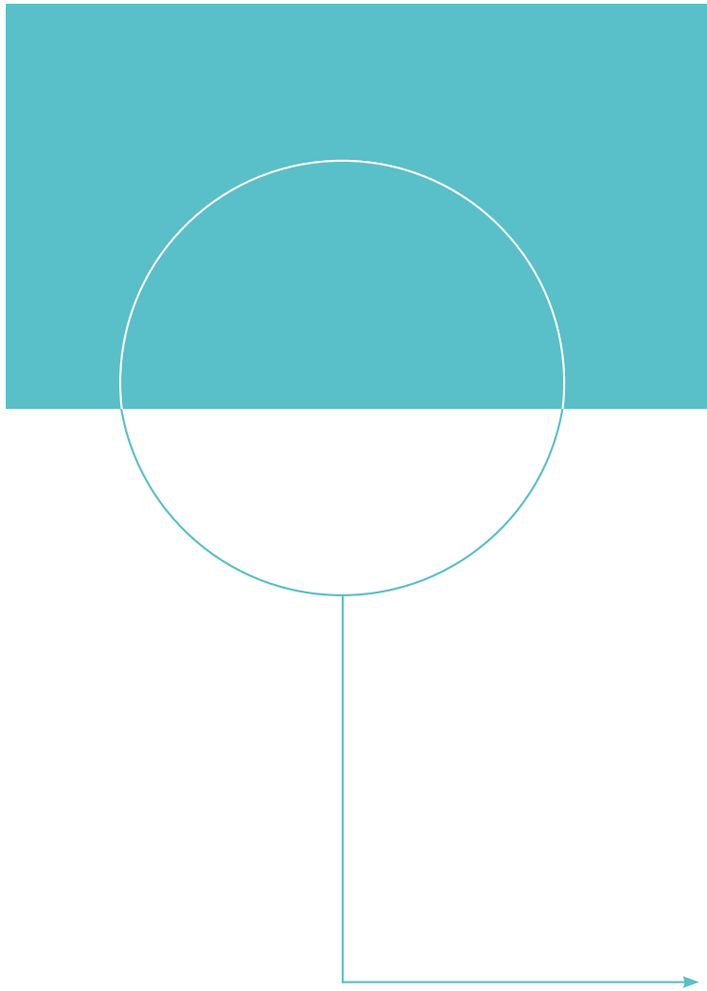


Master's thesis

2019

Martin Mostad

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Martin Mostad

Hardware in the Loop Testing of Explicit Model Predictive COntrol of The CubeSat Selfie-Sat

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Hardware in the Loop Testing of Explicit Model Predictive COntrol of The CubeSat Selfie-Sat

Martin Mostad

Cybernetics Engineering
Submission date: June 2019
Supervisor: Jan Tommy Gravdahl
Co-supervisor: Amund Gjersvik

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Hardware in the Loop Testing of Explicit Model Predictive Control of The CubeSat Selfie-Sat

Martin Mostad

Supervisors: Jan Tommy Gravdahl
Amund Gjersvik

June 3, 2019

Abstract

Tow explicit Model Predictive Controllers where developed for the CubeSat Selfie-Sat. One was intended to be used for camera pointing and the other was intended to be used to keep the satellite in ideal chagrin orientation. In addition a initial design of a Hardware in the Loop test setup for testing the controller was developed. As part of the Hardware in the Loop development disturbance, sensor and actuator models where also developed tailored to the Selfi-Sat. In the end the two controllers where somewhat successful working in ideal conditions, but not when exposed to disturbances. The controller where not tested using the Hardware in the Loop environment as the proposed design could not handle the necessary data flow.

Abstrakt

To eksplisite Model Prediktive Kontrollere ble utviklet for CubeSaten Selfie-Sat. Den ene kontroller skulle bruke til å peke et kamera, mens den andre skulle bruke til å holde satellitten i en ideell orientasjon for lading. I tilleg bel en test benk maskinvare i løken (Hardware in the Loop) utviklet for å teste kontrollene. Som en del av maskinvare i løken ble også utviklet modeller for forstyrrelser, sensorer og aktuatorer som var skreddersydd for Selfie-Sat. Kontrollene som ble utviklet var noenlunde suksessfulle, de fungerte i ideelle tilfeller, men ikke når de ble utsatt for forstyrrelser. Kontrollene ble ikke teste i maskinvare i løkke systemet ettersom det foreslått designet ikke kunne håndtere den nødvendige dataflyten.

Preface

I would like to start this thesis by thanking everybody who has helped me with this thesis in some shape or form. I would especially like to thank Prof Jan Tommy Gravdahl for his continued support, advice, regular meetings and helping me giving this thesis a direction. I would also like to thank Amund Gjersvik for his support and help with procurement of hardware components.

A special thank should go to the MOVE-II project and to the "Lehrstuhl für Raumfahrttechnik" at Technische Universität München for allowing me to use and modify their Hardware in the Loop setup including both their simulation environment and interface which almost is the entire foundation on which this thesis stands. David Messman and Jonis Kiesbye from the MOVE-II ADCS team deserve extra mention, not only have they thought me almost everything I know about ADCS, but they have always been willing to help and answer any question I may have had.

Big thank to Petter Tøndel for allowing me to use his Matlab code for solving eMPC problems. His code has helped my understanding of the problem tremendously. The developer of the Multi-Parametric Toolbox 3.0: M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric will forever have my gratitude for publishing such a great toolbox free of charge. Their toolbox have saved me many hours of work on this project.

Orbit NTNU deserves their thanks as well for allowing me to write this thesis for them and providing me with the necessary lab equipment, a place for me to talk to like-minded people and overall for being an amazing organization to be a part of. The ADCS team shall have a special thank for being such a hardworking group. It has been a pleasure leading the group this last two semesters. Andreas Westre extra recognition for helping with the modification of all the sensor models.

At the very end I would like to thank my friends and fellow students for making such a memorable student experience. My family for all ways being there when I needed them no matter what. Lastly a special thank Melanie Mügler not only for being so supportive and understanding during this whole ordeal but also for the countless hours she has spent helping me proofreading my thesis.

To everybody thanks for everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Orbit is a student organization that aims to promote an interest in space at the Norwegian University of Science and Technology. The main way it is trying to achieve this is through its Selfie-Sat project. The project is building a two unit CubeSat. The main goal of the CubeSat is to take a "selfie" in space. This is achieved by having a camera on a deployable boom with a camera at the end which is pointing back to the satellite. On the face of the satellite is a screen that can display a picture. So a picture can be taken of the screen with the Earth in the background creating a "selfie" in space. A 3D rendering of the satellite created by the PR and mechanical team at Orbit can be seen in Figure 1.1.

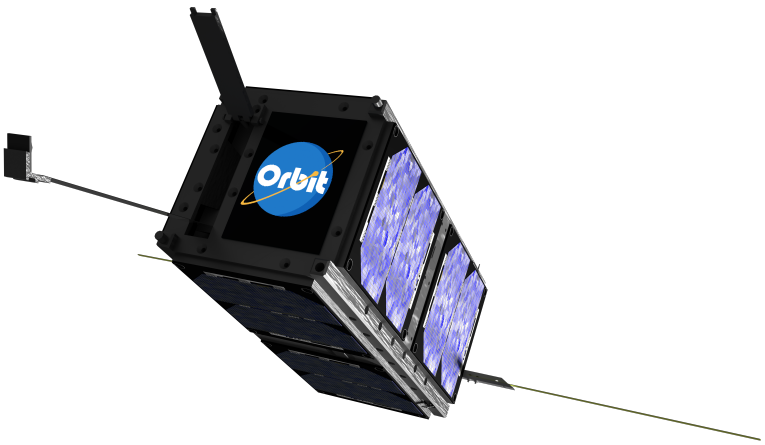


Figure 1.1: 3D model of the Selfie-Sat.

1.1 Selfie-Sat Attitude Determination and Control System

The attitude determination and control system (ADCS) is needed for three reasons: To detumble the satellite after separation from the rocket, to point the camera towards the

Earth so that it is in the background when the "selfies" are being taken and to keep the satellite in optimal charging position.The ADCS consist of several parts to complete its tasks:

- **Attitude Controller:** It is the actual control algorithm that controls the satellite which is developed as part of this thesis.
- **Attitude Determination:** It is estimating the attitude of the satellite using several sensors and environmental models. It is currently being developed as an Extended Kalman filter.
- **Environmental models:** They are different models, for example models of the orbit of the satellite, position of the Sun and magnetic field of the Earth. They are used by the attitude determination system to estimate the attitude of the satellite.
- **Sensors:** Several sensors such as Inertia Measurement Units (IMU) and Sun Sensors are used to gather information about the orientation of the satellite which will be used in the attitude estimate.
- **Actuator:** Three coils are used to create a magnetic moment which interacts with the Earth's magnetic field to create a torque. This is in turn used to control the attitude of the satellite.

1.2 Explicit Model Predictive Control

Orbit tries to build as many systems on the satellite as it can. One of the systems that is almost completely made by Orbit is the ADCS. One of the big challenges when developing a CubeSat is the power budget. The satellite has very strict mass and volume restrictions as it is limited to $1.33\,\mathrm{kg}$ and $0.10\,\mathrm{m}^3$ per unit. This puts hard limit on the battery capacity a CubeSat can have. The limited volume also means limited surface area to put solar cells on. All this factors makes the power budget very tight. As a result of this, it is very important that every subsystem on the satellite is as energy efficient as possible. This especially applies for the ADCS as it is estimated to be one of the most energy consuming subsystems on the satellite. A natural choice of controller to use on the ADCS might therefore be Model Predictive Control (MPC) as this is often good when efficiency is important. The MPC has another advantage, as it is very good at handling constrained control problems, like controlling the attitude of the satellite. Not only does the satellite have very limited actuation possibilities as it only has mangetorquers which can produce a maximum of $0.2\,\mathrm{Am}^2$. It is also very constrained as the torques are a result of the cross product between the magnetic moment and the magnetic field of the Earth. The big downside of the MPC is the huge amount of computing power that is required to continuously solve the optimization problem live fast enough to keep up with the system. This means that traditionally the only systems that could use MPC were slove systems that had access to a lot of computing power. For example in some kind of production setting [25]. Around 2002 [1], [25] development explicit solutions to the MPC problem started to emerge. The idea behind the explicit Model Predictive Control (eMPC) is to find all the explicit solutions to the MPC problem offline and then to create a loop-up table of all the solutions to be used online. This is achieved by transforming the MPC problem into a multiparametric quadratic programming (mpQP) . By use of this technique faster

systems and systems with limited computing power could also get access to the MPC. The eMPC fits perfectly for the ADCS of a CubeSat as it is a very constrained control problem with high demand for efficiency, but with limited computing power. This is not the first satellite that tries to take advantage of the eMPC, it has been written about it in [15] and [8].

