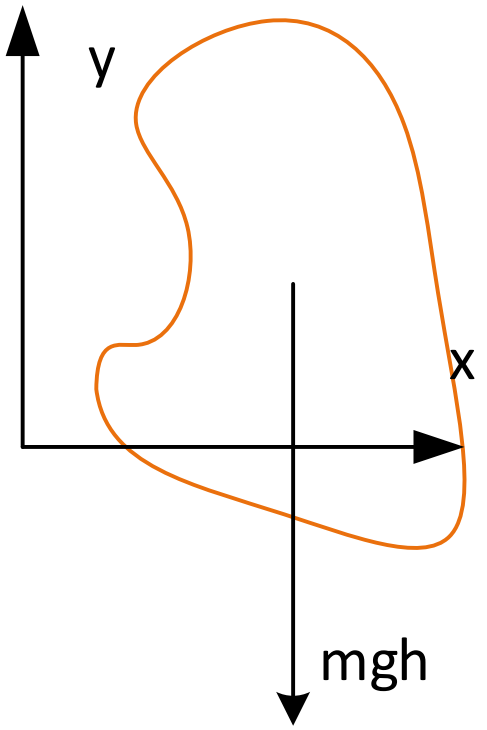


Figure 3.3: A simple example used to demonstrate Lagrangian mechanics

3.6 Kane's Method

An issue using Lagrange method, pointed out in Section 3.5.1, is that the system needs to be holonomic. Kane's method, named by its creator Thomas Kane, can deal directly with non-holonomic constraints [6]. Non-holonomic systems can be fairly simple to their structure, such as a sphere rolling without slip on a plane [10, 11] or a two wheeled robot with a support point in a plane [12].

As with the Lagrangian method, the work flow when using Kane's method begins by choosing a set of generalized coordinates. Then however an auxiliary property called *holonomic generalized speeds*are defined, and in relation to this *holonomic generalized velocities* and *generalized active forces*. All these have non-holonomic counter parts. This creates a transformation of the original problem which can prove very useful. The transformation can be interpreted geometrically [13]. A parallel could perhaps be the Laplace transform that is introduced by definition but proves extremely powerful in for instance linear systems analysis.

3.6.1 Scalar form

Hera follows a presentation of Kane equations on scalar form as found in[6]. A reference frame is denoted A and a body frame B. All motions such as angular velocity and velocity are defined as motions of frame B relative to frame A .
Kane's equations are a set of scalar equations

$$F_r + F_r^* = 0, \quad r = 1 \dots p) \tag{3.22}$$

where p is the number of degrees of freedom for the system, F_r is a *generalized active force* and F_r^* a *generalized inertia force* [6]. Note that a holonomic system is assumed within this context, implying that the number of degrees of freedom, denoted p, are the same the number of generalized coordinates n. The generalized active forces are given by

$$F_r = \underbrace{\sum_{k=1}^{N_b} \left[\boldsymbol{\omega}_r^k \cdot \boldsymbol{T}_k \right]}_{\text{Torque related}} + \underbrace{\sum_{k=1}^{N_b} \left[\boldsymbol{v}_r^k \cdot \boldsymbol{F}_k \right]}_{\text{Force related}} \tag{3.23}$$

were $\boldsymbol{T}_k$ and $\boldsymbol{F}_k$ are net torques and net forces, caused by for instance gravity or friction, acting on each body k of the system. The quantities $\boldsymbol{v}_r^k$ and $\boldsymbol{\omega}_r^k$ are partial velocities which are to be further described in Section 3.6.2. The generalized inertia forces are then determined by

$$F_r^* = \sum_{k=1}^{N_b} \left[\boldsymbol{\omega}_r^k \cdot \boldsymbol{T}_k^* \right] + \sum_{k=1}^{N_b} \left[\boldsymbol{v}_r^k \cdot \boldsymbol{F}_k^* \right] \tag{3.24}$$

where the inertial forces and torques for each body k are determined by

$$\boldsymbol{F}_k^* = -m_k \boldsymbol{a}_k \tag{3.25a}$$
$$\boldsymbol{T}_k^* = -I_k \dot{\boldsymbol{\omega}}_k - \boldsymbol{\omega}_k \times (I_k \boldsymbol{\omega}_k) \tag{3.25b}$$

Inserting (3.23), (3.24) and (3.25) into (3.22) and regrouping yields

$$\sum_{k=1}^{N_b} \left[\boldsymbol{\omega}_r^k \cdot \left(\boldsymbol{T} - I\dot{\boldsymbol{\omega}} - \boldsymbol{\omega} \times (I\boldsymbol{\omega}) \right)_k \right] + \sum_{k=1}^{N_b} \left[\boldsymbol{v}_r^k \cdot \left(\boldsymbol{F} - m\boldsymbol{a} \right)_k \right] = 0 \in \mathbb{R}^{1\times 1} \tag{3.26}$$

It can be seen that (3.25) are Newton's second law and Euler's law of motion. A further elaboration on the formulation in (3.26) can be found in [14], from where the notation has been adopted. Note that k corresponds to the current body index in (3.26), and is not an exponential. In cited literature by Kane [6] inertia dyads are used instead of inertia tensors. Inertia dyads are also treated in [15, 14].

3.6.2 Generalized speed and partial velocities

Generalized speeds are properties that are defined by the user. These relate the generalized coordinates $q_1, \ldots, q_n$ and their derivatives to generalized speeds $u_1, \ldots, u_n$ according to some function

$$u_r(t) \triangleq \sum_{s=1}^{n} Y_{rs}\dot{q}_s + Z_r \qquad (r = 1 \ldots n) \tag{3.27}$$

Here Y_{rs} and Z_r can be chosen freely as long as all generalized coordinates can be solved for uniquely [6].
Then the velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\omega}$ being Cartesian vectors describing the relative motion of frame B in relation to frame A are rewritten using *partial velocities* according to

$$\boldsymbol{\omega} \triangleq \sum_{r=1}^{n} \boldsymbol{\omega}_r u_r + \boldsymbol{\omega}_t \tag{3.28a}$$

$$\boldsymbol{v} \triangleq \sum_{r=1}^{n} \boldsymbol{v}_r u_r + \boldsymbol{v}_t \tag{3.28b}$$

where $r = 1 \ldots n$ are the degrees of freedom.
The partial velocities needs to be found in order to use Kane's method. Note that the partial velocities are the partial derivatives of (3.28) according to

$$\frac{\partial \boldsymbol{\omega}}{\partial u_r} = \frac{\partial(\boldsymbol{\omega}_1 u_1 + \ldots \boldsymbol{\omega}_r u_r + \ldots \boldsymbol{\omega}_n u_n + \boldsymbol{\omega}_t)}{\partial u_r} = \boldsymbol{\omega}_r \tag{3.29}$$

and similarly for the linear velocity

$$\frac{\partial \boldsymbol{v}}{\partial u_r} = \frac{\partial(\boldsymbol{v}_1 u_1 + \ldots \boldsymbol{v}_r u_r + \ldots \boldsymbol{v}_n u_n + \boldsymbol{v}_t)}{\partial u_r} = \boldsymbol{v}_r \tag{3.30}$$

The expressions defined in this section are seemingly abstract, but are often quiet easily found in practice if the generalized speeds are chosen with care.

Example 3. *A central issue when learning Kane's method can be how to to define and use the general speeds and partial velocities. The expressions (3.27) and (3.28) are defined for a general case, but can usually be found easily by inspection.*
Say for instance that the system is a body moving freely in 3D with Euler angles as generalized coordinates. As is shown in Section 4.8 the angular velocity of the body can be related to the Euler angles such that the generalized coordinates can be solved for uniquely, which as required for (3.27) The user could then choose to define the first three generalized speeds to be the angular velocity of the body according to $\boldsymbol{u}_{1:3} \triangleq \boldsymbol{\omega}$.
The scalar generalized speeds can then be related to the angular velocity according to

$$\boldsymbol{\omega} = \boldsymbol{e}_x u_1 + \boldsymbol{e}_y u_2 + \boldsymbol{e}_z u_3 \tag{3.31}$$

where $\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$ are the unit vectors of the frame where the angular velocity is expressed. This corresponds to (3.28)
By applying (3.29) on (3.31) the partial velocities are simply the unit vectors
$\omega_1 = \boldsymbol{e}_x, \quad \omega_2 = \boldsymbol{e}_y, \quad \omega_3 = \boldsymbol{e}_z$
The partial angular velocities where thus found easily by inspection.

3.6.3 Matrix formulation

For many systems the truly useful variant of Kane's equations are on matrix form. The perhaps first derivation of the matrix form in literature seems to have been published by Blajers in 1990 [16]. A similar result is pointed out in a paper from 1991 by Piedbeuf [12], treating a comparison between Kane's method and and an earlier result called Jourdain's principle. However, the treatment doesn't substitute force couples for torques which is useful during practical modeling.
Here we will extend the result in [12] to involve torque. As stated in Section 3.6.2 the partial velocities are just the partial derivatives of the expressions for the velocities expressed using generalized speed as given by (3.30) and (3.29). Expanding the first summation in (3.23) for all degrees of freedom $r = 1 \dots p$ yields

$$\begin{bmatrix} \sum_{k=1}^{N_b} \left[\boldsymbol{\omega}_1^k \cdot \boldsymbol{T}_k\right] \\ \vdots \\ \sum_{k=1}^{N_b} \left[\boldsymbol{\omega}_p^k \cdot \boldsymbol{T}_k\right] \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{\omega}^1}{\partial u_1} \cdot \boldsymbol{T}_1 + \ldots + \frac{\partial \boldsymbol{\omega}^{Nb}}{\partial u_1} \cdot \boldsymbol{T}_{Nb} \\ \vdots \\ \frac{\partial \boldsymbol{\omega}^1}{\partial u_p} \cdot \boldsymbol{T}_1 + \ldots + \frac{\partial \boldsymbol{\omega}^{Nb}}{\partial u_p} \cdot \boldsymbol{T}_{Nb} \end{bmatrix} \in \mathbb{R}^{p \times 1} \tag{3.32}$$

The right hand side of (3.32) can be rewritten as multiple matrix multiplications by applying the definition of the dot product $\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{A}^T \boldsymbol{B}$

$$\begin{bmatrix} \left(\frac{\partial \boldsymbol{\omega}^1}{\partial u_1}\right)^T \\ \vdots \\ \left(\frac{\partial \boldsymbol{\omega}^1}{\partial u_p}\right)^T \end{bmatrix} \boldsymbol{T}_1 + \ldots + \begin{bmatrix} \left(\frac{\partial \boldsymbol{\omega}^{N_b}}{\partial u_1}\right)^T \\ \vdots \\ \left(\frac{\partial \boldsymbol{\omega}^{N_b}}{\partial u_p}\right)^T \end{bmatrix} \boldsymbol{T}_{Nb} \in \mathbb{R}^{p \times 1} \tag{3.33}$$

Then, we state the definition of the differentiation of the angular velocity $\boldsymbol{\omega}^k$ of body k with respect to the vector of generalized speeds $\boldsymbol{u} = (u_1 \dots u_p)$ which yields the Jacobian matrix

$$\frac{\partial \boldsymbol{\omega}^k}{\boldsymbol{u}} = \begin{bmatrix} \frac{\partial \omega_x}{\partial u_1} & \cdots & \frac{\partial \omega_x}{\partial u_p} \\ \frac{\partial \omega_y}{\partial u_1} & \cdots & \frac{\partial \omega_y}{\partial u_p} \\ \frac{\partial \omega_z}{\partial u_1} & \cdots & \frac{\partial \omega_z}{\partial u_p} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{\omega}^k}{u_1} & \cdots & \frac{\partial \boldsymbol{\omega}^k}{u_p} \end{bmatrix} \tag{3.34}$$

By comparing (3.34) and (3.33) it can be seen that the matrices being multiplied with the torques in (3.33) actually can be formed by transposing (3.34), for each body k. The right hand summation of (3.23) containing the partial velocity, can be likewise be formulated on matrix form. Using the derived results, the p scalar

equations of (3.22) can be replaced by the matrix equation

$$\sum_{k=1}^{N_b}\left[\frac{\partial \boldsymbol{\omega}_k}{\partial \boldsymbol{u}}\right]^T(\boldsymbol{T}_k-\boldsymbol{T}_k^*)+\sum_{k=1}^{N_b}\left[\frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{u}}\right]^T(\boldsymbol{F}_k-\boldsymbol{F}_k^*)=\boldsymbol{0}\in\mathbb{R}^p \tag{3.35}$$

where the inertial forces- and torques $\boldsymbol{F}_k^*, \boldsymbol{T}_k^*$ are defined by (3.25).

An advantage of (3.35) over (3.26), apart from the usual benefits of matrix formulations, is a systematic procedure to obtain and organize the partial velocities. This is easily obtained as a single row of code in symbolic handling softwares like `Mathematica` when all other expressions involved are found.
Another matrix formulation is found in the textbook *Fundamentals of multi body dynamics* [7] and a paper from NASA using inertia dyads[14]. Yet another variation of a matrix formulation of Kane's method is used by the creators of Cubli and presented in the paper *The Cubli: A Reaction Wheel Based 3D Inverted Pendulum* [17]. There equation (3.35) is pre-multiplied from the left by a vector of differentials of what can be identified as infinitesimals of generalized speed $(\delta u_1 \dots \delta u_p)^T$. This is a variant of what is called the principle of virtual power and it is elaborated in *Multibody Dynamics with Unilateral Contacts* [18].

3.6.4 Transform to another coordinate frame

Say we wanted to express the equations of motion in some other frame instead of the reference frame A using a rotation matrix R, and all involved vector quantities such as the applied forces etcetera are expressed in some other frame. Let's use the body frame B as an example as that should be the most common choice. This is convenient as Euler's laws of motions are defined in the body frame. By inserting the rotation matrix into the r'th term of (3.26)

$$\begin{aligned}R\boldsymbol{\omega}_r\cdot R(\boldsymbol{F}-\boldsymbol{F}^*)+R\boldsymbol{v}_r\cdot R(\boldsymbol{T}-\boldsymbol{T}^*)=(R\boldsymbol{\omega}_r)^T\left(R(\boldsymbol{T}-\boldsymbol{T}^*)\right)+(R\boldsymbol{v}_r)^T\left(R(\boldsymbol{F}-\boldsymbol{F}^*)\right)=\\ \boldsymbol{\omega}_r^T R^T\left(R(\boldsymbol{T}-\boldsymbol{T}^*)\right)+\boldsymbol{v}_r^T R^T\left(R(\boldsymbol{F}-\boldsymbol{F}^*)\right)=\boldsymbol{\omega}_r\cdot(\boldsymbol{F}-\boldsymbol{F}^*)+\boldsymbol{v}_r\cdot(\boldsymbol{T}-\boldsymbol{T}^*)\end{aligned} \tag{3.36}$$

For (3.36) the definition of dot product $\boldsymbol{a}\cdot\boldsymbol{b}=\boldsymbol{a}^T\boldsymbol{b}$ and the fact that the multiplication of a rotation matrix by it transpose R^TR yields an identity matrix were used. This shows that all vectors of (3.26) can be expressed in any frame while still describing the motions of frame B in relation to A.

3.6.5 Step by step instruction for applying Kane's method

Kane's method is documented extensively in the book "Dynamics, Theory and Applications" that is co-authored by Thomas Kane himself[6]. However, this material doesn't really outline a concept on how to apply Kane's method practically. Also it is so general that it might be hard to single out the most important information for how to apply the method. A few papers on how to apply Kane's [19, 20, 21] were studied by us. However, all of these lacked the matrix formulation of Kane's

(further described in Section 3.6.3) which we deem very useful. In another paper a procedure for matrix formulation was shown [14], but it seems that method is more well suited for more complex systems. We applied Kane's method as follows

List 3.1: Suggested procedure for applying Kanes method

1. Define a proper reference frame A and body frame B. Kane's method is formulated for some body having a frame B moving relative to a frame A . Frame A does not have to be an inertial frame [6]. Also, although Kane's method is formulated for motions of B relative to the frame A, the vector quantities describing the motions may be expressed in any frame.
2. Find the number of degrees of freedom p and number of bodies N_b the system has. Name all the points defining the center of gravity for each body. Then find expressions in an inertial frame for the position, velocity and acceleration of all centers gravity for each body. A suggestion is to then make a table to summarize these expressions, for instance appearing as Table 4.2.
3. Choose n generalized coordinates $q_1 \dots q_n$ and generalized speeds $u_1 \dots u_n$. Determine if the system is holonomic by asserting if the number or generalized coordinates equals the degrees of freedom implying $n = p$
4. Evaluate Kane's on matrix form, as described in Section 3.6.3

This procedure might be well adapted for other systems having similar structure and complexity too. However, for larger and more complex system see for instance the procedure in [14]. We believe that Step 2 in List 3.1 might be the major challenge when first applying Kane's method. The importance of this aspect is also stressed by Kane [6]. Step 3 can be considered a cases where art mixes into exact sciences. This step will to a large part decide the physical meaningfulness of the results, and how easily these are achieved. Now Step 4 can be carried with the help of using symbolic handling software. These equations might come out on a form that is not practically useful. If these are reworked manually, it is recommended to use a software such as `Mathematica` to verify that the reworked equations comply with this initial form.

3.7 State space representation

For all controller design in this context, linear or linearized state space representations of the system were used. A quick reminder on these follows. Many physical systems can be described using ordinary differential equations. A linear state space is basically a way to organize ordinary differential equations. Doing so can greatly simplify system analysis.
First a state vector is defined as $\boldsymbol{x}(t) = (x_1 \dots x_n)^T$. The scalar elements $x_1 \dots x_n$can represent basically any physical quantity. In a chemical process it might be temperature and concentration, in this context it is usually velocities and orientation. Then the insignal $\boldsymbol{u}(t)$ is identified, which is usually the signal that a controller can manipulate, torque within this context, and measurements $\boldsymbol{y}(t)$. A proper state space in continuous time is then defined as

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= A\boldsymbol{x}(t) + B\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= C\boldsymbol{x}(t) + D\boldsymbol{u}(t) \end{aligned} \tag{3.37}$$

where A, B, C and D are constant matrices . This implies that the measurements are a linear combination of states and inputs, although the D matrix is usually zero. As most control systems are implemented using computers, that operate in discrete time, the continuous state space model is usually discretized for a constant and uniform sampling interval T_s for discrete indexes k to yield

$$\begin{aligned} \boldsymbol{x}_k &= A\boldsymbol{x}_{k-1} + B\boldsymbol{u}_{k-1} \\ \boldsymbol{y}_k &= C\boldsymbol{x}_{k-1} + D\boldsymbol{u}_{k-1} \end{aligned} \tag{3.38}$$

The time index k is subscripted.The insignal, being generated by a discrete system , is constant in the interval $(k-1)T_s \leq t \leq (k)T_s$.
Many systems including the cube is described using non-linear differential equations in a non linear state space

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}) \end{aligned} \tag{3.39}$$

where $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ and can be any non linear function. There is a wide range of theory and tools available for linear systems, such as the Laplace Transform which cannot be used for non-linear systems. Therefore a non linear system is many times *linearized* to approximately appear as linear, the tools for linear systems analysis. Linearization is further described in Section 3.8

3.8 Multivariate linearization

Generally, a vector valued non linear continuous and differentiable function

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x} \\ \vdots \\ f_n(\boldsymbol{x}) \end{bmatrix}, \qquad \boldsymbol{x} = (x_1 \dots x_m) \in \mathbb{R}^m, \qquad\qquad f : \mathbb{R}^m \to \mathbb{R}^n$$

can be approximated using first order Taylor expansion according to [22]

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{x}_0) + \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\boldsymbol{x}}\bigg|_{x_0} (\boldsymbol{x} - \boldsymbol{x}_0) = \boldsymbol{f}(\boldsymbol{x}_0) + \begin{bmatrix} \frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial x_m} \\ \vdots & \ddots & \\ \frac{\partial \boldsymbol{f}_n(\boldsymbol{x})}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{f}_n(\boldsymbol{x})}{\partial \boldsymbol{x}_m} \end{bmatrix}\bigg|_{x_0} (\boldsymbol{x} - \boldsymbol{x}_0) \tag{3.40}$$

Example 4. *Say for instance that* $f(\boldsymbol{x}) = x_1^2 + x_2$ *and that this function is to be linearized around* $\boldsymbol{x}_0 = (a, b)$ *By applying 3.40*

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \left[\frac{\partial f(\boldsymbol{x})}{x_1} \quad \frac{\partial f(\boldsymbol{x})}{x_2}\right]\Big|_{\boldsymbol{x}=\boldsymbol{x}_0} (\boldsymbol{x} - \boldsymbol{x}_0) = a^2 + b + \begin{bmatrix}2a & 1\end{bmatrix}\begin{bmatrix}x_1 - a \\ x_2 - b\end{bmatrix} = \quad (3.41)$$

$$a^2 + 2ax_1 - 2a^2 + x_2 \quad (3.42)$$

Note that x_2 *is mapped equally to* $f(\boldsymbol{x})$ *before and after the linearization and that the linearized function is exactly equal to* $f(\boldsymbol{x})$ *at the linearization point.*
It is rather common to have a vector valued function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ mapping some state variables $\boldsymbol{x}$ and input signal $\boldsymbol{u}$ according to

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}_1(\boldsymbol{x}) + f_2(\boldsymbol{x})\boldsymbol{u} \quad (3.43)$$

where f_2 generally is matrix valued. By concatenating $\boldsymbol{x}$ and $\boldsymbol{u}$ into one vector $\boldsymbol{z} = \boldsymbol{x}^\frown\boldsymbol{u}$ and applying (3.40) it is obtained that

$$\boldsymbol{f}(\boldsymbol{z}) \approx \boldsymbol{f}(\boldsymbol{z}_0) + \frac{\partial \boldsymbol{f}(\boldsymbol{z})}{\partial \boldsymbol{z}}\Big|_{z_0} \Delta z = \boldsymbol{f}(\boldsymbol{z}_0) + \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} \quad \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{u}}\right]\Big|_{z_0} \begin{bmatrix}\Delta \boldsymbol{x} \\ \Delta \boldsymbol{u}\end{bmatrix} \quad (3.44)$$

If $\boldsymbol{f}(z)$ describes a non linear state space $\dot{x} = \boldsymbol{f}(z)$ and linearization is commonly made around a point $\boldsymbol{z}_0$ yielding equilibrium so $\dot{x} = \boldsymbol{f}(z_0) = 0$. Continuing with these assumptions

$$\boldsymbol{f}(\boldsymbol{z}) \approx \cancelto{0}{\boldsymbol{f}(\boldsymbol{z}_0)} + \left[\frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{f}_2(\boldsymbol{x})}{\partial \boldsymbol{x}} u \quad \cancelto{0}{\frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{u}}} + \frac{\partial \boldsymbol{f}_2(\boldsymbol{x})}{\partial \boldsymbol{u}}\right]\Big|_{z_0} \begin{bmatrix}\Delta \boldsymbol{x} \\ \Delta \boldsymbol{u}\end{bmatrix} = \quad (3.45)$$

$$\left(\frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{x_0} + \frac{\partial \boldsymbol{f}_2(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{x_0} u_0\right) \Delta \boldsymbol{x} + f_2(x_0)\Delta \boldsymbol{u} \quad (3.46)$$

which gives the general formula for the first order approximation of (3.43). Similar formulations are found in for instance [23, 24] where it is called Jacobian linearization.

3.9 Linear Quadratic Regulator

The popular and common PID controller has an advantage of being fairly simple and intuitive in its structure. However, tuning can be non intuitive, abstract and time consuming, especially for MIMO systems as Cubli. It is thought that using a Linear Quadratic Regulator (LQR) is much more well suited than PID control for the Cubex platform . When using LQR a static control law is derived by solving

an optimization problem offline. For this a state space model as defined in (3.38) is used. The optimization problem considered in this context is

$$\underset{\boldsymbol{u}}{\text{minimize}} \quad \sum_{k=0}^{\infty}(\boldsymbol{x}_k^T Q \boldsymbol{x}_k + \boldsymbol{u}_k^T R \boldsymbol{u}_k) \tag{3.47}$$

$$\text{subject to} \quad \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k \tag{3.48}$$

From an arbitrary initial condition at the discrete time index $k = 0$ the sequence of states and inputs are to be minimized over a time horizon that tends to infinity. The matrices Q and R are symmetric weight matrices used for tuning. These have to be positive definite. By using state feedback

$$\boldsymbol{u}_k = -\bar{K}\boldsymbol{x}_k \tag{3.49}$$

$$\tag{3.50}$$

the optimal static feedback gain matrix can be found as

$$\bar{K} = (R + B^T \bar{P} B)^{-1} B^T \bar{P} A \tag{3.51}$$

where $\bar{P}$ satisfy the discrete algebraic difference Riccati equation:

$$\bar{P} = Q + A^T \bar{P} A - A^T \bar{P} B (R + B^T \bar{P} B)^{-1} B^T \bar{P} A \tag{3.52}$$

This is called an *inifite horizon linear quadratic regulator*. Stability of the closed loop system is guaranteed provided that a solution to is found. For more information regarding LQR see for instance [25].

3.10 Model predictive control

Common controllers such as PID usually look at the current state(s) (trough the proportional and derivative part) and past state(s) (through the integral part) and make a control move based on this. This is true also for the previously described infinite horizon LQR. What sets MPC from these two aforementioned controllers is that it makes predictions about the evolution of states in the future and tries to find an optimal control input based on this. Consider for instance a temperature control of a room. A PID controller would just consider the current and past states. An MPC on the other hand could for example predict how the temperature will evolve in the following hours, perhaps even based on weather conditions and such. This is implemented in practice by solving a general optimization problem using a dynamic model for the system . As MPC is formulated as a general optimization problem, inequality constraints can be used. These potential benefits are perhaps best illustrated using an example.

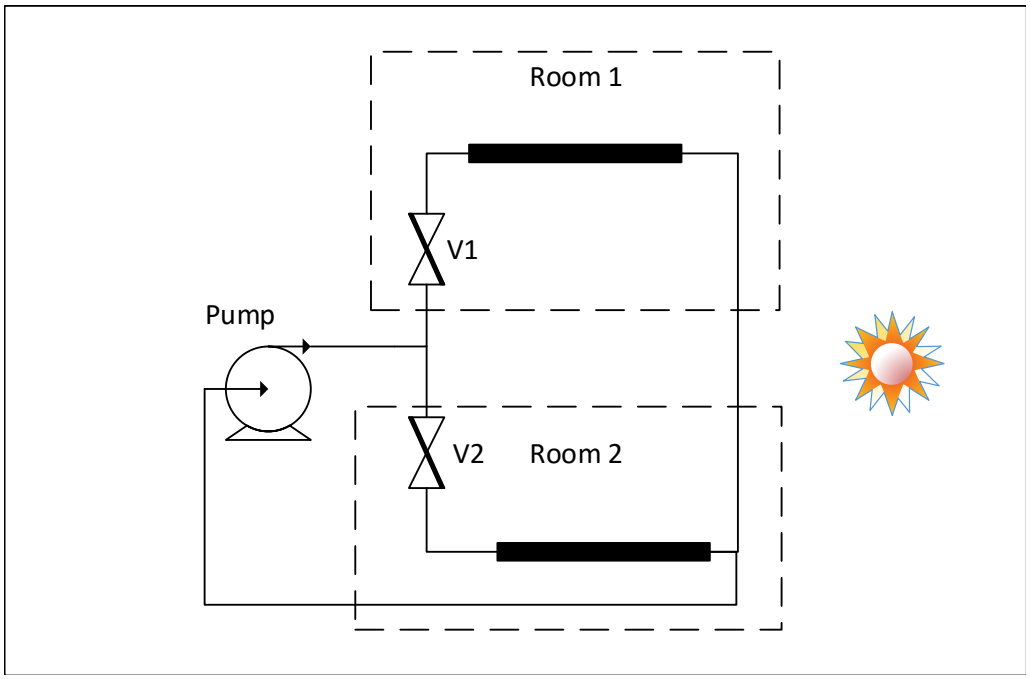


Figure 3.4: A figure used for an example to illustrate how MPC control can be beneficial

Example 5. *Let us expand the room temperature control to an example depicted in Figure 3.4*
Two rooms have temperature control implemented using one radiator in each room. The flow to each room is controlled using separate valves. External disturbances such as weather conditions affect the control loop. Several constraints can be identified. First, the pump can only deliver a certain flow. Then, each radiator can only deliver a certain maximum power. Then, one room might be more important than another. For instance, if room 1 is the living and room 2 is the garage, room 1 should probably be prioritized. Then, the expected weather conditions during the day will likely affect control loop performance. If it is likely to get colder soon, this should be compensated for already at the current time instant. Then, based on heat storage capacity of the building, it might be more economical to run the heating system at certain times. By properly formulating an optimization problem, the MPC can directly deal with all these trade offs. This stands in contrast to traditional PID control, where these problem would have to be dealt with using several discrete conditions, feed forward loops and such.

3.10.1 Mathematical formulation

Given the intuition behind MPC, here follows how it usually described mathematically. Let us say the the current time index is k, and the current controller output is to be determined .
The MPC now *predicts* the state evolution over a horizon N, as depicted in Figure 3.5, usually by using a linear state space formulation of the system by calculating a sequence of control inputs. This is done by formulating a cost function and solving

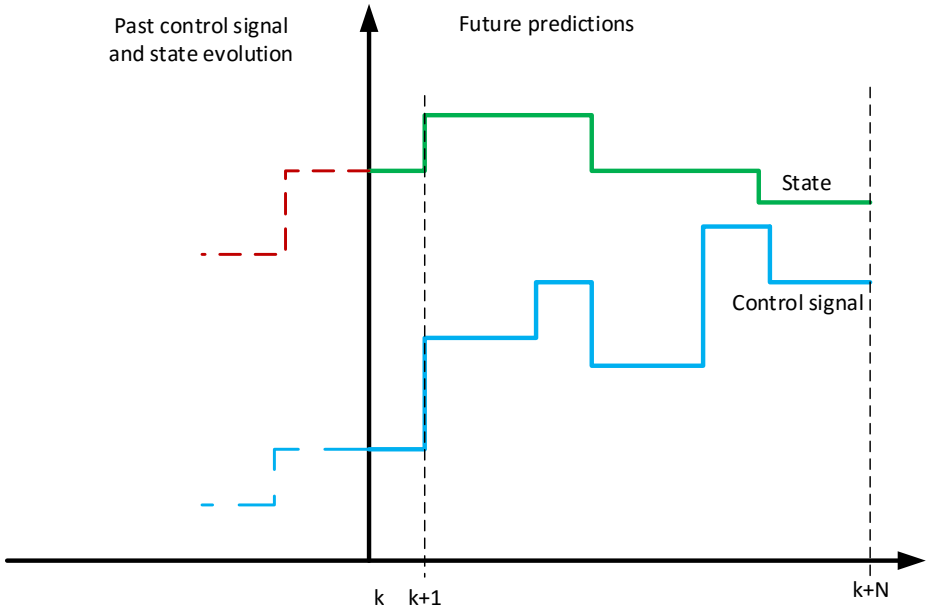


Figure 3.5: Visualization of predictions for MPC for a SISO system

an optimization problem. Then the control input at time index $k+1$ is applied, and the process is repeated at the next time instant [26].
Considering a discrete time state space model as in equation (3.38) we can formulate a quadratic cost function

$$\begin{aligned} \min_{\boldsymbol{x}_{0:N},\boldsymbol{u}_{0:N-1}} \quad & \sum_{k=0}^{N-1} (\boldsymbol{x}_k^T Q \boldsymbol{x}_k + \boldsymbol{u}_k^T R \boldsymbol{u}_k) + \boldsymbol{x}_N^T F \boldsymbol{x}_N \\ \text{s.t} \quad & \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k \\ & \boldsymbol{x}_k \in \mathbb{X} \qquad \boldsymbol{u}_k \in \mathbb{U} \qquad k = 0, 1, ..., N-1 \\ & \boldsymbol{x}_N \in \mathbb{X}_F \subseteq \mathbb{X} \end{aligned} \tag{3.53}$$

for the state and input signal for finite horizon N. The weight matrices Q and R are as for the previously described LQR. The matrix F i used to define a *terminal cost* but is not considered in this context. Here $\mathbb{X}$, $\mathbb{U}$ and $\mathbb{X}_F$ are sets of constraints on the state and input signal. By introducing a optimization vector containing the N state and control variables

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{u}_0^T & \boldsymbol{x}_1^T & \boldsymbol{u}_1^T & \cdots & \boldsymbol{x}_{N-1}^T & \boldsymbol{u}_{N-1}^T & \boldsymbol{x}_N^T \end{bmatrix}.$$

and formulating the optimization problem with equality and inequality constraints, we can formulate the optimization problem on a more compact from

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & \frac{1}{2}\boldsymbol{z}^T H \boldsymbol{z} + \boldsymbol{h}^T \boldsymbol{z} \\ \text{s.t} \quad & D\boldsymbol{z} = \boldsymbol{d} \\ & \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}} \end{aligned} \tag{3.54}$$

where h is the linear contribution of the cost function and the Hessian matrix, H, will have following diagonal form:

$$H = \begin{bmatrix} R & & & & \\ & Q & & & \\ & & \ddots & & \\ & & & R & \\ & & & & Q \end{bmatrix}$$

The system dynamics in equation (3.38) can be included as equality constraints in equation (3.54) by defining the matrices

$$D = \begin{bmatrix} B & -I & & & & \\ & A & B & -I & & \\ & & \ddots & \ddots & \ddots & \\ & & & A & B & -I \end{bmatrix} \qquad \boldsymbol{d} = \begin{bmatrix} A\boldsymbol{x}_k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.55}$$

The inequality constraints $\underline{\boldsymbol{z}}$ and $\overline{\boldsymbol{z}}$ is the lower limit boundary respectably upper limit boundary on the optimization vector. In this project we consider MPC problem on the form presented in equation (3.54).

