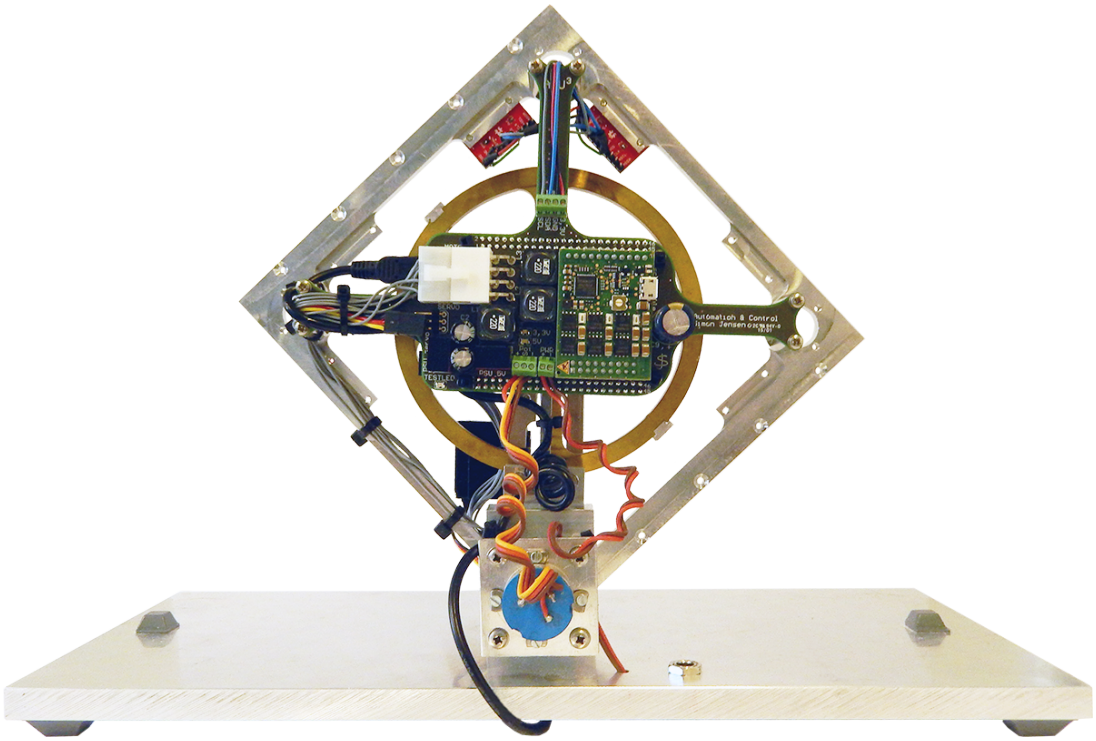
Cubli

Dynamic Control of a Reaction Wheel Inverted Pendulum



6. Semester Project Report

Group 16gr630

Aalborg University
Electronic Engineering & IT
Fredrik Bajers Vej 7
DK-9220 Aalborg

This report is compiled in LaTeX, originally developed by Leslie Lamport, based on Donald Knuth's TeX. The main text is written in *Latin Modern* pt 12, designed by Bogusław Jackowski and Janusz M. Nowacki. Diagrams are made using Inkscape and Tikz.

AALBORG UNIVERSITY
STUDENT REPORT

6th Semester, Bachelor Project

School of Information and Communication Technologies

Electronics and IT

Fredrik Bajers Vej 7C

9220 Aalborg

http://www.sict.aau.dk/electronics-and-it

Title:
Cubli:
Dynamic Control of a Reaction Wheel Inverted Pendulum

Theme:
BSc Project (Control Engineering)

Project Period:
P6, Spring 2016
01/02/2016 - 25/05/2016

Project Group:
630

Participants:
Bjørn Kitz
Julien Bréhin
Mikael Sander
Niels Skov Vestergaard
Noelia Villarmarzo Arruñada

Supervisors:
John-Josef Leth
Palle Andersen

Prints: 8
Pages: 125
Appendices: 15 (37 pages)
Attached: 1 DVD
Concluded: 25/05/2016

Synopsis

The inverted pendulum is used in many applications and it is a classic research area in control theory which is still active.
The aim of this project was to model and analyze the behavior of a reaction wheel inverted pendulum, in the form of a one-frame Cubli; and to design a controller capable of balancing it in equilibrium position.
A controller was designed using classical control methods such as root locus and Nyquist criterion. However, the system had a marginally stable behavior. Not having control of the velocity of the reaction wheel was a problem, so the final controller was done through state space design.
Moreover, a solution for measuring its angular position using only internally mounted sensors was designed to be able to make it portable to a full Cubli. Finally, the performance of the system was tested on the prototype to ensure that it fulfills the needed requirements.

Preface

The purpose of this project is to design and implement a control system that can maintain a reaction wheel inverted pendulum in upright position using only internally mounted sensors.

This report has been written by a group of students on the sixth semester of "Electronics and IT" at Aalborg University, with specialization in Control Engineering. The reader should have a basic knowledge on Electronic Engineering, specially in Modeling and Control Theory, and Convex Optimization, although specific topics will be described in more detail. Code for the implementation is written in `C++` and the reader is assumed to be able to comprehend this programming language.

Special thanks to Simon Jensen (assistant engineer), Simon Vestergaard Johansen (business PhD) and Benjamin Krebs (employee of Cobham SATCOM) for their help along the project. Final thanks to the group supervisors, John-Josef Leth and Palle Andersen (associated professors of the Department of Electronic Systems at Aalborg University), for their guidance during the semester.

Reading Instructions

The report is structured in three parts. Part I contains an analysis of the system, which includes a description of the given setup, the derivation of the dynamic model and the acquisition of the parameters of the system. Part II deals with the design and implementation of the controller and the complementary filter used to measure the angular position of the frame. Part III includes the acceptance tests made to the system and the conclusions that can be derived from the project.

The attached DVD contains a digital copy of this report, all the data and Matlab files needed to plot the figures in the report, data sheets, the code needed to run the controller, the code needed for the implementation of the optimization, the Senstools manual and three videos of the controlled system.

Text by:

Bjørn Kitz

Julien Bréhin

Mikael Sander

Niels Skov Vestergaard

Noelia Villarmarzo Arruñada

Contents

Part I

Pre-analysis

1 | Introduction

The effective control of an inverted pendulum is still an active area of research nowadays.[1] One example of such a system is a setup called Cubli. It consists of a cube controlled with reaction wheels. The Cubli can jump up and balance on one of its edges or on one of its corners, as shown in *figure 1.1*. The Cubli is designed as a simple setup to let control engineers work with an inverted pendulum. A working Cubli can also be an interesting way to show and explain the general public what control engineering is about.[2]

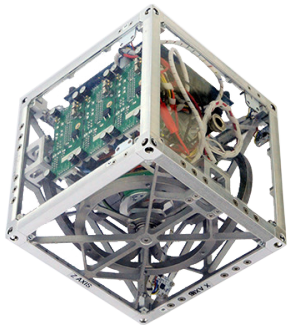


Figure 1.1: A Cubli balancing on one of its corners.[3]

One application of the internally actuated cube has been suggested for alternative locomotion in planetary or asteroid exploration. The internal actuation is not very efficient in higher gravity environments, however, in environments with microgravity such as asteroids the technology becomes very feasible.[4]

In microgravity a Cubli could tumble or even jump across the surface. A traditional rover with wheels would not be able to sufficiently grip or might even push the rover off the surface long enough for it to land upside down. Where such a situation would be fatal for most rovers, a cube with internal actuation would not be immobilized by landing upside down.[5]

This concept was the basis of a small experimental lander called MINERVA, short for Micro/Nano Experimental Robot Vehicle for Asteroids, which was to explore the near earth asteroid Itokawa, see *figure 1.2*. The lander was deployed from its mother spacecraft HAYABUSA in 2005, when it unfortunately missed the asteroid's small gravitational pull and drifted off into space.[6]

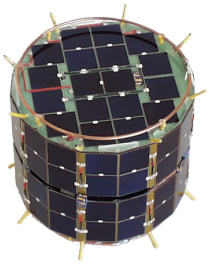


Figure 1.2: MINERVA experimental lander, which was designed for asteroid exploration.[6]

A more recent example of development in this area is NASA's Hedgehog robot, which also is actuated internally with reaction wheels. It has been through several tests aboard an aircraft for microgravity research in June 2015, where it showed its ability to jump out of a sandpit. A picture of the Hedgehog robot can be seen in *figure 1.3*.[5]

Figure 1.3: NASA's Hedgehog robot for asteroid exploration.[5]

It is also possible to take a group of cubes, so they could move together to traverse obstacles or solve puzzles one cube alone could not. A group of cubes can form a structure (*figure 1.4*), and by communicating between each other they can use their reaction wheels to get the structure to move in the desired direction. Since each cube can move independently, a single cube can detach for an assignment or catch up with the main structure if it gets dropped.[7]

Figure 1.4: A number of cube robots (called M-blocks at Massachusetts Institute of Technology (MIT)) making two different structures. These M-blocks stick together with the help of magnets placed in their corners.[8]

2 | Design Considerations

From the introduced applications both in modular robot design as well as planetary or asteroid exploration, a high controllability Cubli design is desired. In a realization of a full functioning Cubli, different considerations regarding design and overall functionalities must be taken into account. Later in this chapter restrictions are considered regarding the features to implement in this project.

2.1 Desired Functionalities

A simple design method, for the Cubli to stand on one of its corners, is to make it jump to this position in two steps. First, it should be made to stand on one of its edges before going to a balancing position on a corner.

Therefore, as a basic ability, the prototype should be able to jump onto any one of its edges and balance. By rotating one of its reaction wheels to a determined and controlled velocity and braking it suddenly, it should be able to raise the cube to an unstable position on an edge. When the jump up controller has raised the Cubli another controller must catch it around equilibrium position.
This second controller is essential to allow the prototype to keep balancing. It should react fast enough when put into action so that the Cubli does not fall again. With some more considerations regarding this controller's robustness, it should be possible to change the inclination of the surface under it to a certain extent.
In a similar fashion, the second step of the process consists of speeding up wheels and braking them one more time to raise the Cubli to one of its corners. The latter should finally be caught by a controller able to maintain it in this unstable equilibrium.

Once standing on a corner, a spinning functionality should allow the prototype to turn around itself. This change in orientation also permits the Cubli to move in its environment by falling and raising repeatedly towards the desired direction.

With a cube, communication and power supply as wired connections are not practical if the prototype has to move around. To ensure a complete autonomy, the prototype should be remotely controllable, i.e. it should be possible to ask it to run pre-defined routines (stand on an edge or a corner, rotate, etc.) from a distant computer.
Moreover, it should be self-sustained by an internal battery able to power the embedded computer, actuators and sensors for a reasonable amount of time.

All these functionalities are potential features to implement. The next section sets some limits to what is actually to be achieved for this project.

2.2 Prototype Restrictions

After some potential functionalities have been described, it is necessary to put some restrictions to the scope of this project.

To achieve a full Cubli design, an intermediary step with a one-side prototype is chosen as the focus for this project. This is to simplify the design process of the model and the controller before scaling it to a full cube.
This simplified prototype consists of a single square face of a cube, later on called frame. It should have one reaction wheel to raise it and a baseplate to which the frame is fixed so it only has one degree of freedom.

With only one frame, the prototype cannot spin around itself nor move as described in *section 2.1*. Instead of balancing up in two steps, the frame can only balance on its corner which is comparable to getting the cube to balance on its edge.
The jump-up of the cube is also set aside, since it is considered out of scope for this Bachelor project.

Concerning the wireless capabilities of the prototype, since it is only going to be built as a single frame in this project, the need for power autonomy and remote communication is not critical for it to work properly. Moreover, wireless communication and hardware design are also out of scope for this Control Engineering project. This means the system will be powered by an external power supply and communication will be done by physically connecting to the prototype.

Noncritical functionalities are now set aside. The remaining focus is put on the control of a balancing single frame that is scalable to a cubic model.

At Aalborg University (AAU), there exists a working setup of a one-side Cubli. The overall goals of this semester are to make a model of this system, to simulate the model and then to design and implement a controller for it to balance around its equilibrium point. Furthermore, the prototype should be able to balance even when its baseplate is inclined. In the next chapter, the available Cubli setup is described in more detail.

3 | System Description

The system is composed of a frame and reaction wheel as the main objects to be controlled.
The control action is done with a brushless DC motor which actuates both on the frame and the wheel.
A board with a microcontroller and a motor control board are mounted on the frame through a connecting and breakout board.
The frame is fixed to a baseplate, connected with a potentiometer that can be used for direct angle measurements of the frame's angular position.
Two additional sensor breakout boards, including an integrated circuit with gyro and accelerometer, are located at the top of the frame. These are to be used for angle and velocity measurements independently of the angle of the baseplate.
A servomotor that controls the brake system is also attached, which has an arm capable of blocking the wheel by hitting one of the two break-blocks mounted on the edge of the reaction wheel.
The different components of the existing system can be seen in *figure 3.1*.

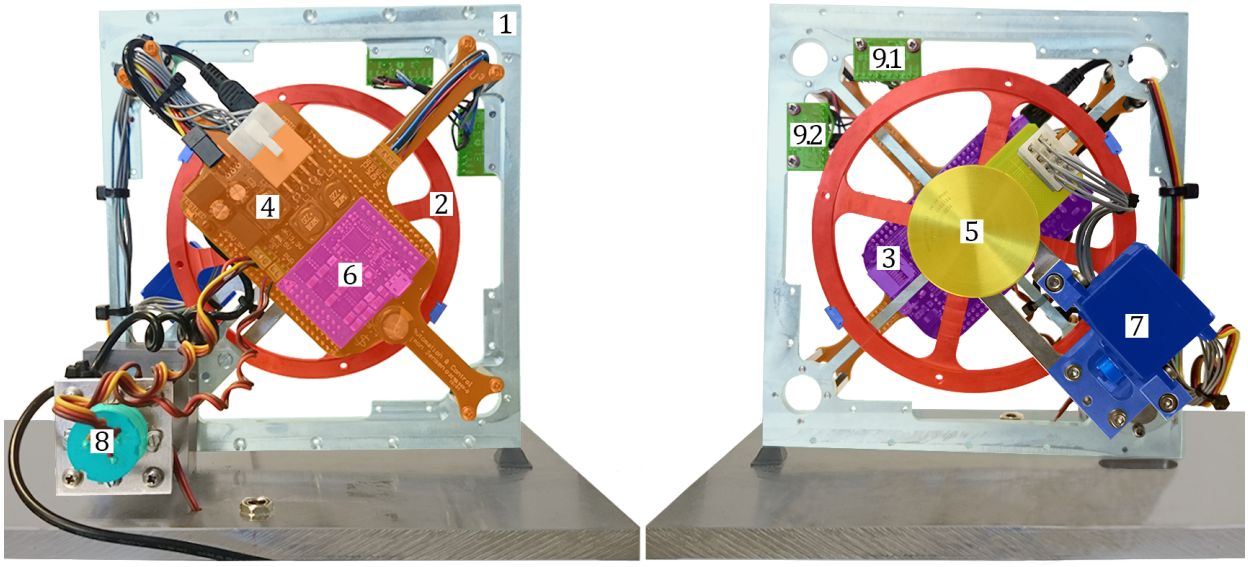


Figure 3.1: Existing setup viewed from the front and the back. There is a color coding to distinguish the different parts: frame(1), wheel(2), microcontroller(3), connecting and breakout board(4), brushless DC motor(5), motor control board(6), servomotor(7), potentiometer(8), Inertial Measurement Units(9).

A detailed description of all the components is presented in the following sections for a better overview of the system.

3.1 Mechanical Components

The frame and the reaction wheel constitute the mechanical parts of the system.

Frame

The frame is made of aluminum with dimensions 17x16x0,5 cm with two cross connections between opposing corners. To keep the weight down on the frame, a large area of the frame is milled out. That composes the body of the Cubli.
As it can only be balanced in one direction, it is attached to the base on one of its vertices to avoid it from falling in any of the other directions.

Reaction Wheel

In the case of the Cubli, the reaction wheel is made of brass with most of the mass in a ring at its outer edge and two cross connections through its center. The reaction wheel is coupled to the axis of a motor, through its center of rotation. When the wheel turns, its change of velocity creates a torque on the system that is transmitted with opposite direction to the body due to the conservation of angular momentum.

3.2 Main Boards

On the prototype, a microcontroller provides the main computing power and controls the system through the connecting and breakout board, directly attached to the frame.

Microcontroller

The microcontroller used on this system is a BeagleBone Black [9], which is in charge of managing the data from the motor and the sensors and calculating the required control action.
It uses an ARM processor at a clock frequency of 1 GHz and has a large amount of general purpose inputs and outputs, of which some of them support the I^2C protocol or include an Analogue to Digital Converter (ADC).
The ADCs of the BeagleBone have a 12-bit resolution that is limited to a range of $0 - 1,8$ V. The fastest sampling time it can provide is 125 ns [10].

Connecting and Breakout Board

This board is used for power distribution with different voltages sent to each unit. There is also a built-in gain for the potentiometer, configured for the range of the BeagleBone ADCs and the possible positions that the frame can have. The schematics can be found in *appendix M*.

3.3 Actuators

There are two actuators in the system, a brushless DC motor as the main actuator, which is used to apply the required control action and aservo motor, which can be used to brake the wheel in order to raise the frame.

Brushless DC Motor

The brushless DC motor is attached to the wheel and it is in charge of providing the required torque to the system. It is an EC 45 flat 50 Watt [11], which has additional features such three Hall effect sensors for angular velocity measurements. Table 3.1 contains some of the characteristics for this motor.

Motor Data	**Value** [Unit]
Nominal current	2,33 [A]
Motor constant (K_t)	33,5 [$N \cdot m \cdot A^{-1}$]
Mechanical time constant	12,4 [ms]

Table 3.1: Important parameters of the brushless DC motor.

Motor Control Board

There is a motor control board, Maxon ESCON Module 50/5 [12], connected between the BeagleBone and the brushless DC motor, and it is specifically made to work with ESCON motors.

The following table shows some of the main characteristics of the board.

Characteristics	**Value** [Unit]
Nominal output current	5 [A]
Peak current (<20 s)	15 [A]
Current control PWM frequency	53,6 [kHz]
Sample Rate of PI current controller	53,6 [kHz]

Table 3.2: Important parameters of the motor control board.

The motor control board can be configured with a program provided by Maxon called ESCON studio.[13]
The board is set to run with a closed loop control for the current. It is assumed that the reference current is the one given to the motor so the loop can be seen just as a unit gain.

This assumption is validated through a test described in *appendix I*, where it can be seen that both currents can be assumed to be equal.

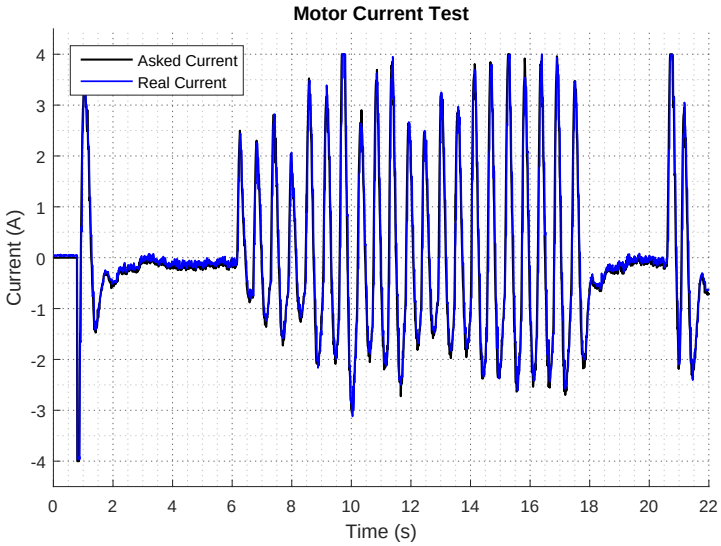


Figure 3.2: Result of the test done to the motor to check that the reference current can be assumed to be the one applied to the motor.

The reference current is sent to the control board as a PWM signal, whose duty cycle is configured within the range of 10 % to 90 % corresponding to 4 A at 90 % and −4 A at 10 %. The actual configuration can be seen *appendix G*.

Braking System

There is a braking mechanism included in the system, which can be used to make the frame go from resting position to vertical position.

To perform this task the brushless DC motor spins up the reaction wheel and when it has enough kinetic energy the braking mechanism suddenly brake it, using for this task a Hitec HS225 Mighty Mini Servomotor. The inertia of the wheel is thus transfered to the frame, in order to raise it to standing position.

3.4 Sensors

The prototype setup is provided with some sensors such a potentiometer for direct angle reference with respect to the baseplate and two Inertial Measurement Units (IMU), containing accelerometer and gyro, for global angle measurements.

Potentiometer

This sensor is a precision potentiometer with continuous turning, linearity within $\pm 1,0$ % and a resolution of $10\,\mathrm{k}\Omega$ ± 10 %.
The potentiometer is placed at the corner of the frame which is fixed to an axis and it can be used to measure the actual position of the frame.
However, its use is restricted to the existing setup, since it is fixed and gives only an angle in relation to the base, which is not present in the full Cubli.
In this project the potentiometer is used to test the dynamics of the Cubli, as feedback in the initial controller design and to check if the calculation of the angle using the IMU is done correctly.
Since some of the analysis and design will depend on the reliability of the potentiometer, different tests are carried out to check its characteristics and behavior.

Linearity Test

To confirm the linearity of the potentiometer a test is done, which is described in *appendix A*. As seen in *figure 3.3*, the result gives an almost straight line within the 1 % linearity of the potentiometer, but at a certain angle the potentiometer has an area where the measurement is deviating. The reason for this deviation lies in the fact that it is a continuous rotating potentiometer and at that point it has a dead zone. A way to correct this problem is to turn the potentiometer and recalibrate its limits. However, in this project it is not necessary since precise measurements are only needed around 0 degrees in the control region.

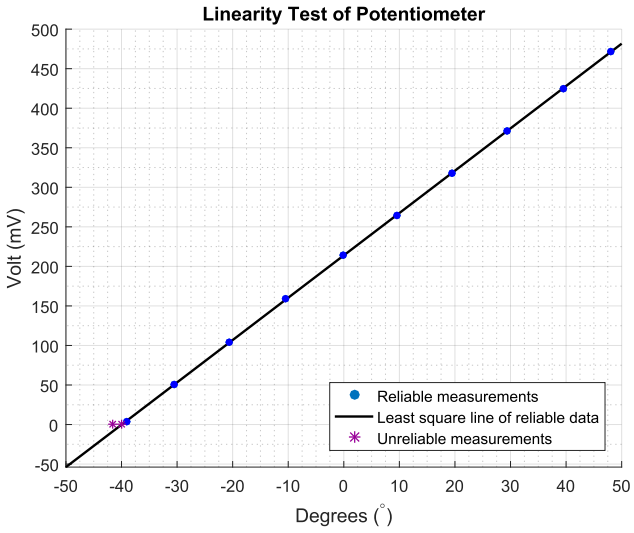


Figure 3.3: Result of the linearity test using a protractor measuring in degrees. It shows that the sensor has a linear behavior around the 0 degrees.

Range Test

An additional test is done to find the conversion from voltage to angle of the potentiometer. It is also tested if there is an offset that has to be taken into account. The detailed description of this test is found in *appendix B*.

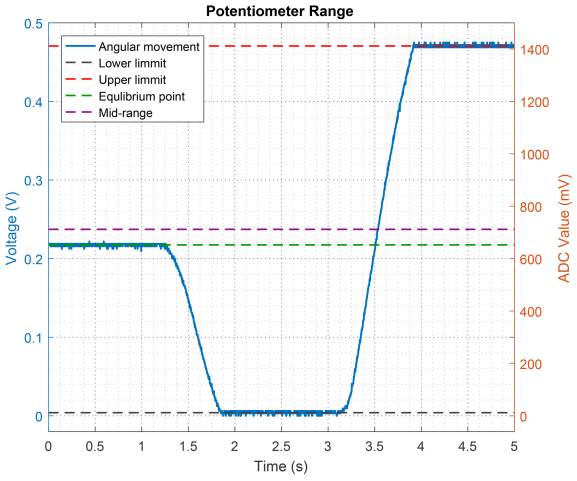


Figure 3.4: Potentiometer measurements in volts and the corresponding values that the ADC provides.

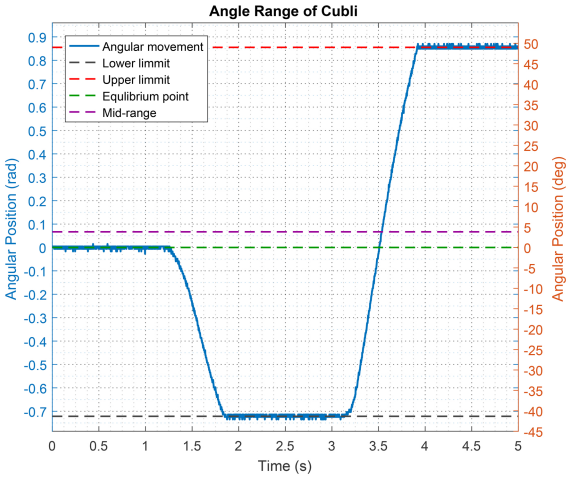


Figure 3.5: Potentiometer measurements converted to radians and degrees.

The results of this test is shown above in *figure 3.4*, where the reference lines reveals an offset between the middle of the range and the equilibrium point of the Cubli frame.
This offset, also seen on *figure 3.5*, exists in the physical position of the frame. When the frame is standing in its equilibrium position it is displaced by approximately 0,068 rad due to uneven distribution of mass around its center.
This results in a 0,853 rad range to one side of the optimal position and 0,717 rad on the other.
It is chosen that the angle-offset must be accounted for such that the equilibrium position of the frame is at angle 0 rad.

Inertial Measurement Unit

The Motion Processing Unit (MPU) contains a triple axis accelerometer and gyro integrated in the same chip [14], mounted on the breakout board from SparkFun [15].

The gyro has a full-scale range of ±250, ±500, ±1000, and ±2000 $\deg \cdot s^{-1}$, while the accelerometer has a programmable full scale range of ±2, ±4, ±8 and ±16 g.

The input voltage can be between 2,3 and 3,4 V, and it includes embedded algorithms for run-time bias and compass calibration.

The unit collects gyroscope and accelerometer data while synchronizing data sampling at a user defined rate, and it uses Inter Integrated Circuit (I^2C) protocol for communication, whose speed can be up to 400 kHz.

3.5 Code Base

More than the physical setup, a certain amount of code allowing to run controllers on the present hardware is also available.
Written in C++, it comprises all the drivers necessary to interface the BeagleBone board with the motor and all the different sensors described above, in *section 3.3* and *section 3.4*.

The actual controllers which make the Cubli stand up on its corner, are all composed of source and header files in a folder named *controller/controller_code/*. Each controller has to implement three functions, as shown in Listing 3.1.

```
/**
 *  Runs the actual controller on the given feedback(s) and the pre-defined input(s).
 *  Takes the sampling time and a 3x1 vector x_hat containing the feedbacks
 *  (processed data from the sensors):
 *              0: angular position of the frame
 *              1: angular velocity of the frame
 *              2: angular velocity of the wheel.
 *  Returns the output which should be applied to the actuator (current -> motor)
 */
extern CONTROLLER_OUTPUT_struct_T AAU3_CUSTOM_CONTROLLER(real_T Ts,
                                                         const real_T x_hat[3]);
/**
 *  Initializes the controller parameters (gain, polynomials coefficients)
 *  Has to be called only once, before running the controller itself.
 */
extern void AAU3_CUSTOM_CONTROLLER_initialize(void);
/**
 *  Does whatever is needed (if needed) to stop the controller
 */
extern void AAU3_CUSTOM_CONTROLLER_terminate(void);
```

Listing 3.1: Code snippet of the standard controller interface, written in C.

This is a default model based on the way MATLAB auto-generates controller code into C++ files. This model shall be used in this project as a general reference, to keep some common structure between the different controllers to be implemented. However, it is possible to adapt the arguments and the returned variables depending on the needs. Moreover, MATLAB auto-generation of code is not used in this project.

The file *controller/controller_test.cpp* contains the core part of the controllers operation. One of its functions, `void ControllerTest::runController(ControllerArgs* args)`, is called at some regular pre-defined intervals which is the desired sampling time. This initializes the available controllers, retrieves and processes the data from the sensors, and finally uses the controller code to compute the output current to send to the motor.

This updated output current is actually sent at the beginning of the next call.

It is important to note that the whole code is intended to run on the Beaglebone Black. Indeed, the latter uses an ARMv7 processor architecture, which requires the program to be compiled either directly on a computer with similar architecture which might be slow, or on another standard PC with a cross-compilation toolchain [16].

In this chapter, the Cubli setup has been described from the mechanical parts to the electronics components and to the software code base. The next chapter presents a mathematical approach for the description and analysis of the system and its behavior.

4 | System Modeling

With the given setup being described, it is necessary to study its natural behavior in more detail by deriving a model of this system. This chapter shows the process used to put up this model.

A mechanical drawing of the Cubli showing angles and coordinate system conventions is seen in *figure 4.1*. A two-dimensional global coordinate system is chosen with its origin on the pivot point of the frame. Moreover, the positive direction of the angles is chosen to be clockwise.

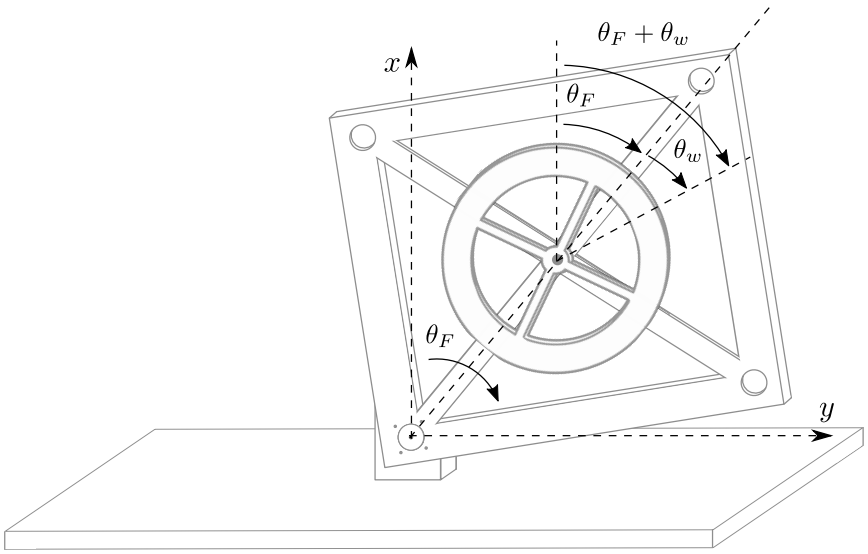


Figure 4.1: Mechanical drawing of the Cubli, including coordinate system and angle conventions. Note that the x axis is pointing upwards.

In next section, a complete model of the given setup is derived from Newton's Second Law of motion and rotation.

4.1 Derivation

To derive the modeling equations for the Cubli system from Newton's Second Law, it is split up into its two moving parts as seen in *figure 4.2* and *figure 4.3*.

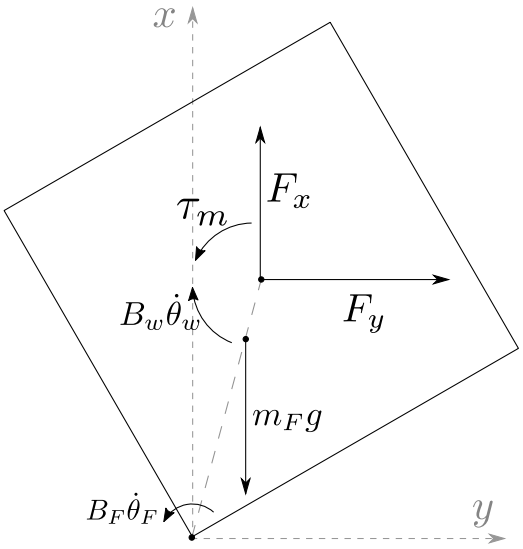


Figure 4.2:
Free body diagram of the frame of the Cubli.

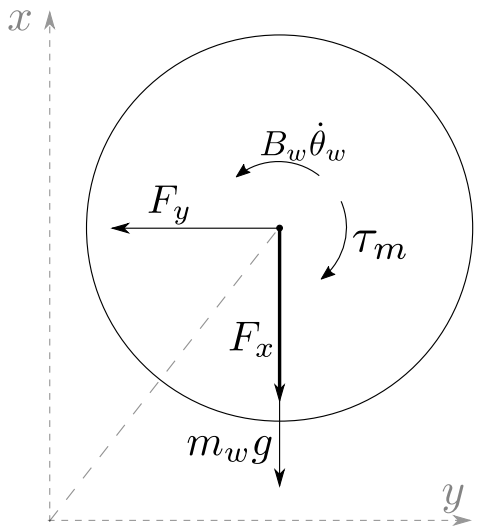


Figure 4.3:
Free body diagram of the reaction wheel of the Cubli.