1.3 Hardware in the Loop

A big challenge when developing systems for space is testing. Any system that is going to space only has one shot at working. If it does not work, it is lost for ever. As a result making a reliable and well tested system is a priority for most teams developing something for space. One big challenge when developing tests for space applications on ground is that the environment on Earth is very different from the space environment which is very difficult to recreate on Earth. To overcome this, many teams relay heavily on simulations to test their satellite. The Hardware in the Loop (HiL) concept tries to take this one step further by linking the simulation and hardware together. By including the true hardware and software in the simulation, tests can be taken to a new level. The system can actually be tested with realistic inputs and outputs. This might revile new errors that could not be found by separate simulations and hardware tests.

1.4 Contributions From This Thesis

- The use of eMPC for controlling the attitude of a satellite with magnetorquers is explored.
- Two eMPCs were created and tested in simulation. One was similar to the one described in [15] and the second was developed using a tracking cost function.
- The simulation environment gotten from the MOVE-II team was updated so it fitted, the "Selfie-Sat" with the intent to use the simulation in a HiL test setup.
- An initial design for a HiL test setup for the satellite based on the work in [14] was developed.

Chapter 2

Background

2.1 Attitude Mechanics

In this section the fundamental theory for attitude mechanics will be presented. It is the basis for the controller that is developed.

2.1.1 Reference frames

When working with satellite mechanics, there are a few reference frames that are useful. They will be described in the following:

Body Frame The Body Frame $\mathcal{F}_b$ is defined to have its origin at the center of mass of the satellite and has its axes aligned so that the inertia matrix of the satellite only has diagonal elements. This means that the Body Frame will always follow the motions of the satellite. Defining the Body Frame in this way should not lose any generality, because one can always go from any Body Frame to this Body Frame with a simple transformation according to the parallel axis theorem [6].

Earth Centered Inertia Frame The Earth Centered Inertia Frame (ECI) $\mathcal{F}_i$ is assumed to be an Inertial Frame so anything in relation to an Inertial Frame will be in relation to this frame. The ECI Frame is defined to have its origin at the center of the Earth. The z-axis goes through the north pole, the x-axis is oriented so that it points directly at the center of the sun on the vernal equinox and the y-axis is defined to complete the right-hand rule [17].

Earth Centered Earth Fixed Frame The Earth Centered Earth Fixed Frame (ECEF) $\mathcal{F}_e$ has its origin at the center of the Earth. Its z-axis goes through the north pole, the x-axis goes through the Earth at zero longitude and latitude while the y-axis is defined to complete the right-hand rule. This means that the frame moves with the Earth. This is useful when talking about things that follow the movement of the Earth like the Earth's magnetic field [17].

Orbit Frame The Orbit Frame $\mathcal{F}_o$ has its origin in the center of gravity of the satellite. Its z-axis is defined to always point directly towards the center of the Earth, the y-axis is perpendicular to the orbit plane and points in the same direction as the negative orbit normal while the x-axis is defined to complete the right-hand rule. For a circular orbit it means that the x-axis is parallel to the velocity vector of the satellite [16].

2.1.2 Attitude Kinematics

Unit quaternions are chosen to represent the attitude of the satellite as they provide a singularity free repression of attitude. Where a unit quaternion is defined as $\mathbf{q} = [\eta \ i\epsilon_1 \ j\epsilon_2 \ k\epsilon_3]^T$ and $|\mathbf{q}| = 1$. The relationship between the unit quaternion and the angular velocity is given by Equation 2.1 [6].

$$\dot{\mathbf{q}}_b^i = \mathbf{T}(\mathbf{q}_b^i)\omega_{b/i}^b \tag{2.1a}$$

$$\mathbf{T}(\mathbf{q}) = \frac{1}{2}\begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \tag{2.1b}$$

Where $\mathbf{q}_b^i$ represents the rotation from the Body Frame to the Inertia Frame and $\omega_{b/i}^b$ is the angular speed between the Body Frame and the Inertia Frame.

2.1.3 Attitude dynamics

The attitude dynamics of the satellite is assumed to be the same as of a rigid body where we only look at the rotational dynamics as the transnational movement has no effect on the attitude. The dynamic equations for the rotation of a rigid body can be seen in Equation 2.2 [6].

$$\dot{\boldsymbol{\omega}}_{b/i}^b = \mathbf{I}^{-1}(\mathbf{S}(\mathbf{I}\boldsymbol{\omega}_{b/i}^b)\boldsymbol{\omega}_{b/i}^b + \boldsymbol{\tau}^b) \tag{2.2}$$

Where $\boldsymbol{\omega}_{b/i}^b$ is the angular rotation speed between the Body Frame and Inertia Frame, $\mathbf{I}$ is the inertia matrix and $\boldsymbol{\tau}^b$ represents all the external torques acting on the satellite including the ones produced by the ADCS. The $\mathbf{S}(\mathbf{a})$ is the skew-symmetric matrix operator and creates a skew-symmetric matrix from the vector $\mathbf{a}$. As one can see from Equation 2.2, the dynamic of the satellite is entirely decided by the inertia matrix. The Body Frame can always be defined in a way where the inertia matrix $\mathbf{I}$ only has diagonal elements so $\mathbf{I} = diag(i_{xx}, i_{yy}, i_{zz})$. Then Equation 2.2 can be written as Equation 2.3.

$$i_{xx}\dot{\omega}_x = (i_{zz} - i_{yy})\omega_y\omega_z + \tau_x \tag{2.3a}$$

$$i_{yy}\dot{\omega}_y = (i_{xx} - i_{zz})\omega_x\omega_z + \tau_y \tag{2.3b}$$

$$i_{zz}\dot{\omega}_z = (i_{yy} - i_{xx})\omega_x\omega_y + \tau_z \tag{2.3c}$$

2.1.4 External Torques

The external torques acting on the satellite can be divided into two categories: Control torque $\boldsymbol{\tau}_c^b$ and disturbance torques $\boldsymbol{\tau}_d^b$ so $\boldsymbol{\tau}^b = \boldsymbol{\tau}_c^b + \boldsymbol{\tau}_d^b$. The control torque is created by a coil producing a magnetic moment that interacts with the magnetic field of the Earth to create a torque according to Equation 2.4.

$$\boldsymbol{\tau}_c^b = \mathbf{S}(\mathbf{m}_c^b)\mathbf{B}^b \tag{2.4}$$

There are several disturbance torques acting on the satellite. The normal ones to consider are gravity gradient, parasitic dipole-moment, aerodynamic drag and solar radiation pressure. For the control design only the gravity gradient torque will be considered. The gravity gradient, parasitic dipole-moment and aerodynamic drag torques will be modeled in the simulation. The solar radiation pressure is considered to be so small that it will not be included in the simulation.

Gravity Gradient Torque Gravity gradient torque is a result of different parts of the satellite experiencing different gravitational forces. Since the gravitational field is dependant on the distance from the center of the Earth, the field is not uniform. The gravity gradient can be expressed in vector form as in Equation 2.5

$$\vec{\tau} = -\int_V \vec{r} \times \mu \frac{(\vec{R}+\vec{r})}{|\vec{R}+\vec{r}|^3} dm \approx \frac{3\mu}{R^5}\vec{R} \times \tilde{I} \cdot \vec{R} \tag{2.5}$$

Where V is the volume of the satellite, $\vec{r}$ is the distance from the satellite center of mass to the small mass element dm, $\vec{R}$ is the vector from the center of the Earth to the gravitation center of the satellite and μ is the gravitational constant of the Earth. The formula can be expressed in matrix form relative to the Body Frame by replacing $\vec{R}$ with the z-axis of the Orbit Frame and its magnitude. The z-axis of the Orbit Frame can be expressed in the Body Frame by a simple rotation $\mathbf{z}_o^b = \mathbf{R}_o^b \mathbf{z}_o^o$. This leads to the matrix form in Equation 2.6 [16].

$$\boldsymbol{\tau}_g^b = 3\omega_0^2 \mathbf{S}(\mathbf{z}_o^b)\mathbf{I}\mathbf{z}_o^b \tag{2.6}$$

ω_0 is the angular speed of the orbit and the relationship $\mu/|R| = \omega_0^2$ is used, which is true for circular orbits.