3.10.1.1 Prediction horizon

Choosing a proper prediction horizon N is a main concern when implementing MPC. It can be shown in certain cases uses Lyapunov function that N affects stability. There is no general method for choosing N though. [26].

3.10.2 Faster MPC solvers

A major problem when implementing MPC is that is require a lot of memory and computational power, a opposed to for instance LQR. Minimizing the computational resources needed is an active area of research and has been so for a long time. For several approaches on how to minimize the computational cost see[27]. Within this context results mainly from a paper to be presented at the 20th IFAC (International Federation of Automatic Control) World Conference of Control 2017 is considered [3]. This paper is to a large part based on a fairly new result on faster MPC solvers called *fast gradient approach* by Stephen Boyd et. al. 2016 [28]. The cornerstones of this are as a follows. First a more general optimization problem is formulated as

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{y}} \quad & f(\boldsymbol{w}) + g(\boldsymbol{y}) \\ \text{s.t} \quad & C\boldsymbol{w} = \boldsymbol{y} \end{aligned} \tag{3.56}$$

where f is a strongly convex cost function, g is a closed convex cost function. A closed convex function is a function that contains all its limit points. Thus it is natural for this function to include inequality constraints. The vector $\boldsymbol{w}$ and $\boldsymbol{y}$ contain optimization variables and C is a matrix defining equality constraints. Boyd et. al shows that using fast gradient method on ill-conditioned problems can cause

unsatisfying performance [29]. To resolve this a so called *pre-conditioner matrix* P is introduced according to

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{y}} \quad & f(\boldsymbol{w}) + g(\boldsymbol{y}) \\ \text{s.t} \quad & PC\boldsymbol{w} = P\boldsymbol{y} \end{aligned} \tag{3.57}$$

This is then to be combined with the cost function of a general MPC optimization problem as formulated in (3.54) which can be achieved by the formulation [28]

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T H \boldsymbol{w} + \boldsymbol{h}^T \boldsymbol{w} + I_{\underline{\boldsymbol{w}} \leq \boldsymbol{w} \leq \overline{\boldsymbol{w}}} \tag{3.58a}$$

$$g(\boldsymbol{y}) = I_{\boldsymbol{y}=\boldsymbol{d}} \tag{3.58b}$$

$$C = D \tag{3.58c}$$

The matrix D is defined in (3.55). Here the indicator function

$$I_{\mathbb{X}} = \begin{cases} 0, & x \in \mathbb{X} \\ \infty, & \text{otherwise} \end{cases}$$

is used to replace inequality constraints. That is, if the inequality constraints are violated cost is infinite. Note how $I_{\mathbb{X}}$ is the previously mentioned closed convex function.

Returning to the paper that is to be presented at IFAC. In Algorithm 1 a fast dual proximal gradient method is restated from [3] as we interpreted and implemented it. It approximately solves a MPC problem on the form (3.54) using the splitting in (3.58). We managed to get the cube to system to balance together with this MPC implementation both in simulation and in practice.

Algorithm 1 Fast dual proximal gradient method

function GETNEXTCONTROLSIGNAL($\boldsymbol{x}_k$)

  procedure INIT ▷ We need to re init. for every new control signal
    $\boldsymbol{d} \leftarrow [A\boldsymbol{x}_k, 0, \dots 0]^T$ ▷ Right hand side of equality constraints
    $\Delta d \leftarrow \infty$ ▷ Scalar break condition for solver
    $\boldsymbol{w}_0 \leftarrow [\boldsymbol{u}_k, \boldsymbol{x}_{k+1}, \dots, \boldsymbol{u}_{k+N-1}, \boldsymbol{x}_{k+N}]$ ▷ Form by shifting last $\boldsymbol{w}_i$ for warm start
    $i \leftarrow 0$ ▷ Internal times index for MPC solver
    $\boldsymbol{\lambda}_0, \boldsymbol{\mu}_0 \leftarrow 0$
  end procedure

  procedure SOLVEOPTIMIZATIONPROBLEM
    while $\Delta d < tol$ AND $i \leq$ Iteration limit **do** ▷ Iteration limit to avoid possible deadlock
      $\beta_i \leftarrow \frac{i-1}{i+2}$
      $\boldsymbol{\mu}_i = \boldsymbol{\lambda}_i + \beta_i(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_{i-1})$ ▷ Internal variable
      $\boldsymbol{w}_i = \text{mid}(\underline{\boldsymbol{w}}, \overline{\boldsymbol{w}}, -H^{-1}\boldsymbol{h} - H^{-1}D^T P^T \boldsymbol{\mu}_i)$ ▷ Median function
      $\Delta d \leftarrow \|D\boldsymbol{w}_i - \boldsymbol{d}\|_\infty$ ▷ Break conditions
      $\boldsymbol{\lambda}_{i+1} = \boldsymbol{\mu}_i + \frac{1}{L}PD\boldsymbol{w}_i - \frac{1}{L}P\boldsymbol{d}$ ▷ Lagrangian multipliers $\boldsymbol{\lambda}$
      $i \leftarrow i + 1$
    end while
    $\boldsymbol{\lambda}_{i-1} \leftarrow \boldsymbol{\lambda}_i$ ▷ For when $i = 0$ next time
  end procedure

return Return $\boldsymbol{u}_k$ from $\boldsymbol{w}_i$
end function

The algorithm is called on time instant k as depicted in Figure 3.5 and returns the control signal vector $\boldsymbol{u}_k$ that transforms the system states to $\boldsymbol{x}_{k+1}$.
Going trough the algorithm. First equality constraints , time indexes and such are updated. The vector with states and control signals $\boldsymbol{w}_0$ was set to zero in our implementation. However, a better approach is probably shifting the last vector $\boldsymbol{w}_i$ one time step instead, to obtain a warm start.
Then the program proceeds to solve the actual optimization problem in an internal loop having time time index i. A max number of iterations are permitted in case the algorithm would diverge. This limit was usually set to 500 iterations. The vectors $\boldsymbol{\lambda}$ and μ are internal variables related to this method of solving the optimization problem. An elementwise operating median function mid[·] is used to incorporate the equality constraints of the problem. A condition for this to be valid is having a diagonal H matrix [3].
Break conditions at convergence of the algorithm is based on an infinity norm. The largest difference between two elements on the same row in the vectors $\boldsymbol{d}$ and $D\boldsymbol{w}_i$ cannot be larger than *tol* . L is the *Lipschitz constant* for the problem. The Lipschitz constant is related to the finding of solutions of initial value problems [30]. In [3] it is proposed to let the pre-conditioner $P = \sqrt{L}R^{-1}$ where R is the Cholesky

factorization that fulfills $RR^T = DH^{-1}D^T$. In this case $L = 1$ so the pre-conditioner was found by

$$P = R^{-1} = \left[\text{cholesky}(DH^{-1}D^T)\right]^{-1} \tag{3.59}$$

where `chol()` is the Matlab function returning a Cholesky factorization. More information on pre conditioners can be found in for instance the paper *Preconditioning in Fast Dual Gradient Methods* [29].
Algorithm 1 was benchmarked against the internal MPC solver `mpcqpsolver()` in `Matlab` during a simulation of edge balancing in a PC environment. The time for determining each control signal at time index k is seen in Figure 3.6 for the respective algorithms.

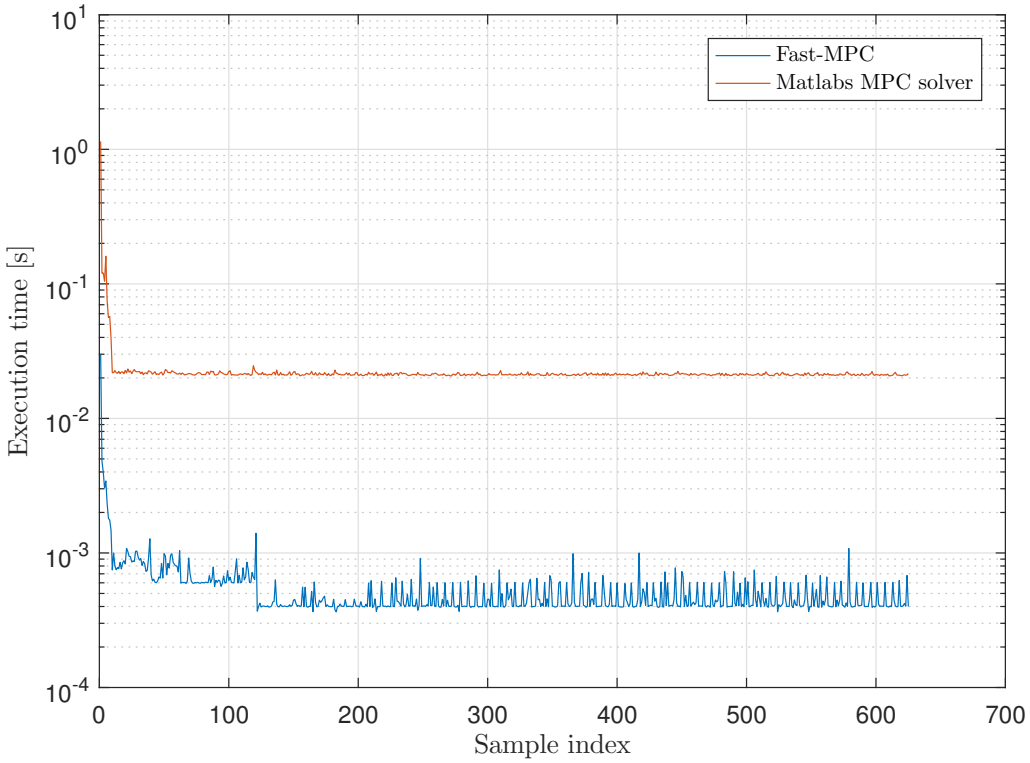


Figure 3.6: Execution times for the MPC solver described in Algorithm 1 for $tol = 10^{-5}$ and the `mpcqpsolver()` command in `Matlab` using default settings. Note the log scale.

The solver from Algorithm 1 on average has an execution time of about 2% of the `mpcqpsolver()` command in `Matlab`. It was implemented using the script in Appendix B.1. Output and performance of the control loops were basically equal for both methods. Evidently, the restated algorithm is significantly faster.

3.10.2.1 Approximation of the preconditioner

In [3] proposal is presented on how to further improve Algorithm 1. The preconditioner P is involved in the computationally most heavy operations in Algorithm 1.

It turns out that many elements of P are close to zero, and it is proven that elements are smaller the longer they are from the diagonal [3]. An example of this is shown in Figure 6.2. Therefore one of main areas of investigation in [3] is investigating if P can be approximated by neglecting elements that have a small relative magnitude. This is done by *m-banding*, which is illustrated on an actual preconditioner matrix in Figure 3.7. The larger elements closer to the diagonal are kept, while the ones outside the red encirclement are neglected.

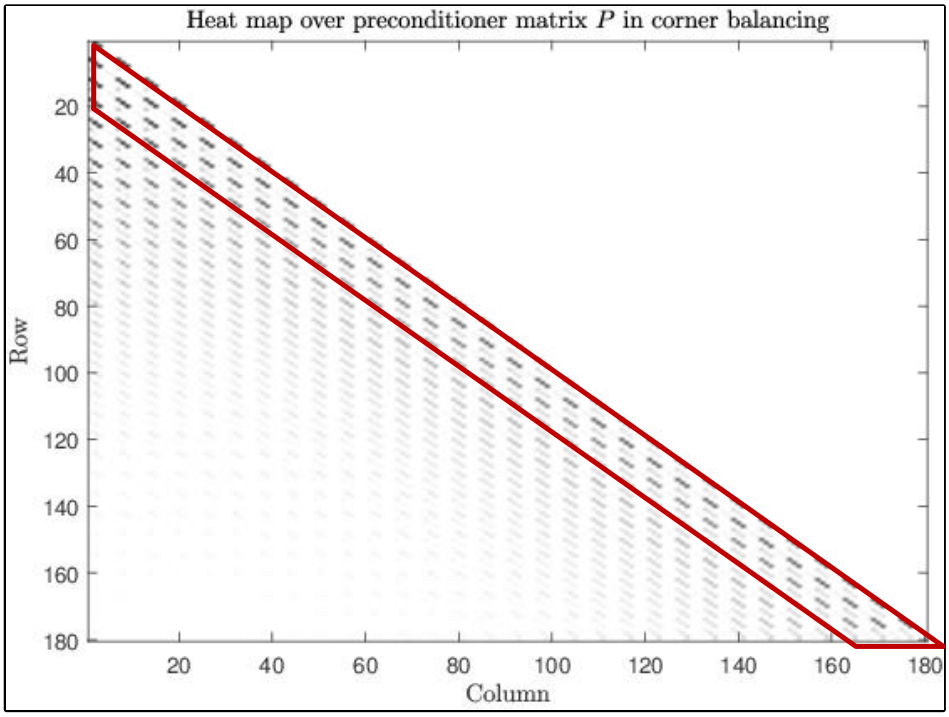


Figure 3.7: An illustration of banding of the preconditioner P for an actual preconditioner within this project. The elements within the red lines are kept.

.

The following banding was used within this thesis

$$\hat{P}_{i,j} = \begin{cases} R^{-1}_{i,j}, & |i-j| \leq m \\ 0, & \text{otherwise} \end{cases} \tag{3.60}$$

for the element on row i and column j in the matrix R^{-1}. That means a smaller m leads to that more elements are neglected and vice versa. The approximated preconditioner $\hat{P}$ then replaces P in Algorithm 1.
Doing such an approximation reduces the number of floating point operations needed for the i'th iteration of Algorithm 1. However, the approximation might lead to slower convergence and and therefore that more iterations are needed which is the trade off [3]. A purpose of this thesis was investigating if this approximation is possible to use on the cube.

3.10.2.2 Lipschitz constant issue

In [3] $\hat{P}$ as defined by (3.60) is also multiplied by the square root of the Lipschitz constant L for this problem. It is unknown by us both how to find the Lipschitz constant L and it is not specified in [3] how to find it. For the non approximated preconditioner P however $L = 1$. For the approximation $\hat{P}$ the Lipschitz constant was set experimentally in the range $1 \leq L \leq 3$. However, multiplying $\hat{P}$ by $\sqrt{L}$ seemed to cause the algorithm to diverge. Therefore this practice was abandoned.

4

Modeling

This chapter describes the modeling of the cube in three dimensions when it is balancing on a corner. Lagrangian mechanics is common to use for multi-body dynamics, and is therefore applied. Though it turns out that Lagrangian mechanics yields equation on a form that is relatively complicated to handle. The remedy for this was applying Kane's method, which is briefly described in Section 3.6. Equations of motion describing the dynamics of the cube are then found in Section 4.5.5.
Both Kane's and Lagrange's approach use the same general definitions of position, velocity, acceleration. These are therefore defined first before proceeding to derive the actual equations of motion.

4.1 Definitions

A body fixed frame (X_B, Y_B, Z_B) is defined as shown in Figure4.1.

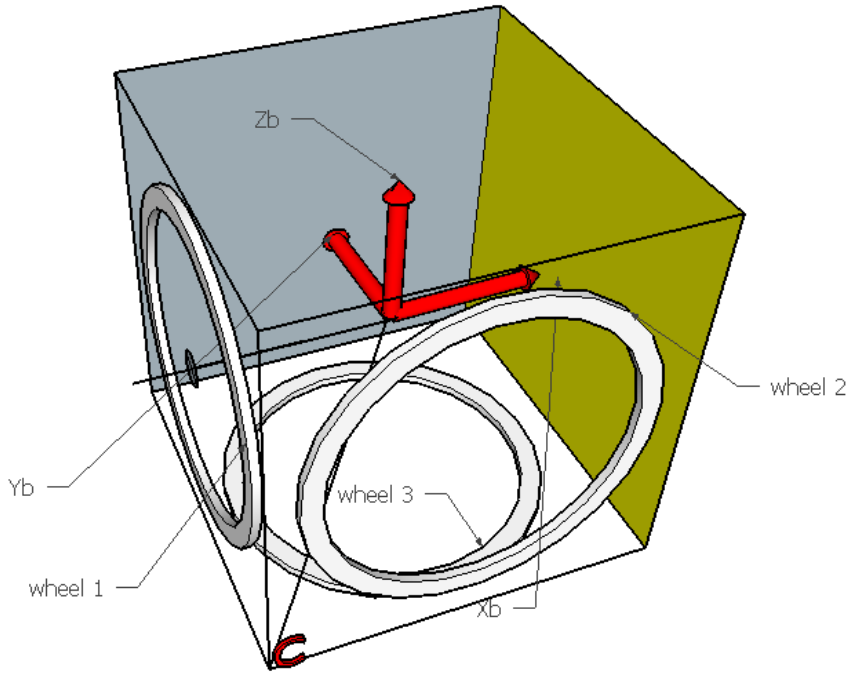


Figure 4.1: Defining the body fixed coordinate system and the indexes for each part. The cube is assumed to balance around the corner c that is marked using red text.

A static contact point c is defined to be the lower left corner in Figure 4.1. The cube is assumed to be standing on this corner when balancing. Each of the four bodies indexed $k = 1, 2, 3, 4$ are assigned to the indexes as shown in Table 4.1

Part	k	Mass	Inertia	Revolves around
Casing	1	$m_1 = m_c$	I_1	-
Wheel 1	2	$m_2 = m_w$	I_2	X_B
Wheel 2	3	$m_3 = m_w$	I_3	Y_B
Wheel 3	4	$m_4 = m_w$	I_4	Z_B

Table 4.1: Assignment of the body indexes $k = 1, 2, 3, 4$ to each body of the cube with associated inertia tensors and masses.

Vectors describing position, velocity and acceleration of each body are described and defined in Section 4.1.1 .

4.1.1 Position, velocity and acceleration

The equations of motion are to be defined for vectors evolving in the body frame. A key motive for this is to obtain constant inertia tensors and position vectors which should yield more compact expressions. In order to generate more compact descriptions, the notation ${}^B\boldsymbol{v}$ for some vector $\boldsymbol{v}$ expressed in the frame B is omitted for the shorter notation $\boldsymbol{v}$.
Generally, as provided in Section 3.2, the velocity and acceleration for one point i on the cube relative to the balancing corner is

$$\boldsymbol{v}_i = \frac{\mathrm{d}\,\boldsymbol{r}_{ci}}{\mathrm{dt}} = \boldsymbol{\omega}_c \times \boldsymbol{r}_{ci} \qquad \text{(3.6, Revisited)}$$

$$\boldsymbol{a}_i = \frac{\mathrm{d}\,\boldsymbol{v}_i}{\mathrm{dt}} = \quad \dot{\boldsymbol{\omega}}_c \times \boldsymbol{r}_{ci} + \boldsymbol{\omega}_c \times \dot{\boldsymbol{r}}_{ci} = \quad \dot{\boldsymbol{\omega}}_c \times \boldsymbol{r}_{ci} + \boldsymbol{\omega}_c \times (\boldsymbol{\omega}_c \times \boldsymbol{r}_{ci}) \qquad \text{(3.7, Revisited)}$$

where $\boldsymbol{\omega}_c$ is the angular velocity of the cube casing and r_{ci} are vectors going from the balancing corner to the point i. As is shown in Section 3.2.1 the two aforementioned vectors can be expressed in the body frame, resulting that for $\boldsymbol{v}_i$ and $\boldsymbol{a}_i$will be expressed there too.
By inspection of Figure 4.1 position vectors from the corner c can be determined. The angular velocity of the casing relative to the inertial frame is defined $\boldsymbol{\omega}_c \triangleq (\omega_x, \omega_y, \omega_z)^T$ and is expressed in the body frame. The wheels are modeled to move independently of the cube casing around around their shafts rotation axises, based on observations of the actual system. Let wheel i have the rotation around it's rotational axis $\omega_i \boldsymbol{e}_{zi}$ where $\boldsymbol{e}_{zi}$ is the unit vector describing the axis of rotation in the main reference frame A. This is a formulation to stress the fact that the the wheels are defined to rotate relative to the main reference frame A. However, all vectors are to be expressed in the body frame B. Due to the orthogonality between the defined body frame and each wheels body frame the angular velocity for each wheel is simply found via inspection.

The results, together with name definitions are presented in Table 4.2.
Table

	Position $\boldsymbol{r}$	Ang. vel	Lin. vel.	Lin. acc
Mass c.	$\boldsymbol{r}_{cb} = [r, r, r]^T$	$\boldsymbol{\omega}_c = [\omega_x, \omega_y, \omega_z]^T$	$\boldsymbol{v}_c = \boldsymbol{\omega}_c \times \boldsymbol{r}_{cb}$	$\boldsymbol{a}_c$
Wheel 1	$\boldsymbol{r}_2$	$\boldsymbol{\omega_2} = [-\omega_{w1}, \omega_y, \omega_z]^T$	$\boldsymbol{v}_2 = \boldsymbol{\omega}_c \times \boldsymbol{r}_2$	$\boldsymbol{a}_2$
Wheel 2	$\boldsymbol{r}_3$	$\boldsymbol{\omega}_3 = [\omega_x, -\omega_{w2}, \omega_z]^T$	$\boldsymbol{v}_3 = \boldsymbol{\omega}_c \times \boldsymbol{r}_3$	$\boldsymbol{a}_3$
Wheel 3	$\boldsymbol{r}_4$	$\boldsymbol{\omega}_4 = [\omega_x, \omega_y, -\omega_{w3}]^T$	$\boldsymbol{v}_4 = \boldsymbol{\omega}_c \times \boldsymbol{r}_4$	$\boldsymbol{a}_4$

Table 4.2: Defining some important vectors and and name conventions in the body frame. For instance, the linear velocity of wheel 1 expressed in the body frame is $\boldsymbol{v}_1$

The vectors from the corner to the mass centers of the wheels can be formulated as

$$\begin{aligned} \boldsymbol{r}_2 &= \boldsymbol{r}_{cb} + \mu \begin{bmatrix} -r & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{r}_3 &= \boldsymbol{r}_{cb} + \mu \begin{bmatrix} 0 & -r & 0 \end{bmatrix}^T \\ \boldsymbol{r}_4 &= \boldsymbol{r}_{cb} + \mu \begin{bmatrix} 0 & 0 & -r \end{bmatrix}^T \end{aligned} \tag{4.1}$$

where $\mu \in (0, 1)$. This means that the mass center of each wheel are assumed to lie equidistantly from the geometrical center of the cube, inside the cube, on the X_B , Y_B and Z_B respectively.
The accelerations $\boldsymbol{a}_c, \boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$ are then calculated according to (3.7). No geometric property is needed to express the derivatives of the angular velocities, these are simply evaluated directly as $\dot{\boldsymbol{\omega}}_c$ etcetera. Note that the fact that the wheels move independently of the cube around their rotation axises is captured by the expressions for the wheels angular velocities above. For instance wheel 1, revolving around the cubes X_B axis is not affected by the wheels angular velocity around this axis.

4.2 Generalized coordinates

The cube consists of 4 bodies, the casing and three wheels so a parameter $N_b \triangleq 4$ is defined. Each body would at max need 6 variables to be described, as stated in Section 3.3 meaning that 24 variables would be needed to completely describe the system. However, the wheels are constrained to move together with the casing. By inspection, the whole system has 6 degrees of freedom. The casing can move in three angular directions (ϕ, θ, ψ) and then each wheel can rotate an angle (ρ_1, ρ_2, ρ_3) around its own rotational axis independently of the casing. The system can thus be fully described of a set of generalized coordinates $q_1 \dots q_6$

$$\begin{aligned} \boldsymbol{q}_{1:3} &\triangleq (\phi, \theta, \psi)^T \\ \boldsymbol{q}_{4:6} &\triangleq (\rho_1, \rho_2, \rho_3)^T \end{aligned} \tag{4.2}$$

This implies that the number of generalized coordinates n equals the number of degrees of freedom p so $n = p = 6$ so the system is holonomic, given the descriptions in Section 3.3. This makes applying Lagrangian mechanics more straightforward.

4.2.1 Euler angle definitions

When orientation is described using Euler angles, the order of rotation matters, and around which axises. A common choice of Euler angles for air- and watercraft is the so called (Z, Y', X'') convention, sometimes called the aerospace sequence [31]. However during analysis of rotating bodies such as the spinning top the (Z, X', Z'') or (Z, Y', Z'') seems more common. Although any Euler angle definition completely can describe orientation, the latter two mentioned here seems favorable in this context. The vertical inclination is the single most important parameter when balancing on a corner, so it makes sense to express it as plain and simple as possible. When using the latter two, the inclination is given directly from one of the three Euler angles. With the aerospace sequence a function using two Euler angles are needed. Therefore, we choose the (Z, X', Z'') convention with ϕ representing a rotation around the Z_A axis, θ an around the $X_{A'}$ axis and ψ around the $Z_{A''}$ axis.

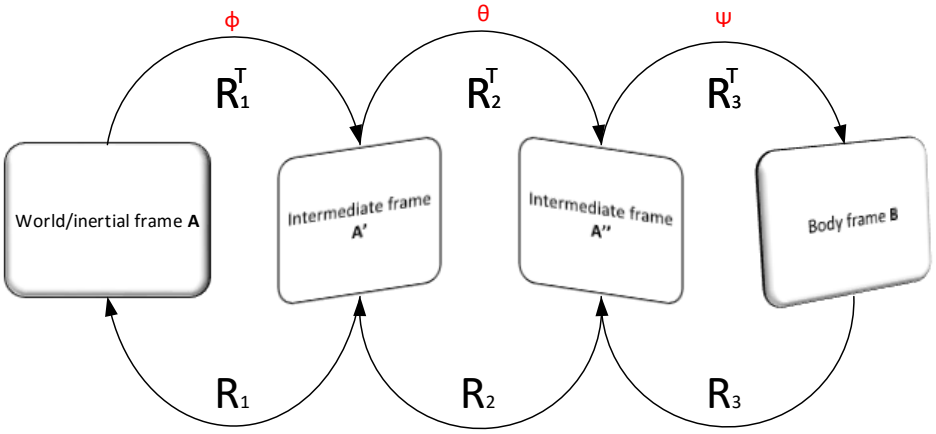


Figure 4.2: Overview of the coordinate frame and rotation matrices for the (Z, X', Z'') Euler angles used

An overview of the frames and rotation matrices used are shown in Figure 4.2. Each of the elementary rotation matrices are found as

$$R_1 = R_z(\phi) \quad R_2 = R_x(\theta) \quad R_3 = R_z(\psi) \tag{4.3}$$

where $R_x(\cdot), R_y(\cdot), R_z(\cdot)$ are described in Section 3.1. The full rotation matrix, used to decribe a vector in the body frame B in the inertial world frame A is then

$$R_B^A = R_1 R_2 R_3 = R_z(\phi) R_x(\theta) R_z(\psi) = \begin{pmatrix} C_\phi C_\psi - C_\theta S_\phi S_\psi & -C_\psi C_\theta S_\phi - C_\phi S_\psi & S_\phi S_\theta \\ C_\psi S_\phi + C_\phi C_\theta S_\psi & C_\phi C_\psi \; C_\theta - S_\phi S_\psi & -C_\phi S_\theta \\ S_\psi S_\theta & C_\psi S_\theta & C_\theta \end{pmatrix} \tag{4.4}$$

For instance, R_1 is used to describe how the frame $Z_{A'}$ is rotated an angle ϕ relative to the inertial world frame.
Later on it will be seen that to complete the modeling process, relations between the angular velocity and the derivatives of the Euler angles are needed. Generally, this can be described by

$$ {}^{B}\boldsymbol{\omega} = \boldsymbol{e}_\phi\dot{\phi} + \boldsymbol{e}_\theta\dot{\theta} + \boldsymbol{e}_\psi\dot{\psi} = \underbrace{\begin{bmatrix}\boldsymbol{e}_\phi & \boldsymbol{e}_\theta & \boldsymbol{e}_\psi\end{bmatrix}}_{\triangleq Q} \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{4.5} $$

where ${}^{B}\boldsymbol{\omega}_c$ is the angular velocity of the cube casing described in the body frame and unit vectors $\boldsymbol{e}$ are the axises of rotation in each of the three intermediate Euler frames described in the body frame [32]. Each of the three unit vectors describing the rotation axises can be found in different rotation matrices as

$$ \begin{aligned} I^{3\times3} &= \begin{bmatrix}\vdots & \vdots & \boldsymbol{e}_\psi\end{bmatrix} \\ (R_3)^T = R^B_{A''} &= \begin{bmatrix}\boldsymbol{e}_\theta & \vdots & \vdots\end{bmatrix} \\ (R_2R_3)^T = R^B_{A'} &= \begin{bmatrix}\vdots & \vdots & \boldsymbol{e}_\phi\end{bmatrix} \end{aligned} \tag{4.6} $$

where the vertical dots denote each of the two columns ignored in each rotation matrix in the current context. As mentioned in Section 3.1, the transpose of a rotation matrix inverses the order of rotation. While $R_1 - R_3$ rotate a vector from the intermediate frames toward the inertial world frame A, their transposes rotate a vector toward the moving body frame B. Note that the column vectors of the rotation matrix Q are non orthogonal in general, that is $\boldsymbol{e}_\phi \times \boldsymbol{e}_\theta \neq \boldsymbol{e}_\psi$. Combining (4.6) and (4.5) yields

$$ {}^{B}\boldsymbol{\omega}_c = \begin{bmatrix}\sin\psi\sin\theta & \cos\psi & 0\\ \cos\psi\sin\theta & -\sin\psi & 0\\ \cos\theta & 0 & 1\end{bmatrix} \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{4.7} $$

This is the relation between the (Z, X', Z'') Euler angles derivatives and angular velocity expressed in the body frame B.

4.2.2 Corner balancing point

The most common case during 3D operation is balancing on a corner. What are the Euler angles $\boldsymbol{E}_0 = (\phi_0, \theta_0, \psi_0)$ when cube is balancing perfectly on the corner? Due to the definition of the body frame, the Euler angles will not be zero here. By inspection, it doesn't matter what the first rotation ϕ_0 around the inertial frames vertical axis Z_A is so ϕ_0 can be anything.
When the cube is balancing perfectly on the corner the vector from corner to the center of gravity will have no components in the horizontal (X_A, Y_A) plane. The nominal

vector to the center of gravity when expressed in the body frame is ${}^{B}\boldsymbol{r}_{cb} = (r, r, r)^T$. During perfect corner balancing i will be ${}^{A}\boldsymbol{r}_{cb} = (0, 0, \sqrt{3})^T$ when expressed in the global reference frame A . However, lets consider a more general case where the center of gravity is offset $\Delta\boldsymbol{r}$, so the vector from corner to the center of gravity can be described

$$\tilde{\boldsymbol{r}}_{cb} = \boldsymbol{r}_{cb} + \Delta\boldsymbol{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}^T \tag{4.8}$$

The problem of finding the set point can thus be formulated as

$${}^{A'}\tilde{\boldsymbol{r}}_{cb} = {}^{A}\tilde{\boldsymbol{r}}_{cb} = R_2 R_3 \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} C_\psi & -S_\psi & 0 \\ C_\theta S_\psi & C_\psi C_\theta & -S_\theta \\ S_\psi S_\theta & C_\psi S_\theta & C_\theta \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ ||\tilde{\boldsymbol{r}}_{cb}|| \end{bmatrix} \tag{4.9}$$

From the first row we have

$$\begin{aligned} \cos(\psi) r_x - \sin(\psi) r_y = 0 \\ \implies \psi = \operatorname{atan2}(r_x, r_y) \end{aligned} \tag{4.10}$$

Substituting this into row 2 yields and solving for θ yields one of multiple possible solutions

$$\begin{aligned} \cos(\theta)\sin(\psi) r_x + \cos(\theta)\cos(\psi) r_y - \sin(\theta) r_z = 0 \\ \implies \theta = \operatorname{atan2}\left(\sin(\psi) r_x + \cos(\psi) r_y, \quad r_z\right) = \operatorname{atan2}\left(\sin(\psi)(r_x^2 + r_y^2), \quad r_x r_z\right) \end{aligned} \tag{4.11}$$

Two expressions are given as one of them might run faster in software. The structure of this geometrical problem implies that row three will be satisfied. For the ideal case when the alignment of the center of gravity is ${}^{B}\boldsymbol{r}_{cb} = (r, r, r)^T$ the angle ψ is simply $\psi_0 = \pi/4$. This then implies

$$\theta_0 = \arctan(\sqrt{2}) = \arccos(1/\sqrt{3}) = \arcsin(\sqrt{2}/\sqrt{3}) \approx 55\,\text{deg}$$

The point $(\phi_0, \theta_0, \psi_0)$ is often of such interest that it is stated explicitly

$$\boldsymbol{E}_0 = \begin{bmatrix} \phi_0 \\ \theta_0 \\ \psi_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \arctan(\sqrt{2}) \\ \pi/4 \end{bmatrix} \tag{4.12}$$

Having found this solution it can be seen that (4.9) can be multiplied from the left by any Z -axis rotation matrix $R_z(\phi)$, leaving the solution unchanged, as was already deduced by inspection.

4.3 Inertia tensors

The inertia tensors for each body of the cube needs to be found in order to find the equations of motion.

4.3.1 Cube casing

The cube casing is assumed to be symmetric. Given the choice of body coordinates as shown in Figure 4.1, the cube will have the inertia tensor

$$I_c = \begin{bmatrix} I_{c0} & 0 & 0 \\ 0 & I_{c0} & 0 \\ 0 & 0 & I_{c0} \end{bmatrix} \tag{4.13}$$

Due to a cubes symmetry, various principal axis coordinate systems can be found. For a coordinate system with the origin at a corner of the cube, all coordinate systems having an axis going from one corner, through the center of gravity, and out in the adjacent corner are principal [8].
And, any coordinate system attached to the mass center of a cube is principal axis. Also, the inertia of the cube for each of these coordinate systems will have all of it's diagonal components equal.
This can be shown by combining (4.13) and (3.14) according to

$$I_T = R_B^T I_c (R_B^T)^T = R_B^T I_{c0} I^{3\times 3} (R_T^B)^T = I_{c0} R_T^B (R_T^B)^T = I_{c0} I^{3\times 3} = I_c \tag{4.14}$$

Here it was used that a transformation matrix multiplied by its transpose yields an identity matrix.