The equation for the frame is deduced from the *figure 4.2*.

$$\mathrm{J_F}\ddot{\boldsymbol{\theta}}_\mathbf{F} = -\mathrm{B_F}\dot{\boldsymbol{\theta}}_\mathbf{F} + \mathbf{l_F} \times (\mathrm{m_F} \cdot \mathbf{g}) + \mathbf{l_w} \times \mathbf{F} - \boldsymbol{\tau}_\mathbf{m} + \mathrm{B_w}\dot{\boldsymbol{\theta}}_\mathbf{w} \quad [\mathrm{N \cdot m}] \quad (4.1)$$

Where:

$\mathrm{J_F}$	is the inertia of the frame	$[\mathrm{kg \cdot m^2}]$
$\ddot{\boldsymbol{\theta}}_\mathbf{F}$	is the angular acceleration of the frame	$[\mathrm{rad \cdot s^{-2}}]$
$\mathrm{B_F}$	is the friction coefficient of the frame	$[\mathrm{N \cdot m \cdot s \cdot rad^{-1}}]$
$\dot{\boldsymbol{\theta}}_\mathbf{F}$	is the angular velocity of the frame	$[\mathrm{rad \cdot s^{-1}}]$
$\mathbf{l_F}$	is the length to center of mass of the frame	[m]
$\mathrm{m_F}$	is the mass of the frame	[kg]
$\mathbf{g}$	is the gravitational acceleration	$[\mathrm{m \cdot s^{-2}}]$
$\mathbf{l_w}$	is the length to center off mass of the wheel	[m]
$\mathbf{F}$	is the force delivered to the frame from the wheel	[N]
$\boldsymbol{\tau}_\mathbf{m}$	is the torque delivered by the motor	$[\mathrm{N \cdot m}]$
$\mathrm{B_w}$	is the friction coefficient of the wheel	$[\mathrm{N \cdot m \cdot s \cdot rad^{-1}}]$
$\dot{\boldsymbol{\theta}}_\mathbf{w}$	is the angular velocity of the wheel with respect to the frame	$[\mathrm{rad \cdot s^{-1}}]$

The following equation is then derived from *figure 4.3*:

$$J_w(\ddot{\boldsymbol{\theta}}_{\mathbf{F}} + \ddot{\boldsymbol{\theta}}_{\mathbf{w}}) = \boldsymbol{\tau}_{\mathbf{m}} - B_w\dot{\boldsymbol{\theta}}_{\mathbf{w}} \quad [\mathrm{N \cdot m}] \quad (4.2)$$

Where:

- J_w is the inertia of the wheel $[\mathrm{kg \cdot m^2}]$
- $\ddot{\boldsymbol{\theta}}_{\mathbf{w}}$ is the angular acceleration of the wheel with respect to the frame $[\mathrm{rad \cdot s^{-2}}]$

In *equation (4.1)* the term $\mathbf{l_w} \times \mathbf{F}$ describes the torque delivered from the wheel to the frame, as it acts around the pivot corner of the frame. The vector $\mathbf{F}$ is decomposed into to forces parallel to the axes, F_x and F_y, as seen on *figure 4.2* and *figure 4.3*. To be able to apply Newton's Second Law, expressions for both the x- and y-coordinate describing the position of the center of mass of the wheel are found.

$$x = l_w \cdot \cos(\theta_F) \quad [\mathrm{m}] \quad (4.3)$$

$$y = l_w \cdot \sin(\theta_F) \quad [\mathrm{m}] \quad (4.4)$$

According to Newton's 2nd law of motion, $\sum F = m \cdot a$. Then to find F_x and F_y, the acceleration of the point at center of mass of the wheel must be known for both the x- and the y-direction. To achieve this the derivatives of the expressions for x and y in *equation (4.4)* are derived.

$$\dot{x} = -l_w \cdot \sin(\theta_F)\ \dot{\theta}_F \quad \left[\mathrm{m \cdot s^{-1}}\right] \quad (4.5)$$

$$\ddot{x} = -l_w \cdot \cos(\theta_F)\ {\dot{\theta}_F}^2 - l_w \cdot \sin(\theta_F)\ddot{\theta}_F \quad \left[\mathrm{m \cdot s^{-2}}\right] \quad (4.6)$$

$$\dot{y} = l_w \cdot \cos(\theta_F)\ \dot{\theta}_F \quad \left[\mathrm{m \cdot s^{-1}}\right] \quad (4.7)$$

$$\ddot{y} = -l_w \cdot \sin(\theta_F)\ {\dot{\theta}_F}^2 + l_w \cdot \cos(\theta_F)\ \ddot{\theta}_F \quad \left[\mathrm{m \cdot s^{-2}}\right] \quad (4.8)$$

Equation (4.6) and *equation (4.8)* can now be used with Newton's 2nd law of motion, while also taking gravity into account in sum of forces, to derive F_x and F_y.

$$\begin{aligned} -F_x - m_w \cdot g &= m_w \cdot \ddot{x} \\ F_x &= -m_w \cdot \ddot{x} - m_w \cdot g \\ F_x &= m_w \cdot (\ l_w \cdot \cos(\theta_F)\ {\dot{\theta}_F}^2 + l_w \cdot \sin(\theta_F)\ \ddot{\theta}_F\ ) - m_w \cdot g \end{aligned} \quad [\mathrm{N}] \quad (4.9)$$

$$\begin{aligned} -F_y &= m_w \cdot \ddot{y} \\ F_y &= -m_w \cdot (\ -l_w \cdot \sin(\theta_F)\ {\dot{\theta}_F}^2 + l_w \cdot \cos(\theta_F)\ \ddot{\theta}_F\ ) \\ F_y &= m_w \cdot (\ l_w \cdot \sin(\theta_F)\ {\dot{\theta}_F}^2 - l_w \cdot \cos(\theta_F)\ \ddot{\theta}_F\ ) \end{aligned} \quad [\mathrm{N}] \quad (4.10)$$

The original objective was to evaluate the term $\mathbf{l_w} \times \mathbf{F}$ in *equation (4.1)*. Since the expressions for the two forces, F_x and F_y, that compose the vector $\mathbf{F}$, are found in *equation (4.9)* and *(4.10)*, the vector product from *equation (4.1)* is evaluated.

$$\mathbf{l_w} \times \mathbf{F} = \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ \mathrm{l_w} \cdot \cos(\theta_\mathrm{F}) & \mathrm{l_w} \cdot \sin(\theta_\mathrm{F}) & 0 \\ \mathrm{F_x} & \mathrm{F_y} & 0 \end{vmatrix} \quad [\mathrm{N \cdot m}] \tag{4.11}$$

$$\mathbf{l_w} \times \mathbf{F} = \begin{bmatrix} (\mathrm{l_w} \cdot \sin(\theta_\mathrm{F}) \cdot 0 - 0 \cdot \mathrm{F_y}) \\ (\mathrm{l_w} \cdot \cos(\theta_\mathrm{F}) \cdot 0 + 0 \cdot \mathrm{F_x}) \\ (\mathrm{l_w} \cdot \cos(\theta_\mathrm{F}) \cdot \mathrm{F_y} - \mathrm{l_w} \cdot \sin(\theta_\mathrm{F}) \cdot \mathrm{F_x}) \end{bmatrix} \quad [\mathrm{N \cdot m}] \tag{4.12}$$

$$\begin{aligned} \mathbf{l_w} \times \mathbf{F} = 0 \cdot \hat{\mathrm{i}} + 0 \cdot \hat{\mathrm{j}} + (\mathrm{l_w} \cdot \cos(\theta_\mathrm{F}) \cdot (\mathrm{m_w} \cdot (\ \mathrm{l_w} \cdot \sin(\theta_\mathrm{F})\ \dot{\theta}_\mathrm{F}{}^2 - \mathrm{l_w} \cdot \cos(\theta_\mathrm{F})\ \ddot{\theta}_\mathrm{F}\ )) \\ -\mathrm{l_w} \cdot \sin(\theta_\mathrm{F}) \cdot (\mathrm{m_w} \cdot (\ \mathrm{l_w}\ \cos(\theta_\mathrm{F})\ \dot{\theta}_\mathrm{F}{}^2 + \mathrm{l_w} \cdot \sin(\theta_\mathrm{F})\ \ddot{\theta}_\mathrm{F}\ ) \\ -\mathrm{m_w} \cdot \mathrm{g})) \cdot \hat{\mathrm{k}} \end{aligned} \quad [\mathrm{N \cdot m}] \tag{4.13}$$

$$\mathbf{l_w} \times \mathbf{F} = (-\mathrm{l_w}{}^2 \cdot \mathrm{m_w} \ddot{\theta}_\mathrm{F}\ (\cos^2(\theta_\mathrm{F}) + \sin^2(\theta_\mathrm{F})) + \mathrm{l_w} \cdot \sin(\theta_\mathrm{F})\ \mathrm{m_w} \cdot \mathrm{g}) \cdot \hat{\mathrm{k}} \quad [\mathrm{N \cdot m}] \tag{4.14}$$

$$\mathbf{l_w} \times \mathbf{F} = (-\mathrm{l_w}{}^2 \cdot \mathrm{m_w} \ddot{\theta}_\mathrm{F} + \mathrm{l_w}\ \sin(\theta_\mathrm{F})\ \mathrm{m_w} \cdot \mathrm{g}) \cdot \hat{\mathrm{k}} \quad [\mathrm{N \cdot m}] \tag{4.15}$$

Since all torques only have a z-coordinate, *equation (4.15)* is inserted in *equation (4.1)*, without vector-notation. Note that $\mathbf{l_F} \times (\mathrm{m_F} \cdot \mathbf{g}) = (\mathrm{m_F} \cdot \mathrm{l_F} \cdot \mathrm{g} \cdot \sin(\theta_\mathrm{F})) \cdot \hat{\mathrm{k}}$.

$$\begin{aligned} \mathrm{J_F} \cdot \ddot{\theta}_\mathrm{F} = -\mathrm{B_F} \cdot \dot{\theta}_\mathrm{F} + \mathrm{m_F} \cdot \mathrm{l_F} \cdot \mathrm{g} \cdot \sin(\theta_\mathrm{F}) \\ -\mathrm{m_w} \cdot \mathrm{l_w}{}^2 \cdot \ddot{\theta}_\mathrm{F} + \mathrm{m_w} \cdot \mathrm{l_w} \cdot \mathrm{g} \cdot \sin(\theta_\mathrm{F}) - \tau_\mathrm{m} + \mathrm{B_w} \cdot \dot{\theta}_\mathrm{w} \end{aligned} \quad [\mathrm{N \cdot m}] \tag{4.16}$$

$$(\mathrm{J_F} + \mathrm{m_w} \cdot \mathrm{l_w}{}^2) \cdot \ddot{\theta}_\mathrm{F} = -\mathrm{B_F} \cdot \dot{\theta}_\mathrm{F} + (\mathrm{m_F} \cdot \mathrm{l_F} + \mathrm{m_w} \cdot \mathrm{l_w}) \cdot \mathrm{g} \cdot \sin(\theta_\mathrm{F}) - \tau_\mathrm{m} + \mathrm{B_w} \cdot \dot{\theta}_\mathrm{w} \quad [\mathrm{N \cdot m}] \tag{4.17}$$

Isolating $\ddot{\theta}_\mathrm{F}$ from *equation (4.17)* gives the final expression for the angular acceleration of the frame.

$$\ddot{\theta}_\mathrm{F} = \frac{-\mathrm{B_F} \cdot \dot{\theta}_\mathrm{F} + (\mathrm{m_F} \cdot \mathrm{l_F} + \mathrm{m_w} \cdot \mathrm{l_w}) \cdot \mathrm{g} \cdot \sin(\theta_\mathrm{F}) - \tau_\mathrm{m} + \mathrm{B_w} \cdot \dot{\theta}_\mathrm{w}}{\mathrm{J_F} + \mathrm{m_w} \cdot \mathrm{l_w}{}^2} \quad \left[\mathrm{rad \cdot s^{-1}}\right] \tag{4.18}$$

The equation above can be rearranged to clarify the effect that each variable exerts on the final acceleration of the frame.

$$\ddot{\theta}_F = -\frac{B_F}{J_F + m_w \cdot l_w^2} \cdot \dot{\theta}_F + \frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g}{J_F + m_w \cdot l_w^2} \cdot \sin(\theta_F) - \frac{1}{J_F + m_w \cdot l_w^2} \cdot \tau_m + \frac{B_w}{J_F + m_w \cdot l_w^2} \cdot \dot{\theta}_w \qquad \left[\text{rad} \cdot \text{s}^{-1}\right] \quad (4.19)$$

Once the acceleration of the frame is described by *equation (4.19)* it is possible to derive an expression for the angular acceleration of the wheel with respect to its axis from *equation (4.2)*.

$$\ddot{\theta}_w = \frac{\tau_m - B_w \cdot \dot{\theta}_w}{J_w} - \ddot{\theta}_F \qquad \left[\text{rad} \cdot \text{s}^{-1}\right] \quad (4.20)$$

Substituting $\ddot{\theta}_F$ by the expression for the angular acceleration of the frame (*equation (4.18)*) into *equation (4.20)* gives the final description for $\ddot{\theta}_w$, as shown in *equation (4.21)*.

$$\ddot{\theta}_w = \frac{\tau_m - B_w \cdot \dot{\theta}_w}{J_w} - \frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g \cdot \sin(\theta_F) - \tau_m + B_w \cdot \dot{\theta}_w - B_F \cdot \dot{\theta}_F}{J_F + m_w \cdot {l_w}^2} \qquad \left[\text{rad} \cdot \text{s}^{-1}\right] \quad (4.21)$$

$$\ddot{\theta}_w = \frac{(J_w + J_F + {l_w}^2 \cdot m_w) \cdot (\tau_m - B_w \cdot \dot{\theta}_w)}{J_w \cdot (J_F + m_w \cdot {l_w}^2)} - \frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g \cdot \sin(\theta_F) - B_F \cdot \dot{\theta}_F}{J_F + m_w \cdot {l_w}^2} \qquad \left[\text{rad} \cdot \text{s}^{-1}\right] \quad (4.22)$$

Equation (4.22) can be rearranged in the same way as *equation (4.19)*.

$$\ddot{\theta}_w = \frac{J_w + J_F + {l_w}^2 \cdot m_w}{J_w \cdot (J_F + m_w \cdot {l_w}^2)} \cdot \tau_m - \frac{(J_w + J_F + {l_w}^2 \cdot m_w) \cdot B_w}{J_w \cdot (J_F + m_w \cdot l_w^2)} \cdot \dot{\theta}_w - \frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g}{J_F + m_w \cdot {l_w}^2} \cdot \sin(\theta_F) + \frac{B_F}{J_F + m_w \cdot {l_w}^2} \cdot \dot{\theta}_F \qquad \left[\text{rad} \cdot \text{s}^{-1}\right] \quad (4.23)$$

The final model of the system can be summarize with the following equations:

$$\ddot{\theta}_F = -\frac{B_F}{J_F + m_w \cdot {l_w}^2} \cdot \dot{\theta}_F + \frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g}{J_F + m_w \cdot {l_w}^2} \cdot \sin(\theta_F) - \frac{1}{J_F + m_w \cdot {l_w}^2} \cdot \tau_m + \frac{B_w}{J_F + m_w \cdot {l_w}^2} \cdot \dot{\theta}_w \quad (4.24)$$

$$\ddot{\theta}_w = \frac{J_w + J_F + m_w \cdot {l_w}^2}{J_w \cdot (J_F + m_w \cdot {l_w}^2)} \cdot \tau_m - \frac{(J_w + J_F + {l_w}^2 \cdot m_w) \cdot B_w}{J_w \cdot (J_F + m_w \cdot {l_w}^2)} \cdot \dot{\theta}_w - \frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g}{J_F + m_w \cdot {l_w}^2} \cdot \sin(\theta_F) + \frac{B_F}{J_F + m_w \cdot {l_w}^2} \cdot \dot{\theta}_F \quad (4.25)$$

4.2 Linearization

Now that a model of the Cubli frame is put forth in *equation (4.24)*, it is apparent that the system is nonlinear due to the term including $\sin(\theta_F)$. In order to proceed with a simulation and controller design, it is convenient to first linearize the model. This is done by use of a Taylor series approximation.

Based on *equation (4.17)* the system is described in an operating point, around which it varies with $\Delta\theta_F$.

$$\begin{aligned}(J_F + m_w \cdot {l_w}^2)(\ddot{\theta}_{F_0} + \Delta\ddot{\theta}_F) = &-B_F \cdot (\dot{\theta}_{F_0} + \Delta\dot{\theta}_F) \\ &+(m_F \cdot l_F + m_w \cdot l_w) \cdot g \cdot \sin(\theta_{F_0} + \Delta\theta_F) \\ &-(\tau_{m_0} + \Delta\tau_m) + B_w \cdot (\dot{\theta}_{w_0} + \Delta\dot{\theta}_w) \qquad [N \cdot m]\end{aligned} \tag{4.26}$$

$$(J_F + m_w \cdot {l_w}^2)(\ddot{\theta}_{F_0} + \Delta\ddot{\theta}_F) = f((\dot{\theta}_{F_0} + \Delta\dot{\theta}_F),\ (\theta_{F_0} + \Delta\theta_F),\ (\tau_{m_0} + \Delta\tau_m),\ (\dot{\theta}_{w_0} + \Delta\dot{\theta}_w)) \quad [N \cdot m] \tag{4.27}$$

The operating point is chosen as θ_{F_0} and θ_{w_0} and their derivatives being equal to 0. This corresponds to the frame being in the upright position, see *figure 4.1*. At this position all the velocities and accelerations are 0, which results in 0 torque τ_m as well. Taking this into account and applying the Taylor series approximation yields the following.

$$\begin{aligned}(J_F + m_w \cdot {l_w}^2)\Delta\ddot{\theta}_F = &\cancelto{0}{f(\dot{\theta}_{F_0},\ \theta_{F_0},\ \tau_{m_0},\ \ddot{\theta}_{w_0})} \\ &+\frac{\partial}{\partial\dot{\theta}_F} f \cdot \Delta\dot{\theta}_F + \frac{\partial}{\partial\theta_F} f \cdot \Delta\theta_F + \frac{\partial}{\partial\tau_m} f \cdot \Delta\tau_m + \frac{\partial}{\partial\dot{\theta}_w} f \cdot \Delta\dot{\theta}_w \qquad [N \cdot m]\end{aligned} \tag{4.28}$$

All the higher order terms are discarded due to their negligible impact on the system when it is near the operating point.

$$\begin{aligned}(J_F + m_w \cdot {l_w}^2)\Delta\ddot{\theta}_F = &-B_F\Delta\dot{\theta}_F + (m_F \cdot l_F + m_w \cdot l_w) \cdot g \cdot \cos(\theta_F)\Delta\theta_F \Big|_{\theta_F = 0} \\ &-\Delta\tau_m + B_w\Delta\dot{\theta}_w \qquad [N \cdot m]\end{aligned} \tag{4.29}$$

$$(J_F + m_w \cdot {l_w}^2)\Delta\ddot{\theta}_F = -B_F\Delta\dot{\theta}_F + (m_F \cdot l_F + m_w \cdot l_w) \cdot g \cdot \Delta\theta_F - \Delta\tau_m + B_w\Delta\dot{\theta}_w \qquad [N \cdot m] \tag{4.30}$$

Equation (4.30) shows the final linearized model.

4.3 Block Diagram

To verify the model in simulation *equation (4.30)* is transformed into the Laplace domain, after which a transfer function of the system can be derived. The proceeding equations

are valid only around the operating point, and so for better overview, in the following $\Delta\theta_F = \theta_F$.

$$(J_F + m_w \cdot l_w{}^2) \cdot \theta_F \cdot s^2 = -B_F\theta_F \cdot s + (m_F \cdot l_F + m_w \cdot l_w)g \cdot \theta_F - \tau_m + B_w\theta_w \cdot s \tag{4.31}$$

The angle of the reaction wheel, θ_w, still features in *equation (4.31)*. It is desirable to have only one input, τ_m, and one output, θ_F. To achieve this, *equation (4.20)* is transformed into the Laplace domain and solved for θ_w.

$$\theta_w \cdot s^2 = \frac{\tau_m - B_w\theta_w \cdot s}{J_w} - \theta_F \cdot s^2 \tag{4.32}$$

$$\theta_w = \frac{-J_w\theta_F \cdot s^2 + \tau_m}{J_w \cdot s^2 + B_w \cdot s} \tag{4.33}$$

Equation (4.33) is now substituted for θ_w in *equation (4.31)*, and the transfer function is of the system is derived.

$$(J_F + m_w \cdot l_w{}^2) \cdot \theta_F \cdot s^2 = -B_F\theta_F \cdot s + (m_F \cdot l_F + m_w \cdot l_w) \cdot g \cdot \theta_F - \tau_m + B_w \cdot \left(\frac{-J_w\theta_F \cdot s^2 + \tau_m}{J_w \cdot s^2 + B_w \cdot s}\right) \cdot s$$

$$\frac{\theta_F}{\tau_m} = \frac{\frac{s}{-J_F - m_w \cdot l_w{}^2}}{s^3 + \left(\frac{B_w}{J_w} + \frac{B_w + B_F}{J_F + m_w \cdot l_w{}^2}\right)s^2 - \left(\frac{(m_F \cdot l_F + m_w \cdot l_w) \cdot g}{\left(J_F + m_w \cdot l_w{}^2\right)J_w} - \frac{B_F B_w}{\left(J_F + m_w \cdot l_w{}^2\right)J_w}\right)s - \frac{(m_F \cdot l_F + m_w \cdot l_w)B_w \cdot g}{\left(J_F + m_w \cdot l_w{}^2\right)J_w}} \tag{4.34}$$

The transfer function from *equation (4.34)* can also be represented in the form of a block diagram, as seen in *figure 4.4*.

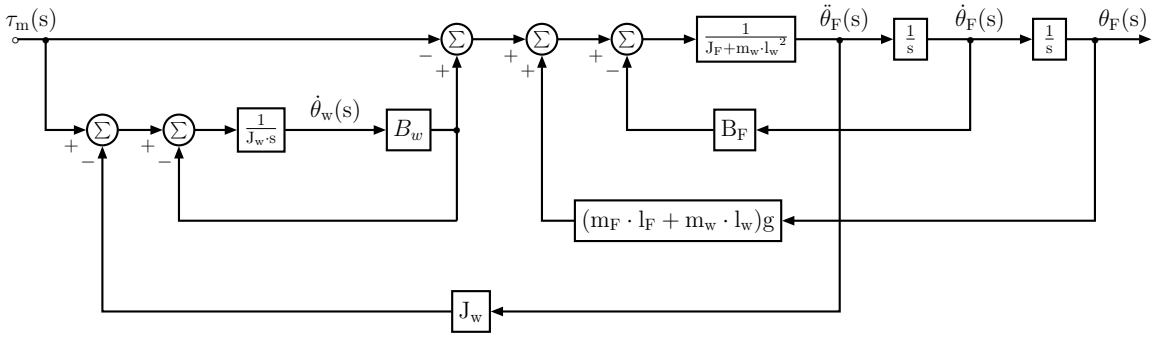


Figure 4.4: Block diagram of the Cubli as a SISO system. The input is the torque applied to the wheel. The output is the angular position of the frame.

5 | Plant Analysis

Once the model of the system has been described, a further analysis can be done. It includes the acquisition and estimation of the parameters, the model testing and the stability analysis.

5.1 Acquisition of Parameters

A method to obtain the parameters from the wheel (mass, distance from its center to the pivoting point of the frame, inertia with respect to its center of rotation and friction) is provided in *appendix L*. However, as these parameters should remain constant and they have been obtained by previous project runners, it is chosen to use these known parameters [17].

Parameter	**Value**	**Units**
m_w	$0,222$	kg
l_w	$0,093$	m
J_w	$0,601 \cdot 10^{-3}$	$kg \cdot m^2$
B_w	$17,03 \cdot 10^{-6}$	$N \cdot m \cdot s \cdot rad^{-1}$

Table 5.1: Parameters of the wheel.

In the previous sections parameters have been found, however, some critical parameters were given from a previous project. When these parameters were found, the Cubli had another mass, due to some physical modifications of the platform, which were performed after the referred project.

Mass of the Frame

The first one to find is the mass of the frame, which can be measured by weighing the setup without the base and substracting the known mass of the wheel. This gives a mass of $0,548$ kg, see *appendix E*.

Center of Mass of the Frame

The new center of mass can be found hanging the frame from different corners and measuring the deviation angle from the vertical position in every direction. This gives a center of mass which can be seen in *figure 5.1*.

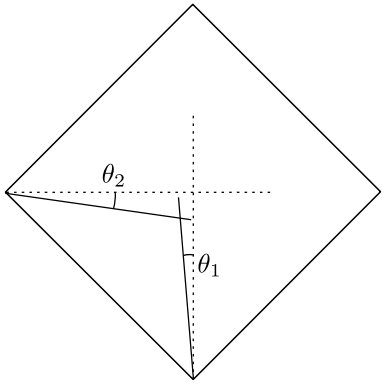


Figure 5.1: Location of the center of mass, where $\theta_1 = 0,043$ rad and $\theta_2 = 0,078$ rad.

The new point is not in the vertical line as it was assumed in the model, but this can be solved correcting the offset in the calculation of the angle inside the control loop and taking this new point as the equilibrium one.

The new l_F can then be obtained projecting the center of mass onto the vertical line, resulting in $8,498$ cm, see *appendix E*.

Inertia and Friction of the Frame

The last parameters, J_F and B_F, can not be measured directly so they have to be estimated. The starting point is given by the parameters from the previous report, as seen on *figure 5.2* [17].

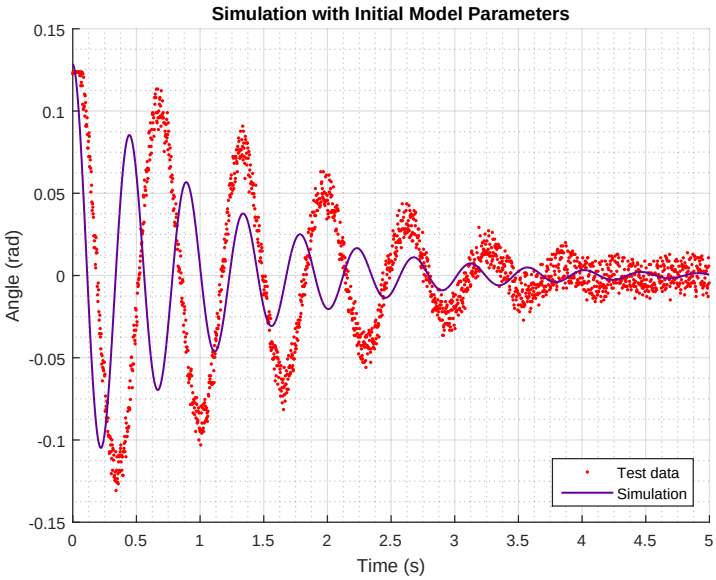


Figure 5.2: A comparison of the model simulation and the initially given parameters ($J_F = 6,08 \cdot 10^{-3}$ kg $\cdot$ m^2 and $B_F = 5,32 \cdot 10^{-3}$ m $\cdot$ s $\cdot$ rad^{-1}). Data is obtained through *appendix D*.

This problem can be solved using optimization, which is the subject of the proceeding

section.

5.2 Parameter Estimation using Optimization

In this section methods for optimization are investigated. The base of the implementation is in this case a Matlab script which is fed with test data along with the model simulation, whose task is to fit the model output to the test data by adjusting one or more parameters in the model. In this case the model representation supplied to the script is a Simulink model, which can then be run by the script whenever needed and the script can modify the parameters to be adjusted. The process is iterative.

The Optimization Problem

The basic scheme for the optimization problem is given in *figure 5.3*.

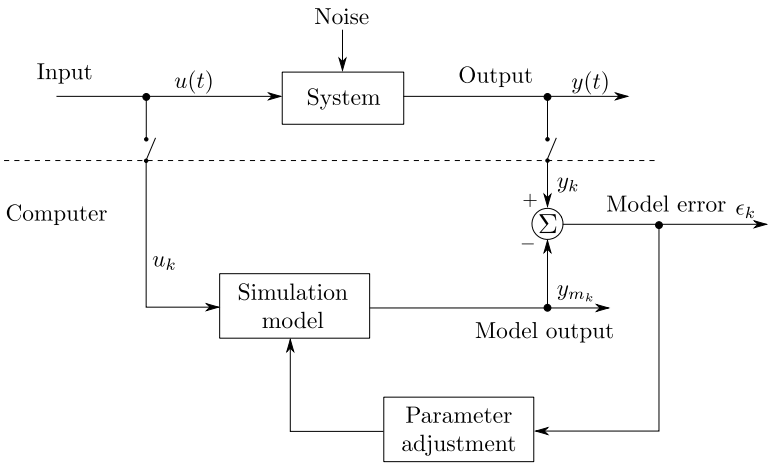


Figure 5.3: Schematic of the optimization problem, where input (u(t)) and output(y(t)) data is logged. The input data is used with the simulation to generate the simulation output ($y_m(t)$). The real output and the simulated output are then compared and an adjustment is made to the parameters of the simulation. Then a new $y_m(t)$ is generated with the u(t) and compared to the y(t). This process is iterated until a satisfactory match between y(t) and $y_m(t)$ is achieved. Inspired from Senstools documentation.[18]

The provided data is taken from an initial value test of the Cubli hanging down like a pendulum, see *appendix D*. If the fit is done in the operating region from $-0,15$ to $0,15$ rad, the simulated behavior in this range will be closer to reality.

Furthermore, the nonlinear model is used to accurately describe the oscillatory behavior of the pendulum. The model is modified such that it describes the system as a regular pendulum without the dynamics of the reaction wheel, in order to match the test conditions under which the data was extracted, as seen in *figure 5.4*.

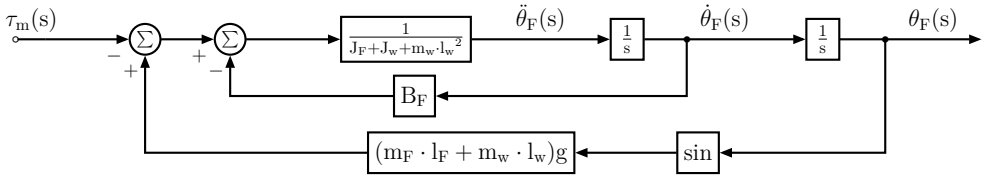


Figure 5.4: Block diagram of the system as a regular pendulum with the wheel fixed to the frame.

In order to minimize the difference between the data points measured in test and the output of the model, a function to describe such a relationship is needed. The cost function used to describe goodness of the fit, is a mean square error function.

$$P(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{k=1}^{N} \left(y_k - y_{m_k}(\boldsymbol{\theta})\right)^2 \tag{5.1}$$

Where:

- $\boldsymbol{\theta}$ is the parameter(s) to be adjusted
- N is the degrees of freedom for each parameter
- k is each sample in time, t = 1T, 2T, ..., kT, ..., NT where T is sampling time
- y_k is the k^{th} sample of the test measurement output vector
- y_{m_k} is the k^{th} sample of the model output vector

A normal mean square error function is only divided by the degrees of freedom, N, but in this case it is divided by 2N to cancel out the factor two which arises when computing its gradient. This does not have any impact on the solution since the minimum maintains its original position.

Optimization using the Gradient

One way of solving the optimization problem is through the use of the gradient. It indicates in which direction the steepest descent (or ascent) is found in an infinitesimal surrounding of a given starting point.

For a function f($\mathbf{x}$) with a change in $\mathbf{x}$ of $\boldsymbol{\delta}$, the following can be obtained from the Taylor series.[19]

$$f(\mathbf{x} + \boldsymbol{\delta}) \approx f(\mathbf{x}) + \mathbf{g}^T\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T\mathbf{H}\boldsymbol{\delta} \tag{5.2}$$

Where:

- $\mathbf{g}$ is the gradient ∇f($\mathbf{x}$)
- $\mathbf{H}$ is the Hessian
- $\boldsymbol{\delta}$ is the change in $\mathbf{x}$

In gradient based methods, only the first order Taylor approximation is used, that is, the last term, $\frac{1}{2}\boldsymbol{\delta}^{\mathrm{T}}\mathbf{H}\boldsymbol{\delta}$ is discarded. If the derivative of the first order approximation is set to zero, the following is obtained.[19]

$$\nabla\ f(\mathbf{x}+\boldsymbol{\delta}) \approx \mathbf{g} = 0 \tag{5.3}$$

That is, if the gradient of the function to be minimized is 0, a minimum or maximum exists as a candidate for a solution in this point. It follows that if standing in some point and computing the gradient in this point, then the gradient, $\mathbf{g}$, is the steepest ascent direction and the negative gradient, $-\mathbf{g}$, is the steepest descent direction. This only takes into account the immediate surroundings of the initially chosen point. A visualization of how the gradient points opposite to the minimum of an arbitrary function is seen on *figure 5.5*, which means that the negative gradient points to the minimum.