Parasitic Dipole-Moment The parasitic dipole-moment is caused by unwanted magnetic moments created by the satellite. These moments can be caused by ferromagnetic materials in the satellite or current loops in the electronics of the satellite. The parasitic dipole-moment creates a torque in the same way the coils do as shown in Equation 2.7.

$$\boldsymbol{\tau}_p^b = \mathbf{m}_p^b \times \mathbf{B}^b \tag{2.7}$$

Aerodynamic Drag In Low Earth Orbit (LEO) there is still sufficient atmospheric density to create drag forces which lead to torques acting on the satellite. The force acting on a flat surface can be calculated using Equation 2.8 [18].

$$d\mathbf{F}_i^b = -\frac{1}{2} C_D \rho v^2 \cos\alpha A_i \tag{2.8}$$

Where C_D is the drag coefficient, ρ is the atmospheric density, v is the magnitude of the relative velocity between the atmosphere and the satellite, α is the angle between the surface norm and the relative velocity and A_i is the area of the surface. Note that the force is only created if $\cos\alpha > 0$, otherwise the surface is not facing in the direction of the relative velocity and no drag is created. The torque generated from this force can be expressed as shown Equation 2.9.

$$\boldsymbol{\tau}_i^b = \mathbf{S}(\mathbf{F}_i^b)\mathbf{l}_i^b \tag{2.9}$$

Where $\mathbf{l}_i$ is the distance from the center of gravity to the center of pressure of the surface A_i. The total aerodynamic drag torque is then given by summing the torque generated by each surface i of the satellite where $\cos\alpha > 0$.

2.2 Hardware in the Loop

Hardware in the Loop (HiL) is a technique for testing a controller on the hardware without having to assemble the complete product and put it in its intended operational environment. This is achieved by having the hardware that needs to be tested interact with a simulation of the plant instead of the actual plant itself. This way of testing has several advantages as testing on the actual plant can be very expensive or sometimes even impossible as for satellites. Once the satellite is launched, it can not come back. HiL normally works by having a simulation of the plant and models of the sensors. The output of the modeled sensors is then given to the hardware through similar interfaces as it would get the real sensor data. Afterwards the controller processes the sensor data and calculates the appropriate actuation. This actuation is then given back to the simulation through similar interfaces as the true actuator uses. This allows the simulation to propagate to the next step while taking into account the actuation of the hardware controller. The relation between hardware and simulation can be seen in Figure 2.1.

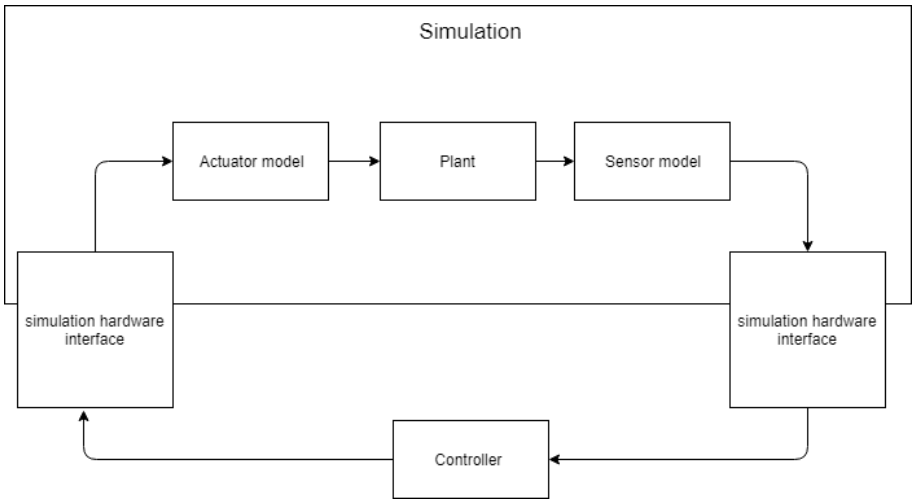


Figure 2.1: Illustration of Hardware in the Loop concept.

The HiL approach has been used by several other CubeSat projects to great effect in [14], [21] and [20]. Each project differs slightly, but the underlying principal is the same. The biggest differences are in how much hardware they included in the loop. In [21] is almost no interaction between the simulation and hardware and it is using almost all of the sensors and actuators. This is achieved by having an extensive lab setup including a Helmholtz cage to create a magnetic field representing the one the satellite will experience in space. This has the main advantage that the system that is actually tested is very similar to the final system. The main drawback is that the test setup is not exactly the same as in space so some modifications have to be made in the software. There are also some problems with evaluating the result as there are a lot more disturbance forces in the test setup than the satellite would experience in orbit. The solution to this problem presented in [21] was to model the disturbances and take them into account when evaluating the result.

In [14] and [20] on the other hand there is much more interaction between the simulation and hardware. They are only testing the main micro-controller and using the simulations to generate all sensor values and to receive the actuation output. The main difference is in how the interface between the simulation and hardware works. [20] has made modules that can generate the current and voltage values the sensors would produce based on input from the simulation. [14] connects to the hardware digitally as almost all of the sensors information is at some point transferred on a digital bus. The main advantage of this approach is that there are no limitations to what kind of tests can be run and the software can be completely identical to the final product. The main drawback is that you get to test less of the complete system and you are still dependent on accurate models.

In all of [14], [21] and [20] it is reported that they found bugs or other errors in their system they would not have found if they did not do HiL tests. This showcases how important it is to do such tests.

2.3 Explicit Model Predictive Control

An explicit model predictive control (eMPC) is a way to get an approximation of the optimal control law normally gained by model predictive control (MPC) without the computa-

tional effort needed to constantly solve the MPC live. This can be achieved by formulating the linear constrained MPC problem as a multiparametric quadratic program (mpQP). The mpQP will give a piecewise affine (PWA) function that can be stored in memory. During any given instant one can look up into the PWA to find the approximate optimal gain [15].

2.3.1 Multiparametric Quadratic Program and Model Predictive Control

Multiparametric programming is finding not only the optimal solution to a single optimization problem, but an explicit solution to a range of similar optimization problems based on a parameter. MpQP is then concerned with finding this explicit solutions to quadratic programming. The general formulation of a mpQP can be seen in Equation 2.10

$$V^*(\boldsymbol{\theta}) = \min_{\mathbf{z}} \mathbf{z}^T\mathbf{H}\mathbf{z} + \boldsymbol{\theta}^T\mathbf{F}\mathbf{z} + \mathbf{c}^T\mathbf{z} \tag{2.10a}$$

$$s.t \quad \mathbf{A}\mathbf{z} \leq \mathbf{b} + \mathbf{S}\boldsymbol{\theta} \tag{2.10b}$$

The optimal solution to an mpQP will be an continuous PWA function dependent on the parameter θ.

According to [15], the MPC problem can easily be formulated as a mpQP. For a linear discreet system defined by Equation 2.11 where $\mathbf{x}_k \in \mathbb{R}^n$ are the states and $\mathbf{u}_k \in \mathbb{R}^m$ are the inputs to the system at time k. $\mathbf{A} \in \mathbb{R}^{nxn}$ and $\mathbf{B} \in \mathbb{R}^{mxn}$ form the stabilizing pair $(\mathbf{A}, \mathbf{B})$.

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{2.11a}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k \tag{2.11b}$$

For controlling the system towards the origin, the MPC is normally formulated as shown in Equation 2.12

$$\min_{\mathbf{U},\mathbf{s}}[J(\mathbf{U}, \mathbf{x}(t)) + \rho||\mathbf{s}||^2_{\mathcal{L}_2}] \tag{2.12a}$$

$$s.t \quad \mathbf{y}_{min} - \mathbf{s} \leq \mathbf{y}_{t+k|t} \leq \mathbf{y}_{max} + \mathbf{s}, k = 1, ..., N \tag{2.12b}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_{t+k} \leq \mathbf{u}_{max}, k = 0, ..., M-1 \tag{2.12c}$$

$$\mathbf{u}_{t+k} = \mathbf{K}\mathbf{x}_{t+k|t}, M \leq k \leq N-1 \tag{2.12d}$$

$$\mathbf{x}_{t|t} = \mathbf{x}(t) \tag{2.12e}$$

$$\mathbf{x}_{t+k+1|t} = \mathbf{A}\mathbf{x}_{t+k|t} + \mathbf{B}\mathbf{u}_{t+k}, k \geq 0 \tag{2.12f}$$

$$\mathbf{y}_{t+k|t} = \mathbf{C}\mathbf{x}_{t+k|t}, k \geq 0 \tag{2.12g}$$

Where $J(\mathbf{U}, \mathbf{x}(t))$ is given by Equation 2.13.

$$J(\mathbf{U}, \mathbf{x}(t)) = \mathbf{x}_{t+N|t}^T \mathbf{P} \mathbf{x}_{t+N|t} + \sum_{k=0}^{N-1} \mathbf{x}_{t+k|t}^T \mathbf{Q} \mathbf{x}_{t+k|t} + \mathbf{u}_{t+k}^T \mathbf{R} \mathbf{u}_{t+k} \tag{2.13}$$

Where $||\mathbf{s}||_{\mathcal{L}_2}$ is the euclidean norm of the slack variable $\mathbf{s}$, ρ is the penalty weight of the slack variables, $\mathbf{U} \triangleq [\mathbf{u}_k^T, \mathbf{u}_{k+1}^T, ..., \mathbf{u}_{k+N-1}^T]^T$ is the vector of inputs at each sample time, $\mathbf{s} \triangleq [\mathbf{s}(k)^T, ..., \mathbf{s}(k+N-q)^T]^T$ is the vector of slack variables at each sample time, $\mathbf{K}$ is the control gain matrix used when there are no constraints, $\mathbf{x}_{t+k|t}$ is the prediction of $\mathbf{x}_{t+k}$ at time t, and N is the output constraint horizon while M is input constraint horizon. The solution to this problem will be a sequence of optimal inputs $\mathbf{U}^*$ given by $\mathbf{U}^* = [\mathbf{u}_k^{*T}, \mathbf{u}_{k+1}^{*T}, ..., \mathbf{u}_{k+N-1}^{*T}]^T$ and $\mathbf{s}^* = [\mathbf{s}^*(k)^T, ..., \mathbf{s}^*(k+N-q)^T]^T$.