4.3.2 Reaction wheels

The reaction wheels (subscripted w) are assumed to have an inertia expressed in their own respective body frame $(X_{wi}, Y_{wi}, Z_{wi}), i = 1, 2, 3$

$$I_w = \begin{bmatrix} I_{w0} & 0 & 0 \\ 0 & I_{w0} & 0 \\ 0 & 0 & I_{wz} \end{bmatrix} \tag{4.15}$$

The two equal elements I_{w0} are due to the wheels symmetry around the two axises X_{wi}, Y_{wi}, and I_{wz} is the inertia around each wheels shaft rotation axis Z_{wi}.
By applying (3.14) we find the inertia matrix for the wheels expressed in the body frame of the cube

$$I_k = {}^B I_{wi} = R_{wi}^B I_w {R_{wi}^B}^T \tag{4.16}$$

where the short hand notation I_k for the bodies $k = 2, 3, 4$ enumerating the wheels is used, as defined in Section 4.1. Due to the orthogonality mounting of the wheels in

the cube, ${}^{B}I_{wi}$ will be I_w with the elements shuffled around the diagonal. Evaluating (4.16) for each wheel yields

$$\begin{aligned} I_2 &= {}^{B}I_{w1} = \text{diag}\begin{bmatrix} I_{wz} & I_{w0} & I_{w0} \end{bmatrix} \\ I_3 &= {}^{B}I_{w2} = \text{diag}\begin{bmatrix} I_{w0} & I_{wz} & I_{w0} \end{bmatrix} \\ I_4 &= {}^{B}I_{w3} = \text{diag}\begin{bmatrix} I_{w0} & I_{w0} & I_{wz} \end{bmatrix} \end{aligned} \tag{4.17}$$

where diag[·] is a function returning a diagonal matrix from an input vector.

4.4 Lagrangian approach

The cube was modeled using the Lagrangian approach to mechanics with the aid of the symbolic mathematics software package `Wolphram Mathematica` . However, the results were extremely hard to handle. Each degree of freedom had an equation that was around a full page long, full of coupled trigonometric expressions. With the lead of ETH in Switzerland obtaining really compact equations [1] using Kane's method, the Lagrangian approach was abandoned. However, for reference, a simplified version is presented here where the cube is modeled as a spinning top. After all the cube behaves is a spinning top if the reaction wheels are neglected. While still unknown if this is a decent approximation here, approximating mechanics systems with a spinning top has found applications in areas from astronomy to nuclear physics [33]. This to give an idea about how the Lagrangian approach to mechanics differ to Kane's method and what significance it can have.
First, the body fixed coordinate system in Figure 4.1 is rotated $-45\,\text{deg}$ around the Z_B axis and $a = -\arccos(\frac{1}{\sqrt{3}}) =\approx -55\,\text{deg}$ around the X_B axis to obtain the coordinate system as depicted in Figure 4.3

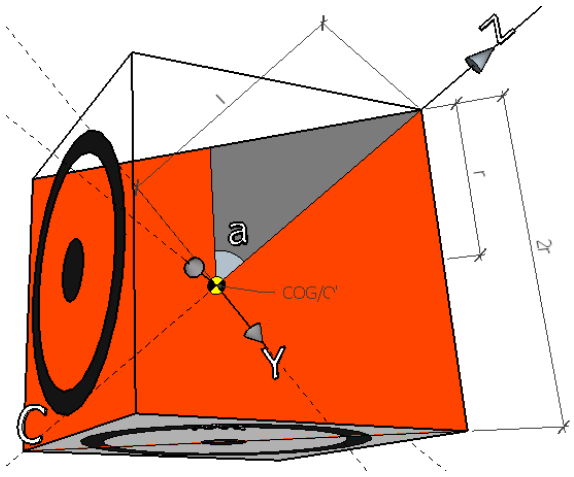


Figure 4.3: Some dimensions and definitions when modeling the cube similar to a spinning top using (Z, X, Z'') Euler angles. Note that this coordinate definition is not used for the final equations of motion. For those the definitions in Figure 4.1 are used.

With the choice of coordinates as depicted in Figure 4.3 takes the simpler form $r_{cb} = (0, 0, \sqrt{3})^T$ and the expression for kinetic energy becomes

$$T_c = \frac{1}{2} m_c ||v_c||^2 + \frac{1}{2} \boldsymbol{\omega}_c^T I_c \boldsymbol{\omega}_c = \frac{1}{2}(\omega_x^2 + \omega_y^2 + \omega_z^2) I_{c0} + \left(\omega_x^2 + \omega_y^2\right) 3 m_c r^2 \tag{4.18}$$

where velocity and angular velocity have been described in the cube frame, by applying the results shown in Section 3.4 and 3.2.1 . Note that the rightmost term in 4.18 is a contribution due to the center of gravity having to translate during a rotation in the around the X_B, Y_B axis. Already here we get a hint about how the Lagrangian approach implicitly handle motion constraints. To get more compact equations, we do the substitution

$$I_{c1} \triangleq I_{c0} + 3 m_c r^2 \tag{4.19}$$

and obtain

$$T_c = \frac{1}{2} \left(I_{c0} \omega_z^2 + I_{cxy} (\omega_x^2 + \omega_y^2) \right) \tag{4.20}$$

Now the kinetic energy is formulated on a form that can be found in various places literature covering the spinning top. For instance in Classical mechanics by Tai Chow [33], which is deemed to have an excellent treatment of the spinning top according to Goldstein [34] . For the Lagrangian

$$L = T_c - U_c$$

the conservative energy U_c is just the potential energy. With the contact point as reference,

$$U_c = m_c g \sqrt{3} r \cos(\theta) \tag{4.21}$$

As is stated in Section 3.5, the Lagrangian needs to be formed using generalized coordinates that are holonomic and complete. These criteria are fulfilled by the Euler angles $q_{1:3} = (\phi, \theta, \psi)$ defined and described in Section 4.2. Therefore the relations between the Euler angle derivatives and angular velocity in (4.7) are substituted into (4.18) and combined with (4.21) to yield

$$\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} I_{c1} \left({\dot{q_1}}^2 \frac{1}{2} (1 - \cos(2q_2)) + {\dot{q_2}}^2 \right) + \frac{1}{2} I_{c0} \left(\dot{q_1} \cos(q_2) + \dot{q_3} \right)^2 - \sqrt{3} g M r \cos(q_2) \tag{4.22}$$

Beginning with the generalized conservative forces

$$\frac{\partial \mathcal{L}}{q_1} = \frac{\partial \mathcal{L}}{q_3} = 0$$

$$\frac{\partial \mathcal{L}}{q_2} = - I_{c0} \dot{q_1} \sin(q_2) \left(\dot{q_1} \cos(q_2) + \dot{q_3} \right) + \frac{1}{2} I_{c1} {\dot{q_1}}^2 \sin(2q_2) + \sqrt{3} g M r \sin(q_2)$$

The partial derivatives for the generalized coordinates q_1 and q_2 are zero. This means that generalized momenta [5] as briefly described in Section 3.5 will be preserved, and thus constant for generalized coordinates q_1 and q_3. However along the q_2 coordinate, describing inclination, the gravity torque performs work meaning the generalized momenta will not be constant along q_2. The magnitude of the gravity induced torque is identified as $\sqrt{3}gMr\sin(q_2)$.
Then, by applying (3.20) in Section 3.5 the full differential equations describing the cube operating as a spinning top are found as

$$\begin{aligned}\frac{1}{2}\ddot{q_1}\left(3mr^2\left(1-\cos(2q_2)\right)+2I_{c0}\right)+I_{c0}\cos(q_2)\ddot{q_3}-\\ I_{c0}\dot{q_2}\sin(q_2)\dot{q_3}+3mr^2\dot{q_1}\dot{q_2}\sin(2q_2)=&Q_1\\ I_{c1}\ddot{q_2}+I_{c0}\dot{q_1}\sin(q_2)\dot{q_3}-\frac{3}{2}mr^2\dot{q_1}^2\sin(2q_2)-\sqrt{3}gMr\sin(q_2)=&Q_2\\ I_{c0}\left(\ddot{q_1}\cos(q_2)-\dot{q_1}\dot{q_2}\sin(q_2)+\ddot{q_3}\right)=&Q_3\end{aligned} \tag{4.23}$$

were Q_1 , Q_2 and Q_3 are generalized non conservative forces acting along each generalized coordinate q_1, q_2 and q_3. This would have to be related to the motor torques. Some simplifications could be made due to I_{c1} being a function of I_{c0}. However, by inspection of (4.23) the system has rather complex dynamics. For one $\cos(q_2)$ is being multiplied by the highest derivatives in the first and third equation. Different approaches exist in literature for how to deal with these equations. In some lecture notes from MIT [35] the constant general momenta for two of the generalized coordinates are utilized and substituted into Q_2. In a publication found at the servers of the University of Surrey an approach where an equation system similar to (4.23) is combined is presented. The highest derivatives are then solved for and can then be integrated numerically. An approach in this context would be to assume rather perfect corner balancing so $q_2 \approx 0$ to yield a small angle approximation $\cos(q_2) \approx 0$ and $\sin(q_2) \approx q_2$. However, this at this point this scalar approach was abandoned. Remember, this is the cube without the reaction wheels When the reaction wheels are included (4.23) expand to extremely complicated expressions. It's not for nothing that these kind of problems are described in literature as hard, cumbersome or even very hard to solve[35, 5, 33]. The approach to circumvent this problem is the application of Kane's method as described in Section 4.5. This yields compact equations that describe angular velocity rather than orientation. In a following stage this is then combined with various kinematic relations to descriptions of orientation.

4.4.1 Transformation using Lie algebra

In the first paper covering modeling released from the original creators of Cubli at ETH [36] a very compact model on matrix form of the system is released. It was derived using Lagrangian mechanics, but includes a neat variable transform.
The derivation can be found in a separate appendix of the same report [37]. Some comments are made on this derivation. First, let $S(\boldsymbol{\omega})$ denote the skew symmetric

matrix that can replace a cross product involving $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$ such that

$$\boldsymbol{\omega} \times \boldsymbol{r} = S(\boldsymbol{\omega})\boldsymbol{r} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \boldsymbol{r}$$

It turns out that this skew symmetric matrix has a multitude of interesting relations to a rotation matrix R, see for instance the comprehensive summary [38]. One such relation is

$$S(\boldsymbol{\omega}) = R^T S(R\boldsymbol{\omega})R = R^T \dot{R}$$

This is part of a larger field in mathematics called *Lie algebra* [39]. The rotation matrix R_B^A between the body frame B and the inertial reference frame A contains the information regarding the cubes orientation. The matrix $S(\omega)$ on the other hand contains the information regarding the velocity of the cube. Denoting the wheels angular orientation around their rotating axises $\boldsymbol{\rho}$ the Lagrangian $\mathcal{L}(R, \boldsymbol{\rho}, \dot{R}, \dot{\boldsymbol{\rho}})$ can be formed. As mentioned in Section 3.5.1 the Lagrangian is a function of velocity but has to be evaluated using expressions for orientation. This is due to Lagrangian mechanics being based on the principle of virtual work, which in its turn needs virtual displacement to be evaluated. However, in the released appendix [37] various algebraic results are applied to achieve a coordinate transform. Interestingly, the resulting equations do not involve any generalized coordinates as opposed to 4.23 but angular velocity. This approach was not found in any of the studied material dedicated to classical mechanics [32, 40, 5, 9, 34, 41]. A relation between Eulers Laws of rotation and Lagrangian Mechanics, were most of the analysis was carried out on matrix form was found though [42]. As such, it appears reasonable to state that an advanced and non standard approach was carried out in the mentioned paper from ETH [37].

4.5 Kane's approach

As was seen in Section 4.4 the modeling process using Lagrangian mechanics yielded fairly complicated equations of motion. And, this without even including the dynamics from the reaction wheels. By using Euler's laws of rotation [5], the cube dynamics, excluding the wheels, could be modeled as

$$I\dot{\boldsymbol{\omega}}_c + \boldsymbol{\omega}_c \times \boldsymbol{L} = \boldsymbol{M} \tag{4.24}$$

where $\boldsymbol{\omega}$ and $\boldsymbol{L} = I\boldsymbol{\omega}_c$ are the angular velocity and momentum of the cube and $\boldsymbol{M}$ is net the moment acting on each axis of the cube. All of these vectorial quantities are defined in the body frame of the cube. The moment is then found as

$$M = T_m + \boldsymbol{r}_{cb} \times (R_A^B\, \boldsymbol{g}) \tag{4.25}$$

where T_m are the motor torques acting on the cube and the rightmost term is gravity induced torque by the gravity $\boldsymbol{g} = \begin{bmatrix} 0 & 0 & -g_0 \end{bmatrix}^T$. As the equations of motion are defined in the body frame, the gravity torque vector must be expressed in the body frame using the rotation matrix R_A^B . This formulation is further treated in "Nonlinear Dynamics of the 3D Pendulum" [43], however reaction wheels are not included in that analysis.

The equations of motion (4.24) feature relative simplicity and also a compact vector formulation, as opposed to what was achieved using Lagrangian mechanics. One drawback though is that they don't provide any information by itself about the orientation of the system in space. In Simulation this problem is easily circumvented as described in Section 4.8. Also relations between orientation and angular velocity can be substituted in to the differential equations describing angular velocity later.

Having established a goal for how the equations of motions should appear, the goal is obtaining equations that appear on a form more similar to (4.24) than those presented in Section 4.4. A main lead are the equations of motions for Cubli presented by ETH in a conference paper [1]. These appear on a very compact matrix form. However, as it is a condensed paper dealing with various topics it is not outlined in detail how the results were achieved. Thus, modeling starts by following List 3.1 from Section 3.6.

4.5.1 Generalized and partial speed

The generalized coordinates were already defined in Section 4.2. The scalar quantities generalized speed are then chosen as

$$\begin{aligned} \boldsymbol{u}_{1:3} &\triangleq \boldsymbol{\omega}_c = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T \\ \boldsymbol{u}_{4:6} &\triangleq \boldsymbol{\omega}_w = \begin{bmatrix} \omega_{w1} & \omega_{w2} & \omega_{w3} \end{bmatrix}^T \end{aligned} \tag{4.26}$$

That is, the angular velocity of the cube casing and the rotational speed of each wheel around it's respective rotational axis.

4.5.2 Jacobians

Now with generalized coordinates and speeds being defined the Jacobian matrices in Kane's equation on matrix form (3.35) shown in Section 3.6.3 are to be found. This requires having found expressions for the linear and angular velocity which was done in Section 4.1.1 . The Jacobians are J_k for each body $k = 1, 2, 3, 4$ corresponding to the casing, wheel 1 , wheel 2 and wheel 3 respectively.

The transposed Jacobians $(J_{\omega k})^T \in \mathbb{R}^{6\times 3}$ related to angular velocity are then

$$\left[\frac{\partial \boldsymbol{\omega}_c}{\partial \boldsymbol{u}}\right]^T = (J_{\omega 1})^T = \left[\begin{array}{c} I^{3\times 3} \\ \hline 0^{3\times 3} \end{array}\right] \tag{4.27}$$

$$\left[\frac{\partial \boldsymbol{\omega}_2}{\partial \boldsymbol{u}}\right]^T = (J_{\omega 2})^T = \left[\begin{array}{c} \mathrm{diag}(0,1,1) \\ \hline \mathrm{diag}(1,0,0) \end{array}\right] \tag{4.28}$$

$$\left[\frac{\partial \boldsymbol{\omega}_3}{\partial \boldsymbol{u}}\right]^T = (J_{\omega 3})^T = \left[\begin{array}{c} \mathrm{diag}(1,0,1) \\ \hline \mathrm{diag}(0,1,0) \end{array}\right] \tag{4.29}$$

$$\left[\frac{\partial \boldsymbol{\omega}_4}{\partial \boldsymbol{u}}\right]^T = (J_{\omega 4})^T = \left[\begin{array}{c} \mathrm{diag}(1,1,0) \\ \hline \mathrm{diag}(0,0,1) \end{array}\right] \tag{4.30}$$

where $\mathrm{diag}(\boldsymbol{v})$ in an operator that creates a diagonal matrix of the vector $\boldsymbol{v}$. Continuing with the Jacobians for linear velocity, all bodies k have a linear velocity expressed as $\boldsymbol{v}_k = \boldsymbol{\omega}_c \times r_k$. Rewriting this using the skew symmetric matrix $S(\boldsymbol{r}_k)$ to replace a cross product according to $\boldsymbol{\omega_c} \times \boldsymbol{r}_k = -\boldsymbol{r}_k \times \boldsymbol{\omega_c} = -S(\boldsymbol{r}_k)\boldsymbol{\omega}_c$ we have

$$\frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{u}} = \begin{bmatrix} \frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{u}_{1:3}} & \frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{u}_{4:6}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{\omega}_c} & \frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{\omega}_w} \end{bmatrix} = \begin{bmatrix} \frac{\partial\ (-S(\boldsymbol{r}_k)\boldsymbol{\omega}_c)}{\partial \boldsymbol{\omega}_c} & \boldsymbol{0}^{3\times 3} \end{bmatrix} = \begin{bmatrix} -S(\boldsymbol{r}_k) & 0 \end{bmatrix} \tag{4.31}$$

By the properties of skew symmetric matrices we have $-S(\boldsymbol{r}_k) = S(\boldsymbol{r}_k)^T$ implying $[-S(\boldsymbol{r}_k)]^T = S(\boldsymbol{r}_k)$ so the transposed jacobians are

$$\begin{aligned} \left[\frac{\partial \boldsymbol{v}_1}{\partial \boldsymbol{u}}\right]^T &= (J_{v1})^T = \left[\begin{array}{c} S(\boldsymbol{r}_{cb}) \\ \hline 0^{3\times 3} \end{array}\right] \\ \left[\frac{\partial \boldsymbol{v}_{v2}}{\partial \boldsymbol{u}}\right]^T &= (J_{v2})^T = \left[\begin{array}{c} S(\boldsymbol{r}_2) \\ \hline 0^{3\times 3} \end{array}\right] \\ \left[\frac{\partial \boldsymbol{v}_{v3}}{\partial \boldsymbol{u}}\right]^T &= (J_{v3})^T = \left[\begin{array}{c} S(\boldsymbol{r}_3) \\ \hline 0^{3\times 3} \end{array}\right] \\ \left[\frac{\partial \boldsymbol{v}_{v4}}{\partial \boldsymbol{u}}\right]^T &= (J_{v4})^T = \left[\begin{array}{c} S(\boldsymbol{r}_4) \\ \hline 0^{3\times 3} \end{array}\right] \end{aligned} \tag{4.32}$$

This result should hold true for any definitions of position vectors as long as they are independent of the generalized speed of choice.

4.5.3 Generalized forces

The generalized forces and torques $\boldsymbol{F}_k, \boldsymbol{F}_k^*, \boldsymbol{T}_k, \boldsymbol{T}_k^*$ in (3.35) for each body $k = 1, 2, 3, 4$ need to be determined. It should be noted that reaction forces such as centripetal forces that do no work need not to be included in the analysis [6]. This is one of the main benefits of methods such as Kane's and Lagrange over the Newtonian approach.
Assuming that the cube is not subjected to any external pushes, the only net generalized force working on each body is gravity $m_k \boldsymbol{g}$

Gravity vector in the body frame when using the (Z, X', Z'') Euler angles is

$$\boldsymbol{g} = -g_0 \begin{pmatrix} \sin(\psi)\sin(\theta) \\ \cos(\psi)\sin(\theta) \\ \cos(\theta) \end{pmatrix} \tag{4.33}$$

The generalized inertia force for each body is $m_k \boldsymbol{a}_k$ where $\boldsymbol{a}_k$ is the linear acceleration determined in Section 4.1.1. The cube is subjected to the motor torques

$$\boldsymbol{T}_1 = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{4.34}$$

Due to the definition of coordinates T_x, T_y and T_z correspond to the motor torque from motor 1 to motor 3 respectively. Then, each wheel is subjected to one of each of these motor torques respectively, having opposite sign due to Newtons third law. Furthermore the reaction wheels are affected by viscous friction when spinning in the air. Although some polynomial functions of the of the wheel velocities would be more accurate in a non-linear model. However a linear model for this torque $b\boldsymbol{\omega}_w$ is chosen, where $b(b > 0)$ is a damping coefficient having the unit $[Nm\ s]$. The damping is unknown and based on experience it can be easier to make a good overall model fit using a simpler model structure, rather than for instance assuming polynomial model in this case, when using system identification techniques.
The active torques acting on the wheels are then

$$\boldsymbol{T}_2 = \begin{bmatrix} -T_x + b\omega_{w1} \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{T}_3 = \begin{bmatrix} 0 \\ -T_y + b\omega_{w2} \\ 0 \end{bmatrix} \quad \boldsymbol{T}_4 = \begin{bmatrix} 0 \\ 0 \\ -T_z + b\omega_{w3} \end{bmatrix} \tag{4.35}$$

The direction of the viscous torque is reversed due to the definition of what is the positive rotational direction for the reaction wheels.
As mentioned in Section 3.6.1 the generalized inertia torques are Eulers laws of rotation for each body so

$$T_k^* = I_k \dot{\boldsymbol{\omega}}_k + \boldsymbol{\omega}_k \times (I\boldsymbol{\omega}_k) \tag{4.36}$$

where the inertia tensors I_k for each body are found in Section 4.3 and expressions for angular velocities $\boldsymbol{\omega}_k$ in Section 4.1.1.

4.5.4 Obtaining the equations of motion

While the equations of motions already can be found, this will appear on a form that is not as compact and useful. Here it will be shown step by step how these can be rearranged, and the final results is shown in Section 4.5.5. The equations of motion $\boldsymbol{U}$ are evaluated as a six dimensional vector, that is $\boldsymbol{U} \in \mathbb{R}^{6\times 1}$. To simplify the analysis the transposes of the Jacobian matrices will be split according to the

syntax $\left(J_{\omega k}^T\right)_{i:j}$ and $\left(J_{vk}^T\right)_{i:j}$ meaning that rows i to j are selected. Splitting at the third degree of freedom, which is by the horizontal dashed delimiters in (4.30) and (4.32), yields

$$\sum_{k=1}^{4}\left[\left(J_{\omega k}^T\right)_{1:3}(\boldsymbol{T}_i-\boldsymbol{T}_i^*)+\left(J_{vk}^T\right)_{1:3}(\boldsymbol{F}_i-\boldsymbol{F}_i^*)\right]=\mathbf{0}\in\mathbb{R}^{3\times 1} \tag{4.37}$$

and

$$\sum_{k=1}^{4}\left[\left(J_{\omega k}^T\right)_{4:6}(\boldsymbol{T}_i-\boldsymbol{T}_i^*)+\left(J_{vk}^T\right)_{4:6}(\boldsymbol{F}_i-\boldsymbol{F}_i^*)\right]=\mathbf{0}\in\mathbb{R}^{3\times 1} \tag{4.38}$$

The reason for the split up is because the degrees of freedom r_1-r_3 are directly related to the cube casing and r_4-r_6 to the reaction wheels.

4.5.4.1 Wheel rotation

In (4.38) there are many zero elements in the Jacobians and is rather easily evaluated to be

$$\begin{aligned} I_{wz}\dot{\omega}_{w1}&=T_x-b\omega_{w1}\\ I_{wz}\dot{\omega}_{w2}&=T_y-b\omega_{w2}\\ I_{wz}\dot{\omega}_{w3}&=T_z-b\omega_{w3}\end{aligned} \tag{4.39}$$

That is, the wheel rotations around their rotational axises is governed by the motor torque inputs and some drag resistance. By defining $\Theta_{wz}\triangleq\text{diag}[I_{wz},I_{wz},I_{wz}]$ the model for the wheel rotation can be formulated compactly as

$$\Theta_{wz}\dot{\boldsymbol{\omega}}_w=\boldsymbol{T}_1-b\boldsymbol{\omega}_w \tag{4.40}$$

This model assumes a friction free coupling between the casing and the reaction wheels.

4.5.4.2 Cube casing

If there were no wheels in the cube, all jacobians related to the wheels would be zero matrices. As such, the whole cube dynamics would be described by the terms

$$\begin{aligned}&\left(J_{\omega 1}^T\right)_{1:3}(\boldsymbol{T}_1-\boldsymbol{T}_1^*)+\left(J_{v1}^T\right)_{1:3}(\boldsymbol{F}_1-\boldsymbol{F}_1^*)=\\ &=\boldsymbol{T}_1-\boldsymbol{T}_1^*\ +\ S(\boldsymbol{r}_{cb})\times(\boldsymbol{F}_1-\boldsymbol{F}_1^*)=\\ &=\boldsymbol{T}_1-I_1\dot{\boldsymbol{\omega}}_c-\boldsymbol{\omega}_c\times(I_1\boldsymbol{\omega}_c)+\boldsymbol{r}_{cb}\times m_c\boldsymbol{g}-\boldsymbol{r}_{cb}\times\boldsymbol{F}_1^*\end{aligned} \tag{4.41}$$

Analyzing the last row, T_1 is the vector of the three motor torques. The next two terms are the inertia torque for the cube casing. However due to the cubes inertia matrix I_1 being diagonal with equal elements, the cross product $\boldsymbol{\omega}_c \times (I_1 \boldsymbol{\omega}_c)$ is zero. Then $\boldsymbol{r}_{cb} \times m_c \boldsymbol{g}$ is the gravity induced torque around the balancing corner. Expanding and evaluating the remaining term

$$\begin{aligned} -\boldsymbol{r}_{cb} \times \boldsymbol{F}_1^* = &- \boldsymbol{r}_{cb} \times [m_c a_c] = \\ &- m_c \boldsymbol{r}_{cb} \times [\dot{\boldsymbol{\omega}}_c \times \boldsymbol{r}_{cb} + \boldsymbol{\omega}_c \times (\boldsymbol{\omega}_c \times \boldsymbol{r}_{cb})] = \\ &- m_c \boldsymbol{r}_{cb} \times (\dot{\boldsymbol{\omega}}_c \times \boldsymbol{r}_{cb}) \quad - m_c \boldsymbol{r}_{cb} \times [\boldsymbol{\omega}_c \times (\boldsymbol{\omega}_c \times \boldsymbol{r}_{cb})] \\ &\; m_c S(r_{cb})^2 \dot{\boldsymbol{\omega}}_c - m_c \boldsymbol{r}_{cb} \times [(\boldsymbol{r}_{cb} \times \boldsymbol{\omega}_c) \times \boldsymbol{\omega}_c] \end{aligned} \tag{4.42}$$

Here the cross product identity $\boldsymbol{A} \times \boldsymbol{B} = -\boldsymbol{B} \times \boldsymbol{A}$ was applied on the terms of the next last row. Then the cross products involving $\dot{\omega}_c$ and $\boldsymbol{r}_{cb}$ were replaced by multiplication with a skew symmetric matrix as more thoroughly described in Section 4.4.1.
Continuing, it would be desirable to separate $\boldsymbol{r}_{cb}$, being constant in the body frame from the time varying ω_c as far as possible. The Jacobi identity states that $\boldsymbol{A} \times (\boldsymbol{B} \times \boldsymbol{C}) = (\boldsymbol{C} \times \boldsymbol{A}) \times \boldsymbol{B} + (\boldsymbol{A} \times \boldsymbol{B}) \times \boldsymbol{C}$. Applying this on the last term

$$\begin{aligned} \underbrace{\boldsymbol{r}_{cb}}_{A} \times [\underbrace{(\boldsymbol{r}_{cb} \times \boldsymbol{\omega}_c)}_{B} \times \underbrace{\boldsymbol{\omega}_c}_{C}] &= (\boldsymbol{\omega}_c \times \boldsymbol{r}_{cb}) \times (\boldsymbol{r}_{cb} \times \boldsymbol{\omega}_c) + (\boldsymbol{r}_{cb} \times (\boldsymbol{r}_{cb} \times \boldsymbol{\omega}_c)) \times \boldsymbol{\omega}_c = \\ &= -\underbrace{(\boldsymbol{\omega}_c \times \boldsymbol{r}_{cb}) \times (\boldsymbol{\omega}_c \times \boldsymbol{r}_{cb})}_{=0} + (\boldsymbol{r}_{cb} \times (\boldsymbol{r}_{cb} \times \boldsymbol{\omega}_c)) \times \boldsymbol{\omega}_c = \\ &= \left[S(\boldsymbol{r}_{cb})^2 \boldsymbol{\omega}_c \right] \times \boldsymbol{\omega}_c \end{aligned} \tag{4.43}$$

Combining the results (4.42) and (4.43) into (4.41) we obtain

$$\begin{aligned} \left(J_{\omega 1}^T \right)_{1:3} (\boldsymbol{T}_1 - \boldsymbol{T}_1^*) + \left(J_{v1}^T \right)_{1:3} (\boldsymbol{F}_1 - \boldsymbol{F}_1^*) = \\ \boldsymbol{T}_1 - I_1 \dot{\boldsymbol{\omega}}_c + \boldsymbol{r}_1 \times m_c \boldsymbol{g} + m_c S(\boldsymbol{r}_{cb})^2 \dot{\boldsymbol{\omega}}_c - \left[m_c S(\boldsymbol{r}_{cb})^2 \boldsymbol{\omega}_c \right] \times \boldsymbol{\omega}_c = \boldsymbol{0} \in \mathbb{R}^{3 \times 1} \end{aligned} \tag{4.44}$$

By inspecting the matrix exponential

$$S(\boldsymbol{r}_{cb})^2 = m_c \begin{bmatrix} -2r^2 & r^2 & r^2 \\ r^2 & -2r^2 & r^2 \\ r^2 & r^2 & -2r^2 \end{bmatrix}$$

this can be identified this an inertia tensor having opposite sign, as a proper inertia tensor should have elements greater than or equal to zero on its diagonal. Thus by defining $\tilde{I}_1 \triangleq -m_c S(\boldsymbol{r}_{cb})^2$ and inserting this into (4.44) and re-arranging the terms the final equation of motion defining the dynamics of the cube casing is found as

$$\left(I_1 + \tilde{I}_1 \right) \dot{\boldsymbol{\omega}}_c + \boldsymbol{\omega}_c \times (\tilde{I}_1 \boldsymbol{\omega}_c) = \boldsymbol{T}_1 + \boldsymbol{r}_1 \times m_c \boldsymbol{g} \tag{4.45}$$

We see that (4.45) is Euler's laws of rotation but with the addition of $\tilde{I}_1$. This is due to the cube being constrained to rotate around its corner in combination with the current choice of coordinate system. Neatly, Kane's method in combination with careful analysis gave what would otherwise have required the application of the parallel axis theorem in three dimensions. Combining the parallel axis theorem with Euler's laws of rotation would have been more effective though if the cube had no reaction wheels being considered.
Then a comment on the choice of coordinate system. For a choice of coordinates as during the Lagrangian modeling approach as described in Section 4.4 we would have obtained

$$-m_c S([0,0,\sqrt{3}])^2 = m_c \begin{bmatrix} 3r^2 & 0 & 0 \\ 0 & 3r^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.46}$$

which complies with the inertial components identified in that section. This makes perfect physical sense as the cube would only have higher inertia around the X_B and Y_B axises with those choice of coordinates. This simpler structure of the added inertia would be desirable, compared to the current $\tilde{I}_1$. However, the current choice of coordinate system yielding $\boldsymbol{r}_{cb} = (r, r, r)$ is a trade-off in order to obtain simpler expressions for the position, velocity and acceleration of the wheels.