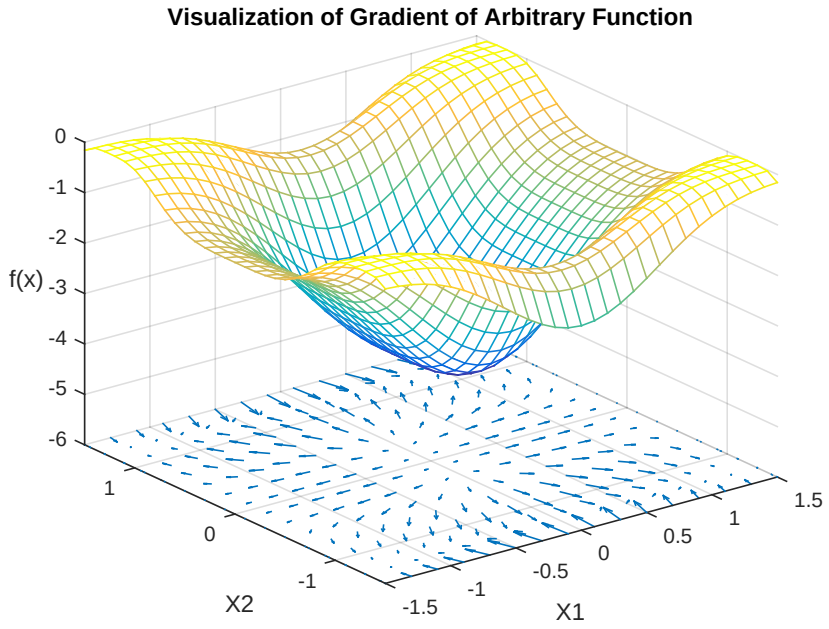


Figure 5.5: Visualization of gradient of an arbitrary function.

One way of implementation is to set a step-size which decides how far in the $-\mathbf{g}$ direction to go. The step-size can then be scaled in each iteration to avoid taking too large steps as shown in *figure 5.6* and 5.7, where x^* is the value of x which minimizes f(x), x_0 is the starting point at which $-g$ is computed and x is the point reached after the step.[18]

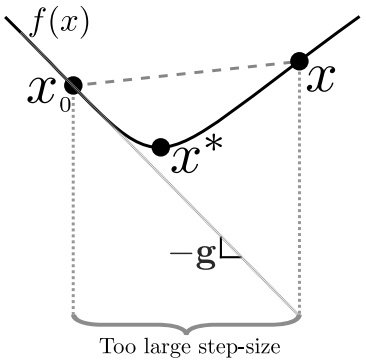


Figure 5.6: A too large step will cause the algorithm to step over the valley, resulting in a larger value of f(x) in the $-\mathbf{g}$ direction.

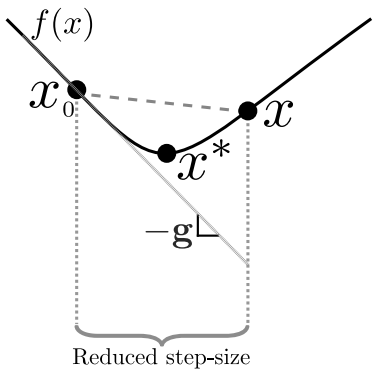


Figure 5.7: By going back and choosing a smaller step, a smaller value for f(x) is obtained.

The method does find a minimum. However, it converges to it rather slowly. An implementation where it is possible to directly retrieve the gradient of the function which is to be minimized, f($\mathbf{x}$), is shown in *figure 5.8*.

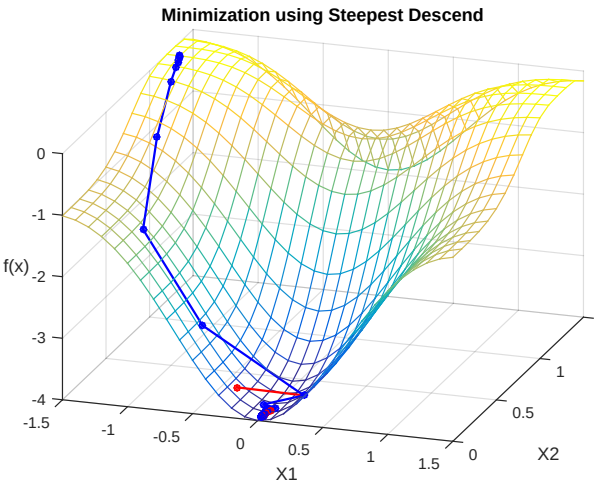


Figure 5.8: An example of a direct implementation of a gradient optimization method. It steps over the valley and the step size is reduced in the red iteration.

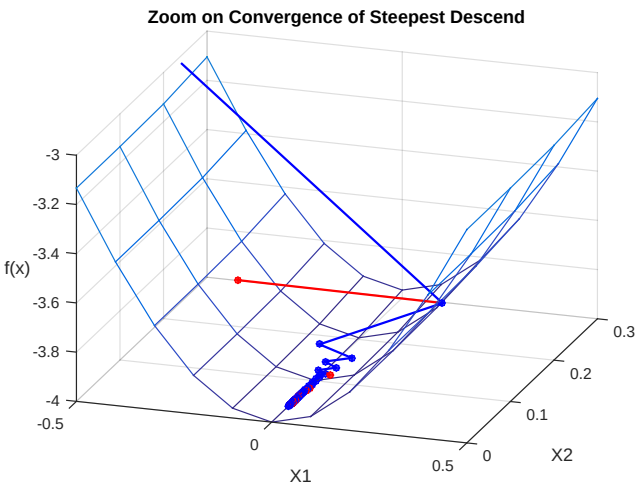


Figure 5.9: From the zoom on the convergence, it is seen that many iterations (here 100) are needed using this method.

Steepest Descend Method

The steepest descent method also uses the gradient to determine in which direction the local minimum lies; the method is therefore also often referred to as the gradient descent method. In the steepest descent method, the problem is minimized along the gradient in each iteration, as opposed to just taking a step as described in the previous.

To achieve this minimization along the gradient, one often uses a line search algorithm. There exists several line search algorithms, such as Dichotomous, Golden Section, Fibonacci, backtracking line search and so on. For the purpose of minimizing the one-dimensional problems in the gradient directions, the Dichotomous and Fibonacci line searchers are further investigated in the following sections. To use the line search however, a region in which the minimum is assumed to reside (the bracket) must be provided.[19] In the following such a region is assumed given, but in the section following the two line searchers, a method for selecting a bracket is discussed.

Dichotomous Line Search

The dichotomous line search is one of the more basic methods. The method locates the middle of the bracket (along x) and finds a point on either side of the middle, this gives the following two x-values at which the cost function must be evaluated.[19]

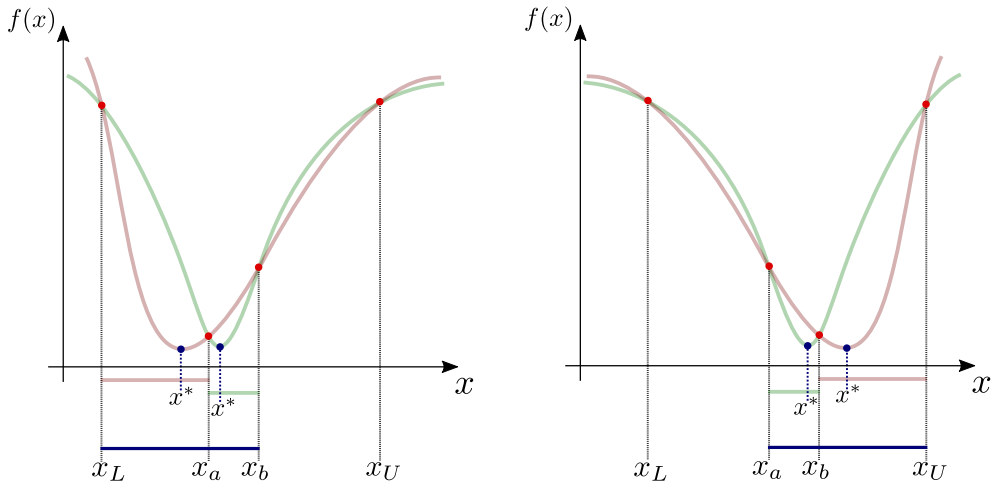
$$x_a = \frac{x_U - x_L}{2} - \epsilon \qquad x_b = \frac{x_U - x_L}{2} + \epsilon \tag{5.4}$$

Where:

- x_L is the lowest value of x in the bracket
- x_U is the highest value of x in the bracket
- x_a is the lower value of x at which f(x) must be evaluated
- x_b is the upper value of x at which f(x) must be evaluated
- ϵ is the small change in positive and negative direction from the middle

When the cost function, f(x) is evaluated at x_a and x_b, the two results are compared and two new possible intervals with possibility of containing the minimum can be determined as a consequence. These two intervals can be combined such that a new smaller bracket containing the minimum is found.[19]

This process of reducing the range of the bracket is illustrated in *figure 5.10*, where the green and red functions are examples of how f(x) could behave given the red known points. This leaves the two possible intervals for the minimum x^*. In the case of $f(x_a) < f(x_b)$, in *figure 5.10a*, the possible intervals, $x_L < x^* < x_a$ and $x_a < x^* < x_b$, can be combined to the new bracket $x_L < x^* < x_b$, shown in blue, which is sure to contain the minimum, x^*, within x_L and x_U. Said in an other way, if $f(x_a) < f(x_b)$, x^* must be contained in $[x_L,\ x_b]$, so $x_b = x_U$ for the next iteration. In the unlikely case that $f(x_a) = f(x_b)$, either x_a or x_b can be set as the new boundary.[19]



(a) Here $f(x_a) < f(x_b)$ resulting in the red interval, $x_L < x^* < x_a$, and green interval, $x_a < x^* < x_b$, which when combined yields the new bracket, $[x_L, x_b]$, shown in blue.

(b) Here $f(x_b) < f(x_a)$ resulting in the green interval, $x_a < x^* < x_b$, and red interval, $x_b < x^* < x_U$, which when combined yields the new bracket, $[x_a, x_U]$, shown in blue.

Figure 5.10: The function, f(x), is only evaluated at the points indicated by red dots. Two examples of how the graph of f(x) could appear is shown in green and red.

In the example implementation provided in *figure 5.11* the method is demonstrated on a rough scale so that the decisions made by the algorithm are clearly seen. Red lines represent x_a, blue lines x_b and for each iteration a new x_L or x_U is selected and indicated by the dotted markings.

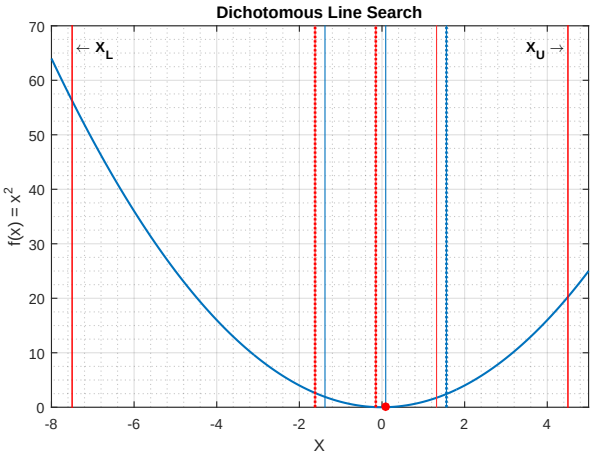


Figure 5.11: The bracket is marked as x_L and x_U, the remaining reference lines marks x_a as red and x_b as lines. The lines which are dotted are the ones selected as new x_L, x_a, or new x_U, x_b in each iteration.

Fibonacci Line Search

The Fibonacci line search and the golden section search are both based on a fundamental method of interval selection. As opposed to the dichotomous line search, these methods

only require one function evaluation per iteration. This is achieved by reusing one of the function evaluations performed in the previous iteration.[19]

An example of this interval system is shown in *figure 5.12* where the first iteration, marked in green, assuming some bracket, $[x_L,\ x_U]$, produces two intervals with equal lengths of I_1. Either the right or the left interval is then chosen as the new bracket. In *figure 5.12* $f(x_a) > f(x_b)$, therefore the right interval must contain the minimizer, so long as there is only one minimum within the bracket. Thus, the new bracket is set: $[x_L,\ x_U] = [x_a,\ x_U]$. In the case that $f(x_a) = f(x_b)$, either x_a or x_b can be set as the new boundary.

Within this bracket one function evaluation has already taken place (the circled red dot), this interval can be reused and only one further evaluation of f(x) is needed to proceed with the next iteration.

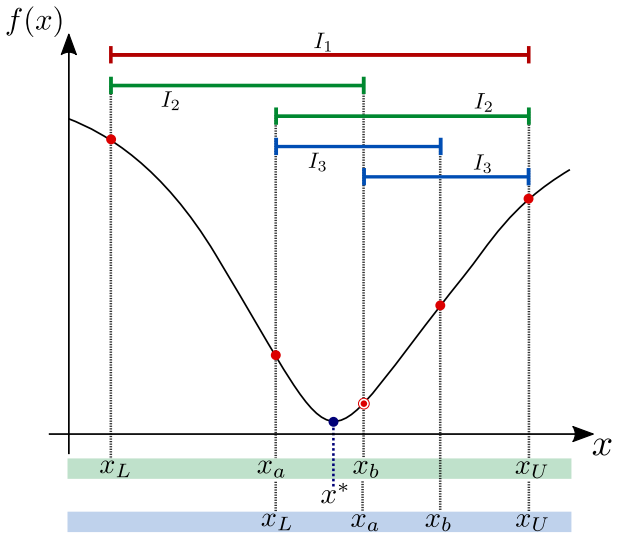


Figure 5.12: The function, f(x), is only evaluated at the points indicated by red dots and the function f(x) is obviously unknown in the process. First iteration is marked in green and the second in blue. x^* denotes the local minimizer and I is used to relate length of the intervals, showing that the two intervals in each iteration are equal.

If this procedure is repeated with numerous iterations, a series of interval lengths is produced. From *figure 5.12* the relation between successive interval lengths can be described as $I_1 = I_2 + I_3$, or in general for any iteration, k, as $I_k = I_{k+1} + I_{k+2}$.
With this as a basis, different ratios between I_{k+1} and I_{k+2} can be chosen, such as the golden section or the Fibonacci numbers.[19]
If for the last n^{th} iteration, I_{n+2} is assumed to be zero, then $I_n = I_{n+1} + I_{n+2} = I_{n+1}$ and

the following sequence of intervals emerges.

$$
\begin{aligned}
I_{n+1} &= & 1I_n &= F_0 I_n \\
I_n &= I_{n+1} + I_{n+2} & 1I_n &= F_1 I_n \\
I_{n-1} &= I_n + I_{n+1} & 2I_n &= F_2 I_n \\
I_{n-2} &= I_{n-1} + I_n & 3I_n &= F_3 I_n \\
I_{n-3} &= I_{n-2} + I_{n-1} & 5I_n &= F_4 I_n \\
I_{n-4} &= I_{n-3} + I_{n-2} & 8I_n &= F_5 I_n \\
&\vdots \\
I_k &= I_{k+1} + I_{k+2} & &= F_{n-k+1} I_n \\
&\vdots \\
I_1 &= I_2 + I_3 & &= F_n I_n
\end{aligned}
\tag{5.5}
$$

Where:

F_n	is the largest Fibonacci number used.
$\{ F_0,\ F_1,\ \ldots\ F_n \}$	is the Fibonacci sequence up to F_n

For the last expression in *equation (5.5)*, the size of the first bracket, I_1, is known and the last interval, I_n, can be chosen as the precision of the search. From this, the largest Fibonacci number needed can be found as $F_n = \frac{I_1}{I_n}$, and the remaining intervals can be calculated progressively during each iteration by use of the appropriate numbers in the Fibonacci sequence.[19]
In *figure 5.13* an example implementation is shown where each iteration contains x_a and x_b. In the iterations where x_a is chosen as the new x_L it is marked with a red circle and for x_b chosen as new x_U a blue circle is used. It shows clearly shows how each iteration only requires one function evaluation.

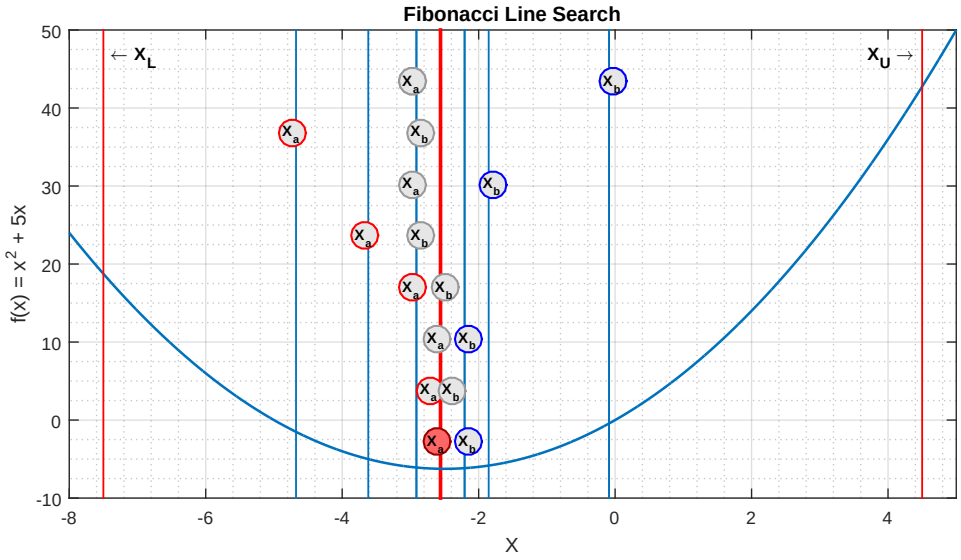


Figure 5.13: Fibonacci line search where the first iteration is at the top of the graph and each subsequent iteration is listed one line at a time downward. New x_L is marked with a red circle and new x_U with a blue circle. The chosen minimizer is marked in red at the bottom.

To see the benefits of using the Fibonacci line search rather than the more rudimentary dichotomous line search, the one-dimentional implementations are compared between *figure 5.15* and *figure 5.14*. The precision intervals determining how close each method is to reaching the true minimum are set to approximately the same value. This is done such that the number of function evaluations used for similar results can be compared in the two methods.

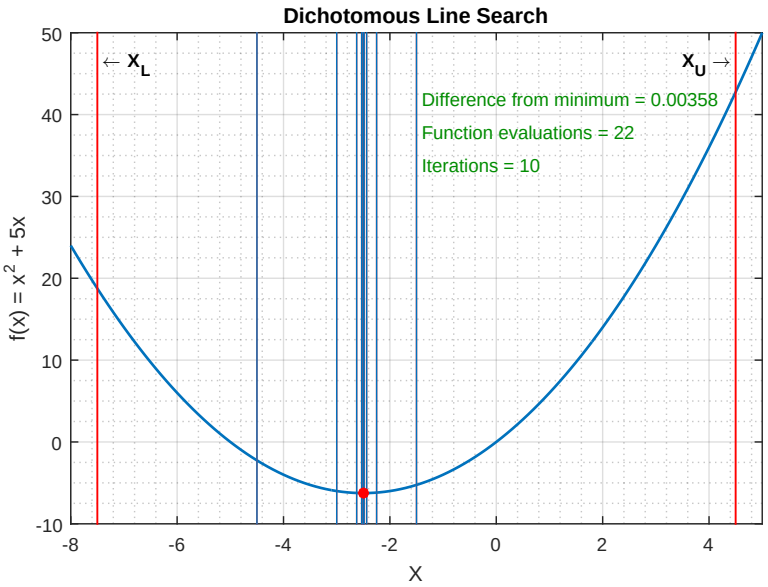


Figure 5.14: Example showing iterations and function evaluations used to obtain a given performance of the dichotomous Line Search.

In the dichotomous line search fewer iterations are used because each iteration almost cuts the bracket in half for each iteration. However, since each iteration requires two evaluations of f(x), the final number of function evaluations exceeds that of the Fibonacci line search. This is of course not always true – the location of the minimum in the bracket plays a role. If the minimum lies close to the middle of the bracket the dichotomous line search might get there faster than the Fibonacci line search. However for a broad spectrum of problems the performance of the Fibonacci line search will likely outperform that of the dichotomous line search.

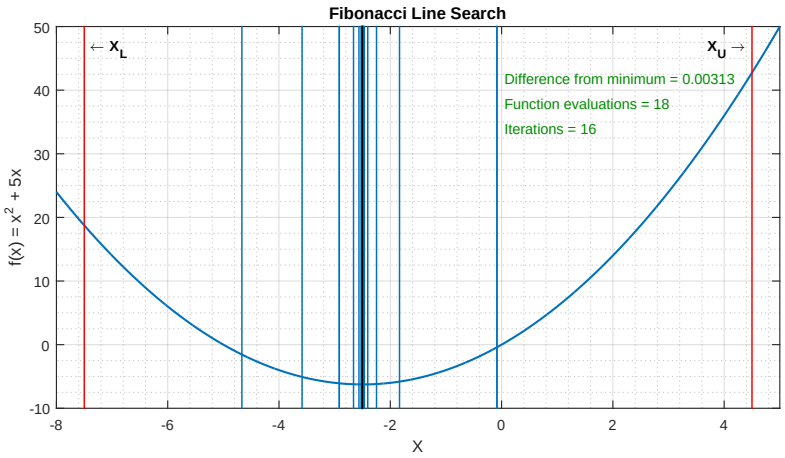


Figure 5.15: Example showing iterations and function evaluations used to obtain a given performance of the Fibonacci Line Search.

The lower number of function evaluations makes the Fibonacci line search superior in this case. In these examples the function can be analytically evaluated and the minimization problem can be solved analytically in one iteration. This is however not the case for a cost function representing mean squared error between simulation and reality for each change in one or more parameters. Such a cost function requires a simulation for each and every function evaluation. This is a time-expensive task which is why Fibonacci line search is chosen for the task at hand.

Forward Backward Method

The gradient shows the search direction and Fibonacci line search finds a minimizer to the one-dimensional problem contained within some bracket in this direction. It follows that some method for finding the bracket to contain the line search must be selected. For this purpose the forward backward method is chosen. The method attempts to find values of the cost function which goes from high to low to high.[20]

The idea is simple: It takes a step forward from the initial point, evaluates the cost function, and if the value is higher than the original one it takes a step backward, otherwise it takes a step forward. If it finds high low low, the step-length is increased. Since it is searching in the gradient direction a low high high geometry indicates that the minimum lies in the found interval so long as the set searched is monotonic; that is, the initial step

did not step over a peak of the cost function.[20]
In *figure 5.16* an implementation of the gradient descend method is illustrated. *Figure 5.17* shows how the gradient method generates a pattern with almost orthogonal lines converging to the minimum of the function.

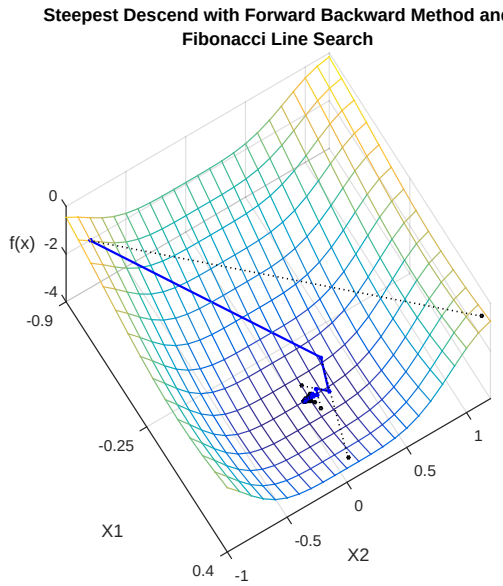


Figure 5.16: An example of a direct implementation of the gradient descent method using the forward backward method to determine the bracket in which the Fibonacci line search is used to solve each of the successive one-dimensional problems. The dotted lines represent the bracket found by the forward backward method in each iteration.

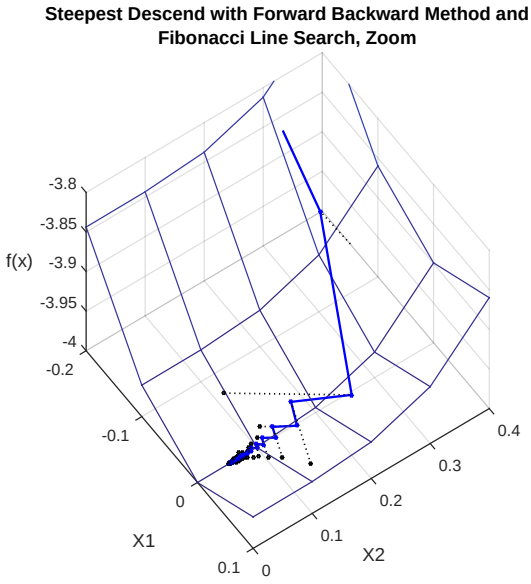


Figure 5.17: Zoom on the convergence shows how the gradient descend method create a pattern of lines converging to the minimum, (43 iterations are shown).

Implementation of Steepest Descent

To better understand the problem at hand the cost function to be minimized is shown in *figure 5.18*. It might seem strange to do optimization when one could just directly find the minimum by graphical inspection of the cost function. However, this graph took over 5 hours to render and contains the results of 90 000 simulations, which makes it a very inefficient and unpractical to work with in general. It is therefore desirable to obtain an optimization algorithm which can give a result in a relatively short time in case of any changes, be it new test data or changes to the system model. In this case the cost function is saved in a data-file, such that it can be plotted during implementation as a graphical reassurance that the optimization algorithm is behaving normally.

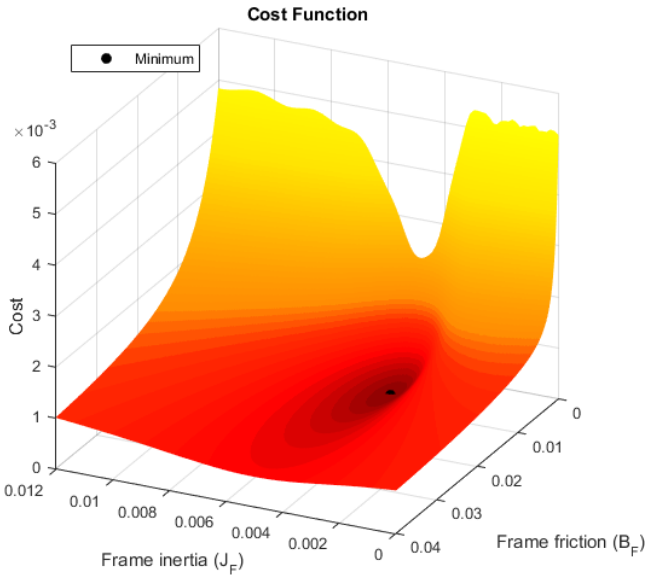


Figure 5.18: The cost function to be minimized. The parameters are shown in the xy-plane and the deviation between simulation and test data, cost, for each two parameters is shown on the z axis. To generate the graph 90 000 simulations were run, one for each point on the surface, and the rendering time exceeded 5 hours on a regular PC. This is not practical to work with in general, but valuable in the implementation process of the optimization algorithm.

When implementing the steepest descent method the gradient of the cost function in needed, which includes a numerical differentiation of the model.

$$\nabla\ P(\boldsymbol{\theta}) = \mathbf{G}(\boldsymbol{\theta}) = \nabla\left(\frac{1}{2N}\sum_{k=1}^{N}\left(\mathbf{y}_k - \mathbf{y}_{m_k}(\boldsymbol{\theta})\right)^2\right) \tag{5.6}$$

$$\mathbf{G}(\boldsymbol{\theta}) = -\frac{1}{\mathrm{N}}\left(\nabla\ \mathbf{y}_{\mathrm{m_k}}(\boldsymbol{\theta})^{\mathrm{T}}(\mathbf{y}_{\mathrm{k}} - \mathbf{y}_{\mathrm{m_k}}(\boldsymbol{\theta}))\right) \tag{5.7}$$

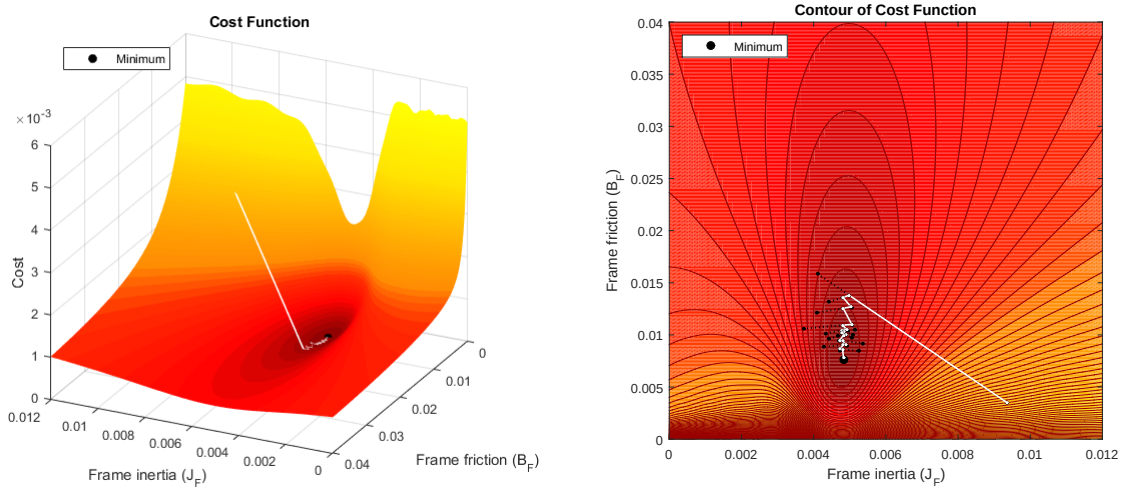
The implementation is seen in Listing 5.1.

```
%------ CALCULATING THE GRADIENT OF THE COST FUNCTION ----------------------

% gradientOfC = 1/N  * ( Y - Ym ) * Ym'
% Where:
%   Y       is the data recorded from test
%   Ym      is the model simulation output
%   Ym'     is the partial derivative of the simulated model with respect to
%           each of the parameters to be estimated.

% parameters: J_f B_f
% Small magnitude deviation from these parameters is set
delta = 0.01;

% Calculating the deviation
deltaJ_f = J_f+delta*J_f;
deltaB_f = B_f+delta*B_f;


```

```
19
20 % Running the simulation again, now with J_f the deviating parameter
21 deltaYmJf = simCubli( deltaJ_f, B_f, J_f, B_f );
22
23 % Running the simulation again, now with B_f as the deviating parameter
24 deltaYmBf = simCubli( J_f, deltaB_f, J_f, B_f );
25
26 % Finally calculating the partial derivatives of the model in (J_f, B_f)
27 YmDiffBf = ( deltaYmBf - Ym )/ delta;
28 YmDiffJf = ( deltaYmJf - Ym )/ delta;
29
30 % This however is just for the model.. and we need
31 % the gradient of the cost function, as stated earlier:
32 % gradientOfC = 1/N  * ( Y - Ym ) * Ym'
33 gJf = -(1/N)* ((Y - Ym )'*YmDiffJf);
34 gBf = -(1/N)* ((Y - Ym )'*YmDiffBf);
35
36 % The gradient of the cost function in the current point (J_f, B_f) is then:
37 g = [ gJf
38       fBf ];
```

Listing 5.1: Algorithm for the approximation of the gradient of the cost function.

With the gradient calculated the search direction is known, and implementation of the forward backward method as described above determines the search interval. This enables the above described Fibonacci line search to minimize the one-dimentional problem in each iteration. In *figure 5.19a* the process of the minimization is shown on the cost function and a contour plot is shown in *figure 5.19b*. In the contour plot the black dotted lines shows the search direction and region determined by the gradient and the forward backward method.



(a) The parameters are shown in the xy-plane and the cost for each set of parameters is shown on the z axis.

(b) A contour plot of the cost function with the optimization iterations. Here the black dotted lines shows the search.

Figure 5.19: Here each iteration of the steepest descend using Fibonacci and forward backward method is seen directly on the cost function.

The final result achieved in by the algorithm is shown in *figure 5.20*, where the normed RMS error between test results and simulation with final parameters, $J_F = 4,8 \cdot 10^{-3}$ kg $\cdot$ m^2 and $B_F = 7,8 \cdot 10^{-3}$ m $\cdot$ s $\cdot$ rad^{-1}, is also shown.