The MPC problem can be formulated as a mpQP as shown in Equation 2.14 according to [25], where $\mathbf{z} \triangleq \mathbf{U} + \mathbf{H}^{-1}\mathbf{F}^T\mathbf{x}(t)$. $\mathbf{x}(t)$ is the current state and will be treated as the parameter to the optimization problem.

$$V_{\mathbf{z}}(\mathbf{x}) = \min_{\mathbf{z}} \frac{1}{2}\mathbf{z}^T\mathbf{H}\mathbf{z} \tag{2.14a}$$

$$s.t \quad \mathbf{G}\mathbf{z} \leq \mathbf{W} + \mathbf{S}\mathbf{x}(t) \tag{2.14b}$$

The transformation is made by using the fact that $\mathbf{x}_{t+k|t} = \mathbf{A}^k\mathbf{x}(t) + \sum_{j=0}^{k-1} \mathbf{A}^j\mathbf{B}\mathbf{u}_{t+k-1-j}$ to transform Equation 2.12 to Equation 2.15 and then completing the square to get Equation 2.14. $\mathbf{H} \succ 0$ as $\mathbf{R} \succ 0$ [15]

$$\mathbf{V}(\mathbf{x}(t)) = \frac{1}{2}\mathbf{x}'(t)\mathbf{Y}\mathbf{x}(t) + \min_{\mathbf{U}}\Big\{\frac{1}{2}\mathbf{U}'\mathbf{H}\mathbf{U} + \mathbf{x}'(t)\mathbf{F}\mathbf{U},\ s.t \quad \mathbf{G}\mathbf{U} \leq \mathbf{W} + \mathbf{E}\mathbf{x}(t)\Big\} \tag{2.15}$$

$\mathbf{H}$, $\mathbf{F}$,$\mathbf{Y}$, $\mathbf{G}$, $\mathbf{W}$ and $\mathbf{E}$ are obtained from the MPC formulation and $\mathbf{Q}$ and $\mathbf{R}$. $\mathbf{S}$ is given by $\mathbf{S} = \mathbf{E} + \mathbf{G}\mathbf{H}^{-1}\mathbf{F}'$ [1]

2.3.2 Solving the MpQP

To solve the mpQP the work of [25] will be followed and for a more detailed description of the solution, the reader is referred to his work. The solution to the mpQP problem Equation 2.14 can be solved by applying the Karush-Kuhn-Tucker (KKT) conditions as shown in Equation 2.16.

$$\mathbf{H}\mathbf{z} + \mathbf{G}^T\lambda = 0, \quad \lambda \in \mathbb{R}^q \tag{2.16a}$$

$$\lambda_i(\mathbf{G}^i\mathbf{z} - \mathbf{W}^i - \mathbf{S}^i\mathbf{x}) = 0, \quad i = 1, ..., q \tag{2.16b}$$

$$\lambda \geq 0 \tag{2.16c}$$

$$\mathbf{G}\mathbf{z} - \mathbf{W} - \mathbf{S}\mathbf{x} \leq 0 \tag{2.16d}$$

Where q is the number of inequality constraints and a matrix with superscript i indicated the ith row of the matrix. In Equation 2.16 it can be seen that $\mathbf{z}$ can be given by Equation 2.17 as $\mathbf{H}$ has full rank.

$$\mathbf{z} = -\mathbf{H}^{-1}\mathbf{G}^T\lambda \tag{2.17}$$

Suppose that $\mathbf{z} * (x)$ is the optimal solution. We can then define active constraints as constraints where $\mathbf{G}^i\mathbf{z}*(x) - \mathbf{W}^i - \mathbf{S}^i x = 0$ and inactive constraints as constraints where $\mathbf{G}^i\mathbf{z} * (x) - \mathbf{W}^i - \mathbf{S}^i x \leq 0$. We also define $\mathcal{A}$ as a set of all indices of active constraints, $\mathcal{A} = \{i \mid \mathbf{G}^i\mathbf{z} * (x) = \mathbf{W}^i - \mathbf{S}^i x\}$. From $\mathcal{A}$ we can form the matrices $\mathbf{G}^{\mathcal{A}}$, $\mathbf{W}^{\mathcal{A}}$ and $\mathbf{S}^{\mathcal{A}}$ and the Lagrange multipliers $\lambda^{\mathcal{A}} \geq 0$ corresponding to the active constraints. Assuming that $\mathbf{G}^{\mathcal{A}}$ has full row rank $\lambda^{\mathcal{A}} = -(\mathbf{G}^{\mathcal{A}}\mathbf{H}^{-1}(\mathbf{G}^{\mathcal{A}})^T)^{-1}(\mathbf{W}^{\mathcal{A}} + \mathbf{S}^{\mathcal{A}}\mathbf{x})$. This gives the final solution Equation 2.18.

$$\mathbf{z} = \mathbf{H}^{-1}(\mathbf{G}^{\mathcal{A}})^T(\mathbf{G}^{\mathcal{A}}\mathbf{H}^{-1}(\mathbf{G}^{\mathcal{A}})^T)^{-1}(\mathbf{W}^{\mathcal{A}} + \mathbf{S}^{\mathcal{A}}\mathbf{x}) \tag{2.18}$$

As long as a given $\mathcal{A}$ stays the optimal active set, the solution in Equation 2.18 remains the optimal solution as a function of $\mathbf{x}$. A given $\mathcal{A}$ remains the optimal active set as long as $\mathbf{z}$ stays feasible and the Lagrange multipliers remains non-negative. These conditions are given by the inequalities Equation 2.19. These inequalities describe a polyhedron in the state space called the critical region CR_0.

$$\mathbf{G}\mathbf{H}^{-1}(\mathbf{G}^{\mathcal{A}})^T(\mathbf{G}^{\mathcal{A}}\mathbf{H}^{-1}(\mathbf{G}^{\mathcal{A}})^T)^{-1}(\mathbf{W}^{\mathcal{A}} + \mathbf{S}^{\mathcal{A}}\mathbf{x}) \leq \mathbf{W} + \mathbf{S}\mathbf{x} \tag{2.19a}$$

$$-(\mathbf{G}^{\mathcal{A}}\mathbf{H}^{-1}(\mathbf{G}^{\mathcal{A}})^T)^{-1}(\mathbf{W}^{\mathcal{A}} + \mathbf{S}^{\mathcal{A}}\mathbf{x}) \geq 0 \tag{2.19b}$$

To solve the mpQP, an expression for $\mathbf{z}$ and the region in which it is valid needs to be found. As shown by Equation 2.18 and Equation 2.19, they are both completely defined by the optimal active set. This means that finding a solution to the mpQP is equivalent with finding all the optimal active sets of the mpQP.

There are several algorithms that can be used to solve the mpQP, but they all follow the same principal of searching for the optimal active sets. They only differ in how they search for it. The algorithm proposed by [1] can be explained as the following. Solve the Linear Programming (LP) in Equation 2.20 to find a feasible $x_0 \in X$ where X is the set of parameters for which the mpQP is to be solved.

$$\max_{\mathbf{x},\mathbf{z},\epsilon} \ \epsilon, \tag{2.20a}$$

$$s.t \quad \mathbf{T}^i\mathbf{x} + \epsilon||\mathbf{T}^i|| \leq \mathbf{Z}, \tag{2.20b}$$

$$\mathbf{G}\mathbf{z} - \mathbf{S}\mathbf{x} \leq \mathbf{W} \tag{2.20c}$$

Using $x = x_0$ in Equation 2.14 to solve the mpQP as a normal Quadratic Programming (QP) and to find the optimal active set for x_0. Use Equation 2.19 to describe the region called CR_0 where the solution is valid. Then use the hyperplanes of CR_0 to define new regions in X as shown in Figure 2.2. For each of the new regions repeat this process until the entire space of X is explored.

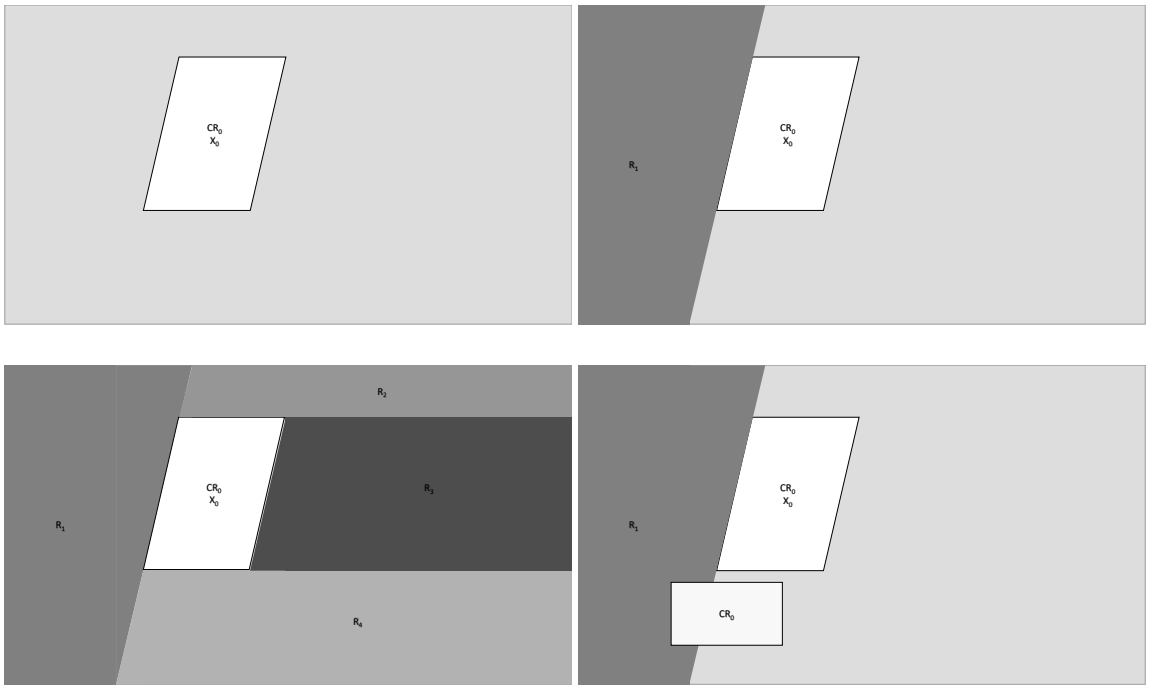


Figure 2.2: Illustrations showing how the search regions are created and that that a critical region might belong to two search regions. Inspired by [25]

This method has several draw backs. The first is that the new regions do not take into account where the critical regions will be so there can exist critical regions which are in multiple regions. This means that they will be found multiple times by the algorithm. The second problem is that LP problem has to be solved multiple times which takes a lot of time.