4.5.4.3 Wheel impact on cube casing

Having obtained (4.45) the analysis is continued with the objective of finding terms similar in structure. This alleviates the process of formulating the final equations of motions on a compact form.
Beginning by investigating the first terms related to wheel 1

$$\left(J_{\omega 2}^T\right)_{1:3} (\boldsymbol{T}_2 - \boldsymbol{T}_2^*) = \underbrace{\text{diag}(0,1,1)\boldsymbol{T}_2}_{=\boldsymbol{0}} - \underbrace{\text{diag}(0,1,1)}_{\triangleq D_2} \boldsymbol{T}_2^* = \\ -D_2 \left[I_2 \dot{\boldsymbol{\omega}}_2 + \boldsymbol{\omega}_2 \times (I_2 \boldsymbol{\omega}_2) \right]$$

Continuing by rewriting the angular velocity for wheel 1 as $\boldsymbol{\omega}_2 = D_2\boldsymbol{\omega}_c + \boldsymbol{\omega}_2^*$ by defining $\boldsymbol{\omega}_2^* \triangleq (-\omega_{w1}, 0, 0)^T$ to obtain

$$-D_2 \left[I_2(D_2\dot{\boldsymbol{\omega}}_c + \dot{\omega}_2^*) + (D_2\boldsymbol{\omega}_c + \boldsymbol{\omega}_2^*) \times (I_2(D_2\boldsymbol{\omega}_c + \boldsymbol{\omega}_2^*)) \right] =$$

$$-D_2 \left[I_2(D_2\dot{\boldsymbol{\omega}}_c + \dot{\omega}_2^*) + \underbrace{(D_2\boldsymbol{\omega}_c \times (I_2 D_2 \boldsymbol{\omega}_c)}_{=\boldsymbol{0}} + (D_2\boldsymbol{\omega}_c) \times (I_2\boldsymbol{\omega}_2^*) + \boldsymbol{\omega}_2^* \times (I_2 D_2 \boldsymbol{\omega}_c) + \underbrace{\boldsymbol{\omega}_2^* \times (I_2\boldsymbol{\omega}_2^*)}_{=\boldsymbol{0}} \right]$$

Here two cross products vanish due to their involved vectors being parallel. It might be worth noting that the leftmost zero resultant crossproduct generally would not vanish if the inertia tensor I_2 had not the current symmetric structure. The terms left are then further evaluated to yield

$$\begin{aligned}\left(J_{\omega 2}^T\right)_{1:3}(\boldsymbol{T}_2 - \boldsymbol{T}_2^*) &= -D_2\left[I_2(D_2\dot{\boldsymbol{\omega}}_c + \dot{\omega}_2^*) + (D_2\boldsymbol{\omega}_c)\times(I_2\boldsymbol{\omega}_2^*) + \boldsymbol{\omega}_2^*\times(I_2D_2\boldsymbol{\omega}_c)\right] \\ &= -I_2D_2\dot{\boldsymbol{\omega}}_c - \boldsymbol{\omega}_2^*\times(D_2\boldsymbol{\omega}_c)\left[I_{w0} - I_{wz}\right]\end{aligned} \tag{4.47}$$

The terms for all wheels have the same structure. By doing the substitution $\boldsymbol{\omega}_k^* = -(I^{3\times3} - D_k)\omega_w$ it can be worked out that

$$\boldsymbol{\omega}_k^*\times(D_k\boldsymbol{\omega}_c) = -\boldsymbol{\omega}_w\times(D_k\boldsymbol{\omega}_c) + [I^{3\times3} - D_k]\boldsymbol{\omega}_w\times\boldsymbol{\omega}_c \tag{4.48}$$

Then by summation, after first defining $\Theta_{w0} \triangleq \text{diag}[2I_{w0}, 2I_{w0}, 2I_{w0}]$

$$\begin{aligned}&\sum_{k=2}^{4}\left[\left(J_{\omega k}^T\right)_{1:3}(\boldsymbol{T}_k - \boldsymbol{T}_k^*)\right] = \\ &-(I_2D_2 + I_3D_3 + I_4D_4)\dot{\boldsymbol{\omega}}_c - \left(-\boldsymbol{\omega}_w\times\left(\left[\sum D_k\right)\right]\boldsymbol{\omega}_c\right) + \left[\sum(I^{3\times3} - D_k\right]\boldsymbol{\omega}_w\times\boldsymbol{\omega}_c\Big)\left[I_{w0} - I_{wz}\right] = \\ &-\Theta_{w0}\dot{\boldsymbol{\omega}}_c - \boldsymbol{\omega}_c\times\boldsymbol{\omega}_w\left[I_{wz} - I_{w0}\right]\end{aligned} \tag{4.49}$$

Continuing in a similar way with the jacobians for linear velocity, yields similarly to for the cube casing

$$\left(J_{v2}^T\right)_{1:3}(\boldsymbol{F}_2 - \boldsymbol{F}_2^*) = \boldsymbol{r}_2\times m_w\boldsymbol{g} + m_wS(\boldsymbol{r}_2)^2\dot{\boldsymbol{\omega}}_c - \left[m_wS(\boldsymbol{r}_2)^2\boldsymbol{\omega}_c\right]\times\boldsymbol{\omega}_c \tag{4.50}$$

All wheels have terms with the same structure as (4.50) By defining $\tilde{I}_k \triangleq -m_wS(\boldsymbol{r}_k)^2$, similar to the cube casing, all these terms related to the linear velocity of the wheels for the degrees of freedom $r = 1, 2, 3$ can be rewritten as

$$\begin{aligned}&\sum_{k=2}^{4}\left[\left(J_{vk}^T\right)_{1:3}(\boldsymbol{F}_k - \boldsymbol{F}_k^*)\right] = \\ &= (\boldsymbol{r}_2 + \boldsymbol{r}_3 + \boldsymbol{r}_4)\times m_w\boldsymbol{g} - (\tilde{I}_2 + \tilde{I}_3 + \tilde{I}_4)\dot{\boldsymbol{\omega}}_c - \boldsymbol{\omega}_c\times\left[(\tilde{I}_2 + \tilde{I}_3 + \tilde{I}_4)\boldsymbol{\omega}_c\right]\end{aligned} \tag{4.51}$$

Defining $\tilde{I}_w \triangleq \tilde{I}_2 + \tilde{I}_3 + \tilde{I}_4$ and moving the the scalar mass m_w makes it possible to rewrite (4.51) more compactly as

$$[m_w(\boldsymbol{r}_2 + \boldsymbol{r}_3 + \boldsymbol{r}_4)]\times\boldsymbol{g} - \tilde{I}_w\dot{\boldsymbol{\omega}}_c - \boldsymbol{\omega}_c\times(\tilde{I}_w\boldsymbol{\omega}_c) \tag{4.52}$$

4.5.5 Final equation of motion

Restating the inertia components defined previously

$$\tilde{I}_1 = -m_cS(\boldsymbol{r}_1)^2 \quad \tilde{I}_2 = -m_wS(\boldsymbol{r}_2)^2 \quad \tilde{I}_3 = -m_wS(\boldsymbol{r}_3)^2 \quad \tilde{I}_4 = -m_wS(\boldsymbol{r}_4)^2$$

where $S(\boldsymbol{r})$ is the skew symmetric matrix replacing a cross product with some vector $\boldsymbol{v}$ according to $\boldsymbol{r}\times\boldsymbol{v} = S(\boldsymbol{r})\boldsymbol{v}$. Here $\boldsymbol{r}_k$ are the position vectors for the mass center

of each body $k = 1, 2, 3, 4$ where $k = 1$ corresponds to the cube casing and $k = 2-4$ to wheel 1-3 respectively. The position is taken relative to the balancing corner c, defined in the body frame (X_B, Y_B, Z_B) and thus constant.
Continuing

$$\Theta_{w0} = \begin{bmatrix} 2I_{w0} & 0 & 0 \\ 0 & 2I_{w0} & 0 \\ 0 & 0 & 2I_{w0} \end{bmatrix} \qquad \Theta_{wz} = \begin{bmatrix} I_{wz} & 0 & 0 \\ 0 & I_{wz} & 0 \\ 0 & 0 & I_{wz} \end{bmatrix}$$

where I_{wz} is the scalar moment of inertia around each wheels shaft rotation axis and I_{w0} the scalar moment of inertia for each wheels other axises. This if further described in Section 4.3 Then by defining

$$\begin{aligned} I_{3D} &\triangleq \tilde{I}_1 + \tilde{I}_2 + \tilde{I}_3 + \tilde{I}_4 + I_1 + \Theta_{w0} \\ \tilde{I}_{3D} &\triangleq \tilde{I}_1 + \tilde{I}_2 + \tilde{I}_3 + \tilde{I}_4 \\ \tilde{\boldsymbol{r}}_w &\triangleq r_2 + r_3 + r_4 \end{aligned} \tag{4.53}$$

The final equations of motion can be written as

$$I_{3D}\dot{\boldsymbol{\omega}}_c + \boldsymbol{\omega}_c \times (\tilde{I}_{3D}\boldsymbol{\omega}_c) - \boldsymbol{\omega}_c \times \boldsymbol{\omega}_w \left[I_{wz} - I_{w0}\right] = \boldsymbol{T} + (m_c \boldsymbol{r}_1 + m_w \tilde{\boldsymbol{r}}_w) \times \boldsymbol{g} \tag{4.54}$$

$$\Theta_{wz}\dot{\boldsymbol{\omega}}_w = \boldsymbol{T} - b\boldsymbol{\omega}_w \tag{4.55}$$

4.5.6 Verification and analysis

The equations of motion on the form described in (4.37) and (4.38) are easily obtained using symbolic handling software such as `Mathematica` once all involved expressions for positions etcetera are obtained. Equation (4.54) was obtained manually but verified to comply with (4.37) and (4.38) using `Mathematica`
Due to the assumed strong symmetry of the cube the auxiliary vector $\tilde{\boldsymbol{r}}_w$, being defined by (4.1) and (4.53), is parallel to $\boldsymbol{r}_{cb}$. This implies that the orientation for equilibrium corner balancing where $(m_c \boldsymbol{r}_1 + m_w \tilde{\boldsymbol{r}}_w) \times \boldsymbol{g} = 0$ is not changed by the wheels.
Comparing the obtained equations of motions (4.54) to a report from ETH [44] presenting a model for Cubli, it can be seen that the expressions differ for the inertia tensors. This is due to it being assumed implicitly in [44] that the reactions wheels are constrained to rotate together with the cube casing around all three Cartesian axis. That is not observed for Cubex, which is this version of Cubli. If for instance the cube where to be rotated around its X_B axis it was observed that wheel 1 that is nominally rotating around X_B kept more or less still.
The term $\boldsymbol{\omega}_c \times \boldsymbol{\omega}_w \left[I_{wz} - I_{w0}\right]$ described gyroscopics effect arising due to the wheels rotating around their respective rotation axises. Note how this effect would disappear if the wheels had the same principal inertia component around their rotation axis as the other two principal axises.

4.5.7 Substitution of orientation relations

While the final equation of motion describe the dynamics of the system, they cannot be solved for nor integrated for orientation. This can be solved by substituting in relations between orientation and angular velocity. One such is the relation between angular velocity and (Z, X', Z'') Euler angles described in Section 4.2.1

$$\boldsymbol{\omega}_c = \underbrace{\begin{bmatrix} \sin\psi\sin\theta & \cos\psi & 0 \\ \cos\psi\sin\theta & -\sin\psi & 0 \\ \cos\theta & 0 & 1 \end{bmatrix}}_{\triangleq Q} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = Q\dot{\boldsymbol{E}} \tag{4.56}$$

Differentiating (4.56) with respect to time yields

$$\dot{\boldsymbol{\omega}}_c = \dot{Q}\dot{\boldsymbol{E}} + Q\ddot{\boldsymbol{E}} \tag{4.57}$$

Substituting (4.56) and (4.57) into (4.54) and solving for the highest derivative of the vector of Euler angles $\boldsymbol{E}$ yields

$$\ddot{\boldsymbol{E}} = (I_{3D}Q)^{-1}\left(\boldsymbol{T}_1 + \boldsymbol{T}_G + Q\dot{\boldsymbol{E}} \times \boldsymbol{\omega}_w\left[I_{wz} - I_{w0}\right] - (Q\dot{\boldsymbol{E}}) \times (\tilde{I}_{3D}Q\dot{\boldsymbol{E}}) - I_{3D}\dot{Q}\dot{\boldsymbol{E}}\right) \tag{4.58}$$

where it has been assumed that $I_{3D}Q$ is invertible. Regardless of the choice of Euler angles sequence, Q^{-1} always has a singular point, which is further elaborated in the Section 4.8 that is about Simulation. In this section it is also described how describing system dynamics using Euler angles, as here, can be circumvented in numerical simulations. Within the context of this thesis work (4.58) is mainly used for control systems synthesis after linearization, which is described in Section 6.1.2.

4.6 Motor model

The cube has three Maxon EC45 70W Brushless DC (BLDC) motors with Maxon ESCON 36/3 driver circuits. The driver has three operating modes available, current , speed or open loop control of the connected motor [45]. A BLDC motor is a three-phase motor fed by three alternating currents from the driver circuit. However, the torque output can be modeled as for a DC motor according to

$$T = k_T i \tag{4.59}$$

where k_T is the motors torque constant and i its equivalent DC current [46]. For the EC45 motor it is $k_t = 36.9$ mNm/A [47]. The electrical characteristics can be can also modeled similar to a DC motor according to

$$\frac{di(t)}{dt} = \frac{1}{L}V_{in}(t) - \frac{R}{L}i(t) - \frac{1}{L}E(t) \tag{4.60}$$

where L and R are the stators equivalent inductance and resistance respectively, $i(t)$ is the stator current and $E(t)$ is the electro motoric force (emf)[46]. The current $i(t)$ is governed by an internal PI (Proportional-Integrating) controller in the ESCON motor controller. This is tuned using a system identification procedure in a provided software called Escon studio. The insignal provided by the controller is therefore a current set point reference $i_{ref}(t)$.

The counter emf is a voltage induced due to the relative motion between the stator and the rotor. For this case where the stator is moving, it is found as

$$E(t) = \underbrace{(\omega_i(t) - \omega_j(t))}_{\text{Diff. stator and rotor}} k_\omega \tag{4.61}$$

where ω_i is the scalar rotation of the wheel and rotor in relation to a fix point in in reference frame A and B and ω_j the rotation of the associated stator. Due to the orthogonal configuration of the cube ω_j will be either of one the three components of the angular velocity of the casing $\boldsymbol{\omega}_c = (\omega_x, \omega_y, \omega_z)$ for the three wheels respectively, albeit with the signs corrected where applicable. This is of less concern for practical applications, as the motors factory mounted encoders give this relative velocity directly.

4.6.1 Simplifications

It is assumed that the electrical circuit dynamics described by(4.60) is so fast that in comparison to the mechanical system that it is possible to neglect. It is also assumed that the PI controller governing the current is so fast and accurate such that $i_{ref}(t) \approx i(t)$

Then current and the motor torque is then governed by the static relation

$$T_i = k_t i(t) = k_t \frac{V_{in}(t) - E(t)}{R} = \frac{V_{in}(t) - (\omega_i(t) - \omega_j(t))k_\omega}{R} \tag{4.62}$$

Equation (4.62) determines how large the current can be at a time instant t.

4.7 Two-dimensional modeling

A two-dimensional model was developed for when the cube is balancing on one of its edges, assuming that only one wheel is rotating.

The modeling is done with the basis of a global coordinate system as shown in Figure 4.4.

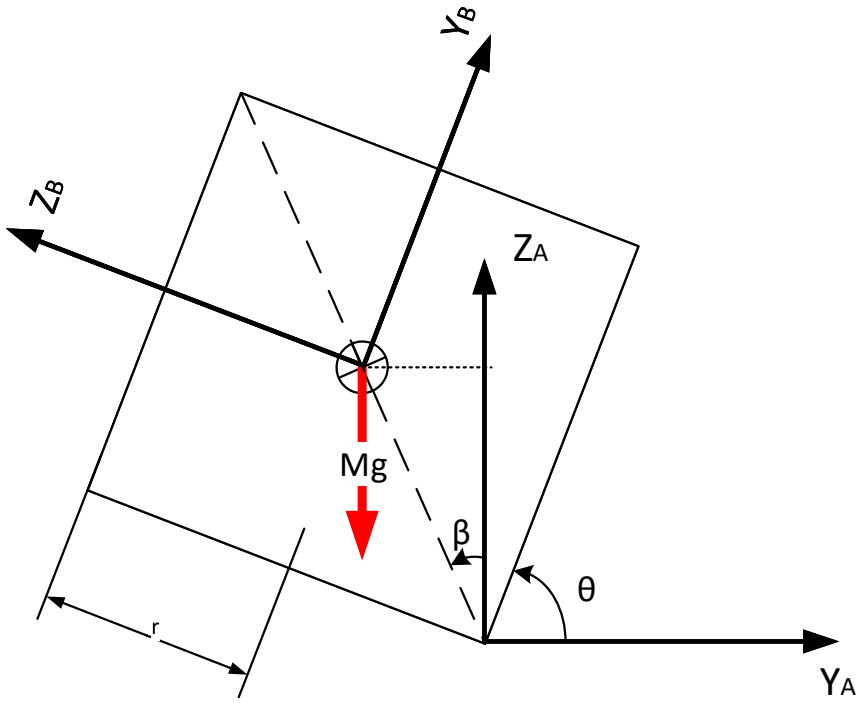


Figure 4.4: Figure used as support for the 2D modeling

Lagrangian mechanics is used to arrive at the equations of motion. As described in Section 3.5. The process is significantly simpler than for the 3D case. The system two degrees of freedom, θ describing how the cube casing is tilted and ρ_2 describing the angle of the wheel as defined for the 3D modeling. So,

$$q_1 \triangleq \theta \tag{4.63}$$

$$q_2 \triangleq \rho_2 \tag{4.64}$$

The cube has the side r meaning that the distance between the corner and the center of gravity is l. The speed of the center of gravity and the wheel is thus

$$v = \dot{\theta} l \tag{4.65}$$

As for the 3D modeling the wheel is assumed to rotate completely independent of the cube casing. The kinetic energy is then

$$T = \frac{1}{2} M v^2 + \frac{1}{2} \omega_c^T I_{c0} \omega_c + \frac{1}{2} \omega_1^T I_{wz} \omega_1 = \frac{1}{2} M l^2 \dot{q}_1^2 + \frac{1}{2} \left(I_{c0} \dot{q}_1^2 + I_{wz} \dot{q}_2^2 \right) \tag{4.66}$$

where M is the total mass of the cube and I_{c0} and I_{wz} inertia tensor components defined in Section 4.3 . It is assumed that center of the whole cube coincides with the geometrical center of the cube around which the wheel rotates. It has been used that both the angular velocity and inertia tensor are the same in both the body frame B as the reference frame A during 2D operation. Symmetry is assumed, such that the center of gravities of the wheel and the body both coincide with the geometrical center of the side of the cube in the (Y_B, Z_B) plane.
The conservative energy U is taken with the contact point as reference and is then

$$U = Mgl\cos(\beta) = Mg\cos(\theta - \pi/4)l \tag{4.67}$$

The auxilliary angle β has been introduced for simpler analysis and denotes how much the center of gravity is tilted in relation to the vertical reference axis X_A . The definition of θ stems from the choice of coordinate system for the 3D modeling, and is the same as θ in the Euler angle sequence chosen.
The partial derivative for all generalized coordinates of the Lagrangian $L = T - U$ are zero except for

$$\frac{\partial L}{\partial q_1} = Mgl\sin(\theta - \pi/4) = Mgl\sin(\beta) \tag{4.68}$$

which is the generalized conservative force representing the gravity induced torque around the balancing edge. Continuing to evaluate the Lagrangian to arrive at equations of motion

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\dot{q}_1}\right) - \frac{\partial L}{q_1} &= \underbrace{\left(Ml^2 + I_{c0}\right)}_{\triangleq I_{2D}}\ddot{q}_1 - Mgl\sin\beta = Q_1 \\
\frac{d}{dt}\left(\frac{\partial L}{\dot{q}_2}\right) - \frac{\partial L}{q_2} &= I_{wz}\ddot{q}_2 = Q_2
\end{aligned} \tag{4.69}$$

Here Q_1 and Q_2 are the generalized net non conservative force performing virtual work along the q_1 and q_2 coordinate respectively. Both the wheel and the cube casing are affected by the motor torque M, with opposite signs. However, the reaction wheels is defined to have a positive direction opposite to the positive direction of θ, inverting the signs again. The casing is assumed not to be affected by viscous friction and such. On the other hand, the wheel seemed so affected by this in practice that it shouldn't be neglected. A simple linear friction model is assumed so

$$\begin{aligned}
Q_1 &= T_1 \\
Q_2 &= T_1 - b\dot{q}_2
\end{aligned}$$

Substituting the generalized forces into (4.69) and solving for the highest derivatives yields the equations of motion

$$\begin{aligned} \ddot{q}_1 &= \frac{Mgl\sin\beta}{I_{2D}} + \frac{T_1}{I_{2D}} \\ \ddot{q}_2 &= \frac{T_1}{I_{wz}} + b\frac{\dot{q}_2}{I_{wz}} \end{aligned} \tag{4.70}$$

By inspection of (4.70) it can be seen that the cube casing isn't affected by the rotation of the reaction wheel nor the wheel inertia, only it's mass. This makes perfect sense as gyroscopic effects won't come into play in this two dimensional system. The mass of the wheel however affects the cube as the cube has to displace the mass of the wheel while moving.

4.8 Simulation

As was seen the final equations of motions (4.54) doesn't contain information about orientation in themselves, but just the angular velocity and acceleration. However, they also contain the transformation of the gravity vector to the body frame which requires knowing the orientation. Then the control loop needs will be based on feedback related to the orientation. As proposed in a paper from Stanford [48], rigid bodies can be simulated by using two blocks . One block contains the dynamics related to angular velocity and the second block contains kinematics relating angular velocity to the derivative of a orientation estimate [48]. An overview of this idea is shown in Figure 4.5

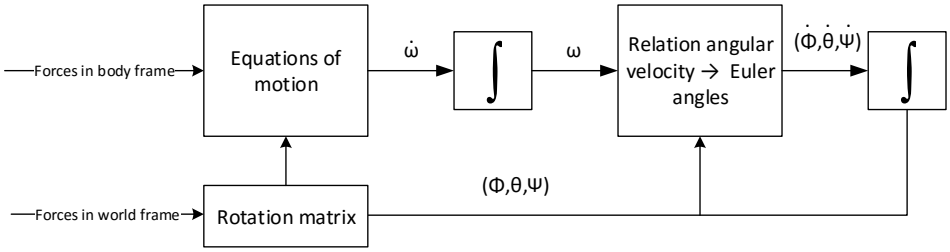


Figure 4.5: Overview of the simulation approach used. The highest derivatives are solved for and integrated numerically

First, the angular acceleration is obtained by using a model where this has been solved for. This is then integrated numerically and fed to a function relating angular velocity to the derivative of the position estimate. By then integrating this, a position estimate is obtain. While there are different ways of implementing this, we did it using a block diagram approach in Simulink structurally looking as the approach depicted in Figure 4.5.
The derivative of the position estimate can be solved for by solving for the Euler angles in (4.7) according to

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = Q^{-1B}\boldsymbol{\omega}_c = \begin{bmatrix} \frac{\sin(\psi)}{\sin(\theta)} & \frac{\cos(\psi)}{\sin(\theta)} & 0 \\ \cos(\psi) & -\sin(\psi) & 0 \\ -\frac{\cos(\theta)\sin(\psi)}{\sin(\theta)} & -\frac{\cos(\theta)\cos(\psi)}{\sin(\theta)} & 1 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4.71}$$

By inspection, Q^{-1} has a singularity for $\theta = 0$. In fact, all choices of Euler angles has an inverse a singular position [40]. Common ways to circumvent this problem seems to be using either so called *direction cosines* or *unit quaternions* [49, 50]. Another simpler way is to use a discrete condition forcing θ to be non zero while being close to zero. This has the drawback of causing inaccuracies while operating close to this point. [50]. However, this was the approach to avoid singularities during simulations in this projct as θ is seldom approaching zero during corner balancing.

5
Hardware and implementation

Some information regarding more practical details of this work.

5.1 Electrical system

The electrical system was modified within the context of this thesis work. The main circuit handling voltage conversion from a battery to the various units in the cube is described.

5.1.1 Schematics and description

All electronics is powered from a single six cell 800 mAh Lithium Polymer battery, having a nominal voltage of $22V$ [volts] designated as ***V1*** in the schematics in Figure5.1. There is a fuse connected directly in series mainly for short circuit protection. A switch is placed before all current consumers, so that no current should be drawn when the switch is off. It is clearly stated in documentation that the motor drivers will break if polarity is reversed [45], and these are quiet expensive. Therefore a polarity protection circuit consisting of the p-Mosfet ***Q1***, the 11 v zener diode ***D1*** and the resistor ***R1*** is employed. As opposed to using simple diode for this purpose, this protection circuitry allows for opposite currents once being activated by a correctly connected battery[51]. This is necessary as the motors drivers employ regenerative braking, meaning currents sometimes have to flow from the motor to the battery. Then the light emitting diode (LED) ***L1*** and ***R2*** form an indicator for

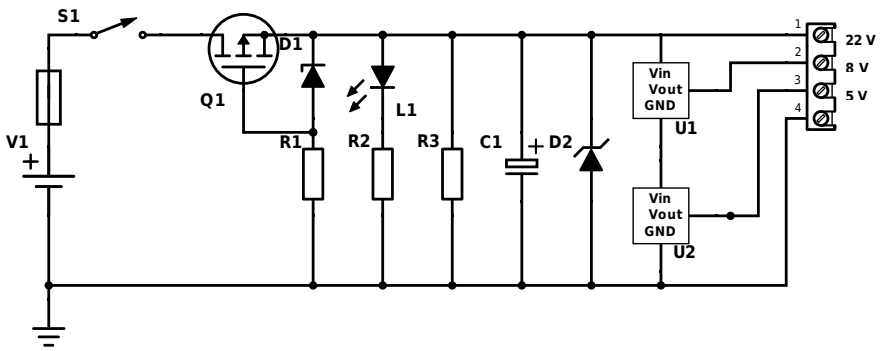


Figure 5.1: Description of the main parts of the cubes electrical systems

operation. The motor drivers are connected to the screw terminal marked $22V$. The capacitor ***C3*** is a fairly large electrolytic capacitor for smoothening current flow to the battery. This is drained to zero voltage after the switch is turned off by ***R3***. A charged capacitor is always a risk, and also the polarity protection circuit will not work if the capacitor remains charged. As the motor drivers are subjected to more than $38V$ they will break. Therefore a $30V$ transient voltage suppression diode ***D2*** is placed. Then two voltage step down regulators are places in series to give a $7V$ output to the servo motors actuating the brakes, and $5V$ for the CPU board.

5.1.2 Battery safety issues

The battery used is typically used for hobby remote controlled applications such as remote controlled quadcopters. Litihium batteries posses a potential fire hazard, which has been brought up in media recently when litihium batteries caught fire and even exploded in airplanes [52]. Both overcharging and over discharging a lithium battery can cause thermal runaways, that can lead to fire and explosions [53].
Charging is done using a separate charger before use of the cube. Over voltage is therefore assumed not to be an issue during normal operation, as no energy added to the system after charging. On average the current draw seemed to stabilize around a net current draw $0.2A - 0.8A$ [amperes] which is well below the maximum continuous discharge rate for the battery being $60C$ implying a discharge rate of $48A$. However the motor drivers have regenerative braking implemented [45], driving a current to the batteries during braking of a load. To mitigate the current flowing to the batteries the rather large capacitor ***C1*** was installed. Regenerative charging is yet not thoroughly investigated in this context. However, according to a study conducted at *Institute for Electrical Energy Storage Technology, Technische Universität München*[54] it is not short time recharging with high current rates leading to degradation of Lithium batteries, but rather long lasting charging periods. Having observed the net current draw of $0.2A - 0.8A$ it is assumed that regenerative current are not much larger than the generally recommended charge rate of $1C$ implying $0.8A$ regenerative current. The issue regarding under voltage is curbed using an alarm system having both visual and audible notifications. The cube is assumed to always be supervised during operation and that the battery is disconnected afterwards.

5.2 Embedded software development

The software for Cubex is implemented trough `Matlab` and `Simulink` using the `Waijung` Blockset. The main controller on Cubex is a STM32F4 Discovery kit that uses a STM32F407 microprocessor with a 32-bit ARM Cortex-M4F core. The STM32F4 Discovery kit has a 1-MB Flash memory and 192-KB RAM.
`Waijung` is an addon to `Simulink` for generating C-code that is then compiled to assembler code for the STM32F4 processor . `Waijung` is designed specifically to support the STM32F4 family. For more information regarding `Waijung`, see [55]
Using `Simulink` with the addon `Waijung` enables for some graphical drag-and-drop programming, which we believe can make software development and debugging faster

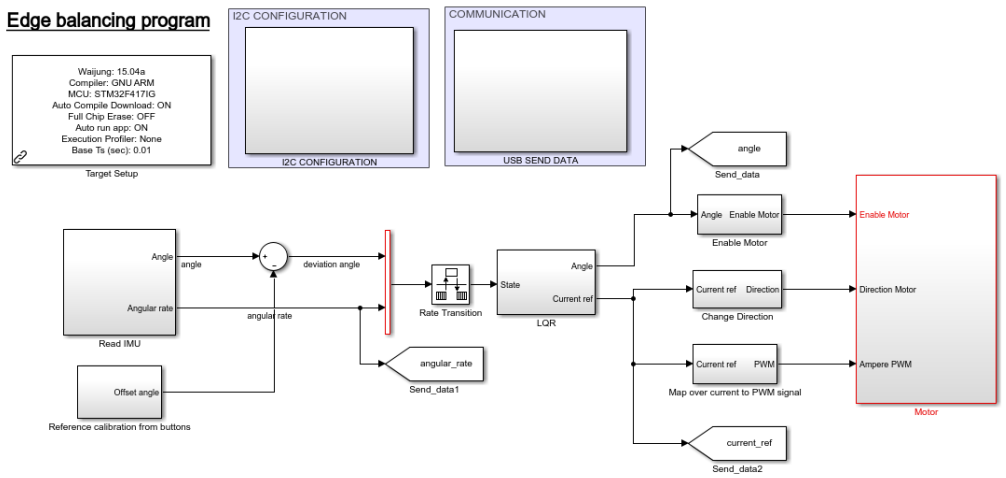


Figure 5.2: An example of a program using `Waijung` Blockset with `Matlab` and `Simulink`.

and easier. Many individual blocks were written using "conventional" code, but then integrated to the rest of the software graphically. An example of a program developed in `Waijung` used on Cubex shown in Figure 5.2. As `Simulink` was used for simulation, a lot of software developed for simulations could be used to generate code for the actual Cubex system. This facilitated software development substantially and allowed for faster software iterations then when for instance using `C/C++` directly for creating the software, in our experience and opinion.

6
Control strategies

While boasting with expensive and advanced hardware, the cube is essentially a mere decoration without having some sort of controller implemented. While creativity probably is what sets the furthest limit of what kind of control strategies and feats that could be implemented on the cube hardware, the project time scope left us with two controllers that balance the cube on either an edge or on the corner. As was seen in Section 4 the 2D modeling case is significantly simpler then the 3D and the same goes for the controller synthesis.

6.0.1 Control system overview

How is Cubex controlled then? The cube has three electric motors connected to reaction wheels, as is seen illustratively Figure 4.1. When the motors are actuated with a proper electrical signal, they produce a torque that act on the cube body and reaction wheels at the same time. The torques have equal magnitude but opposite directions according to Newtons third law. When the cube body is accelerated in one direction by a motor, the associated reaction wheel is then accelerated in the opposite direction. By controlling these torques properly, the cube can be brought to balance. Glancing at the generic overview in Figure 6.1, it is seen that the controller cannot directly control torque. As is detailed in Section 4.6 the motor controllers are configured to receive a reference signal i_{ref} for motor current. During the control system synthesis it was assumed that the motor controller and electrical circuit were ideal and without any dynamics so the torque for motor $u = 1, 2, 3$ would be $T_u = k_t i_{ref,u}$ where k_t is the motor torque constant found in the motor data sheet [47].
The question that follows is then which motor to actuate and when? In our opinion and experience, many simple SISO systems might work well with more or less experimentally tuned PID controllers. Sometimes experimental tuning might be better use of the time than trying some model based control systems approach. Cubli and Cubex on the other hand are systems we cannot imagine would work without a model based approach, or at least some sophisticated learning algorithm. Therefore all control design was done model based with using the of equations of motions (4.54) and (4.58) . The control input is defined as

$$\boldsymbol{T}_m \triangleq T_1 = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = k_t \begin{bmatrix} i_{ref,1} \\ i_{ref,2} \\ i_{ref,3} \end{bmatrix} \tag{6.1}$$

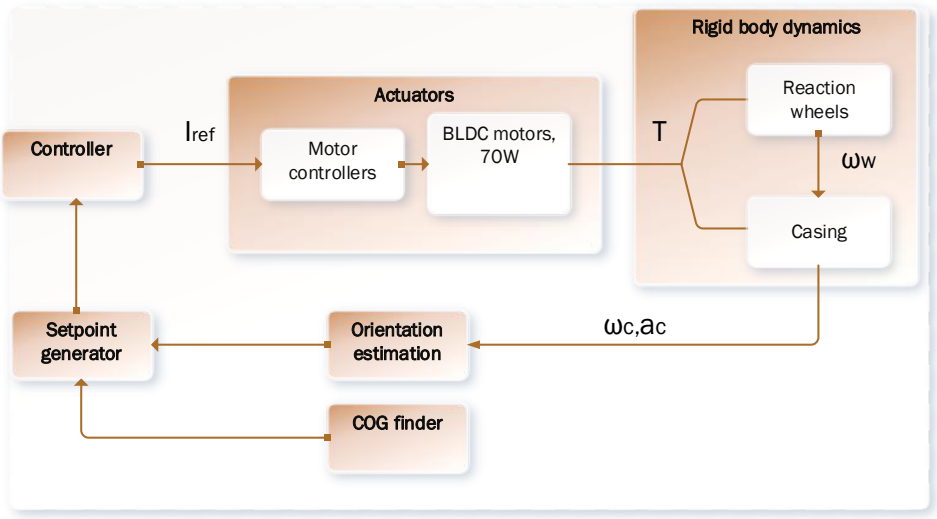


Figure 6.1: A generic overview over how the control systems for balancing were implemented

That is, the current reference signal for each motor times the motor constant, again assuming ideal motor controllers. Note that $\boldsymbol{T}_m$ is a proper three dimensional geometric vector. This is due to the orthogonal configuration of the motors and the choice of body coordinate system shown in Figure 4.1.
As seen in Figure 6.1, an orientation estimate is fed to the controller. The orientation is not measured directly but estimated using Inertial Measurement Units (IMU's) having accelerometers measuring $\boldsymbol{a}_c$and gyroscopes measuring $\boldsymbol{\omega}_c$. This is a large topic in itself and further described in Section 8. Also, during the modeling and control system synthesis it was assumed that the center of gravity perfectly aligns with the geometrical center of the cube. This is generally not the case in practice. Therefore, various COG(Center Of Gravity)-finders were tried out. These have the goal to generate a proper reference signal for an orientation where the cube can balance without applying any motor torque. The cube could hardly not balance without such an algorithm in practice. This is further elaborated in Section 7

6.1 Linearized discrete time state space

As stated in Section 3.7, describing a non-linear system using a linearized state space is a common strategy for control systems synthesis and analysis.

6.1.1 Two dimensional case

The systems only non-linear equations can be seen in equation (4.70). As the system is assumed to be operating close to $\beta = \theta - \pi/4 \approx 0$ it follows that

$$\sin(\beta) \approx \beta \tag{6.2}$$

Substituting 6.2 into (4.70) and formulating on state space form yields

$$\begin{bmatrix}\omega_x \\ \dot{\omega}_x\end{bmatrix} = \begin{bmatrix}0 & 1 \\ \frac{Mgl}{I_{2D}} & 0\end{bmatrix}\begin{bmatrix}\beta \\ \omega_x\end{bmatrix} + \begin{bmatrix}0 \\ I_{2D}\end{bmatrix} T_1 \tag{6.3}$$

Here it as been utilized that the derivatives $\dot{\beta} = (\theta - \dot{\pi}/4) = \dot{\theta}$ are equal. Note that the angular velocity of the wheel is ignored in this model. When the offset angle goes to zero, $\beta \to 0$, so will torque insignal T_1 meaning that the angular velocity of the wheel will decay to zero according to (4.70). This state is thus considered redundant during 2D operation.