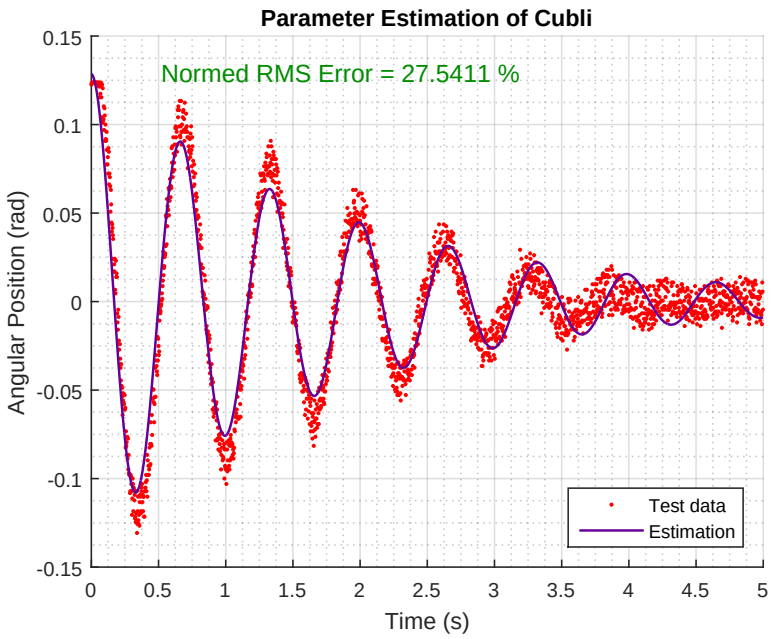


Figure 5.20: The final result of the minimization using steepest descent with Fibonacci line search and forward backward method.

Newton's Method

When approaching the minimum another method can be used, also rooted in the Taylor approximation. It is called Newton's method and uses both the first and second order terms of the approximation.[19]

$$f(\mathbf{x}+\boldsymbol{\delta}) \approx f(\mathbf{x}) + \mathbf{g}^T\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T\mathbf{H}\boldsymbol{\delta} \tag{5.8}$$

In this case the derivative of the approximation is set to 0, and the following is obtained.[19]

$$\nabla\ f(\mathbf{x}+\boldsymbol{\delta}) \approx \mathbf{g} + \frac{1}{2}\ \nabla\ \mathbf{H}\boldsymbol{\delta}^2 \tag{5.9}$$

$$\nabla f(\mathbf{x}+\boldsymbol{\delta}) \approx \mathbf{g} + \mathbf{H}\boldsymbol{\delta} = 0 \tag{5.10}$$

Using this to find an expression for the difference in $\mathbf{x}$, $\boldsymbol{\delta}$, yields the following.[19]

$$0 = \mathbf{g} + \mathbf{H}\boldsymbol{\delta} \tag{5.11}$$

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g} \tag{5.12}$$

The gradient is found in the previous section, the Hessian is also needed and can be related to to gradient as follows.[18]

$$\mathbf{H}(\boldsymbol{\theta}) = \nabla\ (\nabla\ P(\boldsymbol{\theta})) = \nabla\ \mathbf{G}(\boldsymbol{\theta}) \tag{5.13}$$

To avoid the second derivative of the cost function in *equation (5.1)*, the Hessian can be approximated simply by removing the 2nd derivative therm, which leads to the following.[18]

$$\widetilde{\mathbf{H}}(\boldsymbol{\theta}) \triangleq \frac{1}{N}(\nabla\ \mathbf{y_m}(\boldsymbol{\theta}))\,(\nabla\ \mathbf{y_m}(\boldsymbol{\theta}))^T \tag{5.14}$$

This approximation assumes that the model is only linearly dependent on the parameters, $\boldsymbol{\theta}$. As the error term, $(\mathbf{y} - \mathbf{y_m}(\boldsymbol{\theta}))$, approaches zero the approximation becomes increasingly accurate.

Senstools

Some of the background for the implementation above comes from documentation on a Matlab toolbox called Senstools [18]. Since this toolbox includes an extra feature based on parameter sensitivity and frequency domain and uses Newton's method to converge faster, it is a good decision to use this tool to check the previous results.

The same data, Simulink model and initial parameters as the previous case are given to this toolbox and the result of the fit can be seen in *figure 5.21*.

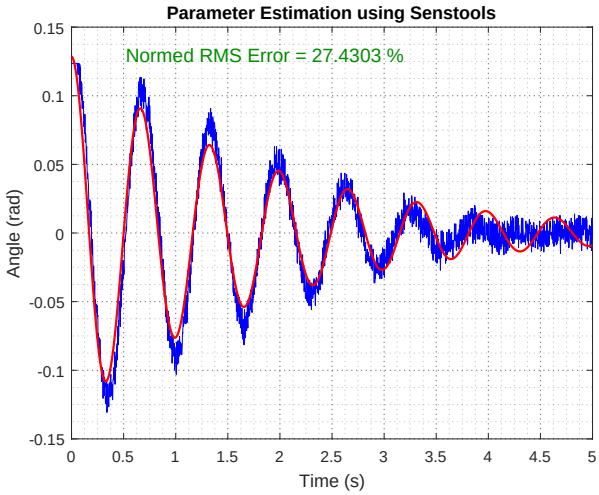


Figure 5.21: Data from the test (red) and final fit with the new parameters (blue) made with Senstools.

Finally, the parameters that Senstools estimates for the model are $J_F = 4,8 \cdot 10^{-3}\ \mathrm{kg \cdot m^2}$ and $B_F = 7,7 \cdot 10^{-3}\ \mathrm{m \cdot s \cdot rad^{-1}}$. The normed RMS error is $27,4\ \%$ compared to the $27,54\ \%$ shown in *figure 5.20*. In *appendix N* the normed RMS error is calculated only for the first part of the region where the fit of the graphs is best, where Senstools get $22,8908\ \%$ and the gradient descend implementation is $23,0713\ \%$. It can be seen that the previously explained implementation provides a satisfactory result as the error is almost the same, but since Senstools gives a slightly better result its parameters are chosen as the final ones.

5.3 Final Parameters

The final parameters of the system can be seen in 5.2.

Parameter	Value	Units
m_w	$0,222$	kg
l_w	$0,096$	m
J_w	$0,601 \cdot 10^{-3}$	$kg \cdot m^2$
B_w	$17,03 \cdot 10^{-6}$	$N \cdot m \cdot s \cdot rad^{-1}$
m_F	$0,548$	kg
l_F	$0,08498$	m
J_F	$4,8 \cdot 10^{-3}$	$kg \cdot m^2$
B_F	$7,7 \cdot 10^{-3}$	$N \cdot m \cdot s \cdot rad^{-1}$

Table 5.2: Parameters of the whole system.

5.4 Model Testing

Substituting all the constants of *equation (4.34)* with the parameters of the real model results in the final transfer function of the system.

$$G(s) = \frac{-148,8 \cdot s}{s^3 + 1,177 \cdot s^2 - 98,19 \cdot s - 2,783} \tag{5.15}$$

Using *equation (5.15)* it is possible to simulate the response of the system to a step input and compare it with the response of the simulation of the block diagram. This is done to verify that the block diagram is in fact showing the system described in *equation (5.15)*, as seen in *figure 5.22*.

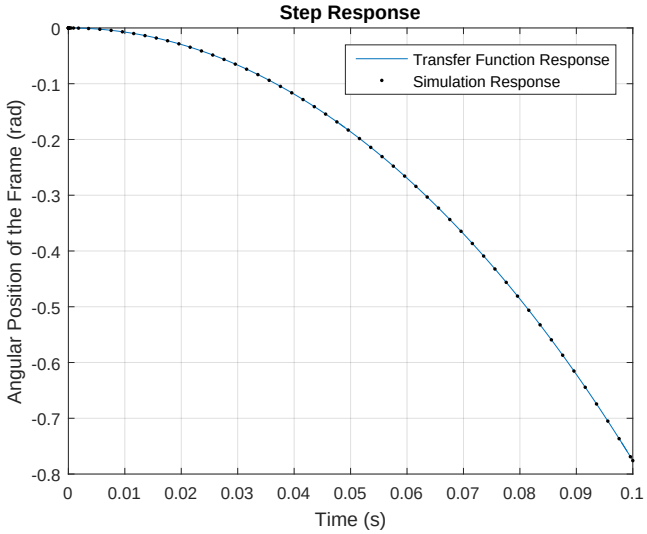


Figure 5.22: Step response comparison between the transfer function from *equation (5.15)* and the block diagram from *figure 4.4*. The conclusion form this is the blockdiagram of the system was made correctly.

In *figure 5.23*, the effect of the linearization is apparent. In the simulation the frame is placed in upright position very slightly off 0 rad.
Figure 5.23 shows the behavior around the frame's pivot point and does not include the platform itself. For this reason, the simulation allows for the frame to fall down and act as a normal pendulum. The nonlinear model shows how the pendulum dampens around its natural equilibrium point, while the linear model keeps increasing the angle.

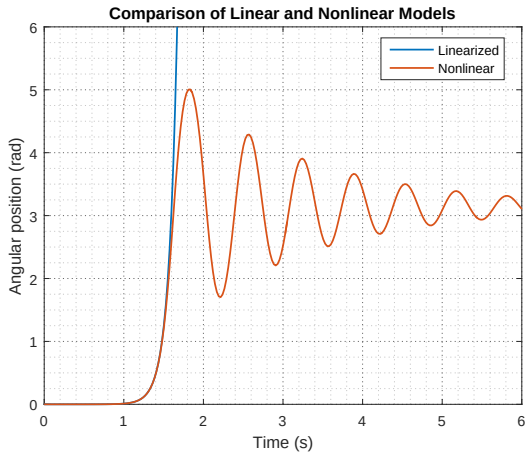


Figure 5.23: Simulation of the linearized model compared to the nonlinear model.

Another simulation can be made to see how the linearization behaves around 0 rad, see *figure 5.24*.

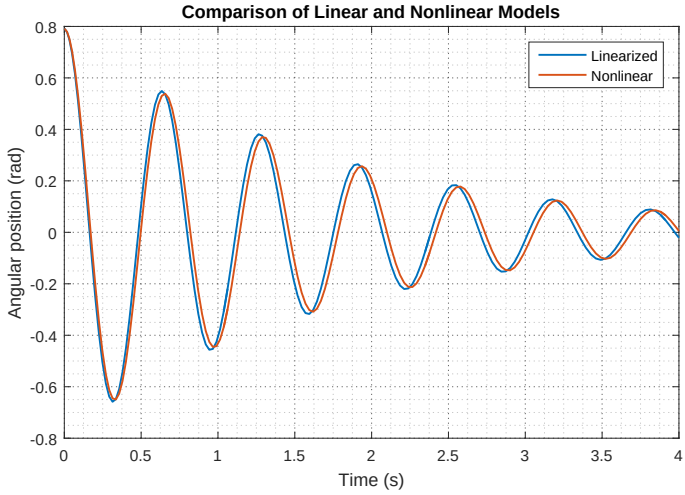


Figure 5.24: Simulation of the linear and non linear models, when they oscillate around equilibrium position. It can be seen that the approximation is very closed to the nonlinear model in the range between $-0{,}79$ rad and $0{,}79$ rad.

From *figure 5.24*, the linearized model is considered a good approximation of the system's behavior in the operational region of $\pm 0,79$ rad.

To further investigate the model, another test is made, see *appendix C*, to determine weather the fall response of the nonlinear model and the real system matches.

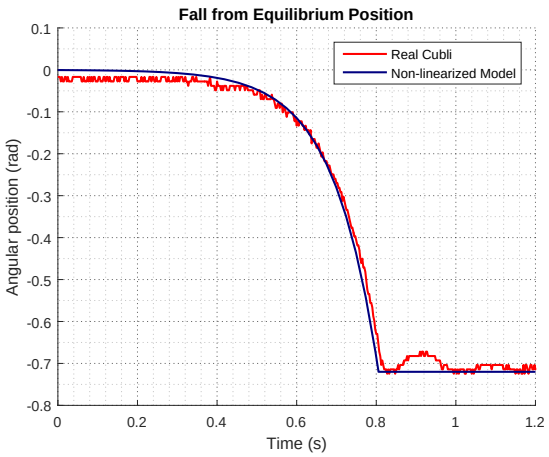


Figure 5.25: Comparison of test and nonlinear model of frame falling from equilibrium position.

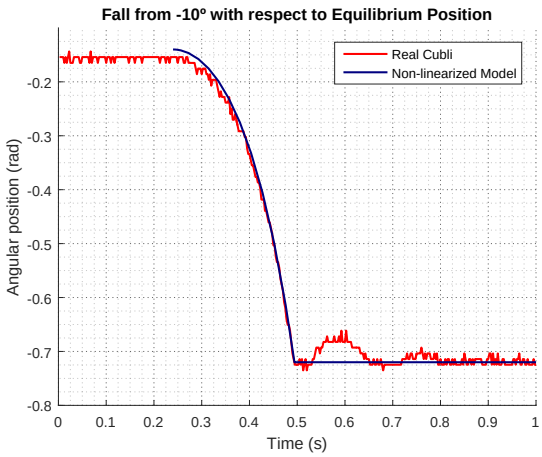


Figure 5.26: Comparison of test and model of frame falling from initial condition $-0,174$ rad.

In *figure 5.25* the Cubli falls from equilibrium position given a very small impulse. When plotting the two data sets the time of the fall is aligned to see if the characteristics of the

simulated fall matches reality.
To show that the simulation matches reality regardless the initial condition, the test is repeated starting from $-0,174$ rad ($-10°$) in both simulation and test, the result is shown in *figure 5.26*.

5.5 Stability Analysis

The linearized model is considered valid within the discussed range of angles ($\pm 0,79$ rad), so it can be used to do a deeper analysis on the behavior of the system.

Root Locus

The Root Locus plot gives information about the location of the poles and zeros in open loop, and how the poles in closed loop will change as the gain of the whole system increases.

As seen in *figure 5.27* and *figure 5.28* the system has one zero ($s = 0$), and three poles ($s = -10,5014$; $s = -0,0283$ and $s = 9,3531$). This means that the system is unstable as it has one pole in the Right Half Plane (RHP) and, moreover, as one of the branches never crosses over to the Left Half Plane (LHP), the system can not be controlled just with a proportional controller.

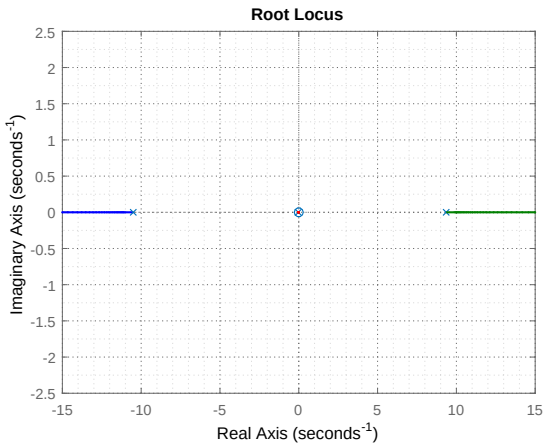


Figure 5.27: Root Locus of the system.

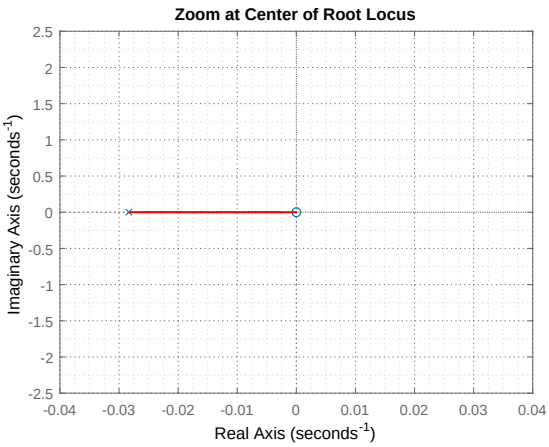


Figure 5.28: Zoom of *figure 5.27* from $s = -0,04$ to $s = 0,04$.

Nyquist Plot

In any transfer function, the zeros of 1+L(s) (L(s) being the open loop function) become the poles of the closed loop system. That is why it is interesting to look at the Nyquist Stability Criterion, which can give information about this topic.

The number of zeros in the RHP of 1+L(s) (Z_{RHP}) is given by the number of poles of L(s) (P_{RHP}) and the number of clockwise encirclements of the Nyquist plot around -1 (N) (a counterclockwise circle has a negative sign in this equation).

$$Z_{RHP} = N + P_{RHP} \tag{5.16}$$

For the system to be stable (Z_{RHP}) has to be zero, which means that there is no pole of the close loop function in the RHP.

In the case of this plant, the number of poles of L(s) = G(s) in the RHP is one and there are no encirclements of -1 in the Nyquist plot (*figure 5.29*). That results in one zero in the RHP. As this zero will be a pole in the closed loop, the system is confirmed as unstable.

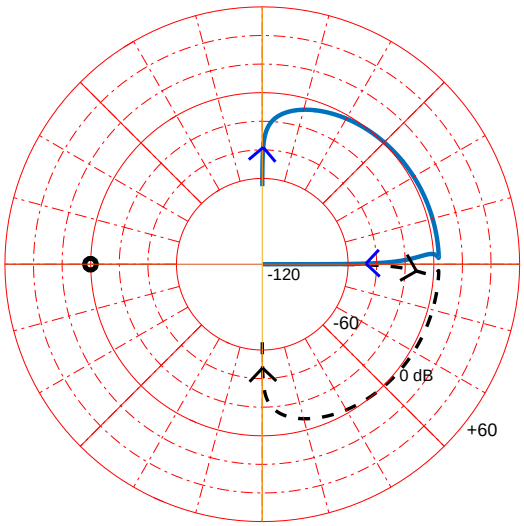


Figure 5.29: Nyquist plot of the plant.

6 | Design Specifications

6.1 Requirements

Based on the design considerations and limitations explained earlier in this report, a list of requirements has been developed.

1. **The Cubli should be able to balance starting from an unstable equilibrium position and null velocity.**

   Considering only naturally occurring perturbations, the Cubli needs to be able to regulate its position by itself, around 0 rad, without falling directly.

2. **The prototype should be able to balance around 0 rad, even though the angle of inclination of the baseplate is changed within a reasonable range, using internally mounted sensors.**

   This ensures that the 2D design of the Cubli can keep its upright position independently from its base plate and therefore, can be more easily translated to a 3D model.

From these established requirements, a controller allowing the Cubli to stabilize in an upright position is to be designed and implemented.

6.2 Further Capabilities Analysis

Moreover, a further investigation on the behavior of the controlled system is to be done in order to determine the capabilities of the final prototype. This includes:

1. **Maximum recovery angle**

   Once the Cubli is balance, its position is forced to change to different angles and the capability of it to go back to the equilibrium position is checked.

2. **Maximum catching angle with no initial velocity of the wheel**

   The maximum starting angle the system can have and still be able to balance is to be tested.

Part II

Design & Implementation

7 | Classical Controller Design

The controller design is a wide field that goes from classic linear PID controllers to more advanced techniques using numerical optimization.

In this chapter, a controller is designed using classical techniques. It is further analyzed to verify if the results fulfilled the requirements.

7.1 Root Locus Design

The Root Locus of a transfer function is a plot of all the possible positions of the closed loop poles when applying a proportional gain K. The number of poles will be the same as the ones in the open loop case and each of them will have an associated branch that shows how it moves along with K. The plot has the following main characteristics:

- The plot is symmetric with respect to the real axis.
- The number of branches, defined as n, is equal to the number of poles in the open loop function.
- There are m branches that end in the zeros of the open loop function, m being the number of zeros.
- There are n-m branches going to infinity.
- The position of the poles changes the behavior of the system as seen in *figure 7.1*

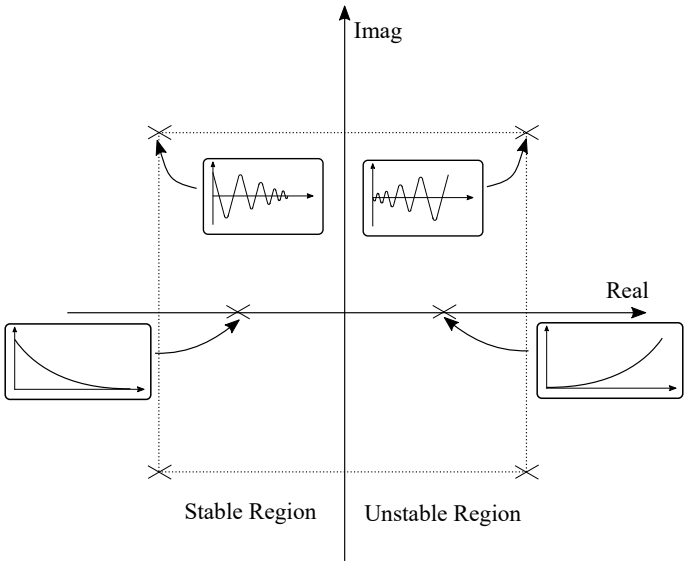


Figure 7.1: Response of a system depending on the position of its poles.

Root Locus design of controllers is based on the fact that the closed loop poles of a system will depend on the poles, zeros and gain of the open loop function. Knowing how the plot changes with the addition of poles and zeros, the branches can be modified to place the poles of the closed loop function where needed.

7.2 Design of the Controller

The design of the controller is based on the root locus plot, where the final location of the poles in closed loop can be seen.

The root locus of the system with a proportional controller can be seen in *figure 7.2*.

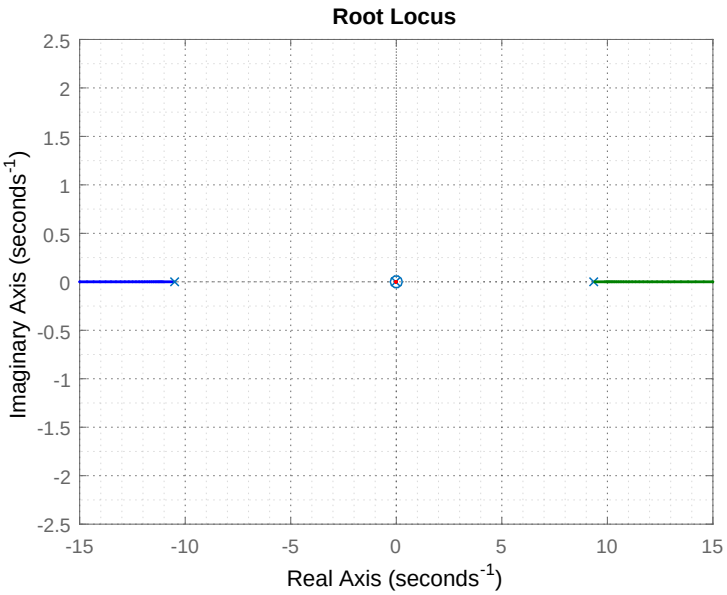


Figure 7.2: Root Locus of the system. It has one zero ($s = 0$), and three poles ($s = -10,5014$; $s = -0,0283$ and $s = 9,3531$).

Looking at the root locus it can be derived that the system can not be controlled using only a proportional controller because there is always a pole in the RHP no matter the gain.

However, a first approximation of the system's behavior in closed loop can be done through a proportional controller. Such a system is tested with a gain of 10 as a controller. The final closed loop poles are placed at $-40,4198$; $-0,0018$ and $39,2448$. In *figure 7.3* it is shown the response of the simulation and the one from the real setup with this proportional controller.

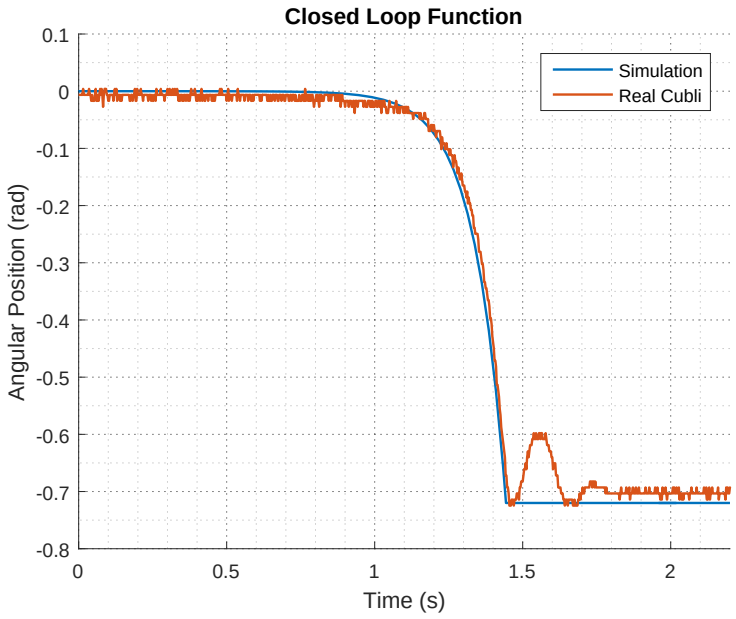


Figure 7.3: Behavior of the closed loop function with a proportional controller, both in simulation and reality.

It is clear that, upon application of a 0 rad reference and a minimal initial offset, the closed loop function with only a gain of 10 has an unstable response. Then the branches of the root locus need to be changed by means of the addition of poles and zeros in the controller.

As there exists one pole in the Right Half Plane (RHP), the controller must also have one there to create two branches which can be attracted to the Left Half Plane (LHP).

Then, two zeros must be placed in the LHP to make the branches enter in the stable region of the plot.

It is also important for the number of poles to be greater than the number of zeros so that the controller is feasible in reality. This means that two poles need to be placed somewhere so they don't affect the behavior of the final system. This can be achieved if they are placed in the LHP and far from the imaginary axis.

Finally, the gain must be adjusted to make the closed loop poles to be in a stable location. The resultant Root Locus can be seen in *figure 7.4*.

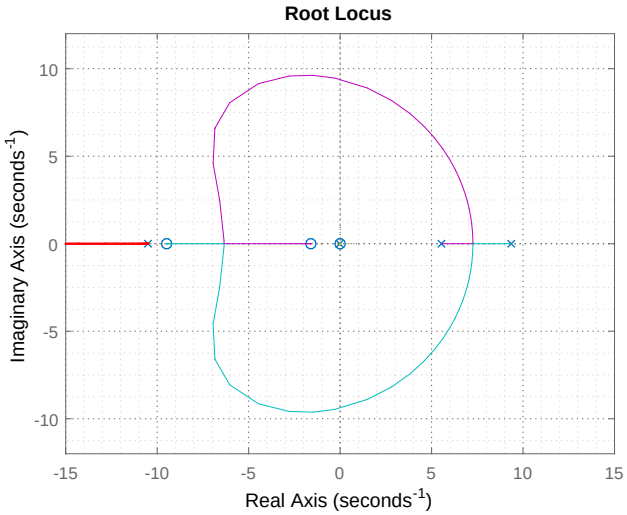


Figure 7.4: Root Locus of the final system.

The final system looks like *figure 7.5* and the transfer function of the controller is give by *equation (7.1)*.

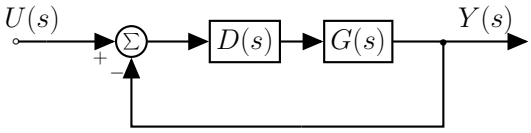


Figure 7.5: Block diagram of the final controlled system.

Where

$$\mathrm{D(s)} = 0,62737 \cdot \frac{(0,1054\ \mathrm{s}+1)\cdot(0,6254\ \mathrm{s}+1)}{(0,1805\ \mathrm{s}-1)\cdot(0,01\ \mathrm{s}+1)\cdot(0,005\ \mathrm{s}+1)} = \\ = -4582,2 \cdot \frac{(\mathrm{s}+9,488)\cdot(\mathrm{s}+1,599)}{(\mathrm{s}-5,54)\cdot(\mathrm{s}+100)\cdot(\mathrm{s}+200)} \tag{7.1}$$

The stability of the controlled system can be then analyzed using the Nyquist plot of the controller and the plant together in open loop, as seen in *figure 7.2*.

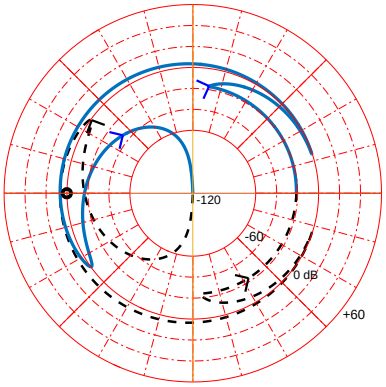


Figure 7.6: Nyquist plot of the system with the controller.

Now there are two poles in the RHP and the number of encirclements around -1 equals -2 (they are counterclockwise). This means that the system should be stable (see *section 5.5*), as the number of zeros in the RHP becomes 0.

7.3 Discretization of the Controller

Since the controller appears stable from the root-locus and Nyquist plots analysis, the controller calculations are to be implemented for testing on the system.
All the controller computations are made on the Beaglebone Black board, which is a computer and cannot run continuously. This means that the controller has to go through a discretization, i.e. that an approximation of the continuous-time controller is made in the discrete-time domain.

The discretization process of a controller consists of mapping frequencies from the continuous domain to the discrete domain (z-domain), with respect to the sampling time, T, of the feedback control system:

$$s = j\omega \rightarrow z = e^{sT} \tag{7.2}$$

The sampling time is chosen to be 0,01 s. This decision is based on the results from *appendix J*, where it can be seen that the code takes around 0,008 s, and the presence of a digital filter in the IMU with cut-off frequency 260 Hz, see *appendix F*. To avoid timing issues and aliasing, a frequency of 100 Hz is chosen and results in the sampling time previously mentioned.

Since, by definition, the discrete domain cannot represent the complete behavior of a system through time, approximations are used to estimate this behavior. Different methods are briefly described and compared hereafter.

The most direct method is the Zero-Order Hold (ZOH) approximation. It mimics a Digital to Analog Converter's output behavior by holding a value of a certain amplitude during a time as long as the defined sampling time. Although it matches reality correctly as long as the signal does not change too much between each sample, it does not fit as well at high frequencies. Thus, it translates into phase lag in the frequency domain and can cause trouble in closed-loop systems.
Another of the most commonly used aproximations is the bilinear transform (or Tustin's method) which is based on the trapezoidal integration principle. A reason to use this method instead of others is that it maps the entire LHP (stable area in the continuous domain) into the unit circle (stable area in the discrete domain)[21].
The bilinear approximation of z is defined as:

$$z \approx \frac{1 + s\frac{T}{2}}{1 - s\frac{T}{2}} \tag{7.3}$$

The inverse transformation of *expression (7.3)* is given by:

$$s \approx \frac{T}{2} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{7.4}$$

In general, the *expression (7.4)* is used to replace s in the continuous-time transfer function of the designed controller. However, due to the non-linear mapping induced by the discretization, a pre-warp of the frequencies can be used before discretizing. This avoids effects of phase lag near the cross-over frequency and thus, also avoids unwanted reduction of gain and phase margins.[22, 23]

Moreover, Matlab has an available function, `c2d()`, designed to convert a continuous system's transfer function into the discrete domain by specifying the sampling time (here, $T = 0,01$ s) and the desired method (`'zoh'`, `'tustin'` or `'prewarp'`). When using the `'prewarp'` option, a supplementary argument is needed, corresponding to the critical fequency, [24].
Here, $33,5 \text{ rad} \cdot \text{s}^{-1}$ is chosen as the critical pre-warp frequency, as it corresponds to the point at which the phase lag is at its maximum on the open loop Bode plot of the continuous plant and controller, see *figure 7.7*.

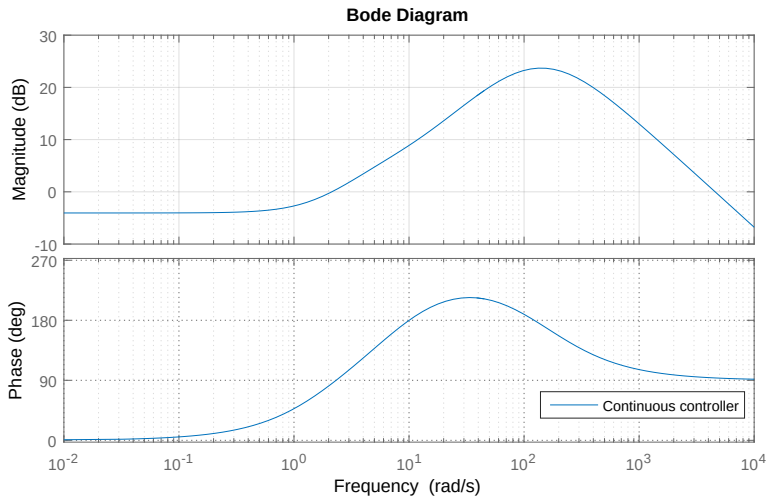


Figure 7.7: Bode plot of the continuous open loop system.

Figure 7.8 shows Bode plots comparing open loops with the original continuous controller against the discretized controller's with ZOH, normal Tustin's method and pre-warping.