Another approach has been developed by [25]. This method takes advantage of the hyperplanes separating the critical regions. He defines two types of separating hyperplanes $\mathcal{H}$ separating two critical regions CR_0 and CR_i. Where CR_0 and CR_i are neighboring critical regions and CR_0 is defined by the optimal active set $\{i_1, i_2, ..., i_k\}$ and Equation 2.19. The two types are:

Type 1 *If $\mathcal{H}$ is given by $\mathbf{G}^{i_{k+1}}\mathbf{z}_0^*(\mathbf{x}) = \mathbf{W}^{i_{k+1}} + \mathbf{S}^{i_{k+1}}\mathbf{x}$, then the optimal active set in CR_i is $\{i_1, ...i_k, i_{k+1}\}$ or $\{i_2, ...I_k, i_{k+1}\}$*

Type 2 *If $\mathcal{H}$ is given by $\lambda_0^{i_k}(\mathbf{x}) = 0$, then the optimal active set in CR_i is $\{i_1, ...i_k, i_{k-1}\}$ or $\{i_2, ...I_k, i_{k-1}\}$*

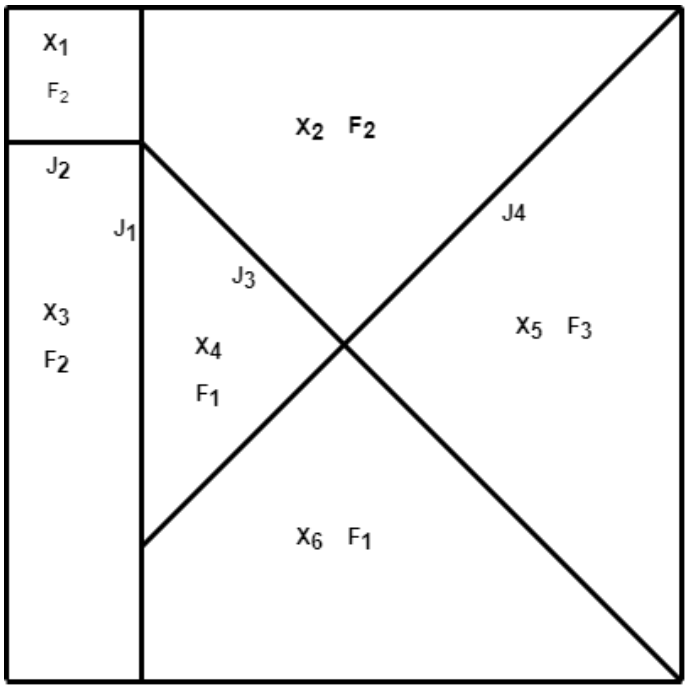
So the main idea for the algorithm presented in [25] is to find an initial optimal active set with the LP in Equation 2.20 as in the previous algorithm. For each of the hyperplanes defining CR_0 based on the optimal active set, investigate which type of hyperplane it is and thereby determine what the active set is in the neighboring critical region CR_i. Use the active set to define CR_i. Repeat this process until the entire space X is explored. This algorithm has several advantages. The first is that it uses the property of the critical regions to find future optimal active sets. This means that there is no need to solve multiple QPs to find new active sets. It also searches the space of X in a better way so it does not find the same critical region multiple times. It should be noted that this is a very simplified

way of describing the algorithm only outlining the general idea. For a more in-depth view of the algorithm the reader is referred to [25].

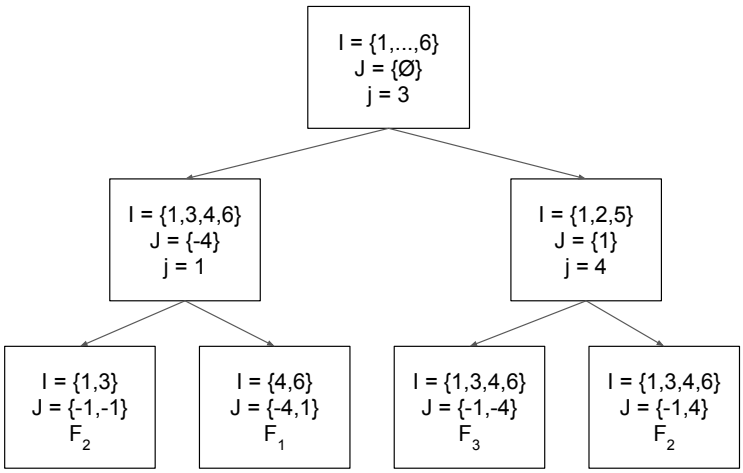
2.3.3 Representing the Explicit Model Predictive Controller as a Binary Tree

The solution to the eMPC problem is a PWA control law that needs to be included on the computer implementing the controller. To achieve that, the controller needs a way to determine in which region of the PWA the current state is. The simplest way of evaluating the PWA is a sequential search through all the regions of the PWA. This is however not very efficient as you might have to search through the entire PWA to find a solution. The efficiency can be increased somewhat by starting the search with the same region as in the last solution and then expanding the search to more and more neighboring regions. In [25] there are multiple reasons stated why this is not always a good idea, which all revolve around sudden changes to the state. This can be caused by changes in the set point, mode switching, disturbances and slow sampling. [25] suggests a much faster way off evaluating the PWA by turning it into a binary search tree. The leaf nodes in the tree contain the appropriate control law and all the parent nodes contain an inequality to determine if the search should continue to the left or right. In the following only the relation between the tree and the regions of the PWA will be explained. For information about the algorithm to generate the tree, the reader is referred to [25].

For each hyperplane j given by $a_j^T x = b_j$ defining one of the regions in the PWA, a function $d_j(x) = a_j^T - b_j$ is defined. Each parent node contains an index to one of the $d_j(x)$ functions to be evaluated. When searching the tree at each node, evaluate if $d_j(x) < 0$. If $d_j(x) < 0$, go to the left child node, if not, go to the right child. To get an intuition for how the search tree divides up the state-space, one can also imagine that each node contains two sets $\mathcal{I}$ and $\mathcal{J}$. $\mathcal{I}$ is a set of indexes containing regions that x can still be a part of. $\mathcal{J}$ contains indexes of previously tested functions $d_j(x)$ and a sign to show what they evaluated to. So a node with $\mathcal{I} = \{3, 5\}$ and $\mathcal{J} = \{-3, 2\}$ means, that x could still be in region 3 and 5 and that $d_3(x) < 0$ and $d_2(x) > 0$.



(a) Example of regions dividing up a state space. Inspired by [25]



(b) Example binary search tree based on the regions in the figure above. Inspired by [25]

In Figure 2.3a one can see an example of how a two dimensional state space could be divided into 6 regions, which each have their own control law F_i. In Figure 2.3b one can see an example of a binary search tree for the regions. Some of the regions share a common control law, which is used to make as few nodes as possible.

Chapter 3

ADCS Architecture

The ADCS has three main objectives that should be achieved. The first task is to detumble the satellite after it is separated from the rocket. The second objective is to orient the satellite in a way that maximizes charging capabilities of the satellite. The third is to point the camera towards the Earth when the satellite should take pictures aka the "selfies". What these modes mean in more technical terms is presented in section 3.1. For the detumbling a B-dot algorithm is used that is provided by the magnetorquer board iMTQ. For the other two modes the two self developed eMPC presented in section 3.3 are used.

3.1 ADCS Modes

The three control modes of the ADCS are: detumbling mode, charging mode and camera pointing mode.

Detumbling Mode This mode is mainly used to reduce the angular velocity that the satellite has after it is deployed from the rocket. When the satellite is deployed from the rocket it will have an unknown angular velocity that needs to be reduced before normal operations of the satellite can start and the camera boom can be released. It will also be used as a failsafe to keep the satellite under control, in case it starts spinning up for some reason. For this reason the Detumbling mode uses a simple controller with very few sensor to make sure it always works. The chosen controller is a B-dot controller that comes with the iMTQ magnetorquer board.

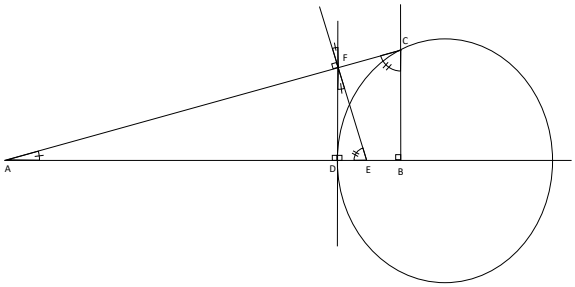
$$\mathbf{m}_d = k\dot{\mathbf{B}}^b \tag{3.1}$$

The B-dot control law works as shown in Equation 3.1 where $\mathbf{m}_d$ is the desired magnetic moment that is used as an input to the mangetorquers, k is the control gain and $\dot{\mathbf{B}}^b$ is the derivative of the measured magnetic field of the Earth in Body Frame. The controller works not by trying to reduce the angular velocity but by trying to reduce the change in the magnetic field. If the magnetic field is assumed to be static in the Inertia Frame, reducing the angular velocity is equivalent to reducing the change in the magnetic field in the Body Frame, because a rotating body is the only way to have a changing magnetic field in the

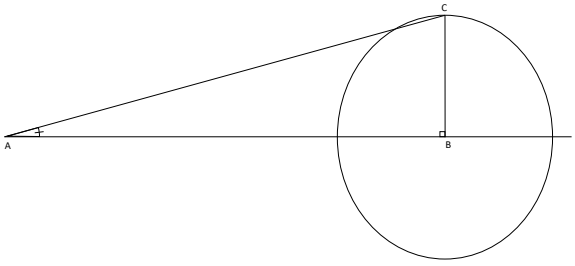
Body Frame. It is a valid assumption that the magnetic field is static in the Inertia Frame, as the magnetic field changes very slowly over one orbit.