6.1.2 Three dimensional case

As usually 3D-modeling is significantly more complicated than 2D-modeling. As we are interested in corner balancing which represents a specific orientation system dynamics described using Euler angles (4.58) is restated and expanded

$$\ddot{E} = (I_{3D}Q)^{-1}\left(T_1 + T_G + Q\dot{\boldsymbol{E}} \times \boldsymbol{\omega}_w\left[I_{wz} - I_{w0}\right] - (Q\dot{\boldsymbol{E}}) \times (\tilde{I}_{3D}Q\dot{\boldsymbol{E}}) - I_{3D}\dot{Q}\dot{\boldsymbol{E}}\right) =$$

$$\underbrace{(I_{3D}Q)^{-1}\left(T_G + Q\dot{\boldsymbol{E}} \times \boldsymbol{\omega}_w\left[I_{wz} - I_{w0}\right] - (Q\dot{\boldsymbol{E}}) \times (\tilde{I}_{3D}Q\dot{\boldsymbol{E}}) - I_{3D}\dot{Q}\dot{\boldsymbol{E}}\right)}_{\triangleq \boldsymbol{f}_1(\boldsymbol{x})} + \underbrace{(I_{3D}Q)^{-1}}_{\triangleq \boldsymbol{f}_2(\boldsymbol{x})} T_1 \tag{6.4}$$

where $\boldsymbol{x} \triangleq (E; \dot{E}; \boldsymbol{\omega}_w) = (x_1 \dots x_9)^T$. It is assumed that all involved matrix inversions are valid. During corner balancing the cube is assumed to be balancing close to a vertical position at equilibrium where the angular velocities, angular acceleration and wheel rotational speed are zero. As no net gravity torque is exerted around the contact point here no insignal is needed so the input torque is zero, $\boldsymbol{U}_0 \triangleq \boldsymbol{T}_0 = \boldsymbol{0}$. For the current definition of body frame given in Section 4.1 two rotations $\theta_0 = \arctan(\sqrt{2}) \approx 55\,\text{deg}$ and $\psi_0 = \pi/4 = 45\,\text{deg}$ are needed to bring the cube from resting flat on a surface to balance on a corner. This is elaborated in Section 4.2.2.
Defining $\boldsymbol{f}(\boldsymbol{x}) = \ddot{\boldsymbol{E}}$ and a linearization point $\boldsymbol{x}_0 = (\boldsymbol{E}_0, \dot{\boldsymbol{E}}_0, \boldsymbol{\omega}_{w0}) = (\boldsymbol{E}_0, \boldsymbol{0}, \boldsymbol{0})^T$ where $\boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{0}$ the general formula for linearization (3.46) can be applied. When evaluating the partial derivative all cross products involving $\dot{\boldsymbol{E}}$ or $\boldsymbol{\omega}_w$ will vanish due to the the choice of linearization point.
Left of $\boldsymbol{f}_1$ are then

$$\frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\boldsymbol{x}}\Big|_{x_0} = \frac{\partial}{\partial \boldsymbol{x}}\Big|_{x_0}\left[(I_{3D}Q)^{-1}T_G - (I_{3D}Q)^{-1}I_{3D}\dot{Q}\dot{\boldsymbol{E}}\right] \tag{6.5}$$

By splitting up the derivative

$$\frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{x_0} = \left[\frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{E}} \quad = \frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \dot{\boldsymbol{E}}} \quad \frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{\omega}_w}\right]\Big|_{x_0} \tag{6.6}$$

$$= \left[\frac{\partial \ (I_{3D}Q)^{-1}T_G}{\partial \boldsymbol{E}}\Big|_{x_0} \quad \frac{\partial(-I_{3D}Q)^{-1}I_{3D}\dot{Q}\dot{\boldsymbol{E}}}{\partial \dot{\boldsymbol{E}}}\Big|_{x_0} \quad 0\right]\Big|_{x_0} \tag{6.7}$$

it is a bit easier to see how many terms directly disappear due to the choice of linearization point and independence of differentiation variables.
Will $\dot{Q}\dot{\boldsymbol{E}}$ evaluate to zero after after being partially differentiated too? First lets note that $-(I_{3D}Q)^{-1}I_{3D}$ is constant in the context and can be lifted out of the analysis.
By describing each element of Q on row m and column n as a scalar function $f_{m,n}(\boldsymbol{E})$ the time derivative of $\dot{Q}$ for each element can be rewritten using total differentials as , by first clearifying that $(x_1, x_2, x_3)^T = (\phi, \theta, \psi)^T$,

$$\dot{f}_{m,n}(\boldsymbol{E}) = \frac{\partial f_{m,n}}{\partial x_1}\dot{x_1} + \frac{\partial f_{m,n}}{\partial x_2}\dot{x_2} + \frac{\partial f_{m,n}}{\partial x_3}\dot{x_3} + \cancelto{0}{\frac{\partial f_{m,n}}{\partial t}} \tag{6.8}$$

The last term is zero due to Q not depending explicitly on time. Then the analyzed multiplication can be evaluated as

$$\dot{Q}\dot{\boldsymbol{E}} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \begin{bmatrix} \dot{f}_{1,1}\dot{x}_1 + \dot{f}_{1,2}\dot{x}_2 + \dot{f}_{1,3}\dot{x}_3 \\ \vdots \\ \vdots \end{bmatrix} \tag{6.9}$$

Investigating the the partial derivative of the scalar element $Q_1$with respect to just $\dot{x}_1$

$$\frac{\partial}{\dot{x}_1}Q_1 = \frac{\partial \dot{f}_{1,1}}{\partial \dot{x}_1}\dot{x}_1 + \dot{f}_{1,1} + \frac{\partial \dot{f}_{1,2}}{\partial \dot{x}_1}\dot{x}_2 + \frac{\partial \dot{f}_{1,3}}{\partial \dot{x}_1}\dot{x}_3$$

where the product rule has been applied. Due to the choice of linearization point at $\dot{x}_1 = \dot{x}_2 = \dot{x}_3 = 0$ all partial derivatives of Q_1 with respect to any variable of $\dot{\boldsymbol{E}}$ will evaluate to zero. As Q_2 and Q_3 have identical structure the will vanish to when partially differentiated and evaluated at $\dot{\boldsymbol{E}} = \boldsymbol{0}$. This should hold true for Q being formed by any Euler angle sequence.
As both I_{3D} and Q are quadratic matrices assumed to be invertible it follows that $(I_{3D}Q)^{-1} = Q^{-1}I_{3D}^{-1}$. We are then left with

$$\begin{aligned} \ddot{E} &\approx \frac{\partial \boldsymbol{f}_1(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{x_0}\Delta\boldsymbol{x} + f_2(\boldsymbol{x}_0)T = \\ &= \left[\frac{\partial \ Q^{-1}I_{3D}^{-1}T_G}{\partial \boldsymbol{E}}\Big|_{x_0} \quad 0 \quad 0\right]\Delta\boldsymbol{x} + Q^{-1}\Big|_{x_0} I_{3D}^{-1}T_1 \end{aligned} \tag{6.10}$$

Due to the choice of linearization point the angular velocity of the wheels $\boldsymbol{\omega}_w$ are decoupled from the Euler angles $\boldsymbol{E}$ and their derivatives $\dot{\boldsymbol{E}}$ in the state vector $\boldsymbol{x}$. The goal of the controller in this context is to achieve an orientation were no input torque $\boldsymbol{T}$ is needed to balance the system and the wheels are still. By finding the right orientation the reaction wheels will not be be affected by any motor torque and will eventually stop spinning due to damping. For the current assumptions and choices, the wheel velocity $\boldsymbol{\omega}_w$ can thus be considered a redundant state. By redefining a reduced state vector $\boldsymbol{x} \triangleq (x_1 \dots x_6)^T = (\boldsymbol{E}; \dot{\boldsymbol{E}})$ the state space matrices and control signals

$$\begin{aligned} A = &\begin{bmatrix} 0^{3\times3} & I^{3\times3} \\ \frac{\partial\ Q^{-1} I_{3D}^{-1} T_G}{\partial \boldsymbol{E}}\Big|_{x_0} & 0^{3\times3} \end{bmatrix} \qquad B = \begin{bmatrix} 0^{3\times3} \\ Q^{-1} I_{3D}^{-1} \Big|_{x_0} \end{bmatrix} \\ \boldsymbol{u} = &T_1 \end{aligned} \tag{6.11}$$

can be identified. The matrices C are not D considered here as all states are assumed to be reconstructed perfectly.

6.1.3 Spinning top approximation

Due to the linearization point and the current model, the wheels angular velocity is not included in the state space. This yields a model that basically describes a linearized spinning top. The model of the spinning top corresponds directly to Eulers Laws of rotation which can be found directly in most textbooks related to rigid body dynamics. It was tried synthesing controller using the spinning top model as described by (4.24) and (4.25) in Section 4.5. The inertia tensor I was set to the diagonal inertia tensor for the casing I_c. An LQR controller was synthesized from this model and was able to succesfully balance the real cube on the corner.

6.1.4 Discretization

Having a linearized continuous time state space formulated, discretization was conveniently using the `c2d()` command built into `Matlab`. The `c2d()` command can discretize a continuous time state space using several different techniques. The default option using *zero order hold interpolation* was used.

6.2 Linear quadratic control

Recalling the Linear Quadratic Regulator (LQR) in Section 3.9 were an overview of LQR is presented. For LQR a weighting matrices Q and R for the states and input signals need to be selected. The weighting matrices used were diagonal , i.e.

$$Q = \begin{bmatrix} q_1 & & \\ & \ddots & \\ & & q_n \end{bmatrix} \qquad R = \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_m \end{bmatrix}$$

where n is the number of states and m is the number of input signals. Here Q set the penalty for deviations from the equilibrium balancing point and R penalty on the currents to the motor. The weighting matrices are usually considered to be tuning parameters and were set experimentally within this context.

6.3 Model predictive control

A dual approximate gradient version of MPC as described in Algorithm1 was used with the linearized models for edge and corner balancing as described in Section 6.1.1 and 6.1.2. What follows is a description of how the parameters described in Section 3.10.2 were chosen.

6.3.1 Weight matrices

The weighting matrices were set to identical as for LQR. A general description of them can be found under 6.2.

6.3.2 Inequality constraints

Recalling that MPC can include inequality contains for the states and control signals. This was formulated as an an upper and lower bond according to

$$\underline{\boldsymbol{z}} = \begin{bmatrix} \underline{\boldsymbol{u}}_0 \\ \underline{\boldsymbol{x}}_1 \\ \vdots \\ \underline{\boldsymbol{u}}_{N-1} \\ \underline{\boldsymbol{x}}_N \end{bmatrix} \qquad \overline{\boldsymbol{z}} = \begin{bmatrix} \overline{\boldsymbol{u}}_0 \\ \overline{\boldsymbol{x}}_1 \\ \vdots \\ \overline{\boldsymbol{u}}_{N-1} \\ \overline{\boldsymbol{x}}_N \end{bmatrix} \tag{6.12}$$

where $\underline{\boldsymbol{z}}$is the lower bond and $\overline{\boldsymbol{z}}$ the upper. Generally, this allows for different constraints at different time indexes over the prediction horizon. However here the same constraints were for the whole prediction horizon. The motor controllers units are internally limited to $\pm 4A$. Depending on if the cube is balancing on an edge. then using one motor, or a corner using three motors, the insignal constraints are

$$\underline{u}_{k,edge} = -4 \qquad \overline{u}_{k,edge} = 4 \qquad \underline{\boldsymbol{u}}_{k,corner} = \begin{bmatrix} -4 \\ -4 \\ -4 \end{bmatrix} \qquad \overline{\boldsymbol{u}}_{k,corner} = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix}$$

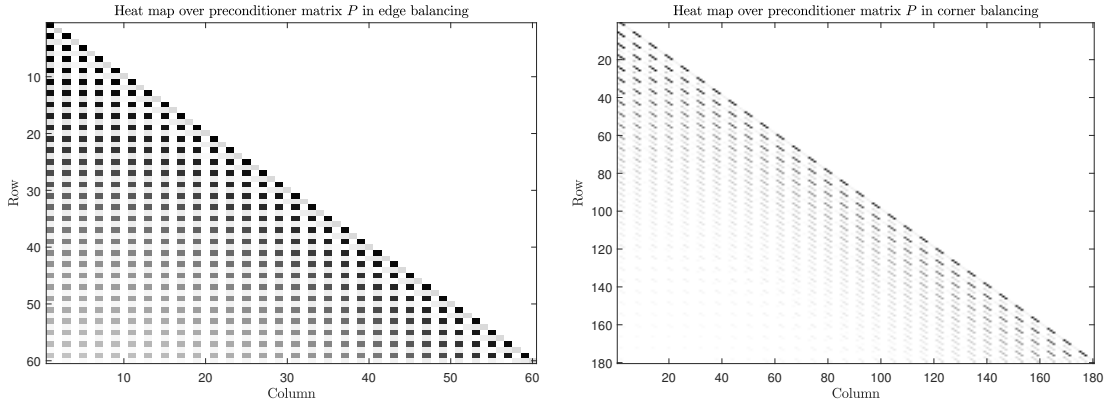
The states were not constrained using inequality constraints. The bounds on the input signal together with the cost function should implicitly assure that the states remain bounded.

6.3.3 Prediction horizon

As mentioned in Section 3.10.1.1 there is no general method for setting the prediction horizon N and this was therefore set experimentally. It was found that $N = 30$ was adequate when operating the controller at the sampling interval $T_s = 16$ ms for both edge and corner balancing. This means that the prediction horizon is 480ms long.

6.3.4 Preconditioner

An important aspect for the MPC implementation within this context is the pre-conditioner matrix as described in Section 3.10.2.Recalling from Section 3.10.2 an idea presented in [3] is basically neglecting smaller elements of P by banding as is illustrated in Figure 3.7. In Figure 6.2a and 6.2b heat maps of the pre-conditioner for two examples of corner and edge balancing are shown.



(a) Two dimensional system **(b)** Three dimensional system

Figure 6.2: Heat map over preconditioner matrix P for both the two dimensional and three dimensional systems

The heat maps basically shows the magnitude of each element were a more solid color means a larger element. A condition for banding is that there are sufficiently many small elements towards the lower left of P . Therefore P in Figure 6.2b should be more well posed for an approximation than P in Figure 6.2a.

6.3.4.1 Lipschitz constant

Inspecting Algorithm 1 it is seen that the scalar Lipschitz constant L is involved. It is not known by us how to find the Lipschitz constant for this problem and it is an issue elaborated in Section 3.10.2.2. Therefore it was set experimentally to somewhere in the range $1.00 - 3.65$.

6.4 State estimation

The Madgwick algorithm, briefly described in Section 8.1, was used for estimating the orientation of Cubex. Then the relation in Section 4.2.1 was used to find the Euler angle derivatives using measurements of angular velocity and the orientation from the Madgwick algorithm as inputs. The angular velocity has first an initial bias estimate $\hat{\boldsymbol{b}}_0$ removed, which is elaborated in Section 8.2.2. Then it was filtered using a first order digital low pass having a time constant of around $0.1s$ before being used to estimate the Euler angle derivatives.
Sometimes the Euler angle ψ describing the first rotation around the Z axis was influenced by slow angular drift due to gyroscopic bias. To make the controller

ignore this effect, this estimate was filtered using a first order digital high pass filter that was experimentally tuned.

7
Offset estimation

In the modeling process it was assumed that the center of mass, as well as the orientation estimate, coincide with the geometrical center of the cube. This is very unlikely to happen in reality, and was a main cause of failure during balancing action. A slight offset would require the motor to deliver a constant torque. This would accelerate the reaction wheels to a velocity where the induced counter EMF (electro motor force, see Section 4.6) in the the motors would start decreasing the maximum torque the motors could output. At this point the cube would start tipping over. The issue has been addressed by the original creators of Cubli too. In the first paper regarding Cubli published 2012 the solution of this issue is integrated into the control law [17]. For the three dimensional case an approach described as heuristic is used. The nominal vector $\boldsymbol{r}_{cb}$ describing the location of the center of gravity is updated in steps by using projections on the vector $\boldsymbol{g}$ describing gravity in the body frame[36]. A few other approaches to address the problem are outlined here.

7.1 Two dimensional case

In Figure 7.1 there is line drawn through the contact point between the underlying surface and the center of gravity.

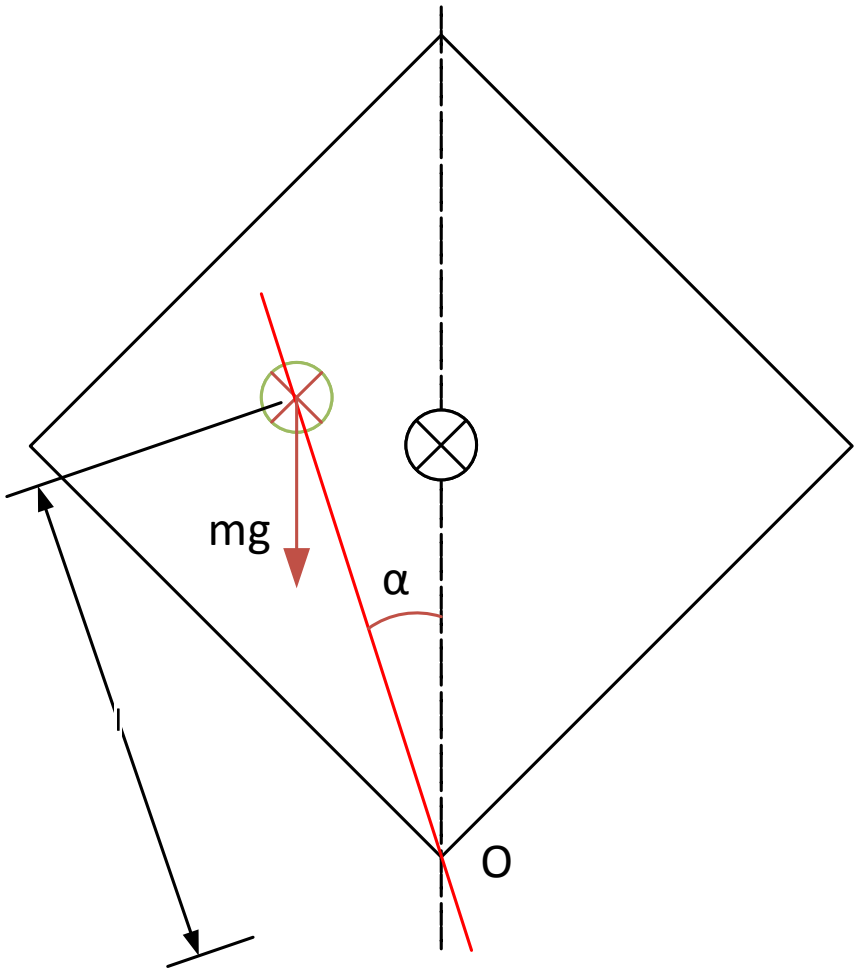


Figure 7.1: Illustrating in red the center of gravity having an offset angle α from the symmetry line of the cube

If the center of gravity coincides with the dashed symmetry line, the cube can balance without any external torque input, as no net torque will be exerted on the cube according to (4.70). However, should center of gravity in reality be slightly offset an angle α to the symmetry line a net disturbance torque $mglsin(\alpha)$ will be exerted around the contact point O. If the control system is to be stable around this point it needs to produce some constant torque $M_c = -mglsin(\alpha)$. This requires that a current is flowing constantly and will affect the wheel according to (4.59). Oscillations of the tilt angle were also observed in practice. This effect was re-created in simulations by introducing first order sensor lag on the tilt angle. This disturbance can be mitigated by estimating α and setting a reference angle $-\alpha$. The cube will be set to balance around a point where no net gravity torque is acting on the cube.
The exact impact of the offset α will vary in practice. However, it's likely that the

average torque $\bar{M}$ the motor will exert on the cube will be affected such that

$$\alpha > 0 \implies \bar{M} < 0$$

It is also reasonable to assume that α will have a negative correlation to $\bar{M}$. Based on this, a gradient descent based algorithm is proposed where an estimate of the offset $\hat{\alpha}$ is updated

$$\hat{\alpha}_{k+1} = \hat{\alpha}_k - C\bar{M} \tag{7.1}$$

where C is a tuning coefficient and $\bar{M}$ roughly would correspond to the gradient.

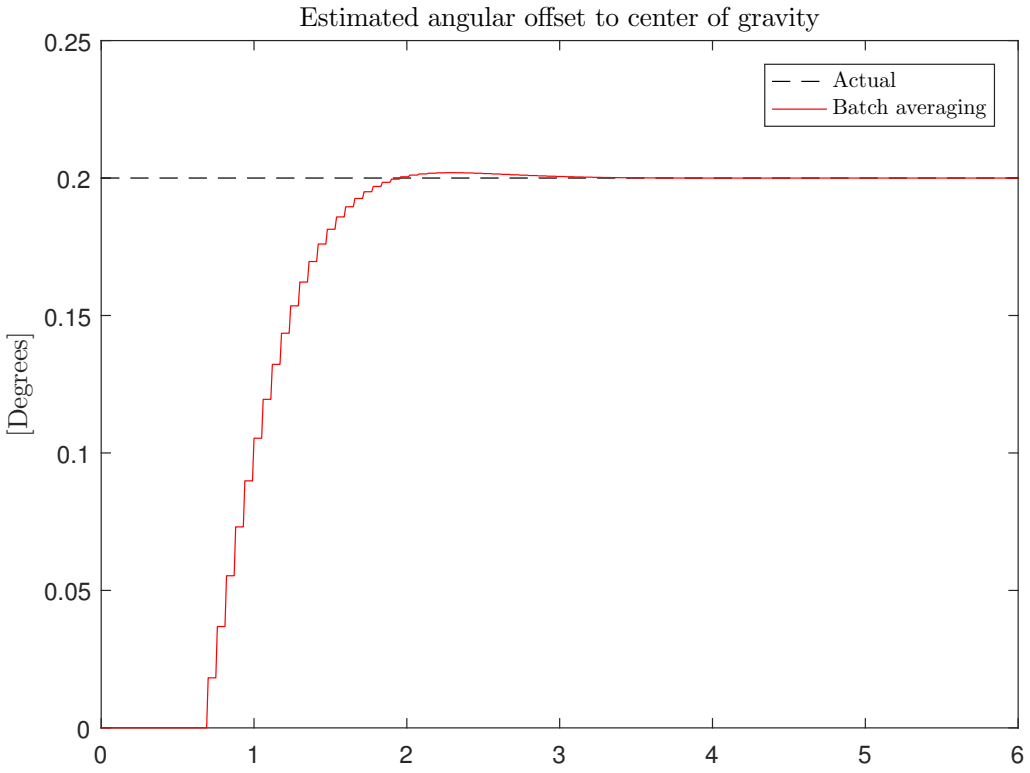


Figure 7.2: Simulation results for various methods of finding the angular offset to the center of gravity during edge balancing in 2D

$$\begin{aligned} \Delta\alpha &= k_{min}C \\ k_{min} &= T_{min}/T_s \\ \implies \frac{\Delta\alpha}{s} &= Nk_{min}C = \frac{1}{T_{min}}\frac{T_{min}}{T_s}C \end{aligned} \tag{7.2}$$

The rate of increase of the offset estimation can thus be made normalized so that it will be independent of the sampling rate T_s by letting

$$C_n = CT_s \tag{7.3}$$

yielding the update equation. In Figure **??** a simulation result can be seen. The algorithm proved to work in practice too.

7.2 Three dimensional case

As usually, things get much more complicated in three dimensions. The problem can be formulated by adding an offset vector mass located at $\Delta \boldsymbol{r}$ from the the nominal vector from the cube corner to the center of gravity $\boldsymbol{r}_{cb}$ as illustrated geometrically in Figure 7.3.

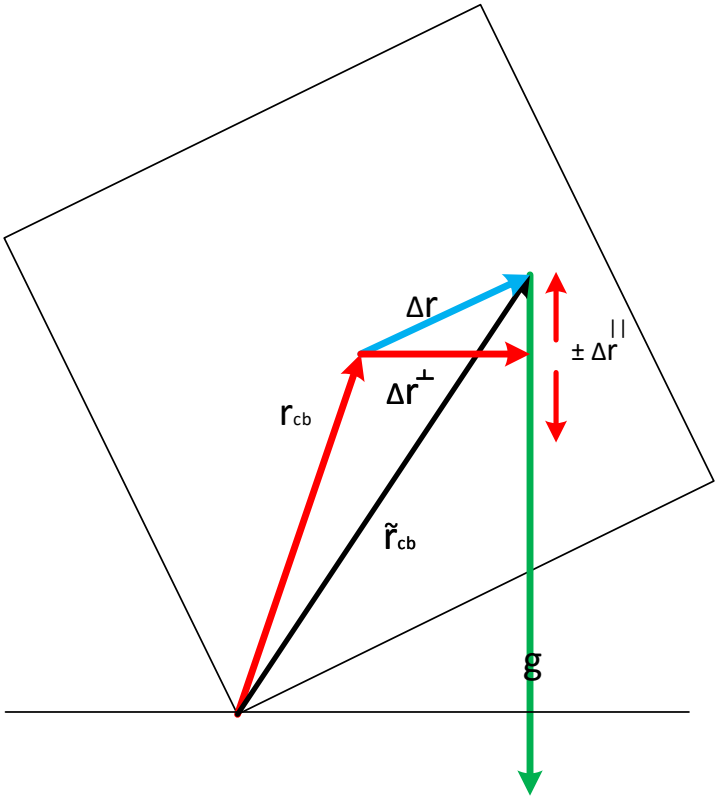


Figure 7.3: Illustration of the nominal vector from the balancing corner to the center of gravity $\boldsymbol{r}_{cb}$ being offset by a vector $\Delta \boldsymbol{r}$. Note that this is a simplified 2D view of a 3D problem

The location of this offset can then be formulated algebraically as

$$\tilde{\boldsymbol{r}}_{cb} = \boldsymbol{r}_{cb} + \Delta \boldsymbol{r} = \begin{bmatrix} r \\ r \\ r \end{bmatrix} + \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{7.4}$$

The cube is highly symmetrical so the auxiliary vector $\tilde{\boldsymbol{r}}_w$ being the sum of the position vectors for the wheels, as defined in Section 4.5.5, can be identified as $\tilde{\boldsymbol{r}}_w = 2\boldsymbol{r}_{cb}$ if making the simplified assumption that $\mu = 1$ in (4.1).
To have a more compact formulation (7.4) is assumed to offset the combined mass center for the cube. Inserting this into the equations of motion for the cube (4.54)

yields

$$I_{3D}\dot{\boldsymbol{\omega}}_c + \boldsymbol{\omega}_c \times (\tilde{I}_{3D}\boldsymbol{\omega}_c) - \boldsymbol{\omega}_c \times \boldsymbol{\omega}_w \left[I_{wz} - I_{w0}\right] = \boldsymbol{T} + m_t(\boldsymbol{r}_{cb} + \Delta\boldsymbol{r}) \times \boldsymbol{g} \tag{7.5}$$

where $m_t \triangleq m_c + 2m_w$. When the gravity vector $\boldsymbol{g}$ is parallel to the vector from the balancing corner to the the center or gravity $\boldsymbol{r}_{cb}$, or $\boldsymbol{r}_{cb} \parallel \boldsymbol{g}$, no gravity induced torque is exerted and the cube can remain at equilibrium. By finding $\Delta\boldsymbol{r}$ it is possible to find the orientation the cube should have in order to balance without any external torque applied. This will indirectly solve the issue.

7.2.1 Vector projection approach

It is prosed by the original creator or Cubli that the vector to the center of gravity should be updated

$$\hat{\boldsymbol{r}}_{cb,k+1} = \mu \frac{\hat{\boldsymbol{r}}_{cb,k} \cdot \boldsymbol{g}}{\|\boldsymbol{g}\|_2} \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2} + (1-\mu)\hat{\boldsymbol{r}}_{cb,k} \tag{7.6}$$

$$\hat{\boldsymbol{r}}_{cb,k+1} = \hat{\boldsymbol{r}}_{cb,k+1} \frac{\|\hat{\boldsymbol{r}}_{cb,k}\|_2}{\|\hat{\boldsymbol{r}}_{cb,k+1}\|_2} \tag{7.7}$$

where the tuning factor $\mu \ll 1$. The last step (7.7) is to ensure that $\hat{\boldsymbol{r}}_{cb,k+1}$ maintains its magnitude [36].
A visualization of an iteration of the algorithm is showed in 7.4 for a two dimensional case. First, a projection is done by the dot product in (7.6). Then $\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$ is identified as the unit vector $\boldsymbol{e}_g$ for the gravity vector. Then a vector parallel to the gravity vector is added to a vector that is parallel to the last estimate, but shorter.

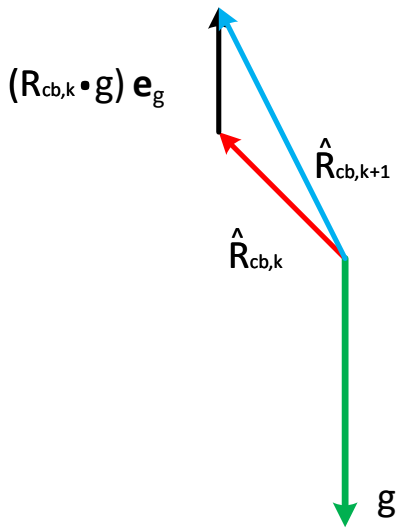


Figure 7.4: Illustration of one iteration of the vector projection approach for finding the center of gravity

As is visualized, $\hat{\boldsymbol{r}}_{cb,k+1}$ is slightly more parallel to the gravity vector then the previous update, as is desired. No name has been given to the algorithm so we choose to call it a vector projection approach.

7.2.2 Orthogonal approach

This algorithm proved to work both in simulation and practice, although it is unknown if it generally converges. The idea is to recover the perpendicular component of $\Delta\boldsymbol{r}$ in Figure 7.3 and use this to update the estimate of the nominal location of the center of gravity. This is done using linear algebra together with a model of the system.
Let us begin by examining some general cross product

$$\boldsymbol{X} \times \boldsymbol{Y} = \boldsymbol{Z}$$

First we remember that for some scalar p

$$(p\boldsymbol{A}) \times \boldsymbol{B} = \boldsymbol{A} \times (p\boldsymbol{B}) = p(\boldsymbol{A} \times \boldsymbol{B})$$

and that some vector always can be decomposed using its norm and unit vector according to $\boldsymbol{X} = \|\boldsymbol{X}\| \, e_{x'}$.
By decomposing $\boldsymbol{Y}$ into a one part thats parallel and one parts that perpendicular to $\boldsymbol{X}$ in the plane where the vectors lie follows by the definition of the cross product

$$\boldsymbol{X} \times (\boldsymbol{Y}^{\perp} + \boldsymbol{Y}^{\parallel}) = \boldsymbol{X} \times \boldsymbol{Y}^{\perp} + \cancelto{0}{\boldsymbol{X} \times \boldsymbol{Y}^{\parallel}} = \boldsymbol{Z} \tag{7.8}$$

As the vector component $\boldsymbol{Y}^{\parallel}$ that is parallel to $\boldsymbol{X}$ can have any length without affecting the cross product, there are an infinite number of solutions for $\boldsymbol{Y}$ satisfying (). This is why we cannot solve directly for $\Delta\boldsymbol{r}$
Then, for the perpendicular unit vectors$(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)$ defining a Cartesian right handed coordinate system we have

$$\begin{aligned} \boldsymbol{e}_x \times \boldsymbol{e}_y &= \boldsymbol{e}_z \\ \boldsymbol{e}_z \times \boldsymbol{e}_x &= \boldsymbol{e}_y \\ &\implies (\boldsymbol{e}_x \times \boldsymbol{e}_y) \times \boldsymbol{e}_x = \boldsymbol{e}_y \end{aligned} \tag{7.9}$$

Analogously this must hold for any two unit vectors that are perpendicular in the same plane. By assuming that the unit vectors $\boldsymbol{e}_{x'}$ and $\boldsymbol{e}_{z'}$ defining the direction for the general vectors $\boldsymbol{X}$ and $\boldsymbol{Z}$ are known in combination with (7.2.2) and the general result (7.2.2) we have

$$\begin{aligned} (\boldsymbol{e}_{x'} \times \boldsymbol{Y}) \times \boldsymbol{e}_{x'} = (\boldsymbol{e}_{x'} \times \left[\left\|\boldsymbol{Y}^{\perp}\right\| \boldsymbol{e}_{y\perp} + \left\|\boldsymbol{Y}^{\parallel}\right\| \boldsymbol{e}_{y\parallel}\right]) \times \boldsymbol{e}_{x'} = \\ \left\|\boldsymbol{Y}^{\perp}\right\| (\boldsymbol{e}_x \times e_{y\perp}) \times \boldsymbol{e}_x = \left\|\boldsymbol{Y}^{\perp}\right\| e_{y\perp} = \boldsymbol{Y}^{\perp} \end{aligned} \tag{7.10}$$

Then by rewriting (7.2.2)

$$\begin{aligned} \boldsymbol{X} \times Y = \|\boldsymbol{X}\| \, \boldsymbol{e}_{x'} \times \left\|\boldsymbol{Y}^{\perp}\right\| \boldsymbol{e}_{y\perp} = \boldsymbol{Z} \\ \implies \boldsymbol{e}_{x'} \times e_{Y\perp} = \frac{\|1\|}{\|X\| \, \|\boldsymbol{Y}^{\perp}\|} \boldsymbol{Z} \end{aligned} \tag{7.11}$$

Combining (7.11) and (7.10) then yields

$$\boldsymbol{Y}^{\perp} = \frac{1}{\|\boldsymbol{X}\|} \boldsymbol{Z} \times \boldsymbol{e}_{x'} = \frac{1}{\|\boldsymbol{X}\|^2} \boldsymbol{Z} \times \boldsymbol{X} \tag{7.12}$$

If the norm $|\boldsymbol{Y}|$ is known the parallel component $\boldsymbol{Y}^{\|}$ can then be found by the Pythagorean theorem, down to sign of it. This is illustrated in Figure 7.3.
How to use this then? Consider the equation of motion as reformulated for the offset finding problem in (7.5). First, let us assume that en estimate of the vector to the center of gravity $\hat{\boldsymbol{r}}_{cb,k}$ is updated at every time step k . Inserting this into (7.5) and assuming steady state balancing so $\dot{\boldsymbol{\omega}}_c = \boldsymbol{\omega}_c = 0$ it can be deduced that

$$\underbrace{\boldsymbol{T} + \hat{\boldsymbol{r}}_{cb,k} \times m_t \boldsymbol{g}}_{\boldsymbol{Z}} = \underbrace{m_t \boldsymbol{g}}_{\boldsymbol{X}} \times \underbrace{\Delta \boldsymbol{r}}_{\boldsymbol{Y}} \tag{7.13}$$

Then by applying (7.12) on (7.13)

$$\Delta \boldsymbol{r}^{\perp} = \frac{1}{m_t^2 g_0^2} (\boldsymbol{T} + [\hat{\boldsymbol{r}}_{cb,k} \times m_t \boldsymbol{g}]) \times m_t \boldsymbol{g} = \left[\frac{1}{m_t g_0} \boldsymbol{T} + \hat{\boldsymbol{r}}_{cb,k} \times \boldsymbol{e}_g \right] \times \boldsymbol{e}_g \tag{7.14}$$

were $\boldsymbol{g} = g_0 \boldsymbol{e}_g$. Now, (7.14) should give us $\Delta \boldsymbol{r}^{\perp}$ as illustrated in Figure 7.3. Note that the dimension of all involved terms become distance, as is anticipated.
So, while it is not possible to reconstruct $\Delta \boldsymbol{r}$ entirely, it is possible to reconstruct the component that is perpendicular to the gravity vector $\boldsymbol{g}$. It lies in the same plane as the plane formed by $\boldsymbol{g}$ and $\Delta \boldsymbol{r}$ that is intersected by the center of gravity, which is valuable information.
Remembering that this is a 3D dimensional problem, lets focus on the plane where the vectors $\boldsymbol{g}$ and $\Delta \boldsymbol{r}$ as illustrated in Figure 7.5. The vectors defined in (7.4) that does not lie in this plane are illustrated by dotted lines.