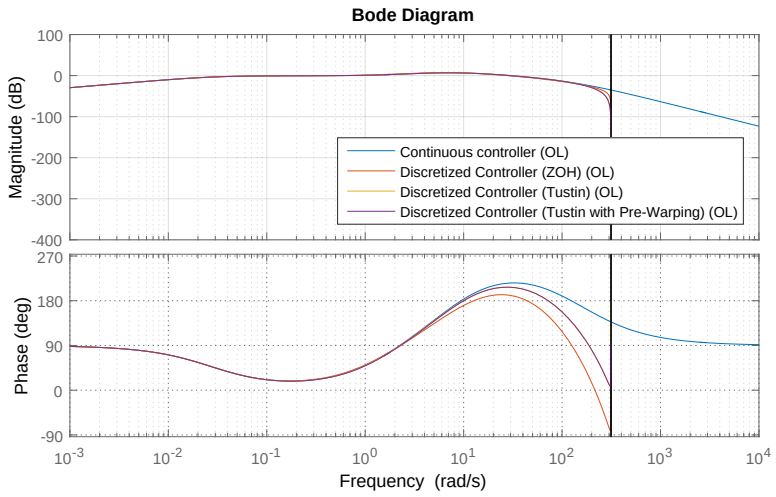


Figure 7.8: Bode plot of the open loop system with the continuous controller (in blue), ZOH discretized controller (in red), Tustin discretized controller (in orange) and pre-warped Tustin discretized controller (in purple).

From *figure 7.8*, it is possible to see that the phase of the ZOH discretized controller diverges from the continuous one's, no later than at $3\,\mathrm{rad}\cdot\mathrm{s}^{-1}$. On the other hand, Tustin discretized controllers match closely with the continuous one until approximately $10\,\mathrm{rad}\cdot\mathrm{s}^{-1}$.
The discrete systems here, are only plotted before the vertical line which represents the Nyquist frequency, i.e. a half of the sampling frequency, chosen earlier in this section. More importantly, the two discrete versions of the controller, the orange one using a simple Tustin method and the purple one using also pre-warping, seem very similar both in frequency and phase.

However, pre-warping should improve the matching between the discretized version and the continuous version of the controller, by re-mapping frequencies to reduce the bilinear transform's distortion. Thus, given some appropriate pre-warp frequency, the discretized system's poles are scaled, so that it becomes realizable.
Thus, the pre-warped discretized controller is chosen for the actual implementation on the Cubli, in the code base, see *section 3.5*, and its discrete transfer function is:

$$\mathrm{D(z)} = \frac{\tau_{\mathrm{m,w}}(\mathrm{z})}{\mathrm{e}_\theta(\mathrm{z})} = \frac{-8{,}314 + 7{,}422\cdot\mathrm{z}^{-1} + 8{,}302\cdot\mathrm{z}^{-2} - 7{,}434\cdot\mathrm{z}^{-3}}{1 - 1{,}382\cdot\mathrm{z}^{-1} + 0{,}3415\cdot\mathrm{z}^{-2} + 0{,}001\,638\cdot\mathrm{z}^{-3}} \tag{7.5}$$

In this section, the controller designed from the root locus has been discretized. The next step is to translate this discretization in a form that can be run on the micro-computer running the calculations, i.e. the BeagleBone Black.

7.4 Implementation of the Controller

To implement a discrete controller in a code environment, the preferred form of equation is the difference equation. This is why *equation (7.5)* is transformed into:

$$\begin{aligned}\tau_{\mathrm{m}}[\mathrm{n}] = &-8{,}314 \cdot \mathrm{e}_\theta[\mathrm{n}] + 7{,}422 \cdot \mathrm{e}_\theta[\mathrm{n}-1] + 8{,}3023 \cdot \mathrm{e}_\theta[\mathrm{n}-2] - 7{,}434 \cdot \mathrm{e}_\theta[\mathrm{n}-3] \\ &+1{,}382 \cdot \tau_{\mathrm{m}}[\mathrm{n}-1] - 0{,}3415 \cdot \tau_{\mathrm{m}}[\mathrm{n}-2] - 0{,}001\,638 \cdot \tau_{\mathrm{m}}[\mathrm{n}-3] \qquad [\mathrm{N}\cdot\mathrm{m}]\end{aligned} \tag{7.6}$$

Where:

τ_{m}	is the wanted motor torque	$[\mathrm{N}\cdot\mathrm{m}]$
e_θ	is the error between wanted and measured frame angle	[rad]
$\mathrm{x}[\mathrm{n}-\mathrm{m}]$	is the m-th previous state of a signal x, m = 0,1,2,3	$[\cdot]$

To match the code convention defined in *section 3.5*, this equation is put into a function named `AAU3_DiscSISOTool()`, as seen in listing 7.1. The coefficients a and b are initialized in the `AAU3_DiscSISOTool_initialize()` function.

```
SISOT_P_Out_Sig_struct_T AAU3_DiscSISOTool(const real_T x_hat[3])
{
  /** Declaration of a temporary variable */
  SISOT_P_Out_Sig_struct_T SISOT_P_U;

  /** Signal shifting */
  SISOT_PComp.e_del[3] = SISOT_PComp.e_del[2];
  SISOT_PComp.e_del[2] = SISOT_PComp.e_del[1];
  SISOT_PComp.e_del[1] = SISOT_PComp.e_del[0];

  SISOT_PComp.taum_del[3] = SISOT_PComp.taum_del[2];
  SISOT_PComp.taum_del[2] = SISOT_PComp.taum_del[1];
  SISOT_PComp.taum_del[1] = SISOT_PComp.taum_del[0];

  /** New calculations */
  // On-the-instant error
  SISOT_PComp.e_del[0] = SISOT_PComp.theta_ref - x_hat[0];

  // Controller job
  SISOT_PComp.taum_del[0] = SISOT_PComp.K * (SISOT_PComp.a[0] * SISOT_PComp.e_del[0] + ↩
      SISOT_PComp.a[1] * SISOT_PComp.e_del[1] + SISOT_PComp.a[2] * SISOT_PComp.e_del[2]↩
       + SISOT_PComp.a[3] * SISOT_PComp.e_del[3] + SISOT_PComp.b[1] * SISOT_PComp.↩
      taum_del[1] + SISOT_PComp.b[2] * SISOT_PComp.taum_del[2] + SISOT_PComp.b[3] * ↩
      SISOT_PComp.taum_del[3]);

  // Current saturation as a preventive protection
  if(TORQUE_2_CURRENT * SISOT_PComp.taum_del[0] > 4)
    SISOT_PComp.taum_del[0] = K_T * 4;//SISOT_PComp.taum_del[1];
  else if(TORQUE_2_CURRENT * SISOT_PComp.taum_del[0] < -4)
    SISOT_PComp.taum_del[0] = K_T * -4;//SISOT_PComp.taum_del[1];

  SISOT_P_U.I_m = TORQUE_2_CURRENT * SISOT_PComp.taum_del[0];

  return SISOT_P_U;
}
```

Listing 7.1: Code for the implementation of the controller designed from root locus in C++.

7.5 Analysis of the Controller

The first step is to simulate both the continuous and discrete controller with the model of the system and analyse the behavior of the whole closed loop system.

This is done not only to see the behavior of the designed controller but also to verify that the discretized controller matches the original continuous one.

With a constant reference of 0 rad and a disturbance in the form of a torque applied to the frame of 0,55 Nm, the responses are the ones shown in *figure 7.9* and *figure 7.10*.

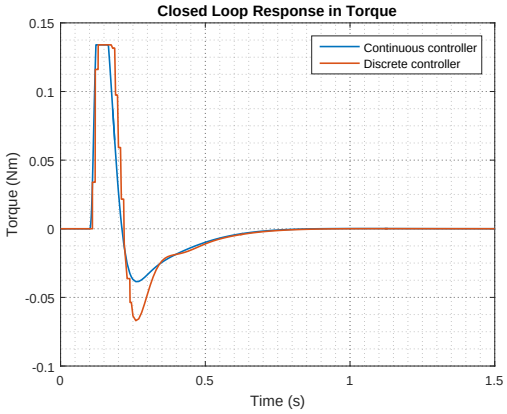


Figure 7.9: Controller's output (torque) response in the control loop with the continuous (blue) and discrete (red) controllers.

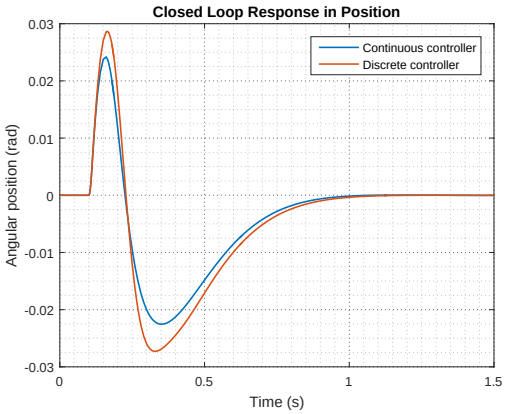


Figure 7.10: Closed loop response of the continuous (blue) and discrete (red) controllers.

Both controllers seem to have a good behavior and both reach the desired final position in simulation when a disturbance is applied.

The implementation of the controller in the real model gives the results in *appendix K*, where it can be seen that the controller is able to balance the Cubli. However, the behavior is marginally stable and the reason seem to be that there is no control on the velocity of the reaction wheel. This means that another kind of controller, which also takes care of the velocity of the wheel, may result in a better behavior.

One way could be using a cascade controller to control both the velocity of the wheel and the position of the frame. In *figure 7.11* it can be seen the block diagram for a cascade control, with the particularity of a feedback from the position of frame to the velocity of the wheel.

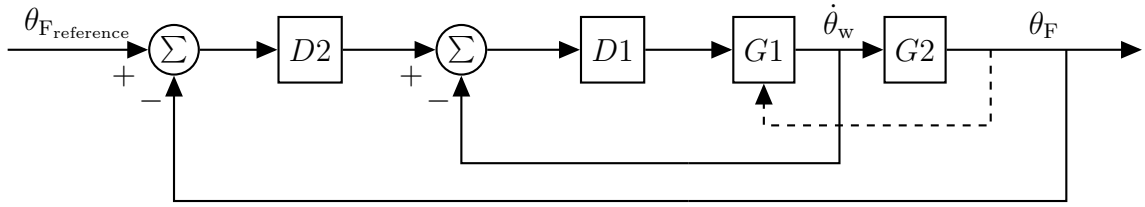


Figure 7.11: Block diagram for a cascade control (the dotted line corresponds to a feedback in the Cubli system that is not present in the normal cascade control).

The problem of this solution in the case of the Cubli is that both variables to control are coupled together, which means that it is not possible to split the plant in a way that there is no feedback between them.[25]

Another way is to use a state space approach, since now the system will have two outputs to control. This controller solution is further detailed in *chapter 8*.

8 | State Space Controller

State space representation describes a system with a set of first-order differential equations. This can be done through the use of state variables, which describe internal states of the system. This representation gives a very convenient way of analyzing systems that have several inputs and outputs.[21]

In this chapter the state space description of the system is derived and a controller capable of maintaining it in equilibrium position is designed.

8.1 State Space Representation of the System

The first step to give a representation of the system in state space is to choose the input, output and state variables. In this case the state variables are chosen to be the position of the frame and the velocities of frame and wheel (θ_F, $\dot{\theta}_\mathrm{F}$, $\dot{\theta}_\mathrm{w}$). The input is the torque from the motor (τ_m) and the chosen outputs are the ones to control, (θ_F, $\dot{\theta}_\mathrm{w}$).

$$\mathbf{x} = \begin{bmatrix} \theta_F \\ \dot{\theta}_F \\ \dot{\theta}_w \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} \theta_F \\ \dot{\theta}_w \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} \tau_m \end{bmatrix} \tag{8.1}$$

The relationship between them is given in the form of first-order differential equations:

$$\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u}) \tag{8.2}$$

$$\mathbf{y} = \mathrm{g}(\mathbf{x}, \mathbf{u}) \tag{8.3}$$

In this case f and g can be derived from *equation (4.25)* and *equation (4.24)*. They are non-linear functions since there exist a sinusoidal term in both.

However, it has been shown in *section 4.2* that a linear approximation near the equilibrium point is possible. Then, the system can be described in the form of linear state space equations as seen in *equation (8.4)* and *equation (8.5)*.

$$\dot{\mathbf{x}}(\mathrm{t}) = \mathbf{A} \cdot \mathbf{x}(\mathrm{t}) + \mathbf{B} \cdot \mathbf{u}(\mathrm{t}) \tag{8.4}$$

$$\mathbf{y}(\mathrm{t}) = \mathbf{C} \cdot \mathbf{x}(\mathrm{t}) + \mathbf{D} \cdot \mathbf{u}(\mathrm{t}) \tag{8.5}$$

Where:

- $\mathbf{A} = \frac{\partial}{\partial \mathbf{x}} \mathrm{f}(\mathbf{x_o}, \mathbf{u_o})$ is the 3x3 state matrix
- $\mathbf{B} = \frac{\partial}{\partial \mathbf{u}} \mathrm{f}(\mathbf{x_o}, \mathbf{u_o})$ is the 3x1 input matrix
- $\mathbf{C} = \frac{\partial}{\partial \mathbf{x}} \mathrm{g}(\mathbf{x_o}, \mathbf{u_o})$ is the 2x3 output matrix
- $\mathbf{D} = \frac{\partial}{\partial \mathbf{u}} \mathrm{g}(\mathbf{x_o}, \mathbf{u_o})$ is the 2x1 feedforward matrix

The state space description can be seen also in the form of a block diagram like the one from *figure 8.1*.

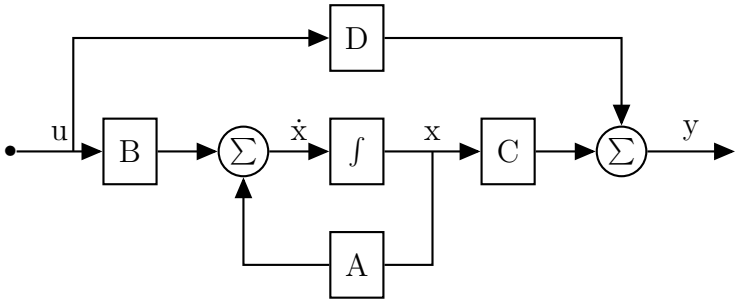


Figure 8.1: Block diagram of the state space representation of the Cubli system.

This matrices can be obtained from the linearized equations of the system from 4.2, given the final system description as *equation (8.6)* and *equation (8.7)*.

$$\dot{\mathbf{x}}(\mathrm{t}) = \begin{bmatrix} 0 & 1 & 0 \\ \frac{(\mathrm{m_F}\cdot\mathrm{l_F}+\mathrm{m_w}\cdot\mathrm{l_w})\cdot\mathrm{g}}{\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2} & -\frac{\mathrm{B_F}}{\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2} & \frac{\mathrm{B_w}}{\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2} \\ -\frac{(\mathrm{m_F}\cdot\mathrm{l_F}+\mathrm{m_w}\cdot\mathrm{l_w})\cdot\mathrm{g}}{\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2} & \frac{\mathrm{B_F}}{\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2} & \frac{(\mathrm{J_w}+\mathrm{J_F}+\mathrm{l_w}^2\cdot\mathrm{m_w})\cdot\mathrm{B_w}}{\mathrm{J_w}\cdot(\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2)} \end{bmatrix} \cdot\mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ -\frac{1}{\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2} \\ \frac{\mathrm{J_w}+\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2}{\mathrm{J_w}\cdot(\mathrm{J_F}+\mathrm{m_w}\cdot\mathrm{l_w}^2)} \end{bmatrix} \cdot\mathbf{u}(\mathrm{t}) \tag{8.6}$$

$$\mathbf{y}(\mathrm{t}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdot\mathbf{u}(\mathrm{t}) \tag{8.7}$$

Substituting the values for the parameter in *equation (8.6)* and *equation (8.7)* gives the final description for the system.

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 98,2208 & -1,1458 & 0,0025 \\ -98,2208 & 1,1458 & -0,0309 \end{bmatrix} \cdot \mathbf{x}(t) + \begin{bmatrix} 0 \\ -148,8077 \\ 1812,7013 \end{bmatrix} \cdot \mathbf{u}(t) \tag{8.8}$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{x}(t) \tag{8.9}$$

8.2 System Analysis in State Space

The stability of the system can be derived from matrix **A**. The denominator of the transfer function of the system is equal to the characteristic polynomial of **A**, which

means that its eigenvalues are the poles of the system. The eigenvalues of $\mathbf{A}$ are, as expected, $-10,5014$; $9,3531$ and $-0,0283$, with a pole in the RHP since the system is unstable by nature.

Once the system is confirmed as unstable, its controllability in state space must be determined. The controllability refers to the ability to go from one state to another in a finite amount of time by means of admissible inputs. To conclude on that, the controllability matrix, $\boldsymbol{\zeta}$, must be analyzed.

$$\boldsymbol{\zeta} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} \end{bmatrix} = \begin{bmatrix} 0 & -149 & 175 \\ -149 & 175 & -14817 \\ 1813 & -226 & 14824 \end{bmatrix} \tag{8.10}$$

The controllability matrix, $\boldsymbol{\zeta}$, in this case is full rank which means that the system is controllable.

Another important characteristic for the design of controllers is the observability, which refers to the capability of inferring the internal states knowing the outputs of the system. In this case there exists a matrix which only includes $\mathbf{C}$ and $\mathbf{A}$, meaning that the observability has no relation with the inputs of the system.

$$\boldsymbol{\mathcal{O}} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ -98,22088 & 1,1458 & -0,0309 \\ 98,22088 & -1,1458 & 0,0025 \\ 115,5755 & -99,5691 & 0,0039 \end{bmatrix} \tag{8.11}$$

This observability matrix, $\boldsymbol{\mathcal{O}}$, also has full rank so the states can be observed.

8.3 Design of the Controller in State Space

The aim of the controller is to maintain the system in equilibrium position, which means that the reference for each state is always zero.

With this assumption a new state matrix can be created such that its poles are placed in the LHP and then the system becomes stable. This is done by adding a state feedback and a 3x1 gain matrix to the whole system, as seen in *figure 8.2*.

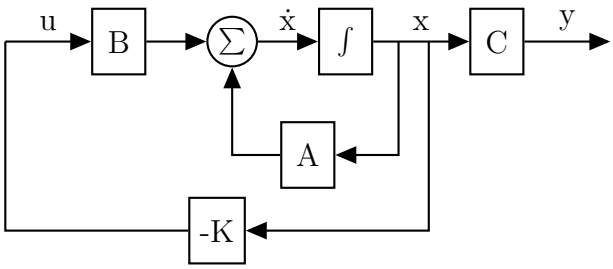


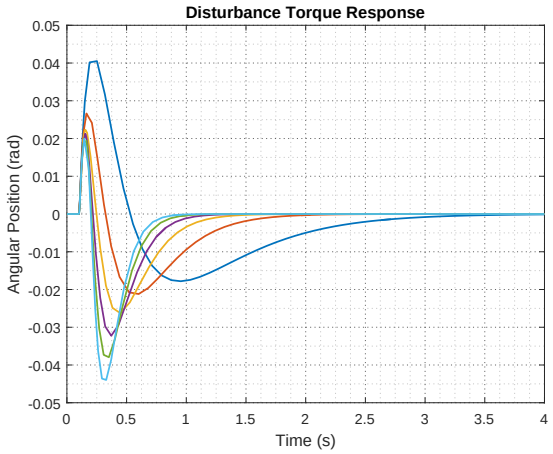
Figure 8.2: Block diagram with the state feedback.

This new configuration gives a new equation for $\dot{\mathbf{x}}$:

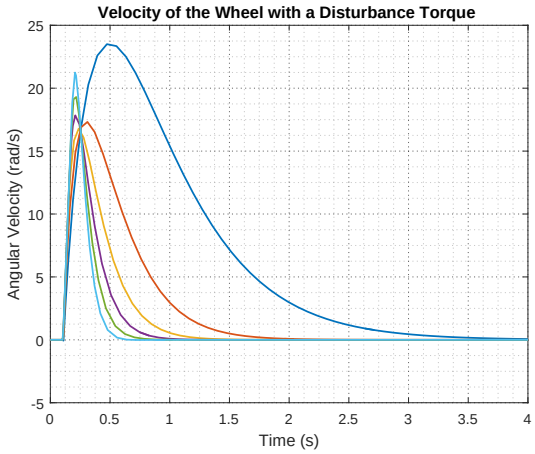
$$\dot{\mathbf{x}}(\mathbf{t}) = (\mathbf{A} - \mathbf{BK}) \cdot \mathbf{x}(\mathbf{t}) \tag{8.12}$$

The objective is to find K such that the eigenvalues of $\mathbf{A} - \mathbf{BK}$ are placed at the LHP. This gain matrix can be found using the Matlab command `place(A,B,P)`, where P is a vector containing the desired position of the new poles.

To choose the position of these poles two simulations are made for different values. In the first one the Cubli starts from equilibrium and a disturbance torque is applied, while in the second one it starts from an initial angle different from 0 rad and with an initial velocity of the frame. It can be observed that the velocity of the frame is in all the cases 0 rad $\cdot$ s^{-1}.

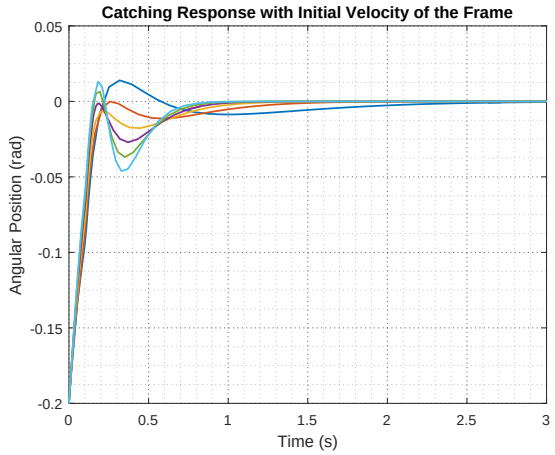


(a) Response of the system to the disturbance starting from angle 0 rad.

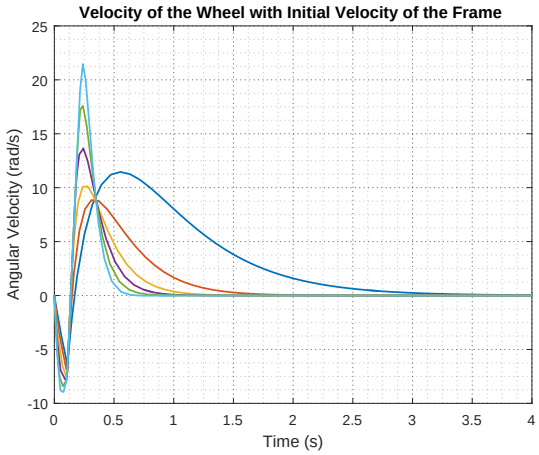


(b) Velocity of the reaction wheel.

Figure 8.3: A small disturbance torque is applied to the frame. The response is simulated with a number of controllers that have different locations of poles. All the controllers slow down the wheel to 0 rad $\cdot$ s^{-1}.



(a) Response starting from $0,2$ rad and an initial velocity of the frame.



(b) Velocity of the reaction wheel.

Figure 8.4: The response is simulated with a number of controllers that have different locations of poles. All the controllers slow down the wheel to 0 rad $\cdot$ s^{-1}.

The final combination of poles is -6, -7 and -15, which gives the yellow responses in *figure 8.3a* and *figure 8.4a*. This seems a good option since its response is quite fast but not with too much overshoot. The gain matrix for this combination results in the following:

$$\mathbf{K} = \begin{bmatrix} -2,3021 & -0,2274 & -0,0039 \end{bmatrix} \tag{8.13}$$

8.4 State Space Controller Implementation

The controller in state space results to be three gains that do not need to be discretized to implement them in the real system. Listing 8.1 contains the code for the function that calculates the action of control. `x_hat` contains the states of the system, being the angle of the frame (`x_hat[0]`), the velocity of the frame (`x_hat[1]`) and the velocity of the wheel (`x_hat[2]`). `x_hat` is the equivalent of vector $\mathbf{x}$. The inner product between $\mathbf{K}$ and $\mathbf{x}$ is calculated on line 11 in code listing 8.1. On line 14 the found output torque is converted to a current, since the motor control board takes a current as reference. The `TORQUE_2_CURRENT` constant is found by taking the inverse of the motor torque constant.

```
/** Runs the controller based on the feedback signal in x_hat */
LSF_COutput_struct_T AAU3_DiscLinFeedback2(const real_T x_hat[3])
{
/** Variable declarations */
LSF_COutput_struct_T LSF_Sig_Out;

/** Controller calculations */
LSF_Controller.tau_m = 0;
for (int i = 0; i < 3; i++)
{
LSF_Controller.tau_m += LSF_Controller.K[i] * x_hat[i];
}

LSF_Sig_Out.I_m = LSF_Controller.tau_m * TORQUE_2_CURRENT;

return LSF_Sig_Out;
}
```

Listing 8.1: Code for the implementation of the State Space Controller. The feedback from the Cubli is contained in the array x_hat.

The important part of the implementation, and in some cases it can give problems, is the measurement of the states that are involved in the feedback. In the case of the Cubli, there are three states to be measured: the position and velocity of the frame and the velocity of the wheel.

The first one can be measured using the values from the potentiometer or using built-in sensors like the IMU, as will be explained in detail in *chapter 9*.

The velocity of the frame is measured using the gyroscopes present in both IMUs, which give an angular velocity value.

Finally, information about the velocity of the wheel is measured with a tachometer attached to the motor and send to the BeagleBone through the motor control board.

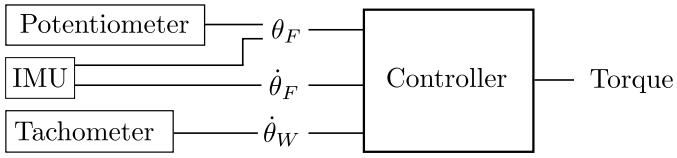


Figure 8.5: Internal states measurements for the implementation of the controller.

9 | Angle Measurement with Built-in Sensors

It is a goal to get the Cubli setup to balance the frame in an upright position independently of the baseplate orientation feasible limits.

As described in *section 3.4* there are two IMUs mounted on the frame of the Cubli (shown on *figure 3.1*). They can be used to achieve this goal since they only depend on the absolute angle and they can also be used in the full Cubli. In this case it has been decided to use IMU number one for the calculations, but number two would be valid too.

9.1 Angle Calculations from the IMUs

The angular position of the Cubli can be calculated using the accelerometer, the gyroscope or combining both. In this section each of the options is described and analyzed.

Angle from the Accelerometer

The accelerometer measures linear acceleration, and if the accelerometer is attached to a object that does not accelerate with respect to the Earth, the accelerometer will measure the gravitational acceleration. This can be used to determine the orientation of the accelerometer sensor with respect to the Earth [26].
The accelerometers used with the Cubli setup have 3-axis detection as described in *section 3.4*. To determine the angle of the frame (accel_θ_F) with the accelerometer it is needed to get the measurements from two of the three axes. The two measurements needed are the ones in line with the frames movement direction, based on the way the IMU is mounted, as shown in *figure 9.1*.

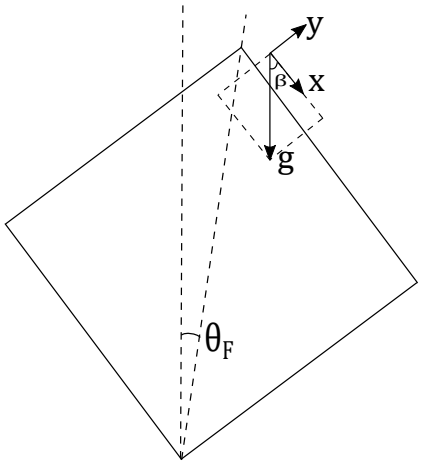


Figure 9.1: Position of the IMU on the setup. The orientation of the accelerometer in relation to the Cubli frame is shown. The sensor is rotated by $\frac{\pi}{4}$ in relation to the angle θ_F. On the IMU the components of the acceleration vector are shown. In case of no movement of the frame these are the components of the gravitational acceleration vector.

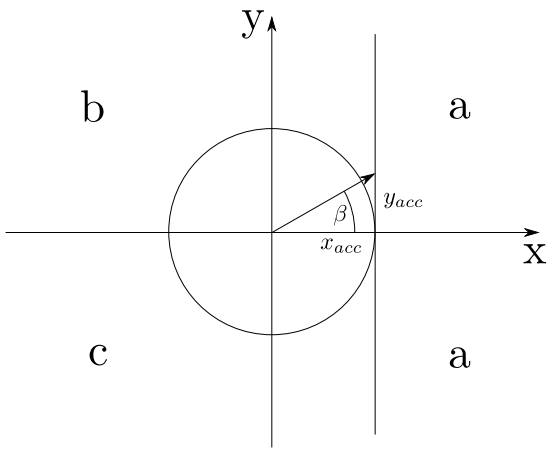


Figure 9.2: atan() is only valid as long as x is positive (a), where the angle (β) is found through the inverse tangent $\beta = \arctan\left(\frac{y}{x}\right)$. If the frame moves more than 45° from equilibrium, then the IMU will have x or y move outside the two areas indicated with a. Angle calculation for area b is $\beta = \pi - \arctan\left(\frac{y}{x}\right)$ and for c it is $\beta = \arctan\left(\frac{y}{x}\right) - \pi$.

Based on the markings on the IMU, those are the x- and y-axis components of the linear acceleration. Taking the two axis measurements shown in *figure 9.1* and the different calculations of β depending on x and y (see *figure 9.2*), the angle can be found using *equation (9.1)* [27].

$$\text{accel\_}\theta_F = \beta + \frac{\pi}{4} \quad \text{[rad]} \quad (9.1)$$

Where:

β is the angle between the x component and the gravity vector [rad]

The offset ($\frac{\pi}{4}$) is added because the IMU is mounted with a $\frac{\pi}{4}$ rotation compared to the orientation of the frame.

A test to measure the angle of the frame with the accelerometer is performed, with the purpose of comparing it to data from the potentiometer. The data from both sensors is read at the same time, to make it possible to compare the accelerometer to the potentiometer to determine if the accelerometer can be used instead of the potentiometer.
A comparison of the two measurements is shown in *figure 9.3*. The accelerometer angle (accel_θ_F) is found by taking the x- and y-axis components and calculating the angle with *equation (9.1)*. The data of the test can be found in *appendix H*.

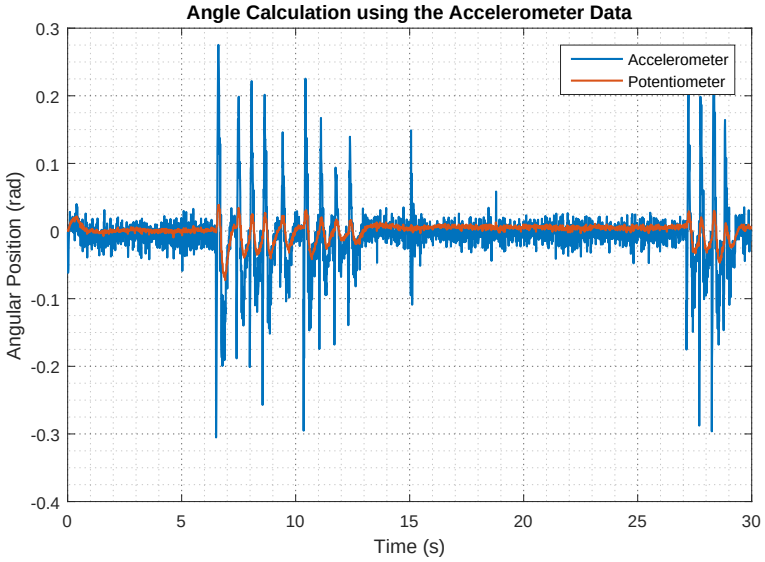


Figure 9.3: The graph shows a comparison of the angle of the frame measured by the potentiometer (orange) and the same angle found with the accelerometer (blue). The accelerometer angle is calculated based on the data from the x- and y-axis components (*appendix H*).

The measurements of the potentiometer and accelerometer are both showing the Cubli frame in the same position. It can be seen that the calculations from the accelerometer has a lot more noise than the potentiometer.

While the Cubli setup tries to balance the frame, the IMU mounted on the top of it will move along with the frame. The spikes are present on the calculated angle from the accelerometer data in *figure 9.3* due to the acceleration of the frame that creates a disturbance on the measurement of the gravitational acceleration [26].
A way to minimize this disturbance would be to move the IMU as close as possible to the point of rotation. This way the accelerometer is still able to measure the gravitational acceleration, but the distance to the pivoting point is shorter and the linear acceleration will be smaller. However, since the design should be portable to the full Cubli, where the pivoting point is always changing, this solution does not seem to be adequate.

Angle from the Gyroscope

The angle of the frame could also be found with the gyroscope (gyro_θ_F), by integrating the measured angular velocity on the axis aligned with the direction of motion of the frame as shown in *figure 4.1*.

$$\text{gyro\_}\theta_F[n] = \int_0^{n \cdot \Delta T} \omega_F \, dt \tag{9.2}$$

Where:

ω_F is the measured angular velocity of the frame $[\text{rad} \cdot \text{s}^{-1}]$

The problem with the data from the gyro is an accumulating error which is caused by the integration done to convert angular velocity into an usable angle. It is also known that the gyros will exhibit a drifting error when experiencing small and slow movement [26]. These problems can be observed in *figure 9.4*.