Charging Mode The charging mode is used to try and maximize the charging of the satellites batteries. The charging can be maximized by two criteria: Maximizing the total area of solar cells facing the Sun at any given time and giving each solar cell equal exposure to the Sun. The first criteria is self evident as the power produced by the solar cells are directly proportional to the solar cell area facing the Sun. The second criteria is important as the solar cells have reduced efficiency if they are too hot and the only way of cooling down the solar cells is having them face away from the Sun. This means that the charging efficiency is the highest, if all the solar cells are kept as cool as possible which is achieved by having equal exposure to the Sun. On the satellite there are only solar cells on the x and y panels. This means that the optimal solution is one where the z-axes is perpendicular to a vector pointing towards the Sun and with some rotation around the z-axis. At any given point in the orbit, multiple orientation can have the z-axis perpendicular to the Sun vector. To make the criteria easier, the following is observed: If the z-axis of the satellite is normal to the orbit plane of the Earth, the satellite's z-axis is perpendicular to the Sun vector when the it crosses the orbit plane as illustrated in Figure 3.1a. This can be used as an approximation for an orientation where the z-axis of the satellite is perpendicular to the Sun vector. Figure 3.1b shows that the angle, which is representing how much this approximation is wrong, depends on θ. The maximum value of θ is illustrated in Figure 3.1c and can be calculated to $\theta = \arctan(\frac{R}{D}) \approx 2.7040$ when $R = 6.971 \times 10^6\,\mathrm{km}$ and $D = 1.476 \times 10^8\,\mathrm{km}$ is the orbital radius and the distance from the Earth to the Sun respectively.

(a) Illustration showing that when the satellite passes through the orbit plane of the Earth, the z-axis is normal to the Sun vector as long as the z-axis is normal to the plane. Point A represents the Sun, point B represents the center of the Earth, the circle represents the orbit of the satellite and the line segment AB represents the orbital plane of the Earth.



(b) Illustration showing how the angle between two lines intersecting in A is equal to the angle between the two lines perpendicular to the first two lines. This can be seen as Δ ABC must be similar to Δ AEF as they have two angles in common. This means that the angle between line AE and line EF has the same angle as the angle between the line AC and line CB. This shows that the angle between DF and EF must be equal to the angel between AB and AC.



(c) Illustration showing that the maximum θ happens when the point is on the intersection between the circle and the line perpendicular to AB when B is placed in the center of the circle.

Figure 3.1: Different figures showing different properties of the angle θ. θ is always the angel marked with one line.

The z-axis of the ECI Frame is not normal to the orbital plane of the Earth, but rather at a 23.5° angle this is called the obliquity of the Earth. This means that the simplest desired orientation for charging is the Body Frame of the satellite rotated 23.5° around the x-axis relative to the ECI Frame. In quaternions this is $q_b^i = [\ \cos\frac{23.5}{2}\ \ \sin\frac{23.5}{2}\ 0\ 0\ ]'$.

Camera Pointing Mode The camera pointing mode is used to point the camera towards the Earth so that the Earth will be in the background for the "selfies". As the camera is pointing in the z- direction, pointing the camera towards Earth is the same thing as pointing the -z-axis of the satellite towards the Earth or so called nadir pointing. Relative to the ECI Frame, this orientation is not easily expressed as the expression would continuously change throughout the orbit. Instead the orientation relative to the Orbit Frame is used as nadir pointing is always expressed as $q_b^o = [\ 1\ 0\ 0\ 0\ ]'$. Therefore the controller for the camera pointing mode is developed using the dynamics of the systems expressed relative to the Orbit Frame. The estimated states are still relative to the ECI Frame, so a rotation matrix to go from ECI to the Orbit Frame is needed. This rotation matrix can be calculated based on the position and linear velocity of the satellite according to Equation 3.2.

$$\mathbf{R}_1 = \mathbf{S}(\mathbf{R}_2)\mathbf{R}_3 \qquad \mathbf{R}_2 = \mathbf{S}(\mathbf{v}^i)\mathbf{r}^i \qquad \mathbf{R}_3 = -\mathbf{r}^i \tag{3.2a}$$

$$\mathbf{R}_i^o = \begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \mathbf{R}_3\end{bmatrix} \tag{3.2b}$$

Where $\mathbf{r}^i$ and $\mathbf{v}^i$ is the unit vector of the satellite position and linear velocity expressed in the ECI Frame.

3.2 Hardware Architecture

The ADCS hardware consists of two PCBs and five Sun Sensors. The first PCB, called the ADCS PCB, contains the main micro-controller and two Inertial Measurement Units (IMU). The second PCB is the iMTQ [11] which contains the magnetorquers and the necessary components to control them. An overview of the ADCS hardware architecture can be seen in Figure 3.2.

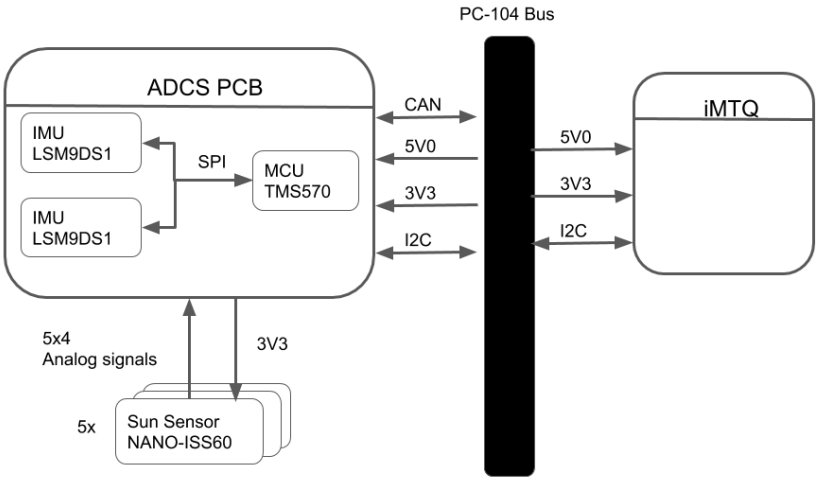


Figure 3.2: Overview of the ADCS hardware. Showing all components and the interfaces between them. The PC-104 represents the main satellite bus connecting all the different subsystems together.

The ADCS uses the CAN bus protocol to talk to the rest of the satellite, and SPI and I2C buses to talk to its internal components. The ADCS PCB also contains all the necessary components to support the MCU and all other components on the board such as voltage regulators, extra memory, CAN transverse and so forth. Only the parts directly related to ADCS as a control system will be discussed future.

ADCS Micro-Controller The ADCS micro-controller is a TMS570LS1224 [24]. It has an ARM Cortex -R4F 32-bits CPU with a double precision FPU. It has two CPUs running in lock step to ensure high reliability. It is the same micro-controller that is used on the On-board Computer. They are both running the same custom made operation system developed by Orbit NTNU to take advantage of the reliability features of the micro-controller. The operation system also comes with all the needed low level drivers and file functionality. This means that all ADCS software will be developed as high level applications.

Inertial Measurement Unit The Inertial Measurement Unit is a nine degrees of freedom unit of type LSM9DS1 [12]. It consists of a accelerometer, gyroscope and magnetometer, all capable of measuring in three directions. It is therefore capable of measuring the acceleration and angular velocity of the satellite as well as the magnetic field surrendering the satellite. Only the gyroscope and magnetometer measurements will be used in the ADCS. The IMU has both a SPI and I2C interface but only the SPI interface will be used.

Sun Sensor The Sun Sensor is a sensor capable of detecting the direction of the Sun. The model used on the ADCS is the Nano Sens Fine Sun Sensor from GOMSpace [19]. The sensor uses the pin hole pricipe. This means that the sensor calculates the direction of the Sun by allowing a small ray of light tto ravel through a small hole. Behind the hole are four light sensitive diodes. The ratio of light between the diodes can then be used to determine the angle of the incoming light which in turn can be used to determine the direction of the Sun. The values of the four pads are transferred to the ADCS-MCU through an I2C bus.

The direction can then be calculated using the equations in Equation 3.3 from the data sheet. Some parts of the equation that relate to calibration parameters are removed to make the equations simpler.

$$x_d = \frac{(A+B)-(C+D)}{A+B+C+D} \qquad y_d = \frac{(A+D)-(B+C)}{A+B+C+D} \tag{3.3a}$$

$$\phi = \arctan\frac{x_d}{y_d} \qquad \theta = \arctan\frac{\sqrt{x_d^2+y_d^2}}{h} \tag{3.3b}$$

$$\mathbf{S}^s = \begin{bmatrix} \cos\theta \\ \sin\theta\cos\phi \\ \sin\theta\sin\phi \end{bmatrix} \tag{3.3c}$$

Where $A,\ B,\ C,\ D$ is the intensity of each diode, h is some calibration parameter and $\mathbf{S}^s$ is the direction of the Sun in the Sensor Frame. The Sun Sensor also comes with a

look-up table for calculating the direction of the Sun based on the diode values. Using the loop-up table should give more accurate results as the table is custom made for each sensor. The sensor also comes with code for using the look-up table so it can be regarded as a black box that gives out the direction of the Sun.

Magnetorquer Board iMTQ The iMTQ from ISIS contains three magnetorquers allowing the board to produce magnetic moments in all directions. Two of the magnetorquers have a soft magnetic core working as an amplifier for the magnetic moment. All the magnetorquers are capable of producing a mangetic moment of $0.2\,\mathrm{Am}^2$. The board also comes with its own micro-controller for controlling the coils and interfacing with the ADCS. The interface between the iMTQ and the ADCS is through an I2C bus. The board has B-dot capabilities and can be commanded to use the B-dot controller to reduce the angular velocity of the satellite. It is this board's B-dot controller that is intended to be used for detumbling the satellite.