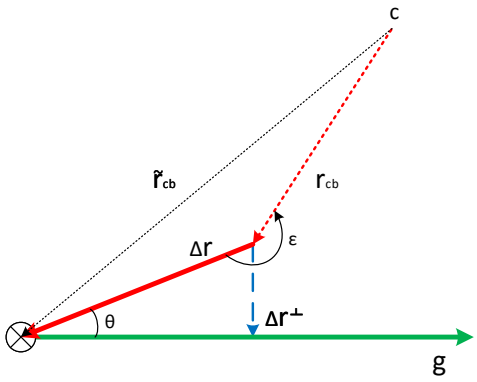


Figure 7.5: Illustration of the vectors involved for the orthogonal approach to the offset estimation. The dotted vectors does not lie in the same plane as $\Delta \boldsymbol{r}$ and $\boldsymbol{g}$

An idea would be to update $\hat{\boldsymbol{r}}_{cb}$ in discrete time steps according to

$$\hat{\boldsymbol{r}}_{cb,k+1} = \hat{\boldsymbol{r}}_{cb,k} + c\Delta \boldsymbol{r}_k \tag{7.15}$$

Heuristically speaking, each update should bring the estimate slightly closer to the real location of the mass center.

7.2.2.1 Issues to be investigated

In simulation (7.14) was used for updating $\hat{\boldsymbol{r}}_{cb,k}$ and results were promising. However, in practice it did not first converge. Without any further justification, a term was crossed out to yield the update of the estimation

$$\Delta \boldsymbol{r}^{\perp} = \frac{1}{m_t^2 g_0^2} \boldsymbol{T} \times m_t \boldsymbol{g} \tag{7.16}$$

It works on the real system and it is unclear why. It could just a be a matter of choosing the right tuning factor.

7.2.2.2 Summary of the orthogonal approach

A summary of the preceding chapter is given in 2

Algorithm 2 Geometric Center of Gravity (COG) finder

procedure UPDATERCB($\dot{\boldsymbol{E}}$, $\boldsymbol{g}$)
  if Init **then**
    $k \leftarrow 0$
    $\tilde{\boldsymbol{r}}_{cb,0} \leftarrow \boldsymbol{r}_{cb}$ ▷ Variable initialization
  end if
  $k \leftarrow k + 1$ ▷ Update time index
  if $\left(\left\|\dot{\boldsymbol{E}}\right\|^2 < E_{tres}\right)$ **then** ▷ Wait until cube is still
    $\Delta \boldsymbol{r}^{\perp} \leftarrow \frac{1}{m_t^2 g_0^2}(\boldsymbol{T} + [\hat{\boldsymbol{r}}_{cb,k} \times m_t \boldsymbol{g}]) \times m_t \boldsymbol{g}$ ▷ Alternatively Equation (7.16)
    $\hat{\boldsymbol{r}}_{cb,k} \leftarrow \hat{\boldsymbol{r}}_{cb,k-1} + c\Delta \boldsymbol{r}^{\perp}$
  end if
end procedure

It is unknown whether the geometric approach generally can be showed to converge and then for which tuning factor c. This factor was tuned experimentally. The treshold for steady state was set to $E_{tres} = 0.2$

7.2.3 Further ideas

Extended Kalman Filter are often used for these kind of problems However, as infinitely many offset vectors $\Delta \boldsymbol{r}$ are possible at a given orientation care has to be taken if formulating such an approach. Probably solutions for an offset vector should be constrained to lie in a plane or such.
Another idea is formulating an optimization problem with quadratic cost on either the squared norm of the torque insignal $\|\boldsymbol{T}\|^2$ or the difference between the actual location of center of gravity and the update $\|\tilde{\boldsymbol{r}}_{cb} - (\hat{\boldsymbol{r}}_{cb,k} + \Delta \boldsymbol{r})\|^2$.
Then there is a proposal inspired by the two dimensional case. For a simple idea, say that the orientation is described by Euler angles such that the inclination relative to the vertical global axis in the reference frame A can be described by an angle θ . The idea is that if the cube is inclined θ it is affected by some constant gravity pull from the offset of the center of gravity in the θ direction. The plane where this occurs will appear exactly as for the 2D case in Figure 7.1. By applying appropriate 3D kinematics a setpoint can be determined that moves the cube in a direction opposite of θ. By doing so iteratively the equilibrium point should be found eventually. Practically it would probably be a good idea to use unit quaternions to describe the orientation for the implementation.

8

Orientation estimation

A crucial part of obtaining proper operation of a system as Cubex is continuously obtaining accurate estimation of the current orientation of the system. A good part of the project time was spent working out practical and theoretical aspect of this.

8.1 Sensor fusion of the inertial measurements

An IMU unit doesn't directly measure orientation but instead commonly acceleration, rotation and in some cases magnetic fields. The latter is for relating the current orientation to the earth magnetic field. All this sensors have their pros and cons. In order to exploit the best properties of each sensor, typically some sort of sensor fusion technique is employed. Tradtionally, Kalman filters based approaches have been very common. A fairly new approach to this is the Madgwick algorithm. Stated benefits are much higher computational efficiency, easier tuning and at least as accurate and responsive for Kalman based approaches [31]. We have previously worked with Extended Kalman Filters (EKF) for orientation estimation and agree on the first two statements. When it comes to dynamic response the paper covering the Madgwick [31] does not specify enough regarding the compared Kalman filter for this to be conclusive in our opinion. However, to our experience the Madgwick algorithm seemed to work very well for estimating the orientation in 3D. Also much computational power was freed as compared to an EKF. Having chosen a ready made approach for sensor fusion, most focus regarding the orientation estimation fell on treating the IMU's properly.

8.1.1 Quaternion conversions

The Madgwick algorithm uses unit quaternions [50] to represent orientation while the current control systems uses (Z, X', Z'') Euler Angles. In the Magdwick implementation found at an official homepage [56] there is a conversion between unit quaternions and (Z, Y', X'') Euler angles supplied, which is incompatible with the definitions in this context. Therefore a relation between (Z, X', Z'') and unit quaterions had to be derived.

A unit quaternion used to represent orientation is a four dimensional number

$$\boldsymbol{q} = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\gamma}{2}) \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\gamma}{2}) \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \cos(\frac{\gamma}{2}) \\ \sin(\frac{\gamma}{2})\boldsymbol{u} \end{bmatrix} \tag{8.1}$$

This represents orientation by a rotation an angle γ around the unit vector $\boldsymbol{u}$ [50]. It is based on Euler's Theorem of rotation which states any orientation can be represented as such [34].
Importantly, the quaternion can then be used to form a rotation matrix between the body frame B and the reference frame A according to

$$\begin{aligned} R_B^A = &\begin{bmatrix} 2v_0^2 - 1 + 2v_1^2 & 2(v_1v_2 + v_0v_3) & 2(v_1v_3 - v_0v_2) \\ 2(v_1v_2 - v_0v_3) & 2v_0^2 - 1 + 2v_2^2 & 2(v_2v_3 + v_0v_1) \\ 2(v_1v_3 + v_0v_2) & 2(v_2v_3 - v_0v_1) & 2v_0^2 - 1 + 2v_3^2 \end{bmatrix} = \\ &\begin{bmatrix} C_\phi C_\psi - C_\theta S_\phi S_\psi & -C_\psi C_\theta S_\phi - C_\phi S_\psi & S_\phi S_\theta \\ -C_\psi S_\phi + C_\phi C_\theta S_\psi & C_\phi C_\psi C_\theta - S_\phi S_\psi & -C_\phi S_\theta \\ S_\psi S_\theta & C_\psi S_\theta & C_\theta \end{bmatrix} \end{aligned} \tag{8.2}$$

The matrix is from Magdwick's internal report [31]. This has been equated to the rotation matrix for the current choice of Euler angles where (ϕ, θ, ψ) perform a rotation around the $Z_A, X_{A'}, Z_{A''}$ axises respectively. Short form notation is used where the input argument a is mapped as $C_a \triangleq \cos(a)$ and $S_a \triangleq \sin(a)$. Although there are 12 different Euler angle representations that can be used to represent the orientation of a body in three dimensions relative to a reference frame, there exists only one rotation matrix for rotating a vector between the body-and the reference frame. [50]. This follows directly by the fact that the rotation matrix contains the unit vectors of the body frame B expressed in the reference frame A.
By using the notation $R_{i,j}$ to denote the element on row i at column j from the rotation matrix we have from (8.2)

$$\begin{aligned} &\frac{R_{1,3}}{R_{2,3}} = \frac{\sin(\phi)}{-\cos(\phi)} = -\tan(\phi) \\ &\implies \phi = -\operatorname{atan2}(R_{1,3}, R_{2,3}) = -\operatorname{atan2}(v_1v_3 - v_0v_2, \quad v_2v_3 + v_0v_1) \\ \\ &R_{3,3} = \cos(\theta) \\ &\implies \theta = \arccos\left(R_{3,3}\right) = \arccos\left(2v_0^2 - 1 + 2v_3^2\right) \\ \\ &\frac{R_{3,1}}{R_{3,2}} = \frac{\sin(\psi)}{\cos(\psi)} = \tan(\psi) \\ &\implies \psi = \operatorname{atan2}(R_{3,1}, R_{3,2}) = \operatorname{atan2}(v_1v_3 + v_0v_2, \quad v_2v_3 - v_0v_1) \end{aligned} \tag{8.3}$$

Here $\operatorname{atan2}(y, x)$ is defined as in most programming languages to operate in four quadrants which is required for finding orientations defined by an angle a when $|a| > \pi/2$. By using $\operatorname{atan2}(y, x)$ singularities associated with $\arctan(y/x)$ when x approaches zero are avoided.

8.2 Inertial measurements units

An IMU generally consist of three axis gyroscope and three axis accelerometer and in some cases a three axis magnetometer measuring linear acceleration, rotation and magnetic fields. The IMU's of choice were the micro-machined electromechanical systems (MEMS) device MPU-9250 from InvenSense.

8.2.1 Error characteristics

There are several error characteristics associated with IMU's. One is misalignment, either of each axis of measurement internally or externally, by mounting. We propose an approach for finding and correcting misalignment by mounting in Section 8.3 . Another important error is gyroscopic bias. This will add a term $\boldsymbol{b}$ to the actual angular velocity $\boldsymbol{\omega}$ according to yielding a measured angular velocity

$$\boldsymbol{\omega}_s = \boldsymbol{\omega} + \boldsymbol{b} \tag{8.4}$$

The term b drifts over time mainly due to self heating. This causes the bias to increase linearly proportional to operation time time. The bias seemed to be the most influential error apart from sensor misalignment in this context, as it makes the system appear rotating even if it is completely still. The aforementioned Madgwick algorithm has internal drift estimation compensation [31]. However, this needed to be completed an extra external bias compensation described in Section 8.2.2. Then, as most real sensors all measurements are affected by white noise. The first step dealing with this was averaging measurements from multiple sensors as described in Section 8.4. This was output but was then low pass filtered when used for feedback of angular velocity in the control loop, but fed raw into Madgwick. For a more in depth review on error characteristics of IMU's see for instance [57].

8.2.2 Bias estimation

Assuming a measurement model for the angular velocity

$$\boldsymbol{\omega}_s(t) = \boldsymbol{\omega}(t) + \boldsymbol{b}(t) + \boldsymbol{w}(t) \tag{8.5}$$

were $w(t)$ is Gaussian noise and $b(t)$ is a bias term. If the bias is approximated to be constant and the angular velocity is zero, the bias can be estimated as

$$\hat{\boldsymbol{b}}_0 = E(\boldsymbol{\omega}_s(t)) \tag{8.6}$$

where $E(\cdot)$ is the operator for expected value. Equation (8.6) is approximated collecting and averaging N samples during the startup of the cube, when it is assumed to be laying still. The values for $\hat{\boldsymbol{b}}_0$ are then stored and subtracted from the gyroscope measurements according to

$$\boldsymbol{\omega}_{k,s,out} = \boldsymbol{\omega}_{k,s} - \hat{\boldsymbol{b}}_0 \tag{8.7}$$

for the remainder of the operation, until the system is reset. Then $\boldsymbol{\omega}_{k,s,out}$ is used for state estimation using the Madgwick filter and for the control loop.

8.3 Procedure to correct IMU misalignment

A major error source for the inertial measurements is misalignment of the Inertial Measurement Unit (IMU). If the IMU is assumed to have a certain alignment but doesn't, measurement will off course be way off. An equivalent metaphor a spirit level that is misaligned in it's casing, as depicted in Figure 8.1. If the spirit level is

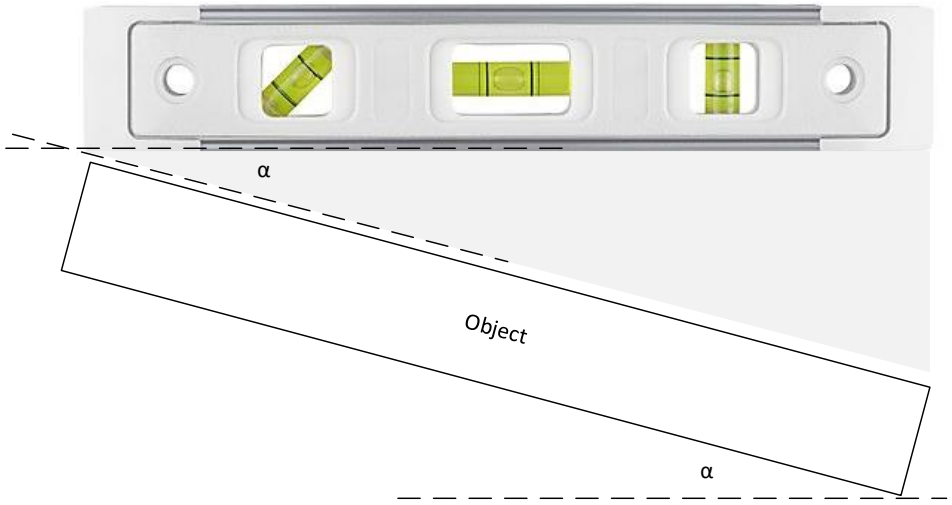
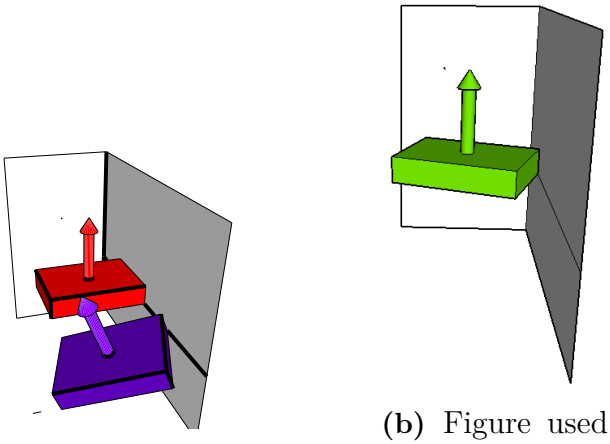


Figure 8.1: Illustrating the impact of IMU misalignment using a spirit level.

assumed to be mounted straight in it's case, but in fact is mounted with an offset α, all measurement will also get this unwanted offset.
The common remedy for a correcting a spirit level such as the one depicted in Figure 8.1 is ensuring to correct the mechanical installation so that the misalignment α approaches zero. It seems that this is also a common solution for IMU's in industry. A lot of resources are spent to ensure that IMU's are mounted mechanically such that they end up at nominal orientation. But even then, misalignment due to uneven soldering and such may occur. [58].
In this particular case, the Cubex frame system had limited possibilities for placing the IMU's freely. Therefore it proved practically difficult to place the IMU units such that they would be perfectly aligned. For this reason, a system identification procedure was used to find the IMU misalignment. This data was then used to determine a rotation matrix for correcting the IMU misalignment. This could be seen as finding the misalignment α in Figure 8.1 and compensating for it, but in three dimensions. The original creators of the Cubli project mention they use principal

component analysis for similar purpose. However, few details are mentioned. It seems that a few manually collected data points were used [17].
An attempt to illustrate misalignment of an IMU in 3D can be found in Figure 8.2a. A misalignment error would be assuming that IMU measurement frame is orientated orthogonally in relation to the cube frame as the red box, while in reality it has an orientation more similar to the blue box. The measurements of acceleration ${}^{S}a$ and



(a) The red box illustrates an IMU appearing perfectly in the frame.

(b) Figure used to illustrate the last rotation

Figure 8.2: Illustrations used to depict misaligned mounting of IMU units.

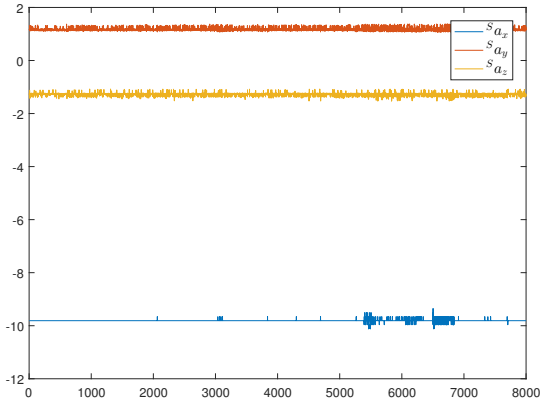
angular velocity ${}^{S}\omega$ are vectors expressed in the sensor frame. If the frame sensor frame is blue frame in Figure 8.2a, a rotation matrix R_S^B can be used to express these vectors in the red frame that is a frame with all axis parallel to the body frame of the cube denoted B according to

$$ {}^{B}\boldsymbol{a} = R_S^{BS}\boldsymbol{a}, \qquad {}^{B}\boldsymbol{\omega} = R_S^{BS}\boldsymbol{\omega} $$

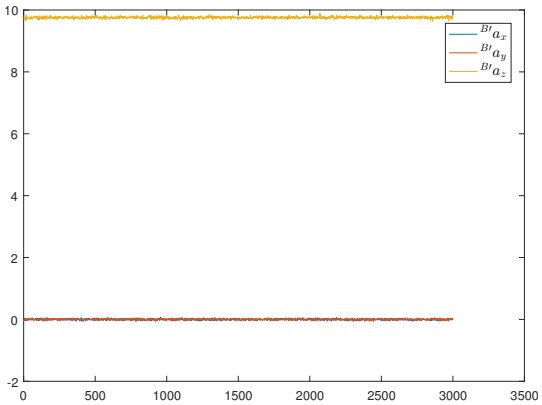
The problem thus lies in finding the rotation matrix R_S^B. Beginning, an accelerometer with an orientation that coincides perfectly with the cube frame should nominally output

$$ {}^{B'}a = \begin{bmatrix} 0 & 0 & \hat{g} \end{bmatrix}^T \tag{8.8} $$

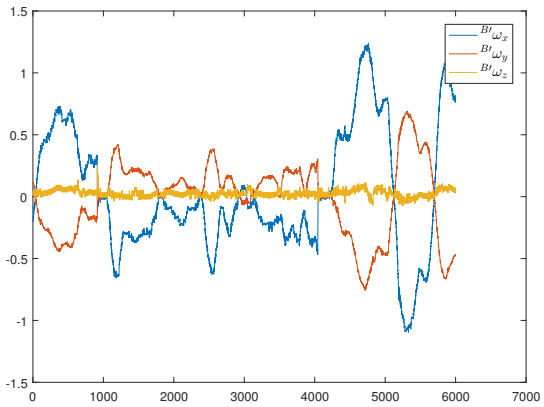
where $\hat{g}$ is the nominal gravity constant, when the cube is laying perfectly flat on a horizontal surface. A misaligned IMU will however output ${}^{S}a = \begin{bmatrix} a_{xs} & a_{ys} & a_{zs} \end{bmatrix}^T$ as is seen in Figure 8.3a.



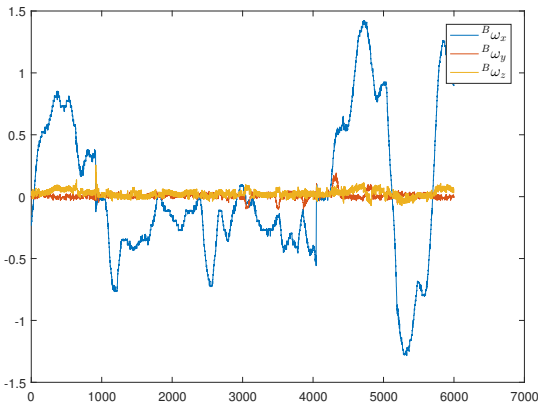
(a) Accelerometer data in sensor frame S



(b) Accelerometer data in B' frame



(c) Gyroscope data in B' frame



(d) Gyroscope data in body frame B

Figure 8.3: Data from the IMU expressed in different frames. First the gravity data is rotated the angles (ϕ_S, θ_S) around the $(x_s, y_{B''})$ axis to obtain the B' frame where gravity is correctly aligned. Gyroscopic data is then expressed in the B' frame and rotated to the cubes body frame B (nominally) by the angle ψ_S

Regardless of the orientation of the IMU, there exists at least two elementary rotations with associated rotation matrices $R(\phi), R(\theta)$ such that ${}^{B'}a = R(\theta)R(\phi){}^{s}\boldsymbol{a}$. This can be derived by geometrical reasoning. Imagine some vector $\boldsymbol{r}$ that is non vertical in the global reference frame. Using only to elementary rotations, this vector can always be rotated so it appears completely vertical. First, ${}^{s}\boldsymbol{a} \in \mathbb{R}^{3\times N}$ is averaged for all N samples over the rows instants to minimize random noise. This yields a single vector ${}^{s}\bar{\boldsymbol{a}} = (a_{xs}, a_{ys}, a_{zs})^T$ The first rotation is to express ${}^{s}\bar{\boldsymbol{a}}$ in an intermediate frame $(X_{B''}, Y_{B''}, Z_{B''})$ being rotated an angle ϕ around the $X_s, X_{B''}$ axis in relation to the frame (X_s, Y_s, Z_s). This is illustrated in Figure 8.4.

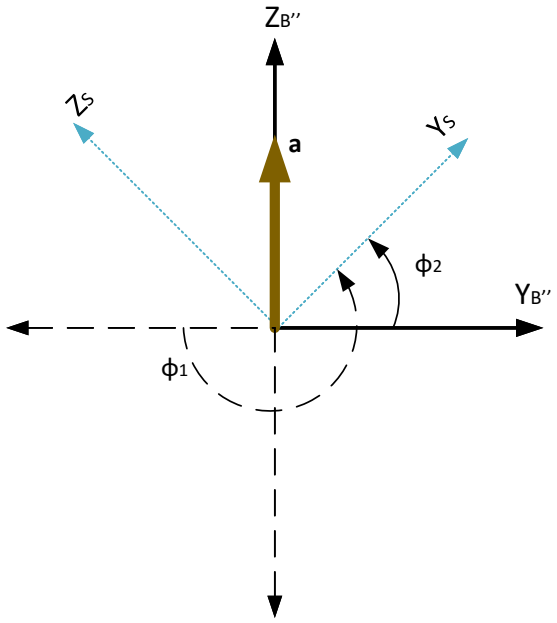


Figure 8.4: Illustrations of the two coordinate rotation angles (ϕ_1, ϕ_2) that can eliminate the x component of the vector $\boldsymbol{a}$

Graphically, it can be seen that there exists some frame $(X_{B''}, Y_{B''}, Z_{B''})$ where the vector $\boldsymbol{a}$ can be expressed without any Y component, or algebraically

$$ {}^{B''}\boldsymbol{a} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} a_{xs} \\ a_{ys} \\ a_{zs} \end{bmatrix} = \begin{bmatrix} a_{xs} \\ 0 \\ a_{zB''} \end{bmatrix} $$

Evaluating the second row yields

$$ \frac{\sin(\phi)}{\cos(\phi)} = \tan(\phi) = \frac{a_{ys}}{a_{zs}} \tag{8.9} $$

Geometrically speaking, this can be interpreted as the angle ϕ_i for the quadrants $i = 1, \ldots, 4$ shown in Figure 8.5.

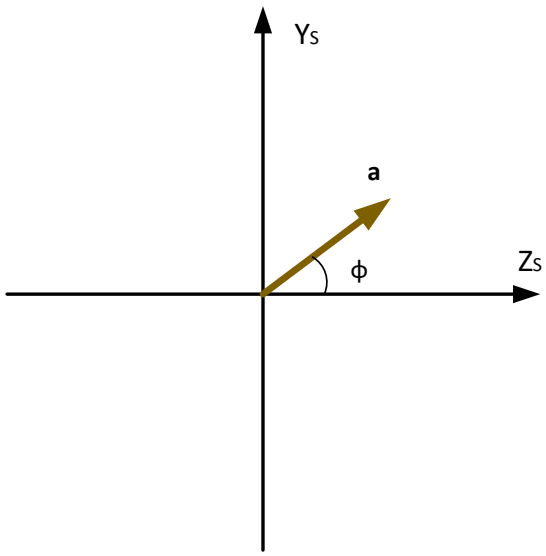


Figure 8.5: The solution to Equation (8.9) can be interpreted geometrically as the angle between the Z_s axis and the $\boldsymbol{a}$ vector

Multiplying a vector a vector by a rotation matrix can either be seen as expressing this vector in another frame, or rotating the vector within its frame. For the time being, it can be more convenient to view is at the latter. By inspecting Figure 8.5 it can be seen that the angle we want us the angle between between the positive part of the Z_s axis and the vector $\boldsymbol{a}$. This is conveniently found by atan2$(\cdot)$,the four quadrant extension of arctan$(\cdot)$, according to

$$\phi = \text{atan2}(a_{ys}, a_{zs}) \tag{8.10}$$

A second rotation can then be formulated around the $Y_{B''}$ in a similar way as

$${}^{B'}\boldsymbol{a} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} a_{xs} \\ 0 \\ a_{zB''} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{\boldsymbol{g}} \end{bmatrix} \tag{8.11}$$

To not invoke the unknown the unknown estimate of the gravity, row 1 of (8.11) is evaluated. It can be found that there are two θ solutions due to the sign inversion according to of one of the components according to

$$\theta_1 = \text{atan2}(-a_{xs}, a_{zB''}) \qquad \theta_2 = \text{atan2}(a_{xs}, -a_{zB''}) \tag{8.12}$$

The solutions are evaluated using (8.11) and the solution yielding $\hat{\boldsymbol{g}} > 0$ is kept. The result of these two rotations ϕ and θ can be seen in Figure 8.3b. Algebraically this is achieved by creating the compound rotation matrix $R_{yx} = R_y(\theta)R_x(\phi)$ multiplying this with the data ${}^s\boldsymbol{a}$ seen in Figure 8.3a according to $R_{yx}{}^s\boldsymbol{a}$. These data will appear as if they were collected from the green IMU unit in Figure 8.2b. This final rotation

cannot be found by accelerometer data still keeping the cube at this orientation. Any rotation ψ around the $Z_{B'}$ will leave the acceleration vector unchanged in this position, and a rotation around another axis is undesired. Other data has to be collected by before proceeding. One approach would be to turn the whole cube platform over and collect data from another orientation, and basically find a similar approach to from the finding of θ and ϕ to find ψ. However, instead the system was rotated around the X_B axis and gyroscope data was collected according to ${}^{B'}\boldsymbol{\omega} = R_{yx}{}^{s}\boldsymbol{\omega}$ which is plotted in Figure 8.3c. This it to find an approach that wo would be more applicable to for instance IMU's in quadcopters, where a rotation axis is usually found more easily than perpendicular surfaces of the system. To find the find this final rotation an optimization problem is formulated as

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{N_x} \left\| \text{diag}[0,1,1] R_z(\psi)^{B'}\boldsymbol{\omega} \right\|^2 = \sum_{i=1}^{N_x} \left\| \begin{bmatrix} 0 \\ \omega_x \sin(\psi) + \omega_y \cos(\psi) \\ \omega_z \end{bmatrix} \right\|^2 \tag{8.13}$$

which basically has the goal to minimize any rotations around the Y_B and Z_B axises. This as we know that according to how we excited the system and collected the data, most or all rotation was around the X_B axis. It is an unconstrained optimization problem. By fully expanding the i'th a term of (8.13), including the sample index i in subscripts

$$C_i = {\omega_{x,i}}^2 + {\omega_{y,i}}^2 + 2\omega_{x,i}\omega_{y,i} \sin(\psi)\cos(\psi) + {\omega_{z,i}}^2 \tag{8.14}$$

it can be seen that summing an arbitrary number of terms will yield an optimization problem with exactly the same structure as for one term. Defining

$$S_x^2 \triangleq \sum \omega_{x,i}^2 \qquad S_y^2 \triangleq \sum \omega_{y,i}^2 \qquad S_{xy} \triangleq \sum \omega_{x,i}\omega_{y,i} \tag{8.15}$$

Setting the partial derivative with respect to ψ in(8.13) to zero using together with the definition (8.15)

$$\frac{\partial}{\partial \psi} \sum C_i = \sin(2\psi)\left({S_x}^2 - {S_y}^2\right) + 2S_{xy}\cos(2\psi) = 0 \tag{8.16}$$

Equation (8.16) has two solutions as it has one global minima and maxima respectively. The global minimizer is

$$\psi = \text{atan2}(2S_{xy}, \quad S_x^2 - S_y^2) \pm n\pi \tag{8.17}$$

By letting $n = 0, 1$ the definition of positive direction for the x direction can be changed. By rotation the raw data from the misaligned in Figure 8.2a using the matrix $R_{zyx} = R_z(\psi)R_y(\theta)R_x(\phi)$ should now appear as if the IMU were mounted as the red IMU. A demonstration on real data for the complete rotation matrix is shown in Figure 8.3d

8.4 Noise reduction by a weighted average

Data from various IMU's units are averaged to reduce random noise. It is shown in Appendix A that averaging to sensors, measuring the same variable and amplitudes

having equal random noise halves the total noise variance. Average the two mounted IMU's on the cube with equal weight reduce the random noise and thus get a more reliable measurement.

9
Results

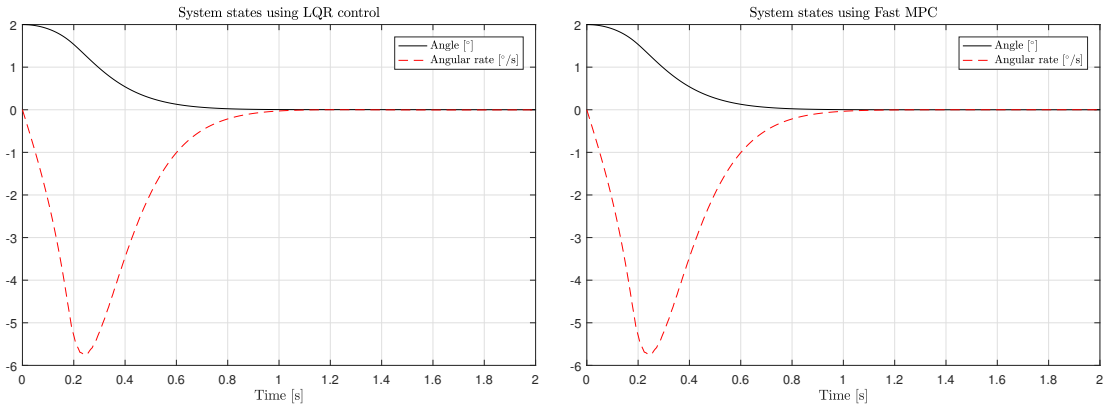
This section summarizes the most important of the quantifiable results from the project. All MPC results are from approximate gradient approach for MPC solvers described in Section 3.10.2. This is referred to as *Fast MPC*. With *real system* we mean the real world mechanical Cubli based platform used in the project. Within this section *equilibrium point* means the point where the cube can balance on either an edge or a corner without any additional torque input, theoretically.