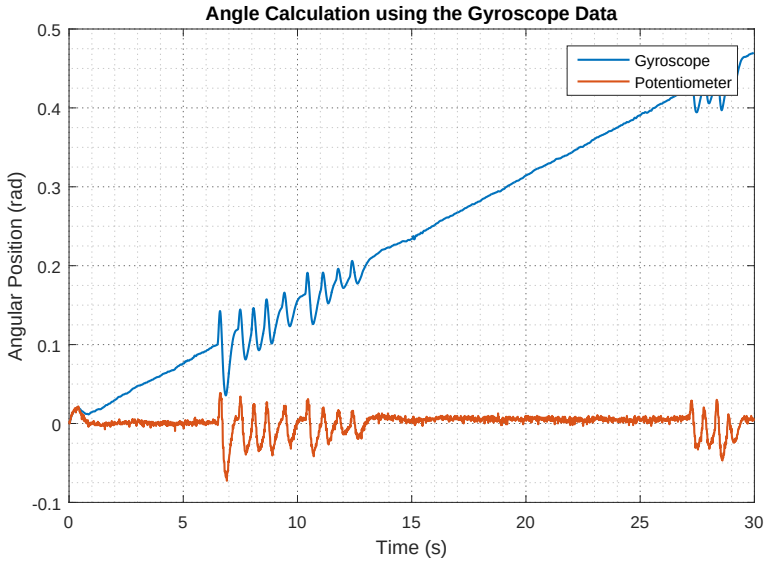


Figure 9.4: Angular position calculation with the data of the gyroscope.

Assuming that the drift is proportional to time (the slope is always constant), a way to minimize its influence on the data would be to do a test while keeping the frame motionless. The inclination of the slope from the data graph multiplied with the sampling time is then the offset that has to be subtracted from each measurement.

It can also be calculated online, while the controller is running. In this case the accelerometer can provide information to compensate this drift. One solution to fuse these two sets of data is to use a complementary filter, explained in detailed in next section.

Data Fusing with Complementary Filter

In order to filter the drift of the gyroscope and the disturbance errors of the accelerometer, a complementary filter can be used to combine both measurements. This is done to get a more reliable angle measurement for both when the frame is moving and when its standing still in the equilibrium point.
The two angle measurements of the IMU are sent through a filter and then summed in order to get an angle of the Cubli's frame. This will result in relying more in the accelerometer data when there is a slow movement and more in the gyroscope when the

movement is fast.[28]

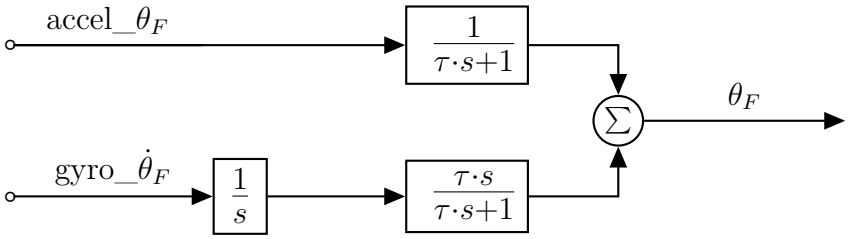


Figure 9.5: Block diagram of the complementary filter setup used for the IMU on the Cubli. The calculated accelerometer angle passes through a low pas filter. The angular velocity measurement from the gyro is integrated and then passed through a high pass filter. Both measurements are then summed to get the angle of the frame. tau refers to the time constant of the filters.

The filter for the IMU is designed as shown in *figure 9.5* with a low pass filter on the measurements from the accelerometer, and an integration followed by a high pass filter on the measurements from the gyroscope [29].[28]

$$\theta_\mathrm{F} = \frac{1}{1+\tau \cdot \mathrm{s}} \cdot \mathrm{accel\_}\theta_\mathrm{F} \tag{9.3}$$

$$\theta_\mathrm{F} = \frac{\tau \cdot \mathrm{s}}{1+\tau \cdot \mathrm{s}} \cdot \frac{1}{\mathrm{s}} \cdot \mathrm{gyro\_}\dot{\theta}_\mathrm{F} \tag{9.4}$$

Combining the two equations yields *equation (9.5)*

$$\theta_\mathrm{F} = \frac{1}{1+\tau \cdot \mathrm{s}} \cdot \mathrm{accel\_}\theta_\mathrm{F} + \frac{\tau \cdot \mathrm{s}}{1+\tau \cdot \mathrm{s}} \cdot \frac{1}{\mathrm{s}} \cdot \mathrm{gyro\_}\dot{\theta}_\mathrm{F} = \frac{\mathrm{accel\_}\theta_\mathrm{F} + \tau \cdot \mathrm{gyro\_}\dot{\theta}_\mathrm{F}}{1+\tau \cdot \mathrm{s}} \tag{9.5}$$

Where τ is the time constant of the filters.

9.2 Discretization of the Complementary Filter

In order to use the complementary filter on the Cubli it needs to be discretized. This is done with the bilinear transformation method, where $\mathrm{s} = \frac{2}{\Delta \mathrm{T}} \cdot \frac{1-\mathrm{z}^{-1}}{1+\mathrm{z}^{-1}}$, with a sample time of 10 ms. Rewriting *equation (9.5)* yields

$$\theta_\mathrm{F} = \frac{1}{1+\tau \cdot \mathrm{s}} \cdot (\mathrm{accel\_}\theta_\mathrm{F} + \tau \cdot \mathrm{gyro\_}\dot{\theta}_\mathrm{F}) \tag{9.6}$$

If s is replaced by its discrete expression the equation can be written in z-domain.

$$\theta_\mathrm{F} = \frac{1}{1+\tau \cdot \frac{2}{\Delta \mathrm{T}} \cdot \frac{1-\mathrm{z}^{-1}}{1+\mathrm{z}^{-1}}} \cdot (\mathrm{accel\_}\theta_\mathrm{F} + \tau \cdot \mathrm{gyro\_}\dot{\theta}_\mathrm{F}) \tag{9.7}$$

$$\theta_\mathrm{F} = \frac{\Delta \mathrm{T} + \Delta \mathrm{T} \cdot \mathrm{z}^{-1}}{(2 \cdot \tau + \Delta \mathrm{T}) - (2\tau - \Delta \mathrm{T}) \cdot \mathrm{z}^{-1}} \cdot (\mathrm{accel\_}\theta_\mathrm{F} + \tau \cdot \mathrm{gyro\_}\dot{\theta}_\mathrm{F}) \tag{9.8}$$

Then *equation (9.2)* is transformed from z-domain to discrete time domain, resulting in the difference equation.

$$\theta_{\mathrm{F}}[\mathrm{n}] = \frac{(2\cdot\tau+\Delta\mathrm{T})}{2\cdot\tau+\Delta}\cdot\theta_{\mathrm{F}}[\mathrm{n}-1] + \frac{\Delta\mathrm{T}}{2\cdot\tau+\Delta\mathrm{T}}\cdot\mathrm{accel\_}\theta_{\mathrm{F}}[\mathrm{n}] + \frac{\Delta\mathrm{T}}{2\cdot\tau+\Delta\mathrm{T}}\cdot\mathrm{accel\_}\theta_{\mathrm{F}}[\mathrm{n}-1] + \frac{\Delta\mathrm{T}\cdot\tau}{2\cdot\tau+\Delta\mathrm{T}}\cdot\mathrm{gyro\_}\dot{\theta}_{\mathrm{F}}[\mathrm{n}] + \frac{\Delta\mathrm{T}\cdot\tau}{2\cdot\tau+\Delta\mathrm{T}}\cdot\mathrm{gyro\_}\dot{\theta}_{\mathrm{F}}[\mathrm{n}-1] \tag{9.9}$$

9.3 Calculation of the Cut-off Frequency

Based on the setup of the complementary filter in *figure 9.5*, the same cut-off frequency is chosen for both low and high pass filter, since it is desired to find the angle of the frame with a gain of 1 (see *figure 9.6*). As it can be seen in *figure 9.7*, if both filters have different cut-off frequencies the gain is less than one at some frequencies, which could be a problem in the calculation of the angular position of the frame.

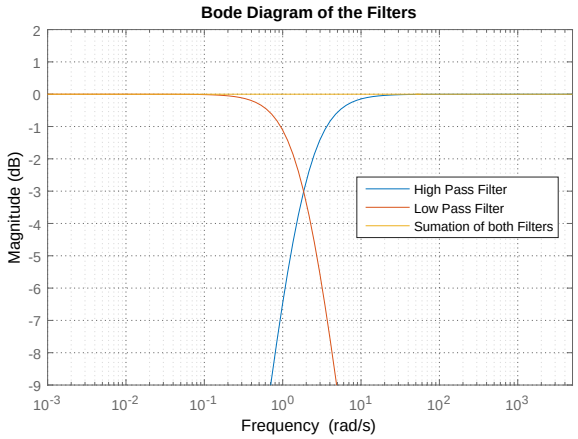


Figure 9.6: Magnitude bode diagrams of the high pass and low pass filters with the same cut-off frequency as well as the summation of both.

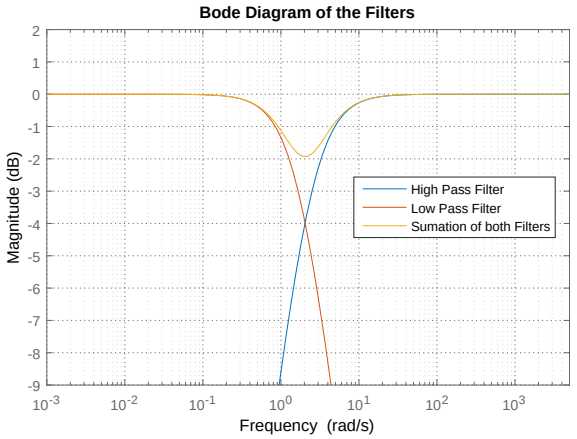


Figure 9.7: Magnitude bode diagrams of the high pass and low pass filters with different cut-off frequencies as well as the summation of both.

The cut-off frequency for the filters can be determined by using Senstools and the data obtained through the test detailed in *appendix H*.

The toolbox uses the data form the potentiometer as the real output of the system, the data from the IMUs as the input and the modeling of the system is done through *equation (9.9)*. With this data Senstools can find the optimal value for τ based on the difference between the angle measured by the potentiometer and an angle calculated from accelerometer and gyroscope measurements done during the same test.

The final fit can be seen in *figure 9.8* and it gives an optimal $\tau = 0{,}5399\,\mathrm{s}$. This results in a cut-off frequency for the filter equal to $1,85\ \mathrm{rad}\cdot\mathrm{s}^{-1}$.

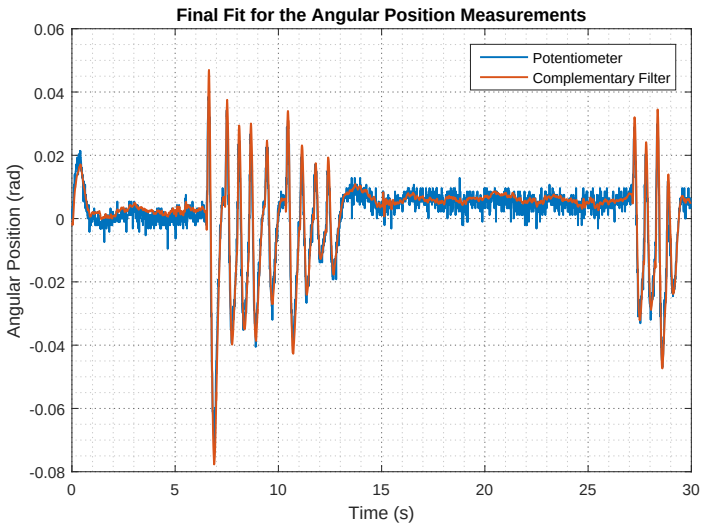


Figure 9.8: Final result of the complementary filter compared with the data from the potentiometer. It can be observed that the two measurements match closely.

9.4 Implementation of the Complementary Filter

Implementing *equation (9.9)* yields to following piece of code.

```
//————————————————————— IMU ——————————————————————//
double Imu::getPosition(double accAngleNow, double gyroVelocityNow, double Ts, int imuNb)
{
    const double acc_off_1 = 0.84;      // accel meass offset
    const double tau = 0.5399;          // cut-off for the complementary filter

    // Coefficients for the complementary equation
    const double K1 = (2 * tau - Ts) / (2 * tau + Ts);
    const double K2 = Ts / (2 * tau + Ts);

    if(imuNb == 1)
    {
        static double acc_angle_1[2] = {accAngleNow + acc_off_1, 0},
        gyro_angle_1[2] = {gyroVelocityNow, gyroVelocityNow},
        comp_angle_1[2] = {acc_angle_1[0], 0};

        // Set old measurement data
        acc_angle_1[1] = acc_angle_1[0];
        gyro_angle_1[1] = gyro_angle_1[0];
        gyro_angle_1[0] = gyroVelocityNow;
        comp_angle_1[1] = comp_angle_1[0];

        acc_angle_1[0] = accAngleNow + acc_off_1;

        //Complementary equation using Tustin
        comp_angle_1[0] = K1 * comp_angle_1[1] + K2 * (acc_angle_1[0] + acc_angle_1[1] + tau * gyro_angle_1[0] + tau * gyro_angle_1[1]);
        return comp_angle_1[0];
    }
}
```

Listing 9.1: Code for the implementation of the complementary filter in C++.

The position is calculated through the function `getPosition()`.

First, the values for the offset of the accelerometer angle and the cut-off frequency are initialized, as well as the two constants for the filter, K1 and K2.

When this function is called the first time, accelerometer, gyroscope and angle arrays are initialized.

The old values from `accel_angle`, `gyro_angle` and `comp_angle` are moved up in their arrays and new data is saved in `gyro_angle` and `accel_angle` on index 0.

Finally, the equation of the complementary filter in implemented to get the angular position of the frame.

Part III

Test & Conclusion

10 | Acceptance Test

This chapter deals with the needed tests that the system with the state space must overcome in order to fulfill the requirements as well as an analysis of the other capabilities, described in *chapter 6*.

10.1 Requirements Test

- The Cubli should be able to balance starting from an unstable equilibrium position and null velocity.

To test this requirement, the Cubli is placed at equilibrium and the controller is then started. The test is done twice, first with the measurements of the potentiometer andthen using the IMU, to ensure that the controller works in both cases.

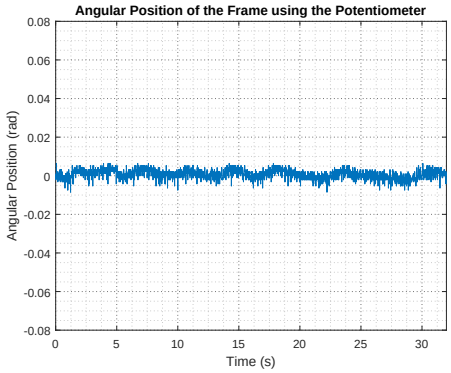


Figure 10.1: Angular position of the frame, measured with the potentiometer, when starting from equilibrium position.

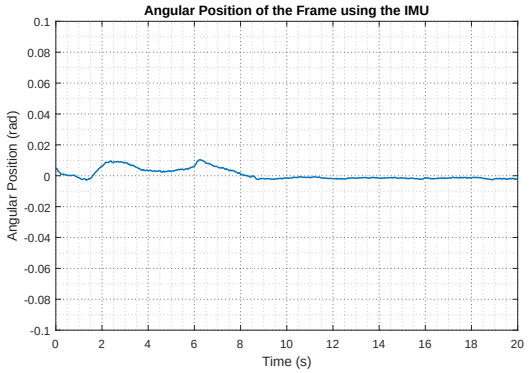


Figure 10.2: Angular position of the frame starting from equilibrium position, calculated with the complementary filter.

The results show that the requirement is fulfilled in both cases since the system is maintained around the equilibrium position.

- The prototype should be able to balance around 0 rad, even though the angle of inclination of the baseplate is changed within a reasonable range, using internally mounted sensors.

To check this requirement a similar test, like the previous one, is done, but is this case the angle of the baseplate is changed to check that the measurements of the IMU are still capable of balance the Cubli. Since the potentiometer is attached to the baseplate its measurements are done in relation to the baseplate's position and can be used to show that baseplate and frame are at different angles.

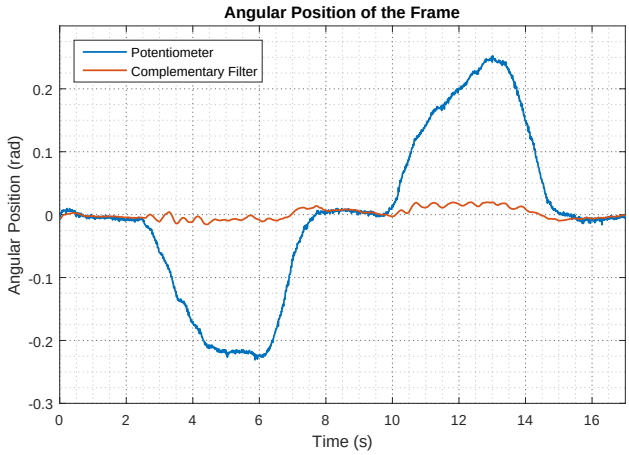


Figure 10.3: Angular position of the frame. The orange curve is the result of the complementary filters angle calculation, which does not depend on the angle of the baseplate. The blue curve shows the measurements of the potentiometer, which are in relation to the baseplate.

The output of this experiment shows that the calculation of the position using the IMU makes the system able to keep in equilibrium position when the angle of the Baseplate is changed. This can be seen on the graph in *figure 10.3*. Here the angle calculated by the complementary filter shows the frame keeping its upright position, while the potentiometer angle changes as the baseplate is lifted up and down.

10.2 Capabilities Analysis

- Maximum recovery angle

To obtain the maximum recovery angle, the Cubli is place in equilibrium at the start. Once the controller has balance it, its position is changed with little disturbances. The result can be seen in *figure 10.4*.

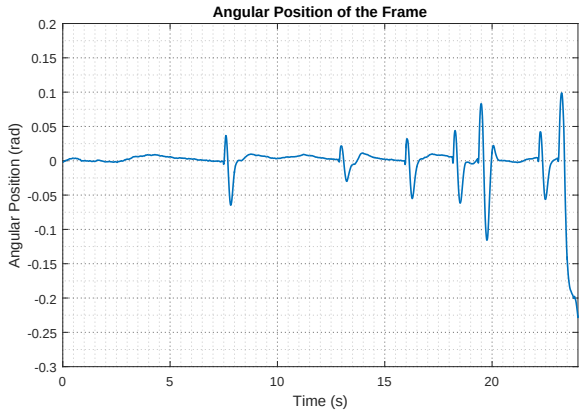


Figure 10.4: Angular position of the frame while disturbances are applied to the frame. The intention is to find the maximum recovery angle of the controller. The controller is not able to recover from a disturbance that pushes the frame over 0,08 rad from resting position, due to the overshoot.

As it can be observed, the controller is able to return to equilibrium position but it creates a remarkable overshoot in the other direction. This behavior limits the recovery angle since the overshoot may drive the Cubli to an angle beyond its handling limit even though the initial disturbance is within it.
It can be seen that the controller is capable of recovering from $0,08$ rad but it fails when the angle is $0,1$ rad. Then, it can be concluded that the maximum recovery angle for the system with the designed controller is $0,08$ rad.

- Maximum catching angle with no initial velocity of the wheel

There exists a limitation in this capability which is given by the maximum current that the motor can provide. As can be seen in *figure 10.5*, if the initial angle is different from 0 rad both the mass of the frame and the mass of the wheel exert an initial torque to the system. This must be overcame by the motor to avoid the Cubli to fall.

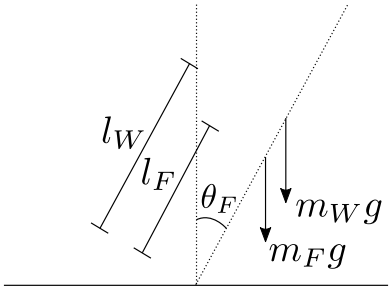


Figure 10.5: Forces acting on the system that create an initial torque when the frame starts from a position different from 0 rad.

The minimum torque (T) that the motor must apply is give by *equation (10.1)*.

$$\mathrm{T} = (\mathrm{m_F} \cdot \mathrm{l_F} + \mathrm{m_w} \cdot \mathrm{l_w}) \cdot \mathrm{g} \cdot \sin(\theta_\mathrm{F}) \qquad [\mathrm{N} \cdot \mathrm{m}] \quad (10.1)$$

Since the torque is restricted by the characteristics of the motor and the control board, the maximum initial angle is derived in *equation (10.2)*.

$$\theta_\mathrm{F} = \operatorname{asin}\left(\frac{\mathrm{T}}{(\mathrm{m_F} \cdot \mathrm{l_F} + \mathrm{m_w} \cdot \mathrm{l_w}) \cdot \mathrm{g}}\right) \qquad [\mathrm{rad}] \quad (10.2)$$

Substituting the values of the maximum torque (see section *section 3.3*) and the parameters of the Cubli (see section *section 5.1*) the maximum possible starting angle is $0,2024$ rad $(11,59°)$. This also applies for the negative angle since the limitation of torque is the same but with opposite sign.

Taking this limit into account some starting angles are tested to check if the controller is able to balance the system with these initial conditions.

As it can be seen in *figure 10.6* and *figure 10.7*, the controller is capable of moving the system to equilibrium position starting from $-0{,}12$ rad and $-0{,}17$ rad (also valid for the

positive angles). It is also noteworthy that as the starting angles goes away from the equilibrium the controller takes more time and there is more oscillations in the response.

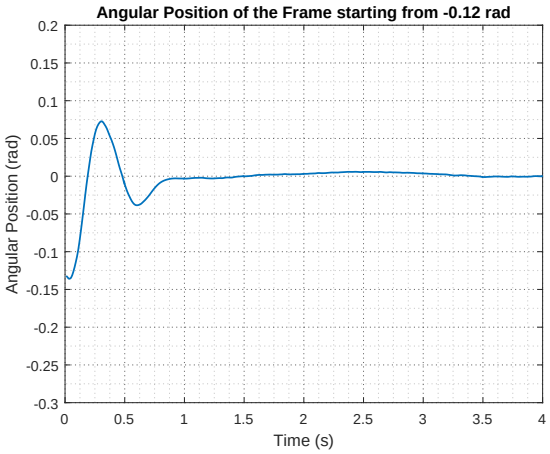


Figure 10.6: Angular position of the frame when the system starts from $-0{,}12\,\text{rad}$ $(-6,87°)$.

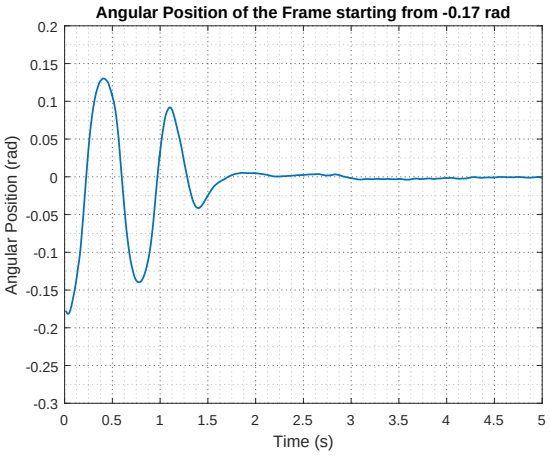


Figure 10.7: Angular position of the frame when it starts from $-0{,}17\,\text{rad}$ rad $(-9,74°)$.

However, when the starting angle is $-0{,}185\,\text{rad}$ $(-10,59°)$ is no longer possible to balance the system in equilibrium position and the frame falls.

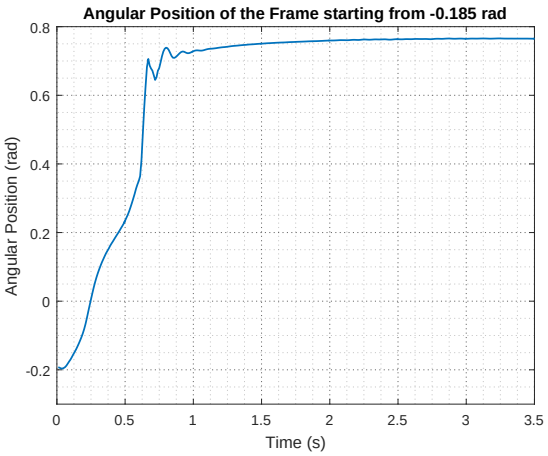


Figure 10.8: Angular position of the frame when it starts from $-0{,}185\,\text{rad}$ $(-10,59°)$.

It can be concluded that the maximum range for the starting angle with zero initial conditions goes from $-0{,}17\,\text{rad}$ to $0{,}17\,\text{rad}$, in which the controller is able to balance the system around equilibrium position.

11 | Discussion

For the state space controller a set of poles were selected, looking at the response of the system for each K matrix, shown in the two simulations, *figure 8.3a* and *figure 8.4a*. Using different pole combinations results in different overshoots and settling time, so another pole combination could be selected if a different behavior of the Cubli was desired. The choice is whether a small overshoot is desired or if the frame is to reach 0 rad faster. Having less overshoot would increase the maximum catching angle with the current system, since the second overshoot with the current used controller is larger than the angle inflicted by a disturbance. A fast response will result in handling better external disturbances if they are applied continuously.

As already mentioned briefly in the complementary filter section, the measurement from the accelerometer in the IMU could be improved by moving the sensor to a position where it would be influenced less by the acceleration of the frame but still be able to measure the angle of the frame. In case of the single frame, that would be as close as possible to the point of rotation fixed to the baseplate.

The fusing of accelerometer and gyroscope measurements can be done using a different type of filter than the used complementary filter. The choice of using a complementary filter was based on the use of it in similar applications [29]. Since the filter showed a good behavior for the problem at hand, no further investigation was needed.

12 | Conclusion

The aim of this project was to work with an unstable nonlinear system and be able to construct an appropriate model and design a controller capable of balancing it around equilibrium position.

First, a pre-analysis of the system has been made, starting from a description of all the components present in the given setup. It also included the derivation of the equations that describe the dynamics and the description of the system in s-domain. Then the parameters of the setup have been found, both with measurements and with an estimation using optimization, to be able to analyze the behavior of the system and compare it with the real model.

Afterwards, a controller to balance the Cubli has been designed using root locus. It has been shown that it was able to keep it in upright position. However, the system had a marginally stable behavior and the reason seemed to be that there was no control on the velocity of the reaction wheel. Thus, it was decided to use a state space design approach, to control both position of the frame and velocity of the wheel.

It was also a requirement to be able to change the angle of the baseplate, which means that the calculation of the angle had to be independent of the inclination. A very convenient option was to use internally mounted sensors for this purpose, so the final chosen solution was to use an IMU present on the setup and calculate the angle through a complementary filter.

Finally, some acceptance tests have been performed to ensure that the final controlled system was able to accomplish the requirements and, similarly, a further analysis has been carried out to check other capabilities of the Cubli.

In conclusion, a control system that can balance the Cubli in upright position independently of the angle of the baseplate, within a reasonable range, has been designed, implemented and tested successfully within the requirements.

13 | Perspective

After designing the one-side prototype, the next step is to build a full cube version. To achieve this goal some changes and extra features must be added to the current design.

The placement of the IMUs on the full Cubli is something that has to be determined. One way to do it if possible is to place just one IMU in the very center of the Cubli.
Alternatively one sensor could be placed in each corner of the Cubli. That would mean using 8 IMUs. This way, there is always at least one sensor very close to the point of rotation, and thus, it should be possible to get measurements with minimal disturbance from linear acceleration. A system that keeps track of the orientation of the Cubli could be implemented, so that the controller receives information from the most reliable sensor.

Each of the three reaction wheels needs to have its own controller. In this case it is not the goal to achieve an upright position, but to perform other maneuvers. This will require a redesigned controller since the chosen one is not able to get a changing reference for the internal states.

The Cubli will have to be able to jump up from a resting position. To do that the reaction wheel has to spin up. Once at the desired speed the wheel will be braked and the resulting torque will raise the Cubli. Now the system has to catch itself. In order to do that a separate controller might have to be designed that handles the jump up part, and then switches to the balancing controller for the balancing functionality.
For the full Cubli, since there are three wheels and three braking systems, all the controllers will need to be coordinated.

Appendix

A | Potentiometer Linearity

Name: Group 630
Date: 15/03 - 2016

Purpose

Finding the linear accuracy of the potentiometer as well as the equilibrium range of the system.

Setup

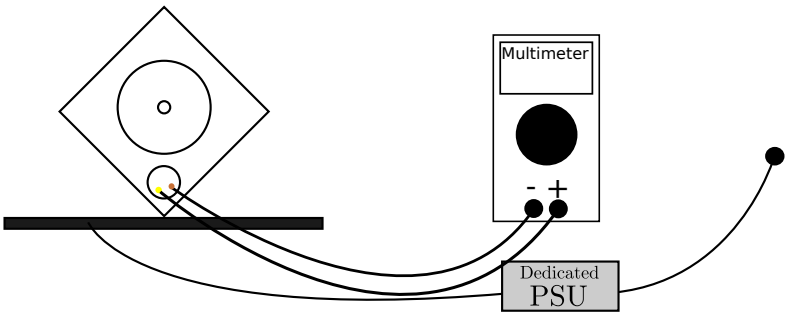


Figure A.1: Setup diagram

List of Equipment

Instrument	AAU-no.	Type
Multimeter	60760	Fluke 189 Multimeter
Dedicated Power Supply of Cubli (24 V - 3 A)	AAU3	XP Power, AEB70US24
Digital Protractor	None	CMT Orange Tools

Procedure

1. Make the setup with connections as seen on *figure A.1*, placing the + connection in the brown cable of the potentiometer and the + connection in the yellow one.
2. Setting the multimeter to measure DC mV.
3. Balance the frame in upright equilibrium position measuring the angle and voltage.
4. Measure the voltage of the potentiometer around Equilibrium point and also min and max angle voltage for every 10°.

Results

Angle from equilibrium in degrees	mV
-41,7	0,066
-40,0	0,067
-38,5	4,25
-30,0	50,55
-20,0	103,66
-10,0	159,20
0	213,64
10,0	264,66
20,0	317,95
30,0	370,74
40,0	425,10
48,65	472,11

Results from Linearity Test

Result of the test shows that below $-39,5°$ the potentiometer has a dead area. The dead area might come from the continuous rotation of the potentiometer, since the measurement are very near to this point where the potentiometer changes. The area have at dead span from 5° to 10°.

The graph shows the measured values according to angle.

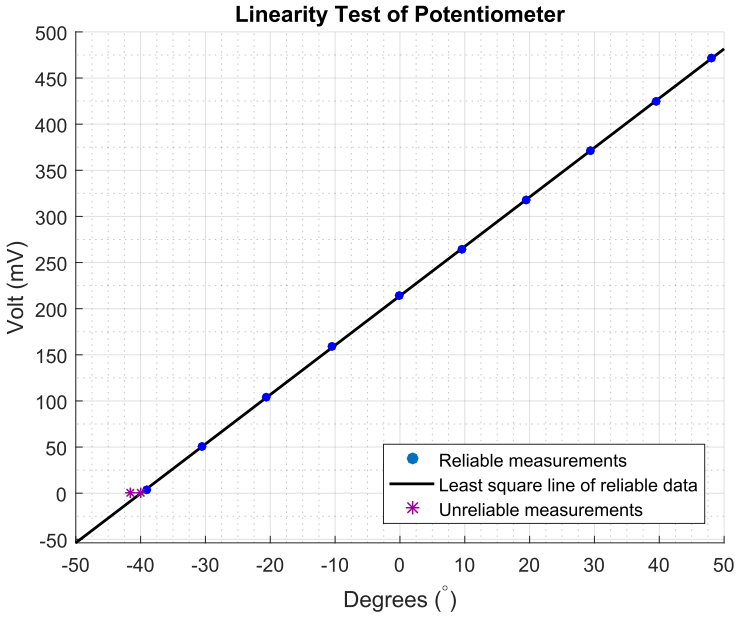


Figure A.2: Result from linearity test

Because of the dead area the potentiometer could be rotated so the frame will be turning in this area, but since the Cubli has been built like this and the code has some hardcoded value of the potentiometer and the area are not used then it will be left as it is. Also because the software is distributed on different machines it has to be changed on every system.

Results of Equilibrium Zone

During the test the equilibrium has varied, and area where it can stand balanced have been measured.

Equilibrium range in degrees	**mV**
-0,44	211,80
-0,05	213,64
0,053	217,00

Since the frame is connected to the baseplate through the potentiometer and this one is kept in place by bearings, the only force keeping the frame standing is the friction between them and potentiometer. This region is about 1°.