3.3 Explicit Model Predictive Control Implementation

For the eMPC implementation the Matlab toolbox MPT3[9] is used. The toolbox has a variety of functions and features that are very useful when developing an eMPC. The most important ones are the "MPCController" and "EMPCController" objects. The "MPCController" object allows for easy formulation of the MPC problem and can transformed into an eMPC problem using a built in function. The MPT3 toolbox has built in solvers for the eMPC problem. The toolbox also has functions for generating C and S-function files for the solved controller. These files are used on the micro-controller and in the simulation to test the micro-controller simulations.

To create the MPC problem, the dynamics of the satellite need to be linearized as they are highly nonlinear. The camera pointing and charging mode have two very different goals and therefore require their own MPC problem. The formulation of the tow problem will be presented in the following.

3.3.1 Camera Pointing Mode eMPC Formulation

The eMPC formulation for the nadier pointing satellite using magnetorquers has already been developed in [15] so the formulation presented there will be used. The kinematic and dynamic equations presented in section 2.1 need to be transformed so they apply to the relation between the Body Frame and Orbit Frame instead of the relation between Body Frame and ECI Frame. For the quaternions this is straight forward as all that is needed is to replace $\boldsymbol{\omega}^b_{b/i}$ with $\boldsymbol{\omega}^b_{b/o}$. The kinematic equation then becomes as in Equation 3.4.

$$\dot{\mathbf{q}}^o_b = \mathbf{T}(\mathbf{q}^o_b)\boldsymbol{\omega}^b_{o/i} \tag{3.4}$$

For the angular velocity we assume a circular orbit and therefore have the relationship in Equation 3.5.

$$\boldsymbol{\omega}^b_{i/b} = \boldsymbol{\omega}^b_{o/b} + \mathbf{R}^b_o \boldsymbol{\omega}^o_{i/o} \qquad \dot{\boldsymbol{\omega}^b_{i/b}} = \dot{\boldsymbol{\omega}^b_{o/b}} + \dot{\mathbf{R}}^b_o \boldsymbol{\omega}^o_{i/o} \tag{3.5}$$

Where $\boldsymbol{\omega}^b_{o/b} = [\ 0\ -\omega_0\ 0\ ]^T$ and ω_0 is the angular velocity of the satellite around the Earth. The external torques that are considered for this system are the gravity gradient torque $\boldsymbol{\tau}^b_g$ and the torque from the magnetorquers $\boldsymbol{\tau}^b_m$ as presented in Equation 2.6 and Equation 2.4. Combining Equation 2.6, Equation 2.4, Equation 2.2 and Equation 3.5 gives Equation 3.6.

$$\begin{aligned}\dot{\boldsymbol{\omega}^b_{o/b}} = \mathbf{I}^{-1}\mathbf{S}(\mathbf{I}(\boldsymbol{\omega}^b_{o/b} + \mathbf{R}^b_o\boldsymbol{\omega}^o_{i/o}))(\boldsymbol{\omega}^b_{o/b} + \mathbf{R}^b_o\boldsymbol{\omega}^o_{i/o}) \\ + \mathbf{S}(\boldsymbol{\omega}^b_{o/b})\mathbf{R}^b_o\boldsymbol{\omega}^o_{i/o} + 3\omega_0^2\mathbf{I}^{-1}\mathbf{S}(\mathbf{z}^b_o)\mathbf{I}\mathbf{z}^b_o + \mathbf{I}^{-1}\mathbf{S}(\mathbf{m^b})\mathbf{B}^b\end{aligned} \tag{3.6}$$

This is then linerized around the point $\mathbf{x}_0 = [\ \mathbf{q}^o_{b\_0}\ \ \boldsymbol{\omega}^b_{o/b\_0}\ ]^T = [\ 1\,0\,0\,0\,0\,0\,0\ ]^T$. Given in Equation 3.7.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & -8k_w\omega_0^2 & 0 & 0 & 0 & 0 & (1-k_w)\omega_0 \\ 0 & 0 & 6k_y\omega_0^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2k_z\omega_0^2 & (k_z-1)\omega_0 & 0 & 0 \end{bmatrix} \tag{3.7a}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{B^b_z}{i_{xx}} & -\frac{B^b_y}{i_{xx}} \\ -\frac{B^b_z}{i_{yy}} & 0 & \frac{B^b_x}{i_{zz}} \\ \frac{B^b_z}{i_{zz}} & \frac{B^b_x}{i_{zz}} & 0 \end{bmatrix} \tag{3.7b}$$

Where $k_x = \frac{i_{yy}-i_{zz}}{i_{xx}}$, $k_y = \frac{i_{xx}-i_{zz}}{i_{yy}}$, $k_z = \frac{i_{yy}-i_{xx}}{i_{zz}}$ and i_{xx}, i_{yy} and i_{zz} are the diagonal elements of the inertia matrix. This linearization has assumed only elements on the diagonal of the inertia matrix. From the linear model presented in Equation 3.7 it can be seen that the first element of $\mathbf{q^0_b}$ is not controllable. This is not a problem as it can be controlled by the remaining elements of $\mathbf{q^0_b}$ as $|\mathbf{q^0_b}| = 1$ must be true for quaternions used to represent rotation.

The cost function in the MPC formulation will be a quadratic cost function with a final penalty $\mathbf{P}$ calculated by solving the discre-time Riccati equations. In addition all the states and actuators have inequality constraints to form an upper and lower bound. So the full MPC formulation will be as in Equation 3.8.

$$\min_{\mathbf{U}}[J(\mathbf{U}, \mathbf{x}(t))] \tag{3.8a}$$

$$s.t \quad \mathbf{x}_{min} \leq \mathbf{x}_{t+k|t} \leq \mathbf{x}_{max},\ k = 1, ..., N \tag{3.8b}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_{t+k} \leq \mathbf{u}_{max},\ k = 0, ..., N-1 \tag{3.8c}$$

$$\mathbf{x}_{t|t} = \mathbf{x}(t) \tag{3.8d}$$

$$\mathbf{x}_{t+k+1|t} = \mathbf{A}\mathbf{x}_{t+k|t} + \mathbf{B}\mathbf{u}_{t+k},\ k \geq 0 \tag{3.8e}$$

$$\tag{3.8f}$$

$$J(\mathbf{U}, \mathbf{x}(t)) = \mathbf{x}^T_{t+N|t}\mathbf{P}\mathbf{x}_{t+N|t} + \sum_{k=0}^{N-1} \mathbf{x}^T_{t+k|t}\mathbf{Q}\mathbf{x}_{t+k|t} + \mathbf{u}^T_{t+k}\mathbf{R}\mathbf{u}_{t+k} \tag{3.8g}$$

There is one problem with this formulation. The $\mathbf{B}^b$ is actually time-varying and the MPC problem can not have a time-varying $\mathbf{B}$ matrix. The solution to this is the same as presented in [15]. Instead of having a time-varying $\mathbf{B}^b$ the average value is used. The only problem then is for when $\mathbf{B}^b$ changes sign. To solve this eight controllers are created, one for each of the sign combinations of $\mathbf{B}^b$. Then the controller chooses which of this eight controllers to use based on the measured sign of the magnetic field.

In the same way as described in [15], the model is also scaled to try and improve the numerical stability of the solvers. Because of this scaling and the fact that the states in the controller are different from the states of the simulation, some prepossessing is needed on the simulation data before it can be used in the controller. For the simulation implementation of the controller, it can be said that it consists of three parts. The first part transforms the simulation's states into the states used by the controller, the second part selects which of the eight controllers to use based on the magnetic field, and the last part the controller itself. The controllers are the S-functions generated by the MPT3 toolbox while the control selector is a S-function that sends out trigger signals to the appropriate controllers. This means that only one controller will run at a time to make the simulation run a bit faster. This selection mechanism can be seen in Figure 3.3

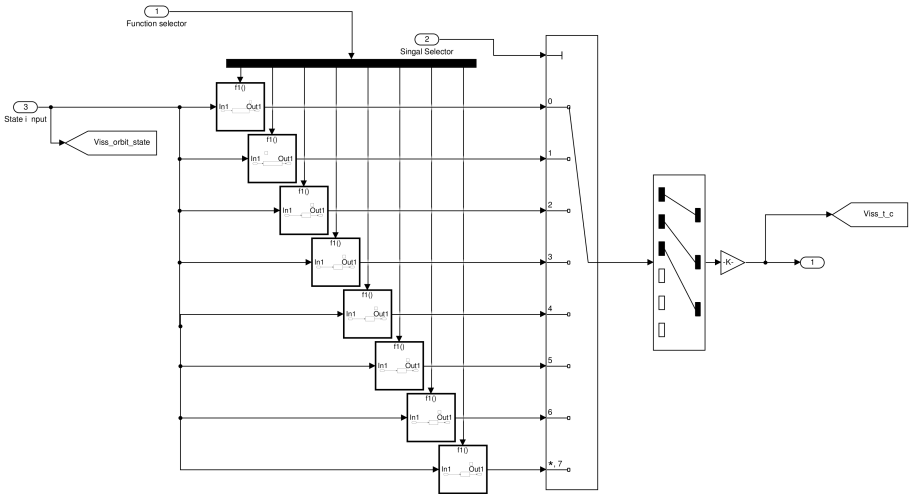


Figure 3.3: Simulink diagram of the control selection. The doted line is the trigger signal that will activate the appropriate sub systems. In each of the sub systems is one of the eight controllers.