9.1 Edge balancing

This feat, as showed in Figure 9.3 was using both an MPC and LQR controller in simulation and on the real system.

9.1.1 Simulation

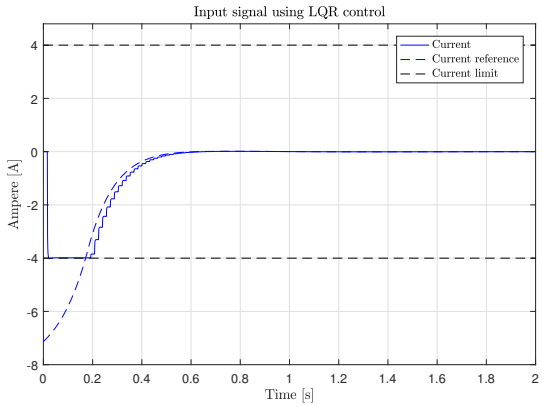
Figure 9.1 shows the response of the system in simulation when starting from a initial orientation of a 2 deg offset from the equilibrium point. It can be seen that both the MPC controller and LQR controller have very similar performance and behavior.



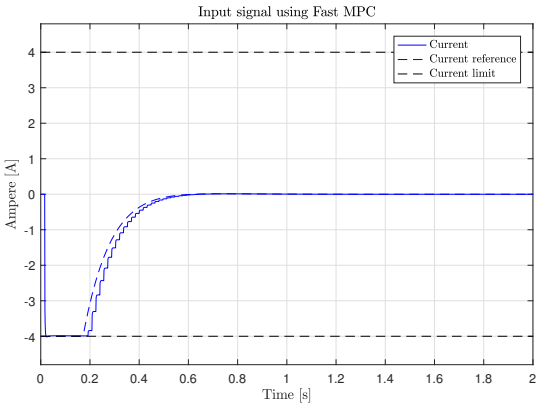
(a) Cube angle and angular rate using LQR controller

(b) Cube angle and angular rate using MPC

Figure 9.1: The response of the cubes angle and angular velocity from a initial orientation of 2 degrees using a LQR controller and MPC



(a) Current and the reference current using LQR controller



(b) Current and the reference current using MPC

Figure 9.2: The response of the current reference to the motor from a initial orientation of 2 degrees using a LQR controller and MPC.

As said in Section 6.0.1, the system is actuated by torque, by setting a current reference i_{ref} to internal controllers for the motors. Thus the outsignal from the LQR and MPC controller is i_{ref} is cascaded with the motor controllers internal control loops. Figure 9.2 show the reference current from both using LQR and Fast MPC for the initial response in Figure 9.1. The motor controllers are internally clamped to max output ± 4 ampere, which was used as the only constraint specified for the MPC optimization problem. However, the non-linear saturation at his amperage in series with the LQR controller appears to make it similar to MPC.

9.1.2 Real system

The real system can be seen edge balacning in Figure 9.3

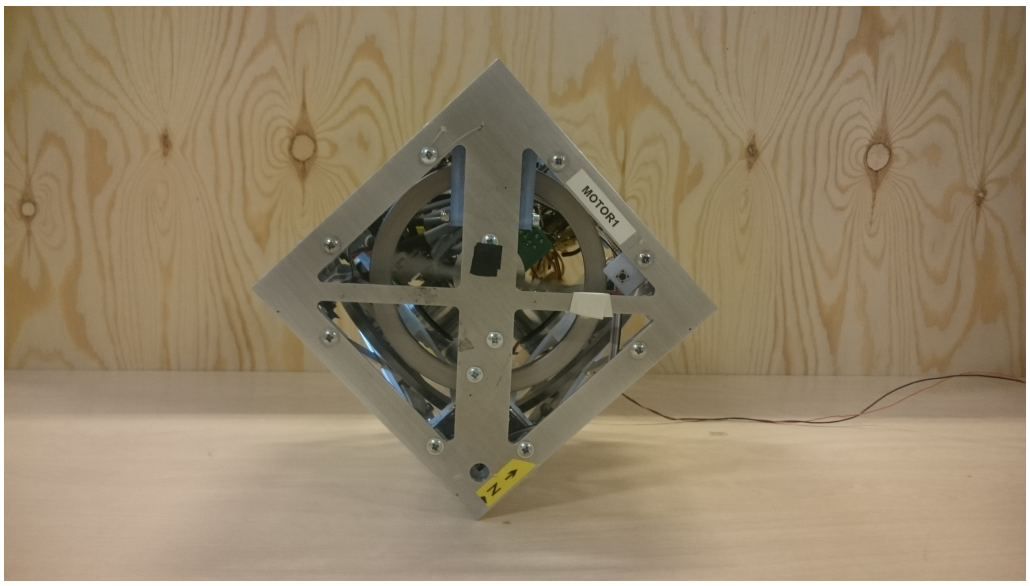
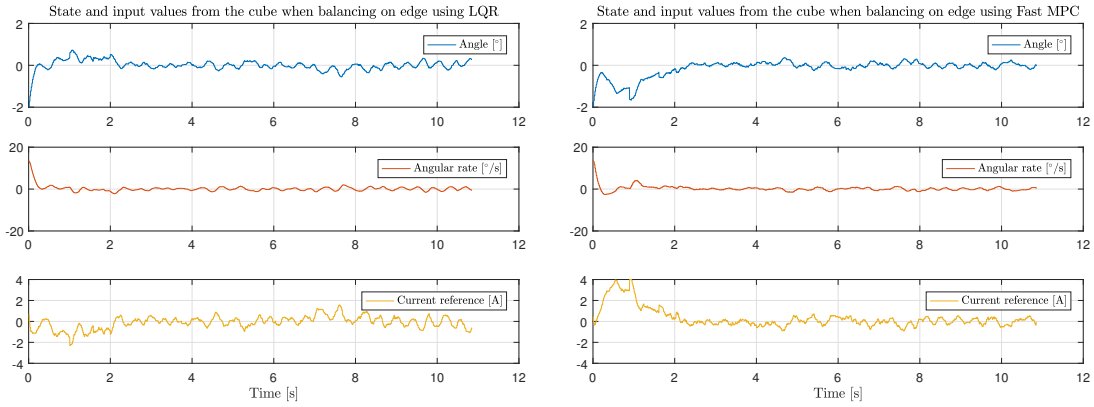


Figure 9.3: The cube balancing on its edge

Figure 9.4 the system states along with the current reference to the motor from running LQR and Fast MPC on the real cube.



(a) System states and current reference signal when using LQR

(b) System states and current reference signal when using Fast MPC

Figure 9.4: System states and current reference signal to motor when the real system is balancing on its edge with LQR and Fast MPC.

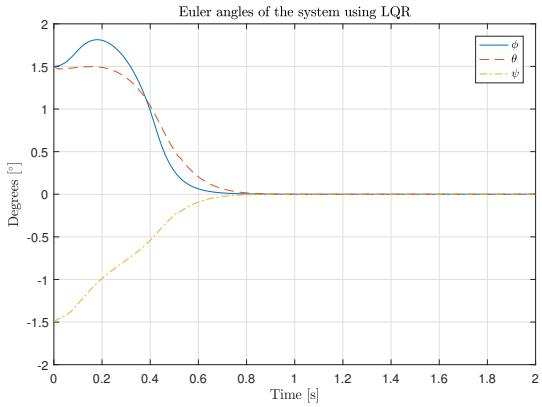
Just as for the simulation, both controllers achieve edge balancing. However, in practice the system is influenced by sensor noise which was not included in the simulation. Also, an algorithm for finding the center of gravity as described in Section 7 was always used. This algorithm was as crucial as the controller itself for successful balancing. Therefore, it is hard to directly compare the LQR and MPC approaches for the real system during edge balancing.

9.2 Corner balancing

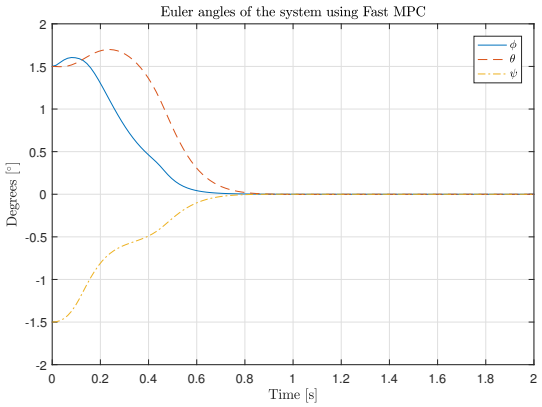
This is considered a main feat of the system and is considerably harder to achieve than edge balancing.

9.2.1 Simulation

Simulation results of the cube balancing on its corner, starting from non-equilibrium initial conditions, can be seen in Figures 9.5-9.9. The equilibrium point is where all Euler angles are zero. In Figure 9.5 shows the system response when using LQR and Fast MPC respectively. Input signals to the system can be seen in Figure 9.6. The response from the reaction wheels can be seen in Figure 9.9.

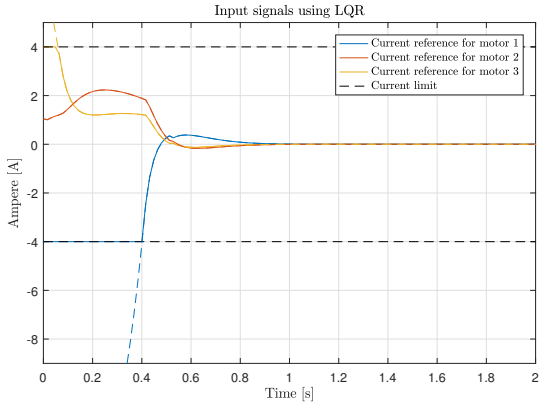


(a) Response on the cube Euler angles from an initial condition using LQR

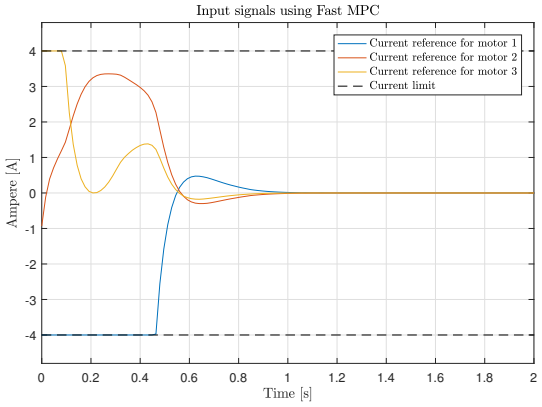


(b) Response on the cube Euler angles from an initial condition using MPC

Figure 9.5: The response of Response on the cube Euler angles from an initial condition using different controllers

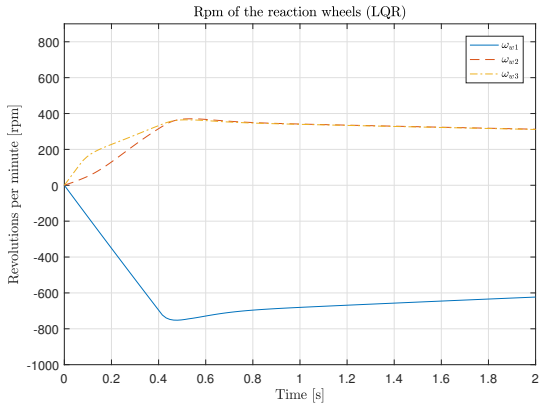


(a) Reference signals to the system from LQR

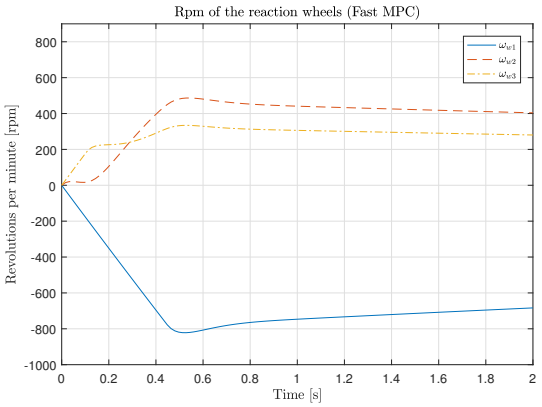


(b) Reference signals to the system from MPC

Figure 9.6: The reference signals response from a initial condition on the Euler angles using different controllers.



(a) The reaction wheels angular rate using LQR



(b) The reaction wheels angular rate using MPC

Figure 9.7: The reaction wheels angular rate response from a initial condition on the Euler angels using different controllers.

Investigating Figure 9.6 and Figure 9.9 it can be seen that the MPC utilizes reaction wheel 2 more and reaction wheel 3 less compared to the LQR controller. Also, the resulting orientation during the initial transient is different. However, both the controller complete to steers the orientation of the cube to its equilibrium.

9.2.2 Real system

Corner balancing in action can be seen in Figure 9.8.

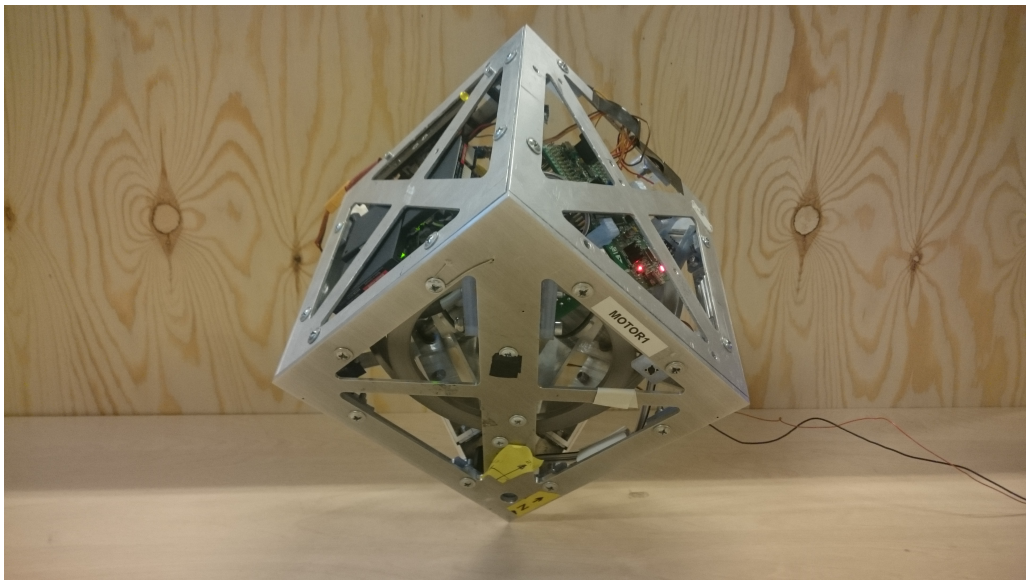
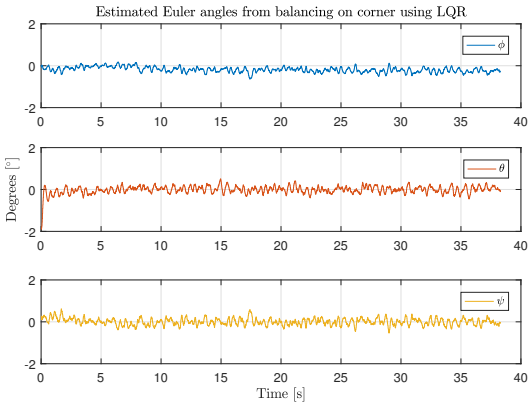
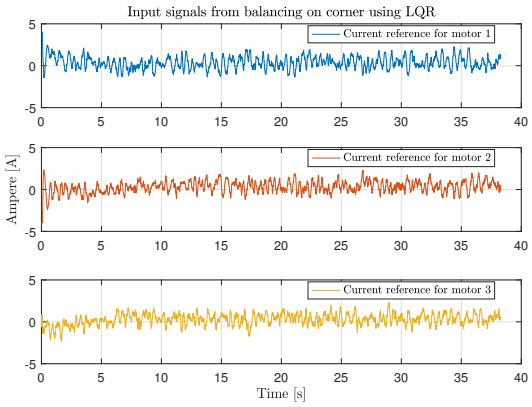


Figure 9.8: The cube balancing on its corner

Figure 9.9 show the estimated Euler angles and the current reference signals to the actuator from the cube balancing on its corner. As in simulation, the equilibrium point for balancing on the corner is at 0 degrees.



(a) Estimated Euler angles of the cube when balancing.



(b) Current reference signals to the motors.

Figure 9.9: Estimated Euler angels and current referance signals to the motors when the physical cube is balancing on its corner.

Subtle oscillations around the equilibrium point where observed, which is reflected in the measurement data. It is not completely clear why these occur. Sensor and actuator errors and a sub optimal estimate of the location of the center of gravity are reasonable explanations.

9.3 Model predictive control execution times

In this section we will present the result of the proposed Algorithm 1 using different bandwidth m on the preconditioner matrix P when the cube is balancing on its edge and corner in simulation. The same method proposed in [3] is used for evaluating the performance for the approximation of the preconditioner . That is calculating the ratio between the average number of floating point operation require for convergence of Algorithm 1 when using a banded matrix, against using the full preconditioner, i.e

$$\theta_m = \frac{c_m i_{avg,m}}{c_{full} i_{avg,full}} \tag{9.1}$$

where $i_{avg,m}$ is the average number of iteration needed for convergence and c_m is the total number of floating point operations needed for one iteration of the algorithm. A $\theta_m < 1$ means that the approximated preconditioner requires fewer floating points operations than the full preconditioner and vice versa.
The number of floating points operations per iteration when approximating the preconditioner as a m-banded matrix is found by

$$c_m = 4M + 6Nn \tag{9.2}$$

where M Is the number of nonzero elements in the matrix PD, N is the length of the prediction horizon and n is the dimension of the state matrix.

9.3.1 Experiment settings

For the simulation to be closer to a real use scenario it is set up so the cube is starting from an offset angle and is to be brought to its equilibrium orientation. The simulation time was set to 2 seconds. It was always verified that the controller managed to bring the states to zero properly. A controller sampling time $T_s = 16\ ms$ was used meaning that 125 iterations were averaged for each $i_{avg,m}$
The stopping criterion used for the algorithm under simulation is when the infinity norm of the dynamic constrain formulated in equation (3.54) is smaller then a threshold, i.e.:

$$||D\boldsymbol{w}_i - \boldsymbol{d}||_\infty \leq 10^{-5} \tag{9.3}$$

As stated in Section 3.10.2.2 it is an issue finding the Lipschitz constant L when using the approximated preconditioner $\hat{P}$. This was therefore set experimentally to values $1 \leq L \leq 3.5$ until the fastest convergence was found. In [3] $\hat{P}$ is rescaled by$\sqrt{L}$. This was tried but seemed to cause the algorithm to never converged so this was omitted on an experimental basis.

9.3.2 Two dimensional case

In the two dimensional case, i.e. during edge balancing, the linear state space has two states. Table 9.1 summarizes some metrics for evaluating different m-banded pre-conditioners.

m	L	$i_{avg,m}$	c_m	θ_m
Full	1	5.08	11400	1
50	1	7.62	11224	1.48
40	1.3	8.43	10464	1.52
30	1.5	9.67	9104	1.52
20	1.8	13.30	7144	1.63
10	2.3	21.75	4584	1.72
5	2.65	42.00	3076	2.23
1	3	64.66	1764	1.97
0	3	62.82	1304	1.41

Table 9.1: Results for different m-banded approximations of the preconditioner P for edge balancing. A $\theta_m < 1$ means fewer floating point operations where needed than for the full preconditioner

Investigating the table shows that a smaller banded m result in less floating point operations c_m required for one iteration of the algorithm. However, it increases the number of iterations $i_{avg,m}$ needed before the convergence of the algorithm, resulting in the end that using the full preconditioner matrix gives the lowest computational cost in the context.

9.3.3 Three dimensional case

For the 3D case there were six states in the linearized state space. Approximating the pre-conditioner in the three dimensional case we can see from Table 9.2 that the computational cost is improved with a more narrow m-banded matrix.

m	L	$i_{avg,m}$	c_m	θ_m
Full	1	19.75	67200	1
150	1	20.19	66016	1.01
120	1	20.94	61296	0.97
90	1	21.22	52976	0.85
60	1.59	27.09	41056	0.84
30	2.6	49.38	25536	0.95
15	2.65	76.62	16412	0.95
0	3.5	146.20	5460	0.6

Table 9.2: Results for different m-banded approximations of the preconditioner P for corner balancing. A $\theta_m < 1$ means fewer floating point operations where needed than for the full preconditioner

Thus in this case, the approximated preconditioner yields a benefit.For the 0 banding the number of floating point operations are reduced by 40 % which could mean a reduction of execution time of almost the same magnitude in some cases.

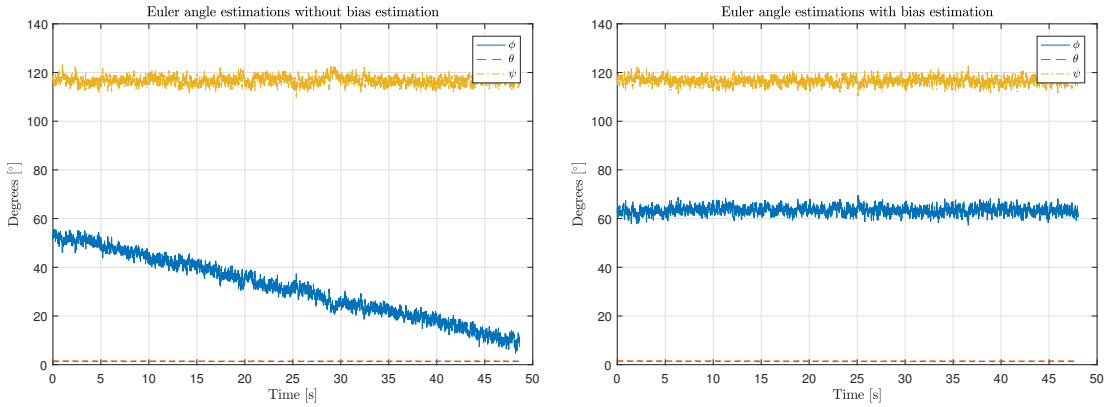
9.4 Orientation Estimation

Accurate estimates of the orientation with low latency might be one of the, if not most important aspects for properly operate a system as Cubex. Section 8 gives details about how this was carried out in this project.

The cube uses two IMU to measure its angular rate and linear acceleration to estimate the orientation using a sensor fusion algorithm, Madgwick filter, which is further described in Section 8.1.

9.4.1 Startup bias estimator

Angular drift during the estimation of orientation was an issue due to to the angular velocities being affected by a bias. A simple bias estimator is described Section 8.2.2 where the bias is estimated by averaging many samples of the gyroscope during startup when the cube is assumed to be still.



(a) Without **(b)** With

Figure 9.10: Output of the orientation estimates from the Magdwick sensor fusion algorithm when the cube was still, so orientation estimates are constant according to ground truth. This with and without subtracting the initial gyroscopic bias estimate $\hat{\boldsymbol{b}}_0$

This estimate of the bias $\hat{\boldsymbol{b}}_0$ was then substracted from all gyroscopic measurements before being fed to the Madgwick sensor fusion algorithm for orientation estimation that was used. This is described in Section 8.1. This simple approach did substantially reduce the angular drift in the output from the Madgwick algorithm and can be seen in 9.10.

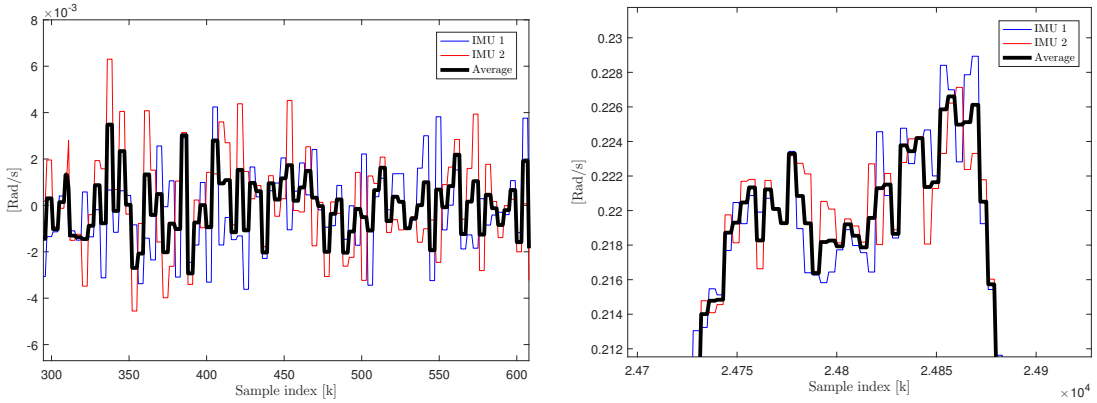
9.4.2 Noise reduction via averaging

Using weighted average to reduce the random noise from the IMU's improves the estimation of the orientation further, disturbed in Section 8.4. For calculating the variance and expected values, equations A.4 and A.8 have to be approximated using a summation. This should be done for data where $\bar{x}(t)$ is constant, else real variations in the measured signal cannot be distinguished from noise. Data in the sample range 1-20000 was used when the cube was laying still, making an angular velocity of 0 a ground truth. Results are shown in Table 9.3

	IMU 1	IMU 2	$(a_1 = a_2 = 0.5)$	$a_1 = 0.54, a_2 = 0.46$
Variance $[\mu rad^2/s^2]$	4.34	5.14	2.48	2.47
Covariance $[\mu rad^2/s^2]$	0.22	-	-	-
Expected value$[\mu rad/s]$	-276	209	-33	-34

Table 9.3: Variance and expected value for some samples 1-20000 in Figure 9.11. Note the μ prefix.

The variance, which is basically the noise amplitude, has reduced significantly for the average with equal coefficients $a_1 = a_2 = 0.5$. This is then compared to a weighted average according to (A.13) and (A.15). As the noise covariances are rather similar, gains are marginal. It turns out the variance shrunk 0.72%



(a) Cube laying still **(b)** Cube balancing

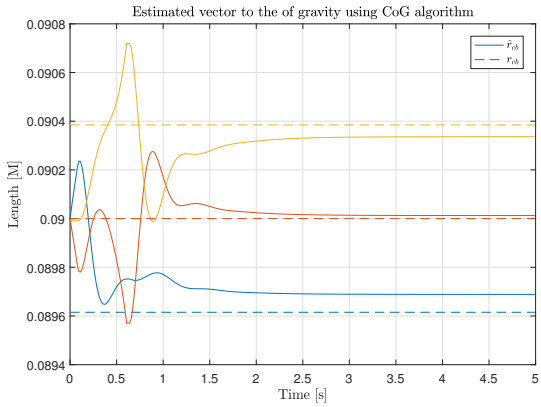
Figure 9.11: Gyroscopic data from two IMU units on the cube first corrected by rotation matrices and then averaged then averaged

Some qualitative analysis can be done by visual inspection of Figure 9.11. Here data from two IMU's mounted in the cube frame are compared. Both IMU's are mounted physically at different orientations and positions but the collected data was first corrected using the rotation matrices described in Section 8.3. It can be seen that the averaging neatly seems to cancels out random noise. Also, it seems that the concept using rotation matrices work really well.

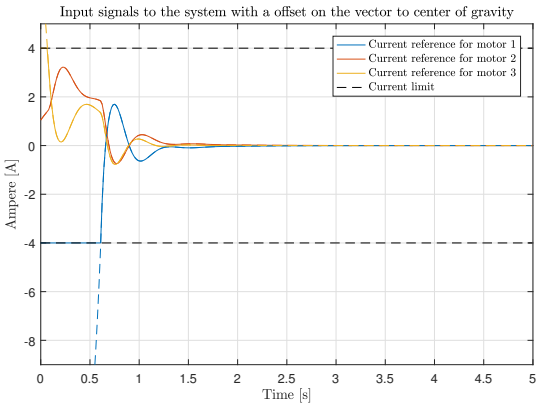
9.5 Offset estimation

This section presents results for estimating the vector to the center of gravity of the cube. Experience obtained during practical experiments showed that Cubex cannot balance properly without a estimating this offset and accounting for it. Further can be read in Section 7.

The development of these algorithms is still ongoing. Convergence is not proved for any of them. Data from practical experiments are influenced by minor oscillations and noise. Therefore simulation data is perhaps what is most informative today. Simulation was carried out by adding an offset vector $\Delta \boldsymbol{r}$ to the nominal position of the center of gravity.



(a) Estimated vector $\hat{\boldsymbol{r}}_{cb}$ and the true vector $\boldsymbol{r}_{cb}$ to the center of gravity



(b) Input signals to the system estimating $\hat{\boldsymbol{r}}_{cb}$

Figure 9.12: Estimating the vector to center of gravity, $\hat{\boldsymbol{r}}_{cb}$, in simulation using the *Orthogonal Approach*

Figure 9.12 show result of estimating the vector $\hat{\boldsymbol{r}}_{cb}$ to the center of gravity using the Orthogonal Approach as described in Section 7.2.2. Most importantly, the algorithm manages to steer the system to a position where no input signal is needed which is the exact purpose of it. Then it also almost manages to find the actual real location of the center of gravity as is seen in Figure 9.12a . However, this is of secondary importance as the orientation where no net torque input is needed is considered more important.

10
Discussion

A general discussion on the topics treated in the report, as well as on involved work methods.

10.1 Fast approximate inverse MPC

An approach for fast approximate inverse MPC was evaluated. This is covered in Section 3.10.2, 6.3 and 9. Although the approximation did not always improve the potential computational complexity, it was seen in Section 9.3 that it potentially could reduce the numbers of floating point operations by 40% on average for a real use scenario. That is off course very promising. However, as stated in Section 3.10.2.2 finding the Lipschitz constant is an issue. Further research should focus on a systematic way of finding this. Then, the reduction of floating point operations is just a potential reduction. Care has to be taken during implementation by possibly using sparse matrices and such for the potential to be fully exploited. It was seen that the feasibility for this approximation to a large part is governed by having most of the elements in th the preconditioner centered around the diagonal. Perhaps it is possible to find the relations that governs the structure of the preconditioner matrix. By doing such, measures could be taken to obtain a preconditioner matrix that has a desired form.

10.2 Software development

In our experience and opinion, the `Waijung` blockset together with `Simulink` greatly facilitated and sped up software development as to compared if "pure" `C/C++` had been used. We have developed several control systems using `C/C++` and never succeeded in doing so many successful software iterations as when using automatically generated code from `Simulink`. Something that we felt was really beneficial was that as simulations were carried out in `Simulink` too, much of the software developed there could be used directly in the real system. Developing a proper simulation environment off course took some effort. However, having one was great for verifying concepts and ideas before trying it on the real system.
We found no easy approach to find a way to store and work with sparse matrices in code generation. It seems `Simulink` does not directly support this. When attempting to run the corner balancing, were the matrices are quiet large if not stored as sparse, it could not be loaded to the real cube due to lack of memory according to the compiler. If the matrices could be stored as sparse this should not be an issue.

Especially as the large preconditioner matrix P is lower triangle and thus half zeros, even when not approximated. It seems that the `Simulink` code generator did not exploit this at all. As far as we know it is not even possible to declare variables as constants, as compared to in `C/C++`. This issue has to be investigated for future successful implementations in this and similar contexts.

10.3 Analytical modeling

Two approaches were tried for modeling the system, namely Lagrangian mechanics on scalar form and Kane's method on matrix form. Kane's method proved to yield substantially more compact and comprehensible equations of motion. However, to arrive at the final equations of motions many manual reduction steps had to be taken. Ultimately, this should be possible to do in software. It would be beneficial to find or develop a software that can identify aspects relevant in the context such as inertia matrices, cross products, gyroscopic effects and such. After all, these usually take a certain structure that could be possible to find by some sort of pattern recognition. It is hard to find straightforward information on how to apply Kane's method we think. Our hope is that our literature review summarized in Section 3.6 will be of help for anybody wanting to apply Kane's method.

10.3.1 Choice of Euler angle conventions

With the lead of the spinning top commonly being analyzed using $(Z - X' - Z'')$ or $(Z - Y' - Z'')$ the Euler Angles the $(Z - X' - Z'')$ was adopted for this project. A benefit of these sequences is that the potential energy of the system can be described using just one of the three Euler angles in the sequence provided a particular choice of body coordinates. However, body coordinates were used that do not exploit this property. In the retrospective it seems far more resources both in litterature and software libraries are available for the (Z, Y', X'') sequence. Therefore, the (Z, Y', X'') convention might be a more convenient choice for this platform if Euler Angles are chosen for orientation.

10.4 Documentation

A significant proportion of most master thesis works worldwide are spent creating documentation. It therefore deserves a few notes.

10.4.1 Online editors need improvements

Seeing the headline, you might fear that this while be a cavalcade of whining and complaints. However, probably anybody using Latex can identify with much what will be mentioned here. Most people probably got used to all issues already. We hope this cry out in open air will be heard by the right person, as to spark a development on how academic writing is carried out in this field. Employers could save money

and researchers could save time while being able to contribute to science in a way more pleasant.