B | Potentiometer Angle Resolution

Name: Group 630
Date: 15/03 - 2016

Purpose

Finding the resolution needed for the conversion of potentiometer voltage to angles, along with possible offsets.

Setup

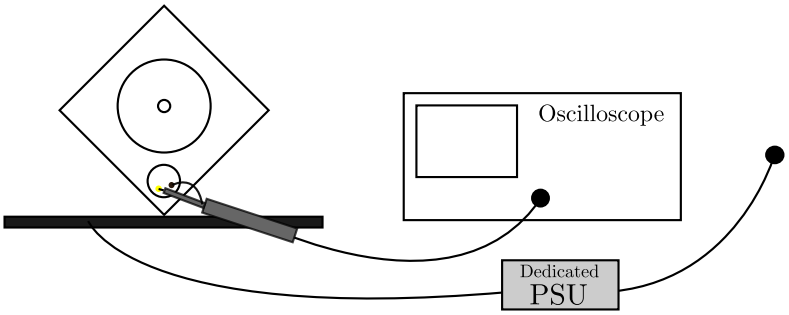


Figure B.1: Setup diagram

List of Equipment

Instrument	AAU-no.	Type
Oscilloscope	61604	Agilent 54621A
Dedicated Power Supply of Cubli (24 V - 3 A)		XP Power, AEB70US24
Probe		1:1

Procedure

1. Make the setup with connections as seen on *figure A.1*, with ground on the brown cable and signal on yellow cable of the potentiometer.
2. Set the oscilloscope on rolling and calibrate so that the full range of the frame movement can be captured on the display.
3. Balance the frame in upright equilibrium position.

4. Move the frame to the left position, hold for a brief duration, then move it over to the right position.
5. Once both the equilibrium, leftmost and rightmost positions are captured on the screen, pushing the stop-button on the scope, to hold keep the measurement.
6. Save the data to the floppy-disk as a CSV file.

Results

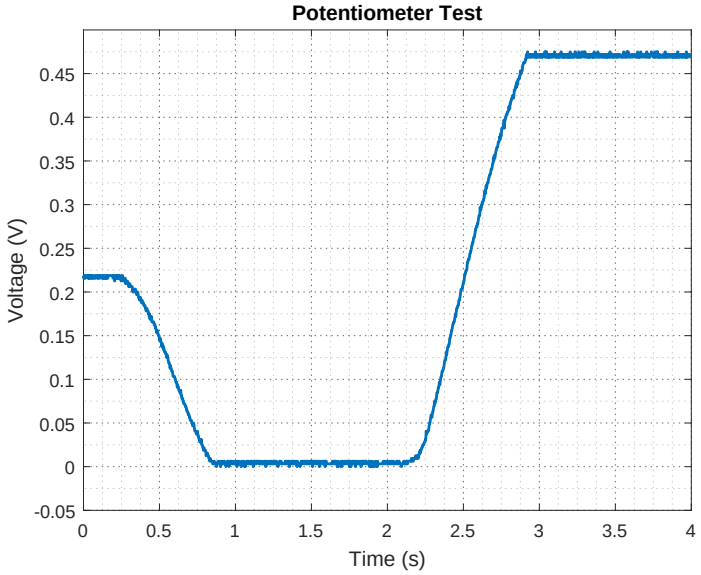


Figure B.2: Raw test data plot, Volt over time.

C | Initial Condition Response from Vertical Position

Name: Group 630
Date: 15/03 - 2016

Purpose

Finding the fall response of the frame from the vertical position, $\theta_F = 0°$, and from a 10° tilted position. Data is used to compare the measured response and the simulation given by the theoretical nonlinear model.

Setup

The wheel is being held in a fixed position with a strip tied to it and the frame. The probe chosen is a 1:1 and is connected to the potentiometer with probe to yellow cable and ground clamp to brown cable. The power supply has to be turned on in order to get readouts from the potentiometer. A sponge is placed on the rubber pad in order to damp the impact of the frame.

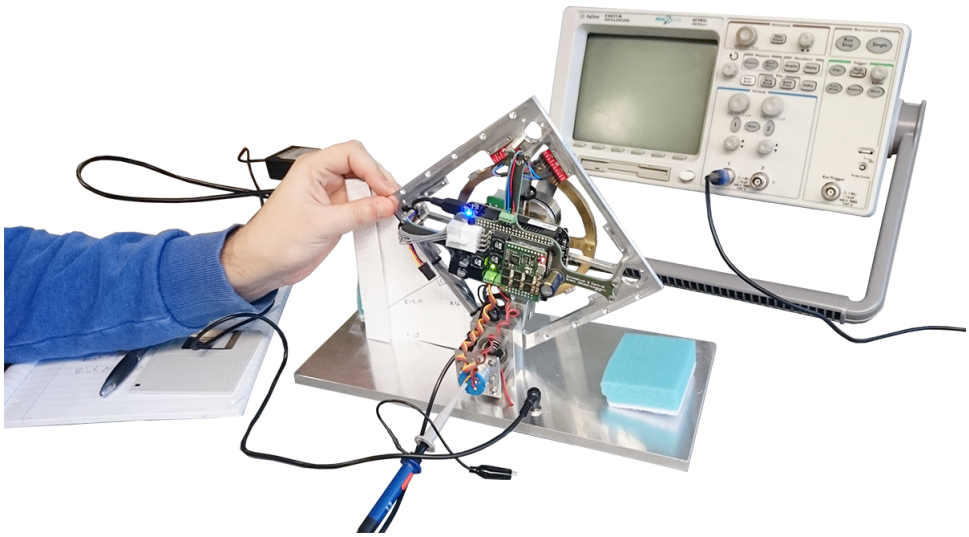


Figure C.1: Picture of the setup for the fall response test

List of Equipment

Instrument	AAU-no.	Type
Oscilloscope	61604	Agilent 54621A
Dedicated Power Supply of Cubli (24 V - 3 A)		XP Power, AEB70US24
Probe		1:1
Sponge	5P0N63	

Procedure

1. Keep the Cubli in the starting position (0° or 10°).
2. Let the Cubli fall over.
3. Use the oscilloscope to measure the voltage changes in the potentiometer and save them.
4. Take the measurements and plot them in Matlab.

Results

Data from the test is presented in the following graphs, showing both the measured voltage and calculated angular position. The position is obtained using the conclusions from *appendix B* explained in Section 3.4.

Fall response starting from 0°

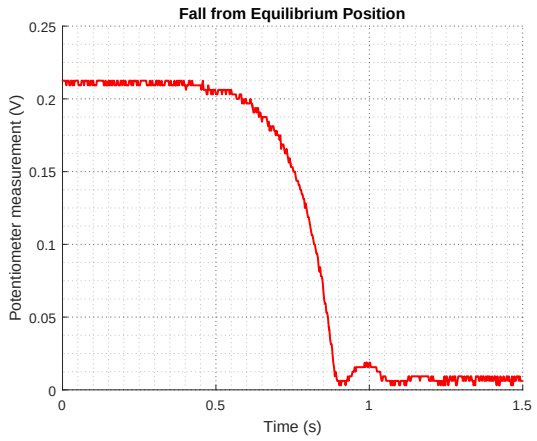


Figure C.2: Raw data taken from the potentiometer

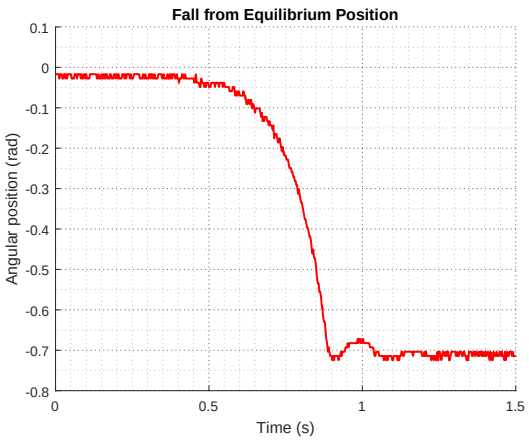


Figure C.3: Angular position of the frame

Fall response starting from 10°

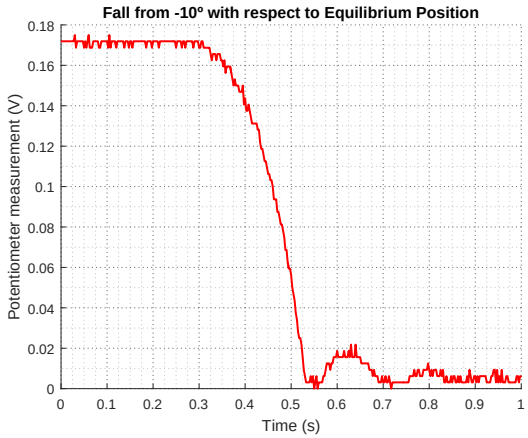


Figure C.4: Raw data taken from the potentiometer

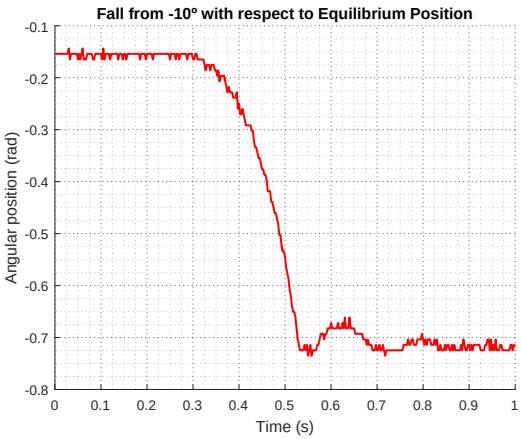


Figure C.5: Angular position of the frame

Note

The bouncing in the data when the frame reaches the lower position is due to the presence of the sponge.

D | Pendulum Behavior Test

Name: Group 630
Date: 16/03 - 2016

Purpose

Observing the behavior of the Cubli when hanging upside down. Data is then used to estimate some of the parameters of the Cubli.

Setup

The Cubli is put upside down under a table with 2 clamps placed at each side of the bottom plate of the Cubli setup. The wheel is being held in a fixed position with a strip tied to it and the frame. The probe chosen is a 1:1 and is connected to the potentiometer with probe to yellow cable and ground clamp to brown cable. The power supply has to be connected, and turned on, to the Cubli in order to get readouts from the potentiometer.

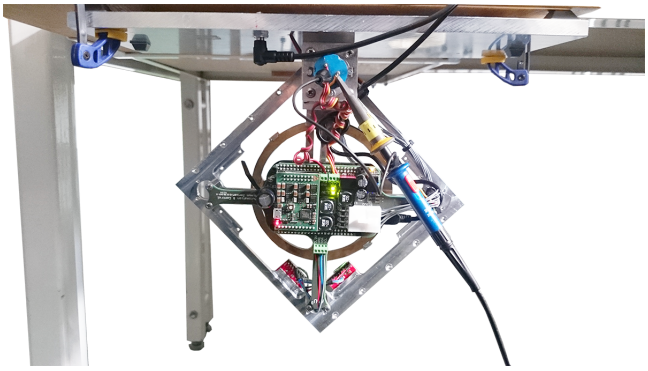


Figure D.1: The Cubli setup hanging upside down beneath a table during the impulse response test

List of Equipment

Instrument	AAU-no.	Type
Oscilloscope	61604	Agilent 54621A
Dedicated Power Supply of Cubli (24 V - 3 A)		XP Power, AEB70US24
Probe		1:1
2x Clamp		

Procedure

1. Place the setup upside-down and place the frame touching the base.

2. Let the Cubli fall and swing until it stops.
3. Use the oscilloscope to measure the changes in the potentiometer.
4. Collect all the data and plot it in Matlab.

Results

The results of the experiment can be seen in *figure D.2* and *figure D.3*. The second graph is obtained using the conclusions from *appendix B* explained in Section 3.4.

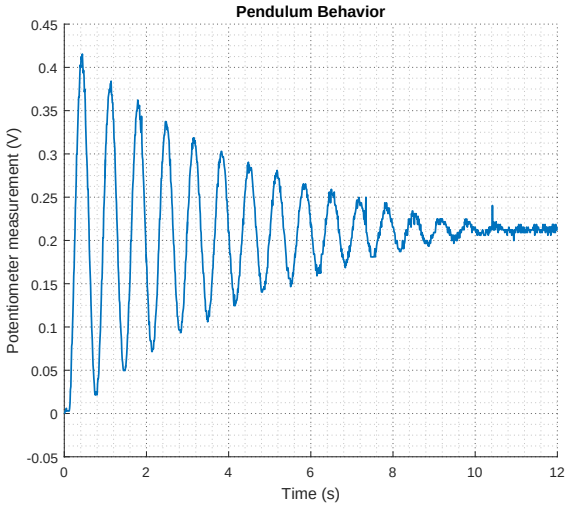


Figure D.2: Raw data taken from the potentiometer

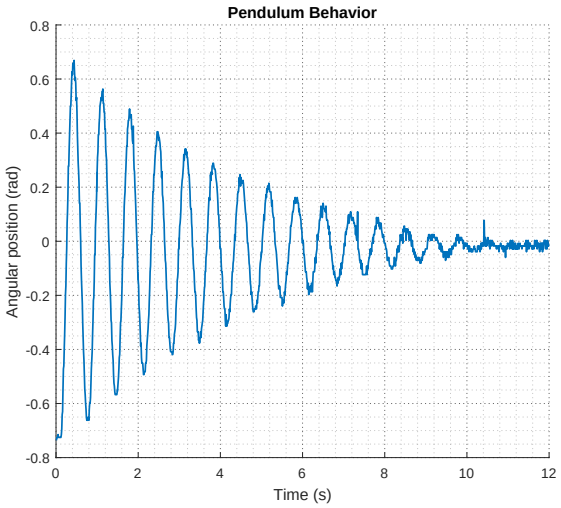


Figure D.3: Angular position of the frame

Note

During this experiment it was observed that if the frame was released from the left position (the right upper side on *figure D.1* since the Cubli is upside down), the frame would hit the rubber pad on the other side. This behavior was not observed when releasing the Cubli from the right position (left upper corner).

E | Measurement of Mass and Position of Center of Mass of the Frame

Name: Group 630
Date: 15/03 - 2016

Purpose

Measuring the mass and center of mass of the frame.

List of Equipment

Instrument	AAU-no.	Type
Scale	86759	
Dedicated Power Supply of Cubli (24 V - 3 A)	AAU3	XP Power, AEB70US24
Digital Protractor	None	CMT Orange Tools

Procedure

1. The Cubli base frame is leveled and the angle of equilibrium point is measured.
2. The frame is dismounted from the base and weight.
3. The frame is mounted back on the base after being rotated 90 degrees and the angle of equilibrium point is measured.
4. The Cubli frame is returned to the original placement on the base frame.

Results of the frame weight

Weight of the frame	kg
Fully mounted frame	0,770
Mass of the wheel	0,222

By subtracting the known mass of the wheel form the fully mounted frame, it gives a frame mass of $0,548$ kg.

Results of center of mass

To measure the center of mass the equilibrium positions of the frame on sides that are orthogonal must be determined. They can be seen in *figure E.1*, and the total center correspond to the crossing of the two lines.

Frame rotation angle	Angle from equilibrium point	Converted angle
0°	$2,5°$	0,043 rad
90°	$4,5°$	0,078 rad

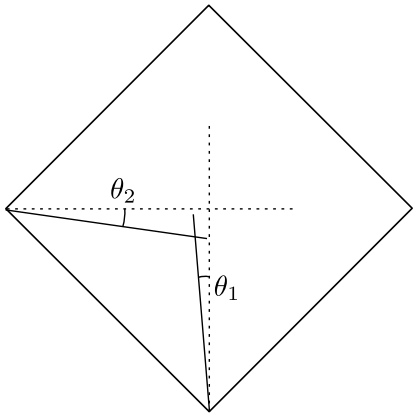


Figure E.1: Location of the center of mass, where $\theta_1 = 0,043$ rad and $\theta_2 = 0,078$ rad

F | IMU Setup Retrieval

Name: Group 630
Date: 30/04 - 2016

Purpose

Determining how the IMU is configured on the Cubli to help analyzing the data that is measured from it.

Setup

The Cubli is plugged through USB to a PC which has the ability to connect via Secure SHell (SSH) to the Beaglebone Black.
A program is written with parts of the given code to communicate with the IMU from the Beaglebone Black and retrieve the registers data from the sensor chip.

List of Equipment

Instrument	Type
Computer	Asus UX301LA

Procedure

1. Plug the Beaglebone Black through USB and wait until the blue LEDs on the Beaglebone Black start blinking slowly.
2. Send the binary compiled program to the board.
3. Connect to a distant terminal on the Beaglebone Black through SSH and launch the program.
4. Read the output of the program.

Results

From the output of the program, the following table is built.

Register	Hex Value on IMU 1	Hex Value on IMU 2
CONFIG	0x00	0x00
GYRO_CONFIG	0x00	0x00
ACCEL_CONFIG	0x00	0x00

Having a 0 in the CONFIG registers means that both IMU use a low pass filter for the noise on their output with the highest possible frequency of 260 Hz for the accelerometers and 256 Hz and that the accelerometers have no delay while the gyroscopes have a rather small delay of 0,98 ms.

The GYRO_CONFIG and ACCEL_CONFIG three most significant bits indicate that no self-test are asked to the sensor. The two next give information on the pre-selected range of measurement of these sensors.
Since there are all set to 0, gyroscopes will measure data within $\pm\ 250\,^{\circ} \cdot s^{-1}$, whereas accelerometers will present values in the range of $\pm$ 2g (g being the Earth's gravitational acceleration).
Eventually, the three least significant bits in ACCEL_CONFIG being set to 0, the digital filter on the accelerometers are disabled and the output value is put to 0 after each sample.

G | Maxon Control Board: ESCON Software

Name: Group 630
Date: 29/04 - 2016

Purpose

Get the configuration parameters through the ESCON software, which is used to configure the Maxon controller.

List of Equipment

Instrument	Type
Computer	HP Elitebook 8570P

Procedure

1. ESCON software is installed on the computer.
2. The control board is connected by a USB cable to the computer and the ESCON software is started.
3. The software automatically detects the controller and the configuration is uploaded to the software.
4. The configuration is then collected.

Results/Data

In the following text there are the most relevant configurations of the Maxon Control Board used in this project. Some of the data has been found on the data sheet of the ESCON module 50/5 and the Maxon Motor data sheet.

PWM controller
The control PWM runs at a frequency of 53,6 kHz and the Maxon motor is customized for this frequency. The PWM controller has been configured to use a Maxon motor EC 45 flat 50 W (part no. 251601). The parameters in the controller are:

Motor characteristics	Value
Speed constant	285,0 rpm · V^{-1}
Terminal time constant winding	17,6 s
Number of pole pairs	8

Controller
The controller has a maximum current drag in continuous operation of 5 A. It has a no load nominal speed of 6710 rpm and a stall current of 23,3 A. The motor has been limited in the configuration software as following

System data	Value
Max. permitted speed	1000 rpm
Nominal current	2,33 A
Max. output current limit	9,00 A

Hall Sensors
There are built-in hall sensors on the motor, used for detecting its position and velocity.

Detection of rotation and speed	Value
Speed sensor	Available Hall sensor
Digital sensor polarity	Inverted

Control Loop
The control can operate in open loop, close loop or current mode. The close loop and the current mode runs at a frequency of 5,36 kHz. The controller has been configured to the following:

Mode of operation	Value
Close loop	Current control
Gain	268

Digital Input
The digital input can be configured, so the motor can be activated or change direction by a signal. The speed or current can be set by a PWM signal. Its configuration can be seen in the following table:

Digital input	**Value**
Digital input 1	PWM
Current at 10%	−4,00 A
Current at 90%	4,00 A
Fixed offset	0,00 A
Digital input 2	Enable motor
On	High active

Analog Output
The analog output can be configured to give actual values of the motor, such speed or current. The analog output resolution is 12-bit and the voltage ranges from −4 V to 4 V. The actual configuration of the analog output is the following:

Analog outputs	**Value**
Analog output 1	Actual current
Current at: 0 V	−4 A
Current at: 4 V	4 A
Analog output 2	Actual speed
Speed at: 1 V	−1000 rpm
Speed at: 3 V	1000 rpm

H | Measurements from IMU

Name: Group 630
Date: 2/05 - 2016

Purpose

Get data from the IMU and the potentiometer when running the state-space controller, and used it to design the complementary filter.

List of Equipment

Instrument	AAU-no.	Type
Dedicated Power Supply of Cubli (24 V - 3 A)		XP Power, AEB70US24
Computer		Asus A55V

Procedure

1. Plug the power supply given of the Cubli.
2. Wait until the blue LEDs on the Beaglebone Black start blinking slowly and connect the USB cable to the PC.
3. Send the binary compiled program of the controller to the board.
4. Connect to a distant terminal on the Beaglebone Black through SSH and launch the program.
5. Let the program run with the state-space controller.
6. Apply small disturbances to the Cubli so there is some variations in the angle
7. Stop the program (by pressing Q and *ENTER*).
8. Retrieve the log file from the Cubli setup with the recorded data.

Results

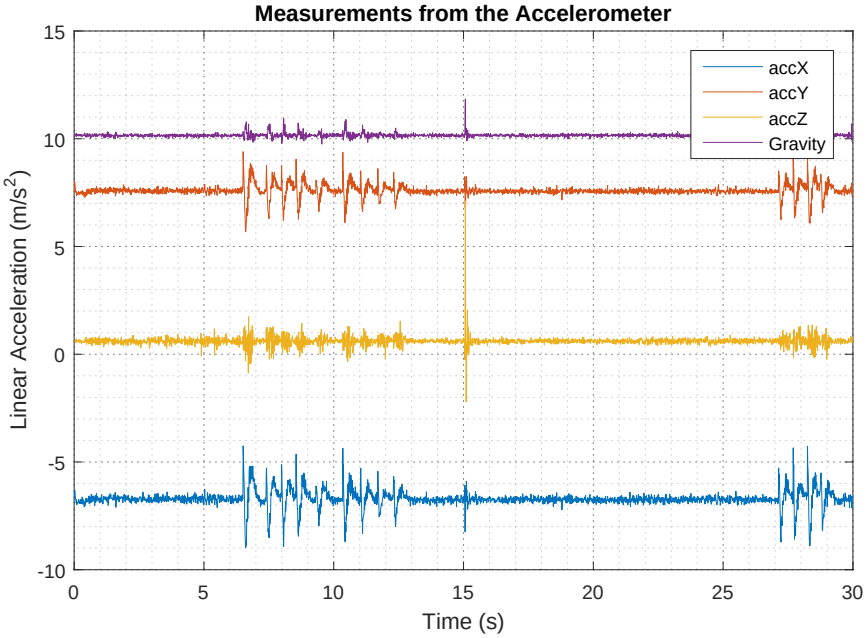


Figure H.1: Linear acceleration measurements from the accelerometer and the calculated gravity magnitude

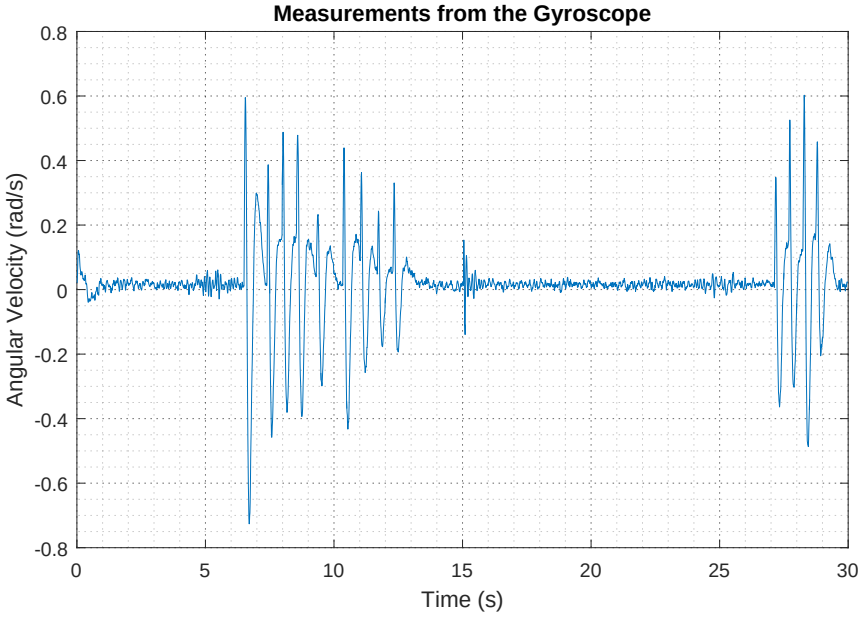


Figure H.2: Angular velocity measurements from the gyroscope

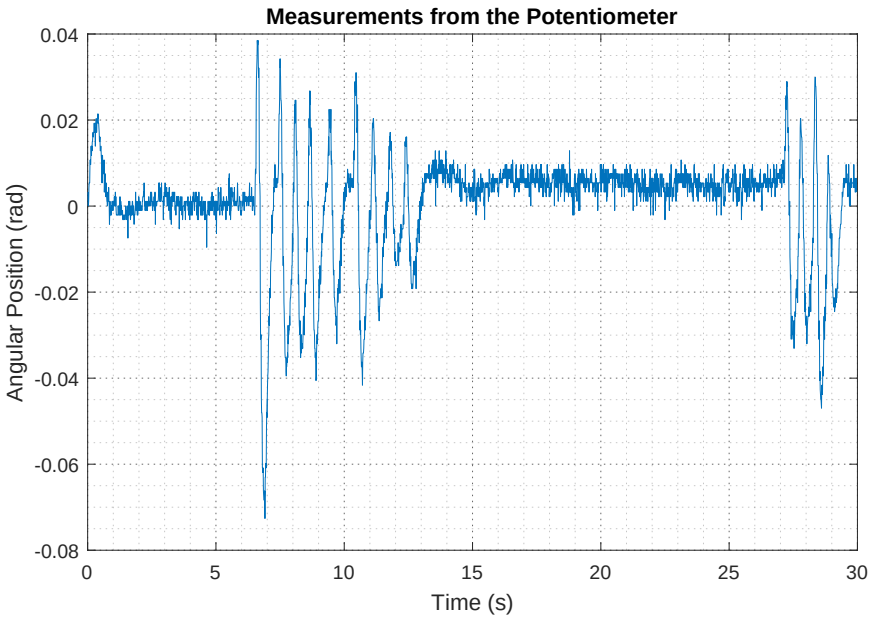


Figure H.3: Angle measurements from the potentiometer

I | Motor Current Test

Name: Group 630
Date: 02/05 - 2016

Purpose

Check the assumption that the asked motor torque is the one that it gives in reality.

Setup

The Cubli is put in equilibrium at approximately 0 °. It is plugged to a computer from which the code can be sent, started and to which the data can later be retrieved, through USB over an Secure SHell (SSH) connection.

List of Equipment

Instrument	Type
Dedicated Power Supply of Cubli (24 V - 3 A)	XP Power, AEB70US24
Computer	Asus A55V

Procedure

1. Plug the power supply given with the Cubli setup to a 220V power outlet.
2. Wait until the blue LEDs on the Beaglebone Black start blinking slowly and connect the USB cable to the PC.
3. Send the binary compiled program of the controller to the board.
4. Connect to a distant terminal on the Beaglebone Black through SSH and launch the program.
5. Let the program run.
6. Stop the program (by pressing *Q* and *ENTER*).
7. Retrieve the log file from the Cubli setup with the recorded data.

Results

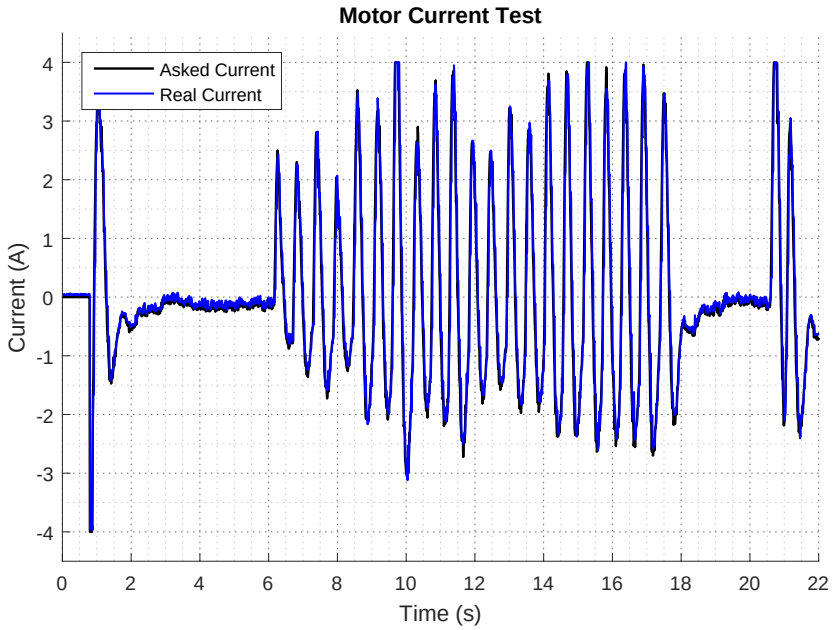


Figure I.1: Result of the test done to the motor to check that the reference current can be assumed to be the one applied to the motor.

J | Timing Characteristics of the Program

Name: Group 630
Date: 11/05 - 2016

Purpose

Check the proper behavior of the program running on the microcontroller to ensure proper operation of the control system.

Setup

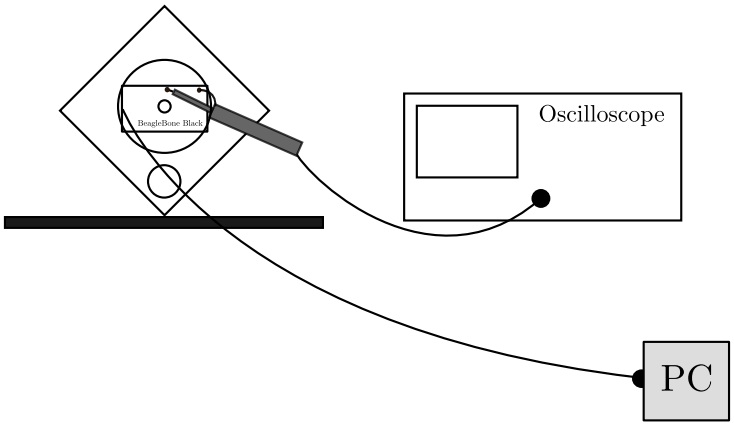


Figure J.1: Setup diagram

To measure timing characteristics, it is necessary to connect to external pins of the BeagleBone Black and raise and lower the pin around the interesting program area in the source code. Here, the section of interest is the whole function `runController()` which performs the sensor readings and runs the controller computations.
It should be made sure that the current sent to the motor controller is null, so that the wheel stands still while doing the measurement.

List of Equipment

Instrument	AAU-no.	Type
Dedicated Power Supply of Cubli (24 V - 3 A)		XP Power, AEB70US24
Oscilloscope	3386	Agilent 54621A
Probe		1:1
Computer		Asus UX301LA

Procedure

1. Make the setup with connections as seen on *figure J.1*, with ground on pin 2 and signal on pin 26 of the P8 header on the BeagleBone Black.
2. Launch the appropriate program over a USB SSH connection.
3. Set the oscilloscope on rolling and calibrate it so that a few periods of the pulse signals can be clearly seen on the display.
4. Stop the acquisition and save the data to a floppy-disk as a CSV file. Gather the CSV log file generated by the program, from the BeagleBone Black.

Results

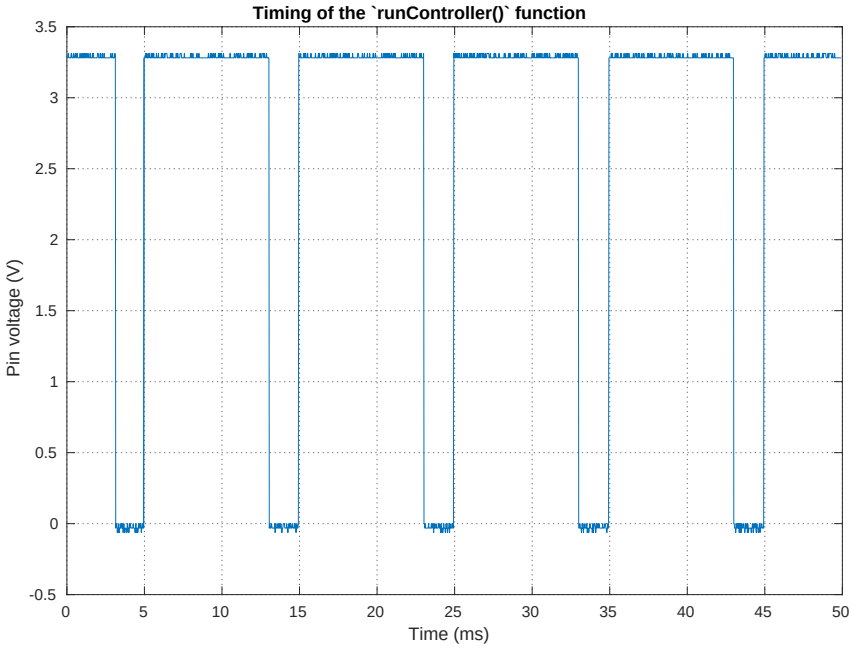


Figure J.2: Pulse signal of the monitored code region executing the main controller operations.

Figure J.2 shows that the function `runController()` is called every 10 ms along the few periods shown on the graph. This is the expected behavior since the function runs in a scheduled thread and it seems to respect, rather precisely, the timing period it is asked in the code to run at.
It is also interesting to note that each call made to the function lasts approximately 8 ms which means this critical section of the code has the time to run completely before the next call. However, this does not leave a very large margin to add other potential features in this code environment.
By running a few other similar tests and removing parts of the code, the sensor readings are identified to be the most time consuming feature. Furthermore, jumps in the potentiometer readings, which can be seen on *figure J.3* and *figure J.4* seem to arise from the way the ADC values are read by the program through the operating system's I/O system.

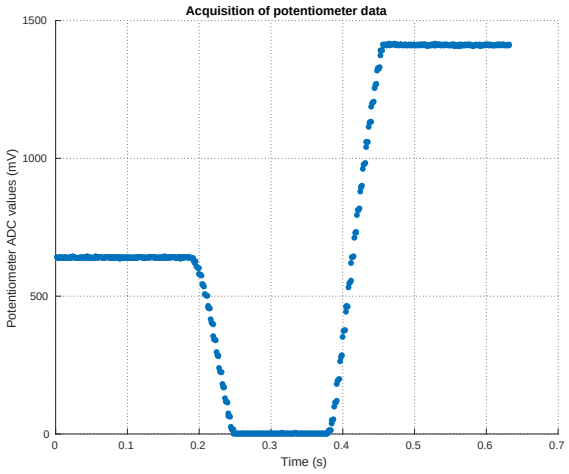


Figure J.3: Potentiometer readings from a range test. Groups of data points can be observed.

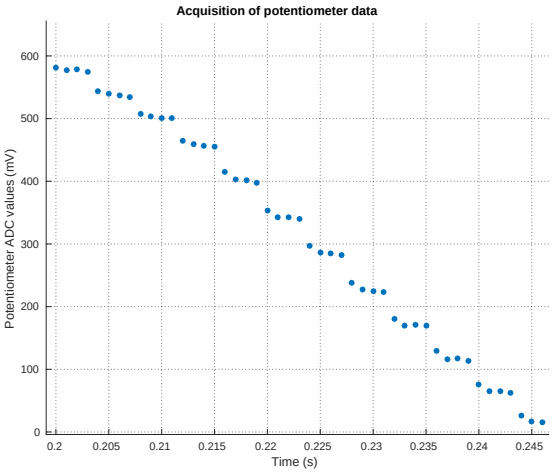


Figure J.4: Zoom-in on the potentiometer readings. Irregularities can be seen in the ADC values.

The area at issue in this situation seems identifiable, but since it is decided to eliminate the potentiometer in this project, and since the measured data is only negligibly affected in any case, no further investigation is to be made on this matter during this project.

K | Root Locus Designed Controller Response

Name: Group 630
Date: 24/05 - 2016

Purpose

Checking the stability of the controller upon basic stimulation.

Setup

The Cubli is put in an unstable equilibrium at approximately 0°. It is plugged to a computer from which the correct code can be sent, started and to which the data can later be retrieved, through USB over an Secure SHell (SSH) connection.

List of Equipment

Instrument	AAU-no.	Type
Computer		Asus A55V
Dedicated Power Supply of Cubli (24 V - 3 A)	AAU3	XP Power, AEB70US24

Procedure

1. Plug the power supply given with the Cubli setup to a 220V power outlet.
2. Wait until the blue LEDs on the Beaglebone Black start blinking slowly and connect the USB cable to the PC.
3. Send the binary compiled program of the controller to the board.
4. Connect to a distant terminal on the Beaglebone Black through SSH and launch the program.
5. Let the program run..
6. Stop the program (by pressing Q and *ENTER*).
7. Retrieve the log file from the Cubli setup with the recorded data.

Results

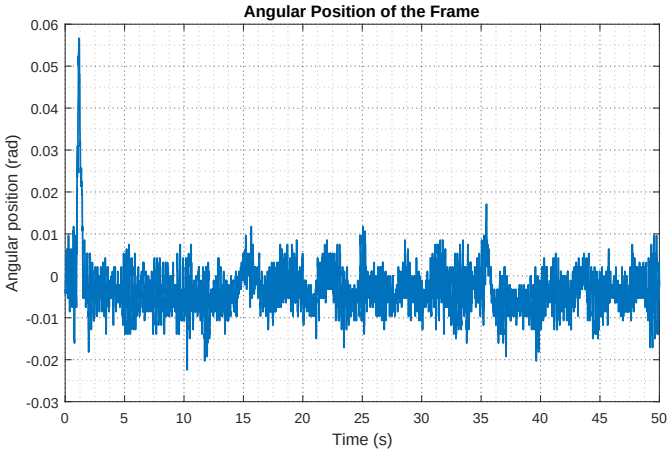


Figure K.1: Angular position of the Cubli

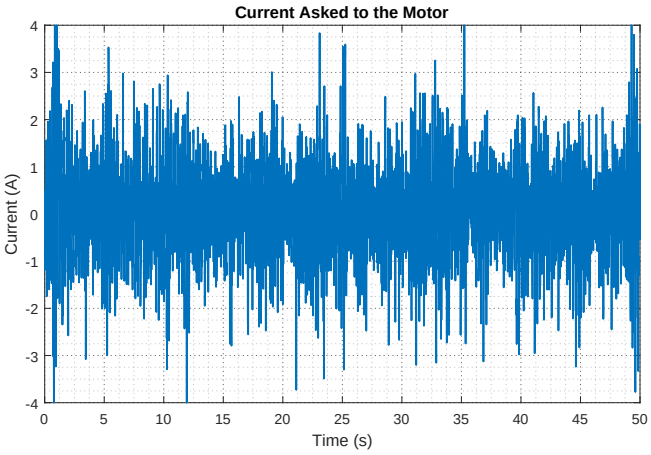


Figure K.2: Current asked to the motor

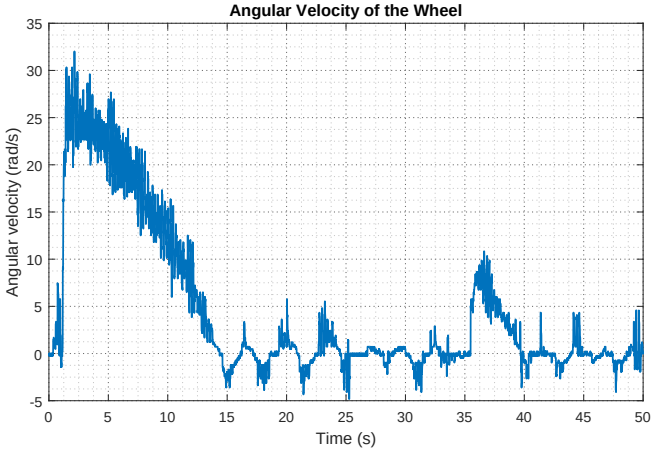


Figure K.3: Angular velocity of the wheel

L | Parameters of the Reaction Wheel

Group 630

Purpose

Provide a method to measure and estimate the wheel's mass, distance to the frame's pivoting point, inertia and friction to include these parameters in the model instead of previous groups'.

Setup and Principle

To measure the mass of the wheel, a scale can be used if the wheel is dismantled from the setup. To get the distance from the center of the wheel to the pivoting point of the frame, a precise ruler can be used. The setup needed to estimate the wheel's inertia and friction can be seen on *figure L.1*.

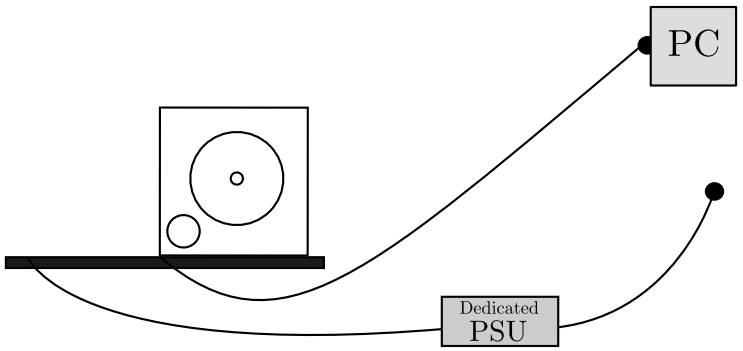


Figure L.1: Setup diagram

By applying a current step to the motor, and measuring the output velocity, as shown on *figure L.2*, it is possible to use Senstools in a similar way to how it is used in *section 5.2* with a Simulink model of the wheel and the attached motor. The inertia and friction parameters of the simulation model will be fitted to match the reality on this test.

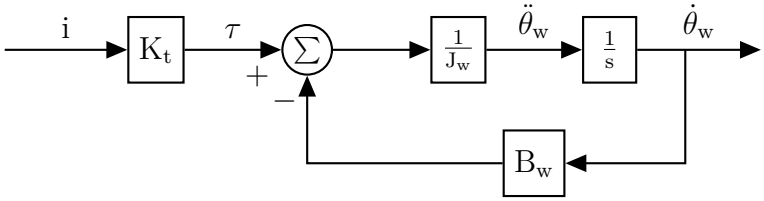


Figure L.2: Block diagram of the wheel used for the simulation

List of Equipment

Instrument	Type
Dedicated Power Supply of Cubli (24 V - 3 A)	XP Power, AEB70US24
Probe	1:1
Computer	

Procedure

1. Plug the dedicated Power Supply Unit to the Cubli setup, then to the computer.
2. Prepare the code so that a current step is applied to the motor after a certain amount of time and the velocity of the wheel and the actual current are logged into a file. Upload this program to the BeagleBone Black.
3. Fix the frame tightly so that it doesn't move during the test.
4. Launch the test program and wait a few seconds after the activation of the motor so that the wheel's velocity seems to saturate.
5. Stop the program. Gather the CSV log file generated by the program, from the BeagleBone Black.
6. It is possible to repeat the last two steps multiple times to have a set of tests to compare.
7. Use the data to fit the model in Simulink with Senstools.

M | Connecting and Breakout Board Schematic

The connecting and breakout board present in the system has been done by Simon Jensen, Electronic Technician at Aalborg University.

Input and Output Connections

The BeagleBone Black input and output connections diagram *figure M.1*:

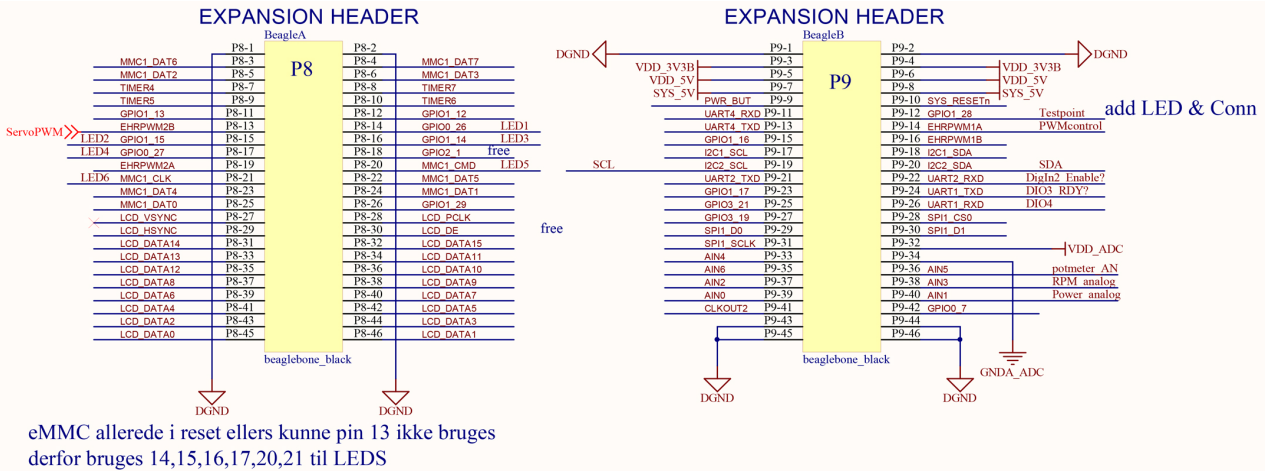


Figure M.1: Expansion header diagram.

Voltage Information

The power supply that supplies the entire system is 24 V. There has been built voltage regulators in to the connecting and breakout board to supply the different units with operation voltage. Many connections between the different units and the BeagleBone Black has to be in the same voltage level, for the BeagleBone Black to be compatible with the units.

Many of the sensors are operating on different voltage. Below is a list of the different voltage usage:

Unit	Voltage (V)
BeagleBone Black	5 V
BeagleBone Black ADC	Max. 1,8 V
BeagleBone Black logic level	3,3 V
Maxon Controller Board	10 V to 50 V
Maxon Motor	24 V
Potentiometer	Max. 50 V
ServoMotor	4,8 V to 6 V
IMU PMU6050	3,3 V

Potentiometer Connections

The Potentiometer connections diagram *figure M.2*:

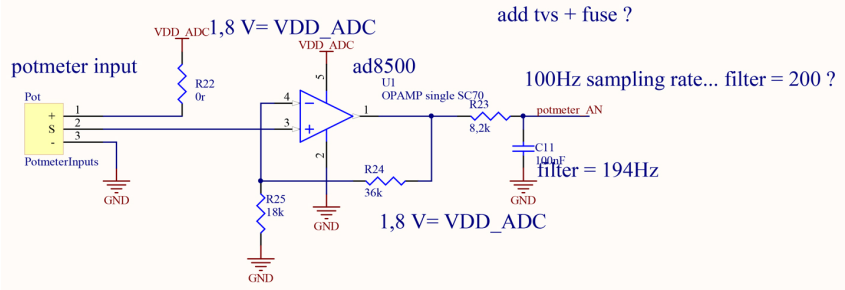


Figure M.2: Potentiometer diagram.

The Potentiometer is connected to the BeagleBone's A/D and only a small area is used on the Potentiometer, and for this reason the signal is gained and it is done by using an Operational Amplifiers (op-amp) which can only gain the voltage to 1,8 V, because of the supply voltage to the op-amp is only 1,8 V.

The Potentiometer is only using a small area of rotation on the Cubli, about ¼ of the full rotation. The gain is calculated from the max. rotation voltage and the result is a gain of 3 V.

To verify the gain, the potentiometer voltage is measured and the ADC value is read from the BeagleBone Black.

Unit	Voltage (V)
Measured before the gain	0,470 V
0,470 V with a gain of 3	1,410 V
Measured after the gain	1,417 V
ADC value	1,419

After the signal has been gained, a low-pass filter with a frequency of 194 Hz is added to damp noise on the signal.

Motor Control Board Connections

The Maxon Motor Driver Board connections diagram *figure M.3*:

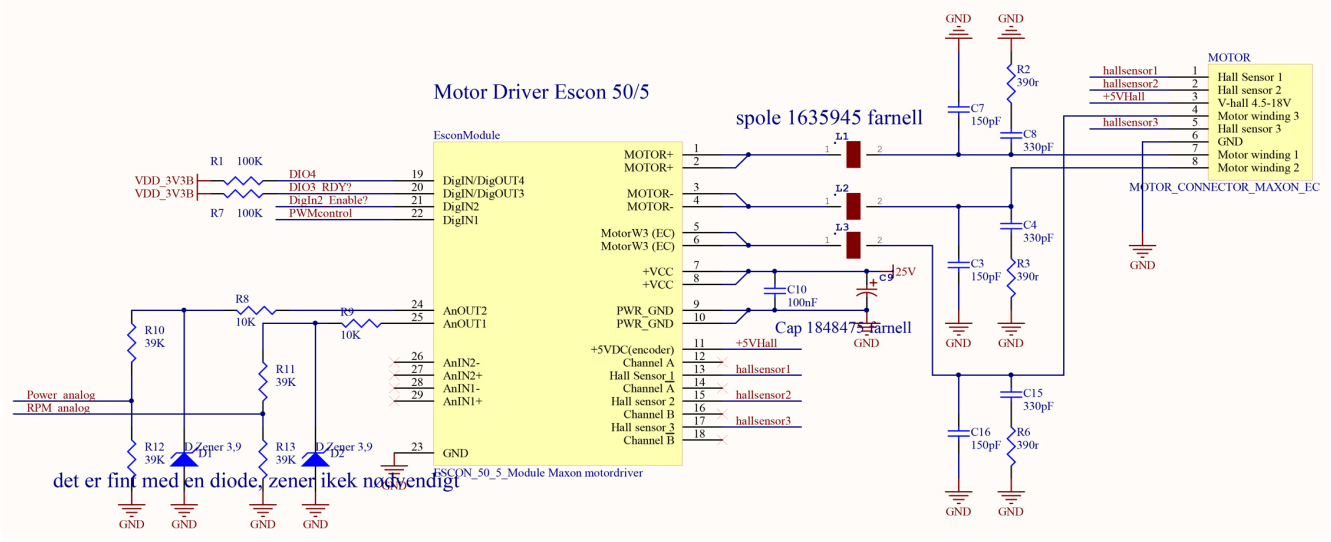


Figure M.3: Motor Driver diagram.

Since the BeagleBone Black ADC only operates up to 1,8 V and the Maxon control board operates at −4 V to 4 V, a voltage divider has been put in from the Maxon motor control to the Beagle Bone so the BeagleBone's A/D limit is not exceeded so the voltage is multiplied by 0,44382. Then if the Maxon controller operates at a maximum of 4 V it becomes 1,775 28 V below the BeagleBone A/D limit of 1,8 V.

Extra Connections

MPU6050's I2C and 3,3 V connections to the BeagleBone Black.

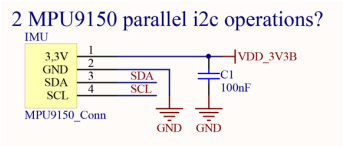


Figure M.4: PMU6050 connections diagram.

5 V power supply for the ServoMotor on Connecting and Breakout Board, and 3,3 V to 5 V level converter is needed to get the BeagleBone Black PWM signals to the ServoMotors logic circuit.

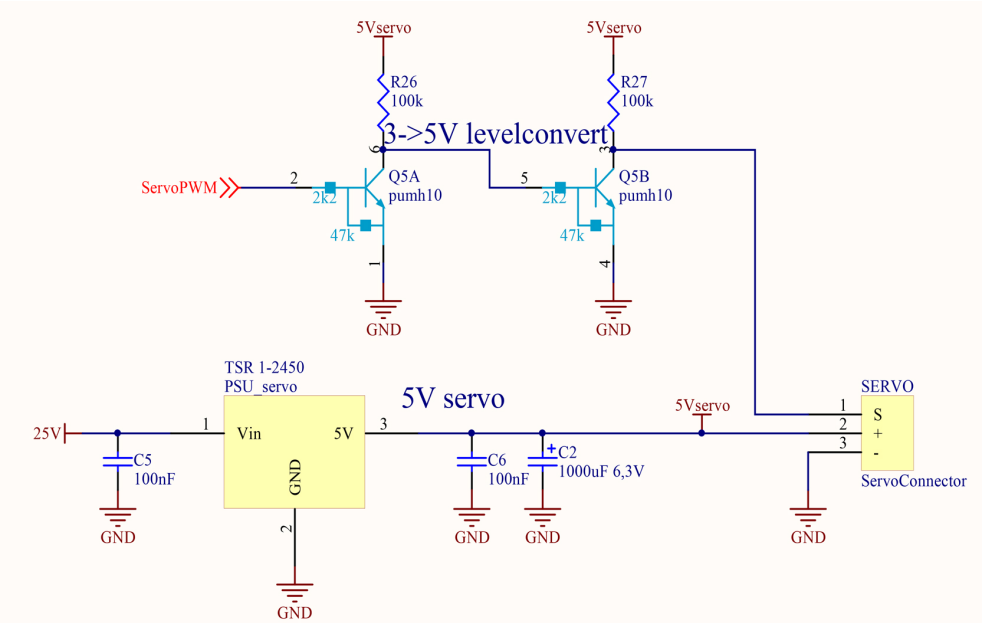


Figure M.5: ServoMotor diagram.

5 V power supply for the BeagleBone Black and the LED's on Connecting and Breakout Board.

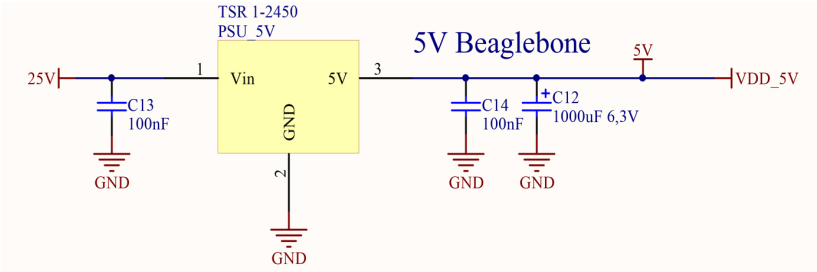


Figure M.6: Power converter from 25 V to 5 V for the BeagleBone Black.

25 V power and grounding connections and 5 V and 3,3 V LED's connections.

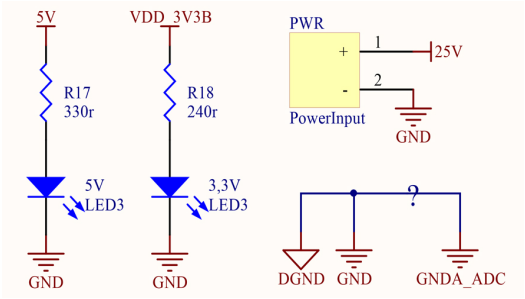


Figure M.7: Power diagram.

5 V LED's connections to the BeagleBone Black.

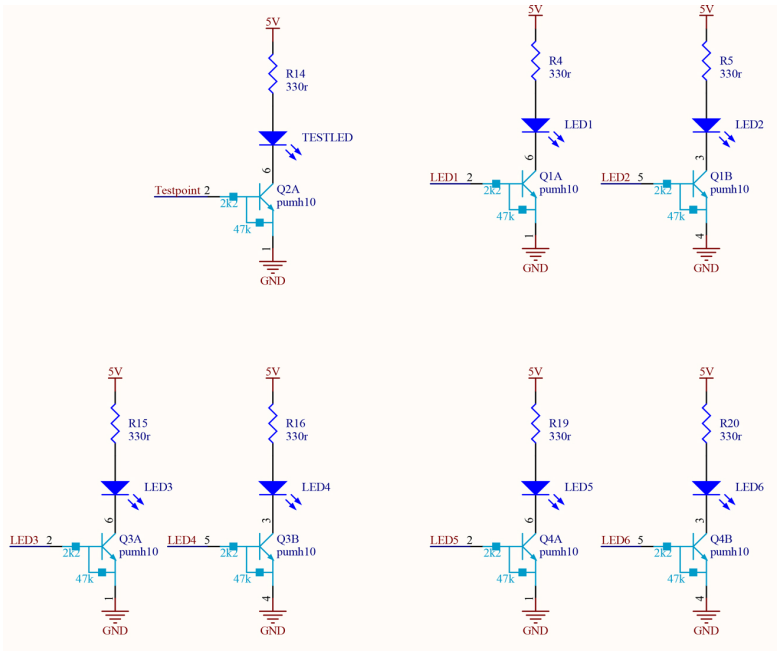


Figure M.8: LED's diagram to the BeagleBone Black.

N | Error Comparison between Senstool and Line Search

Name: Group 630 Date: 24/05 - 2016

Purpose

Checking how the error changes for both estimation of parameters if the last data is not taking into account.

Results

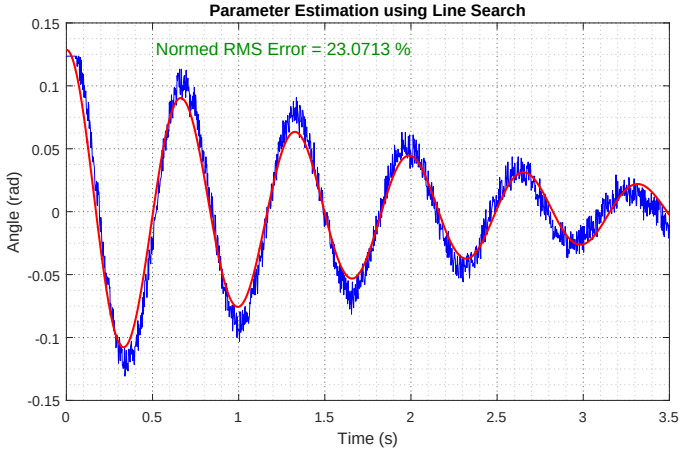


Figure N.1: Normed RMS Error removing the last part of the data, using the parameters that Line Search gives

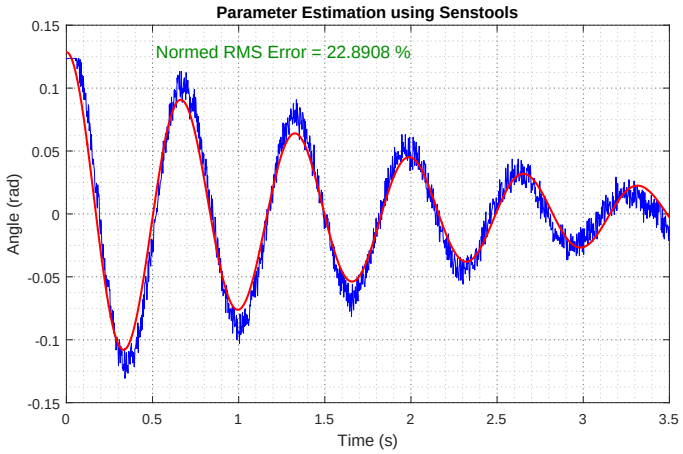


Figure N.2: Normed RMS Error removing the last part of the data, using the parameters that Senstools gives

O | Attached DVD Content

The DVD contains a digital copy of the report and seven extra folders. The content of each one is the following:

1. **Code**

   It contains a compressed file with the Eclipse protect needed to run the controllers.

2. **Data Sheets**

   It contains the data sheets of the BeagleBone, the brushless motor, the motor control board and the IMU.

3. **Matlab Files**

   It includes the files needed to get most of the pictures of the report that do not come from a test.

4. **Optimization Code Files**

   It contains the code and data files needed to implement the optimization code.

5. **Senstool Documentation**

   It contains the manual for Senstool and a compressed file which includes the Matlab files needed to run the toolbox.

6. **Test Files**

   It contains the Matlab and .csv files needed to get all the results of the test.

7. **Videos**

   It includes three videos made to the system, both with the classical and the state space controller.

Bibliography

[1] J. Huber et al. „Online trajectory optimization for nonlinear systems by the concept of a model control loop – Applied to the reaction wheel pendulum“. In: (2013), pp. 935–940. DOI: 10.1109/CCA.2013.6662871. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6662871.

[2] Igor Thommen Mohanarajah Gajamohan Michael Merz and Raffaello D'Andrea. „The Cubli: A Cube that can Jump Up and Balance“. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 2012). URL: https://www.ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Research_DAndrea/Cubli/Cubli_IROS2012.pdf (visited on 24/05/2016).

[3] *Cubli_Corner*. Feb. 29, 2016. URL: http://raffaello.name/projects/cubli/ (visited on 02/29/2016).

[4] Ross Allen et al. „Internally-Actuated Rovers for All-Access Surface Mobility: Theory and Experimentation“. In: *International Conference on Robotics and Automation (ICRA)* (May 6, 2013). Department of Aeronautics and Astronautics, Stanford University, Stanford. URL: https://web.stanford.edu/~pavone/papers/Allen.Pavone.ea.ICRA13.pdf (visited on 04/25/2016).

[5] Elizabeth Landau. *'Hedgehog' Robots Hop, Tumble in Microgravity*. Ed. by Tony Greicius. Jet Propulsion Laboratory, Pasadena, Calif. Oct. 1, 2015. URL: http://www.nasa.gov/feature/jpl/hedgehog-robots-hop-tumble-in-microgravity (visited on 04/25/2016).

[6] Tetsuo Yoshimitsu, Takashi Kubota, and Ichiro Nakatani. „MINERVA Rover Which Became a Small Artificial Solar Satellite“. In: *journaltitle* (Aug. 17, 2006). URL: digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1554&context=smallsat (visited on 04/25/2016).

[7] John W. Romanishin, Kyle Gilpin, and Daniela Rus. „M-Blocks: Momentum-driven, Magnetic Modular Robots“. In: *IROS* (2013). URL: http://ppm.csail.mit.edu/sites/default/files/publications/M-Blocks,%20Momentum-driven,%20Magnetic%20Modular%20Robots.pdf (visited on 24/05/2016).

[8] *M-Blocks and Roombots – Self Assembling Microbots With Many Uses*. May 24, 2014. URL: http://www.21stcentech.com/m-blocks-roombots-assembling-microbots/ (visited on 02/29/2016).

[9] Gerald Coley. *BeagleBone Black System Reference Manual*. 2013. URL: https://cdn-shop.adafruit.com/datasheets/BBB_SRM.pdf.

[10] Cameon. *Reading the analog inputs (ADC)*. URL: http://beaglebone.cameon.net/home/reading-the-analog-inputs-adc (visited on 04/01/2016).

[11] Maxon. *EC 45 flat*. 2015. URL: http://www.maxonmotor.com/medias/sys_master/root/8816806854686/15-262-EN.pdf.

[12] Maxon. *ESCON Module 50/5*. 2015. URL: http://www.maxonmotor.com/medias/sys_master/root/8816814850078/15-376-377-380-EN.pdf.

[13] *ESCON*. URL: http://www.maxonmotor.com/maxon/view/content/ESCON-Detailsite (visited on 07/04/2016).

[14] InvenSense. *MPU-6050*. 2011. URL: http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/.

[15] SparkFun. *SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050*. 2015. URL: https://www.sparkfun.com/products/11028.

[16] eLinux.com. *Toolchains*. URL: http://elinux.org/Toolchains (visited on 23/05/2016).

[17] Simon Vestergaard Johansen and Anna Zeeb. „AAU^3“. MA thesis. Aalborg University, 2014. URL: http://projekter.aau.dk/projekter/da/studentthesis/aau3(8d192136-08b3-4c34-8378-1cbc465dd3f9).html.

[18] Morten Knudsen. *Experimental Modelling of Dynamic Systems*. Documentation and software are on the attached DVD. Department of Control Engineering, Aalborg University. 2004.

[19] Andreas Antoniou and Wu-Sheng Lu. *Practical Optimization. Algorithms and Engineering Applications*. Springer US, 2007. ISBN: 978-0-387-71107-2. DOI: 10.1007/978-0-387-71107-2.

[20] Wenyu Sun and Ya-Xiang Yuan. *Optimization Theory and Methods. Nonlinear Programming*. Springer US, 2006. ISBN: 978-0-387-24976-6. DOI: 10.1007/b106451.

[21] Abbas Emami-Naeni Gene F. Franklin J. David Powell. *Feedback Control of Dynamic Systems*. 7th, Global. Pearson, 2015. ISBN: 978-0-13-349659-8.

[22] Guoxiang Gu. *Discrete-Time Linear Systems. Theory and Design with Applications*. 1st ed. Springer US, 2012. ISBN: 978-1-4614-2281-5.

[23] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. Pearson, 2010. ISBN: 978-0131988422.

[24] *Discretize a compensator*. Feb. 22, 2016. URL: http://fr.mathworks.com/help/control/ug/discretize-a-compensator.html (visited on 04/10/2016).

[25] Louis P. Russo and B. Wayne Bequette. „State-Space versus Input/Output Representations for Cascade Control of Unstable Systems“. In: (1997). DOI: 10.1021/ie960677o. URL: http://pubs.acs.org/doi/abs/10.1021/ie960677o (visited on 05/19/2016).

[26] Harald Molle John-David Warren Josh Adams. *Arduino Robotics*. 1st ed. Apress, 2011, pp. 459–464. ISBN: 978-1-430-23183-7.

[27] Christopher J. Fisher. *Using an Accelerometer for Inclination Sensing*. Analog Devices. 2010. URL: http://www.analog.com/media/en/technical-documentation/application-notes/AN-1057.pdf (visited on 05/11/2016).

[28] Subhas Mukhopadhyay Pengfei Gui Liqiong Tang. „MEMS Based IMU for Tilting Measurement– Comparison of Complementary and Kalman Filter Based Data Fusion“. In: (2015). DOI: 10.1109/ICIEA.2015.7334442. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7334442 (visited on 05/11/2016).

[29] *IMU Data Fusing: Complementary, Kalman, and Mahony Filter*. Sept. 17, 2013. URL: http://www.olliw.eu/2013/imu-data-fusing/ (visited on 04/24/2016).