As publication of this kind writing was carried out in a collaborative online implementation of Latex. We don't think graphic editors such as Microsoft Word, LibreOffice and such has the capability and power to neatly represent code, algorithms and equations. We tried the collaborative online version of Word and it was full of bugs and was discarded within a few hours. Latex is a great creation in itself, and there doesn't seem to be another way to really get as good control over documents with many equations. However, our opinion is that Latex in in its current development stage and implementation is a horribly inefficient way of creating documents. All writing is carried out in collaboration, so all offline editors implementing Latex are off the table. Having 20 documents in a folder called `our_report_new22_better` is just not what a sane researcher should want to do in the 21st century. For the record we had to google how to write the underscores you just saw, and that silly too. Also, there version control options as git is just to much of a hassle. While we used git with an online repository for all software development during the project, we did not want to go there with the writing too. We have seen how neat WYSIQYG collaboration is done in for instance Google Docs and we wanted something similar. We used the online platform Overleaf. It has a WYSIWYG editor in beta version that greatly helps, and is a great initiative, but it is not enough. First off, their no really good way to organize and keep track of references. All tags for equations and such are written completely freely, not encouraging any of all good practices with namespace, classed and such we have learned from object orientated programming the last 50 years. Obviously, all references should at least be classified globally as equations, figures and such. Then there should be a better way to easily find references. When wanting to reference for instance a figure or equation, there should be a pop up or something where thumbnails of all figures equations appear. In Overleaf there it at least an autocomplete function, but that doesn't help so much when there are 50+ pages flooding of equations.

Many times Latex documents are created in pure code, which means that valuable times has to be spent on finding where that comma or underscore did go or not. Tables are a pure hassle where what should normally require a control command from a calculator document can set you back an hour in the worst case scenario. On top of this, a lot of the creation, or all, has to be done in code. Stuck deep into code, overview is easily lost and compilation has to be done frequently as to not loose the layout. Code. Compiling. Compilation errors. What should be writing in the 21th century appears to be some demonstration of some command terminal for a room big computer 1965. T Researchers, departments and employers all over the world loose ought to loose ridiculous amounts of time on the current quirky way Latex is implemented and used. There where solutions for this for HTML during homepage creations 30 years ago and it sensational there is no such development rocketing for Latex yet. A mediocre editors already had a decent split interface between code and graphical in the 90's, and it so strange this hasn't been brought to any online Latex platform yet.

10.4.2 Figures

It is hard and takes a lot of time to generate figures that are consistent and informative. For drawing vectors and such we first tried IPE and LatexDraw. While these have the great feature of integrating Latex writing with drawing we felt these software needs some more development. In the end we created most drawings in `Microsoft Visio` which was a decent way to create drawing we felt were aesthically enough pleasant. Plots were exported from Matlab in eps format, to have them represented as zoomable vector graphics. All drawings and pictures were exported as pdf whenever applicable to have these as vector graphics too.
The schematics seen in the report was done in the online `Scheme-It` from `Digikey`. We believe that was a good way to create an aesthetically pleasant schematics in a short period of time.

10.5 Hardware

The current motors and corresponding drivers used are fairly expensive in relation to their output of torque and power. Recently motors suitable for camera gimbals started to become easily available and these are much cheaper than the current motors used. This could be something to look into for anybody wanting to build a Cubex or Cubli platform. Also, perhaps it is possible to find a motor that has such a high torque that the installed brakes, that are used during jump up would not be needed.

10.6 Future work

The Cubex platform is now ready to be used as a benchmarking and evaluation platform in control systems research. During the time of writing it was a discussed that a new apporach to data driven controller was to be evaluated on Cubex. However, a few aspects regarding the platform should probably be investigated in parallell. It is seen that the orientation estimates are affected by noise to a degree that does not seem neglible. Also, other approaches to finding the offset of the mass center may be tried.
Glancing at the Cubli project, it is seen that this platform is capable of performing multiple feats such as jumping up from a resting position. These feats may appear to be rather useless other then for showing off. However, devising and implementing such feats usually spurs a lot of research results that may have direct applications in the field of mechatronics. We thus believe that continuing to develop such feats is benficial for the field. A natural next step is probably then implementing the jump up procedure that Cubli can do to verify that the hardware and sensoring works properly during such transient events. However, the theory and the ideas behind this seems to be investigated rather throughly by ETH so when this is done another feat should be devised to further push development. Perhaps rolling from corner to corner would be a start. It would be interesting to investigate the general properties of the spinning top in junction with the gyroscopic effects of the wheels and see if

this can be used to generate some interesting movement patterns.

11
Conclusion

An optimization solver for Model Predictive control was tried, which seems to outperform Matlabs quadprog with respect to execution time. According to a suggestion in [3], a further possible enhancement was tried. Although some preliminary results have proven successful, more research needs to be done.
Several auxiliary aspects that can be helpful for researchers interested in modeling, simulate and control Cubex or a similar platform are presented.
A developed approach for correcting misaligned Inertial Measurement Units using rotation matrices was shown to work well.
A working approach for finding the offset of the center of gravity has been presented although. However, general convergence has not yet been proven analytically. .

Bibliography

[1] T. W. Mohanarajah Gajamohan, Michael Muehlebach and R. D'Andrea, "The cubli: A reaction wheel based 3d inverted pendulum," *European Control Conference*, pp. 268–274, 2013.

[2] B. P. Erik Bjerke, "Development of a nonlinear mechatronic cube - the jumping and balancing cube," 2016.

[3] S. G. Emil Klintberg, "Approximate inverses in preconditioned fast dual gradient methods for mpc," 2017.

[4] L. V. G. O. Bruno Siciliano, Lorenzo Sciavicco, *Robotics, Modelling, Planning, Control*. Springer, 2010.

[5] W. A. W. S. G. Dietmar Gross, Werner Hauger Jörg Schröder, *Engineering Mechanics 3*. Springer, 2011.

[6] D. A. L. Thomas R. Kane, *Dynamics, Theory and Application*. Mc Graw Hill, 1 ed., 1985.

[7] F. Amirouche, *Fundamentals of Multibody Dynamics Theory and Applications*. Birkhauser, 1 ed., 2006.

[8] S. W. J. Peraire, "Lecture l26 - 3d rigid body dynamics: The inertia tensor," 2008.

[9] D. t. H. Thomas Tallot Taylor, *Mechanics Classical and Quantum*. Pergamon Press Inc., 1 ed., 1976.

[10] K. Vandiver, "15. introduction to lagrange with examples," 2013.

[11] F. H. S. Wei Zhuang, Xiaoping Liu Chenwei, "Dynamic modeling of a spherical robot with arms by using kane's method," 2002. IEEE International Conference on Industrial Technology.

[12] J.-C. Piedbeuf, "A quick tutorial on multibody dynamics,"

[13] M. Lesser, "A geometrical interpretation of kane's equations," vol. 436, p. 69, 1992.

[14] E. T. Stoneking, "Implementation of kane's method for a spacecraft composed of multiple rigid bodies," 2013.

[15] R. B. Gillespie, "Kane's equations for haptic display of multibody systems," vol. 3, no. 2, 2003.

[16] W. Blajer, "A useful matrix form of kanes equations," *Mechanics Research Communications*, vol. 17, no. 5, pp. 311–318, 1990.

[17] R. D. Michael Muehlebach, "The cubli: A cube that can jump up and balance," 2012.

[18] F. Pfeiffer, *Multibody Dynamics with Unilateral Contacts*. WILEY-VCH Verlag, 2004.

[19] Z. Hussain, "Kane's method for dynamic modeling," 2016.

[20] J. A Purushotham, "Kane's method for robotic arm dynamics: a novel approach," 2013.
[21] R. T. D. Luca, "A brief synopsis of kane's method," 1999.
[22] M. Raginsky, "Linearization (university of illinois, lecture notes course ece486," 2017.
[23] D. E. S. Michael A Henson, *Nonlinear Process Control*. Prentice Hall, 1996.
[24] A. Packard, "Jacobian linearizations, equilibrium points (me 132 lecture notes)," 2005.
[25] R. M. M. Karl J. Åström, *Feedback Systems*. Princeton University Press, 2 ed., 2012.
[26] B. Egardt, *Model Predictive Control, Lecture Notes*. Chalmers University of Technology, Department of Signals and Systems, 2016.
[27] D. Q. M. James B. Rawlings, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 1 ed., 2013.
[28] S. B. Pontus Giselsson, "Metric selection in fast dual forward backward splitting," 2016.
[29] S. B. Pontus Giselsson, "Preconditioning in fast dual gradient methods," 2014.
[30] C. C. Tisdell, "What is a lipshitz condition," 2012.
[31] R. V. Sebastian O.H. Madgwick, Andrew J.L. Harrison, "Estimation of imu and marg orientation using a gradient descent algorithm," 2011.
[32] J. Awrejcewicz, *Classical Mechanics Kinematics and Statics*. Springer, 1 ed., 2012.
[33] T. L. Chow, *Classical Mechanics*. Taylor and Francis group, 2 ed., 2010.
[34] J. S. Herbert Goldstein, Charles Poole, *Classical Mechanics*. Addison Wesley, 3 ed., 2000.
[35] S. W. J. Peraire, "Lecture l30 - 3d rigid body dynamics: Tops and gyroscopes," 2008.
[36] G. M. Michael Muehlebach and R. D'Andrea, "Nonlinear analysis and control of a reaction wheel-based 3d inverted pendulum," 2012.
[37] T. W. Mohanarajah Gajamohan, Michael Muehlebach and R. D'Andrea, "The cubli: A reaction wheel based 3d inverted pendulum [appendix related to modeling]," *European Control Conference*, pp. 268–274, 2013.
[38] A. Shiriaev, "Control methods for robotics applications [course notes lec. 10]," 2011.
[39] A. Neumaier, "Classical and quantum mechanics via lie algebras," 2011.
[40] J. Awrejcewicz, *Classical Mechanics Dynamics*. Springer, 1 ed., 2012.
[41] D. Hestenes, *New Foundations for Classical Mechanics*. KLUWER ACADEMIC PUBLISHERS, 2 ed., 1999.
[42] C. K. L. S. Jain, "A quick tutorial on multibody dynamics,"
[43] M. L. N. H. M. Nalin A. Chaturvedi, Taeyoung Lee, "Nonlinear dynamics of the 3d pendulum," *Journal of non-linear science*, vol. 17, no. 5, pp. 3–32, 2011.
[44] T. W. R. D. Mohanarajah Gajamohan, Michael Muehlebach, "The cubli: A reaction wheel based 3d inverted pendulum," 2013.
[45] M. Motors, "Escon 36/3 ec servo controller p/n 414533 hardware reference," 2014.
[46] A. Hughes, *Electric Motors and Drives*. Newsnes, 3 ed., 2006.

[47] M. Motors, "Side from product catalog for the ec45 70w/24 volt motor," 2014.
[48] V. Ganapathi, "Simulation of rigid body dynamics in matlab," 2005.
[49] R. Stengel, "Aircraft equations of motion: Flight path computation (princeton lecture notes)," 2016.
[50] D. R. P. R. B. M. Joseph M. Cooke, Michael J. Zyda, "Npsnet: Flight simulation dynamic modeling using quaternions,"
[51] M. Pürschel, "Automotive mosfets reverse battery protection [application note, 2.0]," 2009.
[52] T. E. A. for Advanced Rechargeable Batteries, "Exploding headphones reignite fears about batteries on planes," 2017.
[53] K. W. R. T. L. Celina Mikolajczak, Michael Kahn, "Lithium-ion batteries hazard and use assessment," 2011.
[54] A. J. Peter Keil, "Aging of lithium-ion batteries in electric vehicles: Impact of regenerative braking," 2015.
[55] A. Co, "Waijung blockset," 2017.
[56] S. Madgwick, "Madgwick library homepage."
[57] O. J. Woodman, "An introduction to inertial navigation," 2007.
[58] M. Looney, "The basics of mems imu/gyroscope alignment," 6 2015.
[59] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 3 ed., 2013.

A

Optimal fusion of multiple measurements

It is shown how noise can be reduces by averaging multiple sensors.
A way to model the sensor readings from the IMU at some time instant t is

$$x(t) = \bar{x}(t) + w(t) \tag{A.1}$$

where $w(t)$ is zero mean Gaussian normally distributed noise

$$w(t) \sim N(0, \sigma^2) \tag{A.2}$$

Note that σ^2 is the variance of the noise, that is a metric describing to the expected amplitude of it. According to [59], the variable $x(t)$ have a probability function as

$$p(x) = N(x; \bar{x}, \sigma^2) \tag{A.3}$$

The expected value of this non- deterministic signal is

$$E(x) = \int_{\infty}^{\infty} xp(x)dx = \bar{x} \tag{A.4}$$

where time index has been dropped. In this context $\bar{x}$ should be thought of as the true value we want to measure. This could for instance be the angular velocity. Say we have two sensors yielding measurements x_1 and x_2 that are averaged, as described above. This average then has the expected value

$$E(\frac{x_1 + x_2}{2}) = \frac{1}{2}\left(E(x_1) + E(x_2)\right) = \frac{\bar{x_1} + \bar{x_2}}{2} \tag{A.5}$$

If both measurements are similar so $\bar{x_1} = \bar{x_2} = \bar{x}$ the averaged measurement will have the same expected value as the two measurements individually.
Variance and covariance are defined as

$$\text{Var}(x) = E\left((x - \bar{x})^2\right) \tag{A.6}$$

$$\text{Cov}(x, y) = E\left((x - \bar{x})(y - \bar{y})\right) \tag{A.7}$$

For the more general case, let's use a weighted average $X_{avg} = a_1x_1 + a_2x_2$ where $a_1 + a_2 = 1$. Taking the covariance for the averaged signal, which is basically a metric for the amplitude of the noise, yields

$$\begin{aligned}
\mathrm{Var}(a_1x_1 + a_2x_2) =& E\left(\left[a_1x_1 + a_2x_2 - E(a_1x_1 + a_2x_2)\right]^2\right) \\
=& E\left(\left[a_1x_1 + a_2x_2 - a_1\bar{x_1} + a_2\bar{x_2})\right]^2\right) \\
=& E\left(\left[a_1(x_1 - \bar{x_1}) + a_2(x_2 - \bar{x_2}))\right]^2\right) \\
=& E\left(a_1^2(x_1 - \bar{x_1})^2 + 2a_1a_2(x_2 - \bar{x_2})(x_1 - \bar{x_1}) + a_2^2(x_2 - \bar{x_2})^2\right) \\
=& a_1^2\mathrm{Var}(x_1) + a_2^2\mathrm{Var}(x_2) + 2a_1a_2\mathrm{Cov}(x_1, x_2) \qquad \text{(A.8)}
\end{aligned}$$

For example, lets assume that both signals are uncorrelated, so the covariance between them is zero and both of them have the variance σ^2 . Then using a regular average so $a_1 = a_2 = 0.5$

$$\mathrm{Var}(X_{avg}) = \frac{1}{4}\left(\mathrm{Var}(x_1) + \mathrm{Var}(x_2)\right) = \frac{1}{2}\sigma^2 \qquad \text{(A.9)}$$

The noise variance of the averaged signal is then half as large. Is that optimal though? For a more general case with different noise variances $\sigma_{x1}, \sigma_{x_2}$ let

$$\begin{aligned}
a_1 =& 1 - \alpha \qquad \text{(A.10)} \\
a_2 =& \alpha \qquad \text{(A.11)} \\
& \alpha \in (0, 1) \qquad \text{(A.12)}
\end{aligned}$$

Inserting this into A.8, assuming zero covariance and taking the partial derivative w.r.t to α equals to zero yields

$$\frac{\partial\left(\mathrm{Var}(X_{avg})\right)}{\partial\alpha} = 2\alpha(\sigma_{x1}^2 + \sigma_{x2}^2) - 2\sigma_{x1}^2 = 0 \qquad \text{(A.13)}$$

Solving for α yields

$$\alpha = a_2 = \frac{\sigma_{x1}^2}{\sigma_{x1}^2 + \sigma_{x2}^2} \qquad \text{(A.14)}$$

$$a_1 = \frac{\sigma_{x2}^2}{\sigma_{x1}^2 + \sigma_{x2}^2} \qquad \text{(A.15)}$$

which is the α that minimizes the variance of the averaged signal as the second partial derivative w.r.t. α of (A.13) is positive given the constraints on α . This means, that for two signal having an equal noise magnitude an average using $a_1 = a_2 = 0.5$ is actually optimal.

A.1 Generalization to multiple multiple sensors

Consider a weighted average for n sensors according to $X_{avg} = a_1x_1 + a_2x_2 + \dots a_nx_n$ The combined variance, according to A.8 will be

$$\sigma^2 = \mathrm{Var}(X_{avg}) = \sum_{i=1}^{n} a_i^2\sigma_i^2 \tag{A.16}$$

if the covariance between each signal is assumed to be 0 .
How can we choose $a = \begin{bmatrix} a_i & \dots & a_n \end{bmatrix}$ such that the total variance σ^2 is minimized? To begin with, we are constrained to

$$\sum_{i=1}^{n} a_i = 1 \tag{A.17}$$

in order for the expected value to be unchanged according to

$$E(X_{avg}) = E\left(\sum a_i x_i\right) = \bar{x}E\left(\sum a_i\right) = \bar{x}\sum a_i \tag{A.18}$$

where $\bar{x}$ is the assumed common true value for each sensor.
By using Lagrangian optimization to minimize the combined variance a Lagrangian can be formulated

$$\mathcal{L}(a,\lambda) = \sum_{i=1}^{n} a_i^2\sigma_i^2 - \lambda C(a) \tag{A.19}$$

$$C(a) = \sum_{i=1}^{n} a_i - 1 \tag{A.20}$$

The optimum is found by setting the gradient of the Lagrangian according to zero, $\nabla\mathcal{L} = 0$, which yields

$$\begin{aligned} 2a_1\sigma_1^2 - \lambda &= 0 \\ 2a_2\sigma_2^2 - \lambda &= 0 \\ &\vdots \qquad (A.21) \\ C(a) &= 0 \qquad (A.22) \end{aligned}$$

Letting $x = \begin{bmatrix} a & \lambda \end{bmatrix}^T$ and formulating (A.22) on matrix form yields

$$\underbrace{\begin{bmatrix} 2\sigma_1^2 & 0 & \cdots & -1 \\ 0 & \ddots & \cdots & -1 \\ \vdots & & 2\sigma_n^2 & -1 \\ 1 & 1 & \cdots & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_1 \\ \vdots \\ a_n \\ \lambda \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{b} \tag{A.23}$$

Here x can be solved for via matrix inversion yielding $x = A^{-1}b$. It turns out that each solution i is found by

$$\begin{aligned} a_i =& \frac{N_i}{D} \\ N_i =& \prod_{k=1}^{k=n} \sigma_k^2, \quad k \neq i \\ D =& \sum_{k=1}^{k=n} \left(\prod_{j=1}^{j=n} \sigma_j^2, \quad j \neq k \right) \end{aligned} \tag{A.24}$$

Note how each coefficient a_i is normalized by a common denominator D. The Lagrangian multiplier can be found by multiplying any of the coefficients by its related noise variance according to

$$\lambda = 2a_i\sigma_i^2 \tag{A.25}$$

For instance, with 3 sensors meanint that $n = 3$, the weighting factor for sensor 1 will be

$$a_1 = \frac{\sigma_2^2\sigma_3^2}{\sigma_2^2\sigma_3^2 + \sigma_1^2\sigma_3^2 + \sigma_1^2\sigma_2^2} \tag{A.26}$$

Note how (A.24) converges too) for $n = 2$. Also note that for equal noise variances the coefficients converge to $a_i = \frac{1}{n}$ that will yield a non-weighted average.

A.2 Proposed algorithm for fusion of gyroscopes

This is an algorithm that was devised but never tried. It uses the result regarding optimal averaging for estimating noise variance during the startup of the system when the noise variance of the sensors differs. At the same time an estimate for the gyroscopic bias is calculated.
Say that two sensors are to be fused were one has twice as high noise variance as the other according to
$\sigma_{x1}^2 = \sigma^2, \sigma_{x2}^2 = 2\sigma^2$
In practice this could probably happen if one of the sensors is mounted closer to a noise source or so. This yields $a_1 = 2/3$ and $a_2 = 1/3$ according to (A.24) Comparing the noise variances for the weighted average and the non-weighted average can be done by division according to

$$\frac{\sigma_{avg,weighted}^2}{\sigma_{avg}^2} = \frac{a_1^2\sigma^2 + a_2^2\sigma^2}{\frac{1}{4}(\sigma^2 + 2\sigma^2)} = \frac{\frac{6}{9}}{\frac{3}{4}} = \frac{24}{27} \tag{A.27}$$

The optimal average yields a noise variance that is $3/27 \approx 11\%$ smaller ithan for a non-weighted average which could be a considerable gain for just choosing two coefficients right.

The suggestion is to collect N sample values when the cube is laying still at startup and use this to estimate bias and noise variance.
Beginning, noise variances can be calculated by

$$\sigma = \frac{1}{N}\sum_{k=1}^{N}(x_k - \bar{x})^2 \tag{A.28}$$

which in its case requires an average $\bar{x}$ over the N measurements. Apart from that, the gyroscopes are known to have biases b It would be possible to first sample some data M data points, and approximate the bias as

$$\hat{b} = \hat{\bar{x}} = \frac{1}{M}\sum_{k=1}^{M} x_k \tag{A.29}$$

then this value could be inserted into (A.28). However, to make better use of the sampling time it is suggested that a running average is calculated and used instead, so both the mean and variance can be approximated at once. Beginning by splitting the sum

$$\hat{\bar{x}}_k = \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{1}{k}\sum_{i=1}^{k-1} x_i + \frac{x_k}{k} \tag{A.30}$$

At the time instant k, the last estimate of the average would be available according to

$$\hat{\bar{x}}_{k-1} = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i \tag{A.31}$$

Multiplying A.31 by $\frac{k-1}{k}$ yields the summation on the right hand side of (A.30). Therefore, a running average equal to (A.30) can be updated recursively at each time step k according to

$$\hat{\bar{x}}_k = \frac{k-1}{k}\hat{\bar{x}}_k + \frac{1}{k}x_k \tag{A.32}$$

Therefore its suggested that from the startup at time index $k = 1$ until $k = N$ a summation

$$\Theta = \sum_{k=1}^{N}\left(x_k - \hat{\bar{x}}_k\right)^2 \tag{A.33}$$

is collected. Then at $k = N$ the bias estimate and approximately optimal weighting factor alpha are found as

$$\alpha = \frac{\frac{\Theta_1}{N}}{\frac{\Theta_1}{N} + \frac{\Theta_2}{N}} = \frac{\Theta_1}{\Theta_1 + \Theta_2} \tag{A.34}$$

$$\tag{A.35}$$

The bias estimate is then taken as the last value of the running average.

$$\hat{b} = \hat{\bar{x}}_N \tag{A.36}$$

That is because the more samples used for the average, the closer the estimated bias should be to the real one according to the law of large numbers. For an arbitrary numbers of sensors, the algorithm can be generalized using (A.24).

B

Program listings

B.1 Fast MPC

Benchmarking of Algorithm 1 during a simulation of edge balancing.

```
%% Clear and read

clearvars

%Load the cube parameters
cubeparameters;

clearvars -except  Ts cube motor

[MPC, fMPC, sys_d] = MPC_Parameters(cube, motor, Ts);
%Initial state
x0 = [deg2rad(2)  0]';

% Simulation time
sec = 10;
T = (1/Ts.controller) * sec;



%% Make it to sparse matrices

fMPC.P          =   sparse(double(fMPC.P));
fMPC.LPD        =   sparse(double(fMPC.LPD));
fMPC.LP         =   sparse(double(fMPC.LP));
fMPC.D          =   sparse(double(fMPC.D));
fMPC.miHDtPt    =   sparse(double(fMPC.miHDtPt));
fMPC.dd         =   sparse(double(fMPC.dd));

%% Heat map over P matrix
figure;
set(0,'defaulttextinterpreter','latex')
imagesc(abs(fMPC.P))
colormap(flipud(colormap('gray')))
```

```
34 title('Heat map over preconditioner matrix $P$')
35
36
37 xlab = xlabel('Column');
38 ylab = ylabel('Row');
39 set(xlab,'Interpreter','latex')
40 set(ylab,'Interpreter','latex')
41 %% Simulation
42 opt = mpcqpsolverOptions('single');
43 xk = x0;
44 lam(:,2) = fMPC.P * fMPC.dd * xk;
45 yvec=[];
46 uvec=[];
47 eTimeFASTMPC=[];
48 eTimeMPCSOLVER = [];
49
50 for k = 1 : T
51
52     beq  = single(MPC.AA)*xk;
53     if fMPC.N < 50
54         mpcsol=tic;
55         [z, ~, ~, ~] = mpcqpsolver(single(MPC.Linv), single(
               MPC.f'), single(MPC.Ain), single(MPC.bin),...
56                                    single(MPC.Aeq), single(beq
                                       ), MPC.iA0, opt);
                                             % Solve MPC
57         eTimeMPCSOLVER = [eTimeMPCSOLVER toc(mpcsol)];
58     end
59
60     d = fMPC.dd * xk;
61     sig = sparse(fMPC.LP * d);
62     i = 2;
63     fmpc=tic;
64     while true
65         beta =  (i-3)/i;
66         mu = lam(:,i) + beta*(lam(:,i)-lam(:,i-1));
67         KK = [fMPC.inCo fMPC.miHDtPt*mu];
68         w = double(median(KK,2));
69         lam(:,i+1) = mu + (fMPC.LPD * w) - sig;
70
71         if norm((fMPC.D*w)-d,Inf) <= 1e-5
72             iter(k) = i - 1 ;
73             lam(:,2) = lam(:,i+1);
74             lam(:,3:end) = [];
75             break;
76         end
```

```
77          i = i + 1;
78      end
79      eTimeFASTMPC=[eTimeFASTMPC toc(fmpc)] ;
80
81      uk = w(1);
82      xk=sys_d.A*xk+sys_d.B*uk;          %Update time
83      yvec=[yvec   sys_d.C*xk];          %Save outsignal
84      uvec=[uvec; uk];                   %Save insignal
85  end
```

Algorithm 1 implmented for code generation using `Simulink` during corner balancing. Was deployed in hardware.

```
1   function [T, iter] = FAST_MPC(states, fMPC_3d)
2
3       % Algorithm 1 Fast dual proximal gradient method
4       % MATLAB code for corner balancing
5
6       persistent lamda lamda_old
7
8
9       %--------- Rename for readability
10      % (1/L) * P * D
11      LPD          =    fMPC_3d.LPD;
12
13      %(1/L) * P
14      LP           =    fMPC_3d.LP;
15
16      % D Matrix in report
17      D            =    fMPC_3d.D;
18
19      % inv(H) * D' * P'
20      miHDtPt      =    fMPC_3d.miHDtPt;
21
22      % d vector in report before state multiplication
23      dd           =    fMPC_3d.dd;
24
25      % [lower w constrain , upper w constrain]
26      inCo         =    double(fMPC_3d.inCo);
27
28      % Number of states
29      nx           =    double(fMPC_3d.nx);
30
31      % Prediction horizon
32      N            =    double(fMPC_3d.N);
33
34       % Absolute stop condition
35      num_of_iter=    500;
```

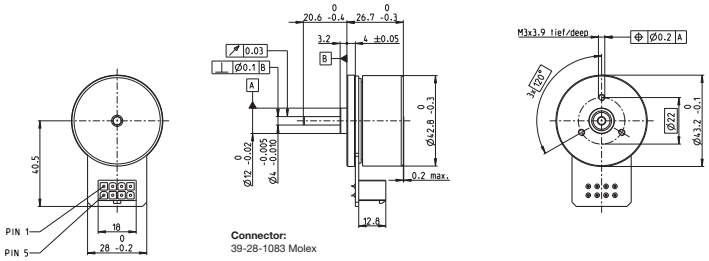
```
36
37
38      % Init one time
39      if isempty(lamda)
40          row_of_lamda = N*nx;
41          assert(row_of_lamda < 4096)
42          lamda = zeros(row_of_lamda,1);
43          lamda_old = zeros(row_of_lamda,1);
44      end
45
46      w = [];
47
48      % d vector from equality constrain
49      d = dd * states;
50
51      % Matrix-vector multiplication here instead in the
            algorithm
52      sig = LP * d;
53
54      % Init index
55      k = 0;
56
57
58      % Solve optimization problem
59      % Use a absolute stop condition to avoid a deadlock
            state
60      % of the algorithm
61      while k < num_of_iter
62
63          beta = (k-1)/(k+2);
64
65          mu = lamda+ beta*(lamda-lamda_old);
66
67          % Shift time index for lamda
68          lamda_old = lamda;
69
70          % Create the inequalty constrain
71          KK = [inCo miHDtPt*mu];
72
73          % Median function
74          w = median(KK,2);
75
76          % Calculating new lamda
77          lamda = mu + (LPD * w) - sig;
78
79          % Stop condtion
```

```
80            if norm((D*w)-d,Inf) <= 1e-5
81                % Save lamda for next iteration
82                lamda_old = lamda;
83                break;
84            end
85
86            k = k + 1;
87        end
88
89        % Amount of iteration of algorithm
90        iter = k;
91
92        % Get optimal solution from w vector to input vector T
93        T = [w(1); w(2); w(3)];
94    end
```

C

Motor data sheet

EC 45 flat ∅42.8 mm, brushless, 70 Watt



Connector:
39-28-1083 Molex

M 1:2

Stock program
Standard program
Special program (on request)

Part Numbers

with Hall sensors	397172	402685	402686	402687
Motor Data (provisional)				
Values at nominal voltage				
1 Nominal voltage V	24	30	36	48
2 No load speed rpm	6110	6230	6330	3440
3 No load current mA	234	194	166	48.1
4 Nominal speed rpm	4860	4990	5080	2540
5 Nominal torque (max. continuous torque) mNm	128	112	108	134
6 Nominal current (max. continuous current) A	3.21	2.36	1.93	0.936
7 Stall torque mNm	1460	1170	1100	915
8 Stall current A	39.5	25.8	20.7	6.97
9 Max. efficiency %	85	84	83	84
Characteristics				
10 Terminal resistance phase to phase Ω	0.608	1.16	1.74	6.89
11 Terminal inductance phase to phase mH	0.463	0.691	0.966	5.85
12 Torque constant mNm / A	36.9	45.1	53.3	131
13 Speed constant rpm / V	259	212	179	72.7
14 Speed / torque gradient rpm / mNm	4.26	5.44	5.85	3.82
15 Mechanical time constant ms	8.07	10.3	11.1	7.24
16 Rotor inertia gcm²	181	181	181	181

Specifications

Thermal data
17 Thermal resistance housing-ambient 3.56 K/W
18 Thermal resistance winding-housing 4.1 K/W
19 Thermal time constant winding 29.6 s
20 Thermal time constant motor 178 s
21 Ambient temperature -40 ... +100°C
22 Max. winding temperature +125°C

Mechanical data (preloaded ball bearings)
23 Max. speed 10000 rpm
24 Axial play at axial load < 4.0 N 0 mm
> 4.0 N 0.14 mm
25 Radial play preloaded
26 Max. axial load (dynamic) 3.8 N
27 Max. force for press fits (static) 50 N
(static, shaft supported) 1000 N
28 Max. radial load, 5 mm from flange 21 N

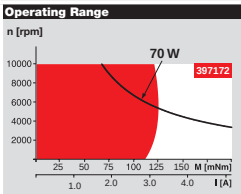
Other specifications
29 Number of pole pairs 8
30 Number of phases 3
31 Weight of motor 141 g

Values listed in the table are nominal.

Connection
Pin 1 Hall sensor 1*
Pin 2 Hall sensor 2*
Pin 3 V_{cc} 4.5 ... 18 VDC
Pin 4 Motor winding 3
Pin 5 Hall sensor 3*
Pin 6 GND
Pin 7 Motor winding 1
Pin 8 Motor winding 2
*Internal pull-up (7 ... 13 kΩ) on pin 3
Wiring diagram for Hall sensors see p. 37

Cable
Connection cable Universal, L = 500 mm **339380**
Connection cable to EPOS, L = 500 mm **354045**

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 20–27

Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 351

Spur Gearhead
∅45 mm
0.5 - 2.0 Nm
Page 353

Encoder MILE
256 - 2048 CPT,
2 channels
Page 379

Recommended Electronics:

Notes	Page **26**
ESCON 36/3 EC	417
ESCON Mod. 50/4 EC-S	417
ESCON Module 50/5	417
ESCON 50/5	418
DEC Module 50/5	420
EPOS2 Module 36/2	424
EPOS2 24/5, 50/5	425
EPOS2 P 24/5	428
EPOS4 Module 50/8	431
EPOS4 Comp. 50/8 CAN	431
MAXPOS 50/5	435

Option
With Cable and Connector
(Ambient temperature -20 ... +100°C)

D

Parameter summary

Numerical values for system parameters.

Variable	Unit
kg	Kilogram
m	Meter
s	Second
rad	Radian
N	Newton
A	Ampere
V	Volt
H	Henry
Ω	Ohm

Table D.1: Acronmyms of SI units relevant for the concerned properties of the system

All inertia parameters were derived using formulas from textbooks for similar bodies. They are likely inaccurate, but the model based control systems derived using them seemed to perform well.

D.1 Cube casing

Parameter	Value	Description
m_c	2.9 kg	Complete cube mass, wheels, batteries and all
r	0.09 m	Half the length of one side
I_{c0}	0.0157 kgm^2	Principle moments of inertia around one axis

Table D.2: Cube casing parameters

D.2 Reaction wheels

For each of three wheels

Parameter	Value	Description
m_w	0.273 kg	Mass
r_{wheel}	0.06 m	Wheel radius
h	0.006 m	Thickness of the reaction wheel
I_{wz}	$0.79607 \cdot 10^{-3}$ kgm^2	Moment of inertia around reaction wheel shaft rotation axis
I_{w0}	$0.24652 \cdot 10^{-3}$ kgm^2	Moment of inertia for reaction wheels other axises

Table D.3: Reaction wheel parameters

D.3 Motors

These are taken from the datasheet for the Maxon EC45 70W flat motor used in the Cubex platform. Resistance and inductance are measured between two connector cables. Values are for one motor.

Parameter	Value	Description
L	0.00463 H	Self-inductance
R	0.6 Ω	Resistance
k_w,	0.0369 Vs/rad	Voltage constant
k_t	0.0369 Nm/A	Torque constant

Table D.4: Motor parameters

The voltage and torque constant are equal as is expected in theory when using SI units.