3.3.2 Charging Mode eMPC Formulation

For the charging mode controller a tracking cost function similar to the one presented in [1] is used in the MPC formulation. The tracking formulation is used because the charging mode wants a specific orientation for the z-axis as well as a specific angular velocity. If there is an angular velocity, there is always a changing quaternion value. This means that a static desired state as used in the camera pointing controller will not work. A tracking cost function can have a changing desired state by changing the reference state. The MPC formulation becomes as shown in Equation 3.9.

$$\min_{\mathbf{U}}[J(\mathbf{U}, \mathbf{y}(t)), \mathbf{r}(t)] \tag{3.9a}$$

$$s.t \quad \mathbf{y}_{min} \leq \mathbf{y}_{t+k|t} \leq \mathbf{y}_{max},\ k = 1, ..., N \tag{3.9b}$$

$$\mathbf{x}_{min} \leq \mathbf{x}_{t+k|t} \leq \mathbf{x}_{max},\ k = 1, ..., N \tag{3.9c}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_{t+k} \leq \mathbf{u}_{max},\ k = 0, ..., N-1 \tag{3.9d}$$

$$\mathbf{x}_{t|t} = \mathbf{x}(t) \tag{3.9e}$$

$$\mathbf{x}_{t+k+1|t} = \mathbf{A}\mathbf{x}_{t+k|t} + \mathbf{B}\mathbf{u}_{t+k},\ k \geq 0 \tag{3.9f}$$

$$\mathbf{y}_{t+k|t} = \mathbf{C}\mathbf{x}_{t+k|t},\ k \geq 0 \tag{3.9g}$$

$$J(\mathbf{U}, \mathbf{x}(t)) = \sum_{k=0}^{N-1} (\mathbf{y}_{t+k|t} - \mathbf{r}(t))^T \mathbf{Q} (\mathbf{y}_{t+k|t} - \mathbf{r}(t)) + \mathbf{u}_{t+k}^T \mathbf{R} \mathbf{u}_{t+k} \tag{3.9h}$$

Where $\mathbf{r}(t)$ is the time varying reference signal. The dynamic and kinematic equations that are used for the charging mode controller are also slightly different. The kinematic equation is the same as the one presented in Equation 2.1, while the dynamic equation is the same as the one in Equation 2.2 but without disturbance forces. This leads to the state equations as shown in Equation 3.10.

$$\dot{\mathbf{q}}_b^i = \mathbf{T}(\mathbf{q}_b^i)\omega_{b/i}^b \tag{3.10a}$$

$$\dot{\boldsymbol{\omega}}_{b/i}^b = \mathbf{I}^{-1}(\mathbf{S}(\mathbf{I}\boldsymbol{\omega}_{b/i}^b)\boldsymbol{\omega}_{b/i}^b + \mathbf{S}(\mathbf{m^b})\mathbf{B}^b) \tag{3.10b}$$

This time the linerization is around the point $\mathbf{q}_b^i = [\ 1\ 0\ 0\ 0\ ]^T$ and $\boldsymbol{\omega}_{b/i}^b = [\ 0\ 0\ \omega_{z\_0}\ ]^T$ where ω_{z_0} is the desired angular velocity around the z-axis to optimize charging. The linerized equation is shown in Equation 3.11.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -\frac{\omega_{z0}}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{\omega_{z0}}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & -\frac{\omega_{z0}}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ -\frac{\omega_{z0}}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & I_x\omega_{z0} & 0 \\ 0 & 0 & 0 & 0 & I_y\omega_{z0} & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.11a}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{B_z^b}{i_{xx}} & -\frac{B_y^b}{i_{xx}} \\ -\frac{B_z^b}{i_{yy}} & 0 & \frac{B_x^b}{i_{zz}} \\ \frac{B_z^b}{i_{zz}} & \frac{B_x^b}{i_{zz}} & 0 \end{bmatrix} \tag{3.11b}$$

Where $I_x = \frac{i_{zz}-i_{yy}}{i_{xx}}$ $I_y = \frac{i_{xx}-i_{zz}}{i_{yy}}$ $I_x = \frac{i_{yy}-i_{xx}}{i_{zz}}$

As the new cost function uses the output, the $\mathbf{C}$ matrix needs to be defined. There are seven states, but one of the states in the quaternion is redundant so only six states are needed to control the attitude and angular velocity of the satellite. The $\mathbf{C}$ is expressed in Equation 3.12.

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

To calculate the reference value, a function to calculate the quaternion representation of a rotation object with the desired angle θ between its z-axis and the z-axis of the Inertia Frame is needed. This is done by combining two rotations. The first one is a rotation θ around the x-axis to get the correct angle between the z-axes expressed as $\mathbf{q}_1$. The second one is a rotation around the new position of the z-axis with an angular velocity ω_{z_0}. This can be expressed as $\mathbf{q}_2$. This two rotations can then be combined using the quaternion product as shown in Equation 3.13.

$$\mathbf{q}_1 = \begin{bmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2} & 0 & 0 \end{bmatrix}^T \tag{3.13a}$$

$$\mathbf{q}_2 = \begin{bmatrix} \cos\frac{\omega_{z\_0}t}{2} & \sqrt{1}\cos(\frac{\pi}{4}-\theta)\sin\frac{\omega_{z\_0}t}{2} & \sqrt{1}\sin(\frac{\pi}{4}-\theta)\sin\frac{\omega_{z\_0}t}{2} & 0 \end{bmatrix}^T \tag{3.13b}$$

$$\mathbf{q}_2 \otimes \mathbf{q}_1 = \begin{bmatrix} \cos\frac{\theta}{2}\cos\frac{\omega_{z\_0}t}{2} + \sin\frac{\theta}{2}\cos(\frac{\pi}{4}-\theta)\sin\frac{\omega_{z\_0}}{2} \\ \cos\frac{\omega_{z\_0}t}{2}\sin\frac{\theta}{2} + \sqrt{1}\cos\frac{\theta}{2}\cos(\frac{\pi}{4}-\theta)\sin\frac{\omega_{z\_0}}{2} \\ \sqrt{1}\sin(\frac{\pi}{4}-\theta)\sin\frac{\omega_{z\_0}t}{2} \\ \sqrt{1}\sin\frac{\theta}{2}\sin(\frac{\pi}{4}-\theta)\sin\frac{\omega_{z\_0}t}{2} \end{bmatrix} \tag{3.13c}$$

Where t is time. Taking the last three elements of the combined quaternion and the desired angular velocity of $\boldsymbol{\omega}_d = [\ 0\ 0\ \omega_{z\_0}\ ]^T$ gives $\mathbf{r}(t) = [\ \mathbf{q}_2 \otimes \mathbf{q}_1(2:4)^T\ \boldsymbol{\omega}_d^T\ ]^T$.

The controller is in similar manner to what was done in subsection 3.3.2 where eight different controllers are created, one for each combination of the signs of $\mathbf{B}^b$.

Chapter 4

Hardware in the Loop Architecture

4.1 Simulation

The simulation that is used for the Hardware in the Loop is completely based on the simulation environment developed by the ADCS team for the CubeSat MOVE-II at Technische Universität München. As most of the simulation was developed by the MOVE-II team, only a short introduction to the overall simulations will be given. For a more in depth view, the reader is referred to [14] and [18]. There are some changes that needed to be made to the simulation, as it has a few parts that are specific to their satellite. The changes are discussed in the following sub sections.

The simulation consists of mainly five different subsystems:

- **Plant**: Contains all the dynamics and kinematics of the satellite as described in section 2.1. In addition it contains all the environment models that are needed, like the orbit propagator, model of the Earth's magnetic field and Sun position model. The orbit propagator is the simplified perturbations model SGP4 [27] that is available in matlab. The Earth's magnetic field is modeled using the International Geomagnetic Reference Field (IGRF) [13], also released as an official matlab function. The Sun position model is carried out by implementing the approximate equations presented in Astronomical Almanac [7]
- **Controller**: Contains the implementation of the eMPC as described in section 3.3. It takes in the attitude and angular velocity and outputs the desired magnetic moment. It will later, when the attitude estimator is a bit more mature, take the estimated attitude and angular velocity from the estimator.
- **Sensors**: Contain models of all the sensors on the satellite. The models take in the necessary environmental and state data and output data on a similar format as it would be produced by the sensor.
- **Actuator**: Has the desired magnetic moment from the controller and the magnetic field as inputs and it outputs the resulting torque.
- **Disturbances**: Take in the attitude information and environmental information to calculate the different disturbance torques in the Body Frame.

4.1.1 Sensor Models

As described in section 3.2, the ADCS has two IMUs and four Sun Sensors. Only the gyroscope- and magnetometer-part of the IMU is used.

Gyroscope The gyroscope is modeled as a system with bias and white noise as shown in [17] . In addition, the bias of the gyroscope is assumed to be a random walk process. The mathematical model of the gyroscope is shown in Equation 4.1

$$\tilde{\boldsymbol{\omega}}^m_{b/i} = \boldsymbol{\omega}^m_{b/i} + \mathbf{b}^m_{gyro} + \boldsymbol{\eta}^m_{gyro\_v} \tag{4.1a}$$

$$\dot{\mathbf{b}}^m_{gyro} = \boldsymbol{\eta}^m_{gyro\_u} \tag{4.1b}$$

Where $\tilde{\boldsymbol{\omega}}^s_{b/i}$ is the measured angular velocity between the Inertia Frame and the Body Frame expressed in the Sensor Frame $\mathcal{F}_m$, $\boldsymbol{\omega}^m_{b/i}$ is the true angular velocity between the Inertia Frame and the Body Frame, $\mathbf{b}^m_{gyro}$ is the bias of the gyroscope and $\boldsymbol{\eta}^m_{gyro\_v}$ and $\boldsymbol{\eta}^m_{gyro\_u}$ are independent zero-mean Gaussian white noise processes. It is assumed that there is no movement between the IMU and the satellite, so that $\boldsymbol{\omega}^b_{m/i} = \boldsymbol{\omega}^b_{b/i}$.

The model is different from the one used in [14] as the gyroscope bias $\mathbf{b}^m_{gyro}$ is not constant, but varies like a random walk process. The model with a none constant bias is needed for testing of the Extended Kalman filter (EKF) which is used on the satellite for attitude estimation, as this filter uses a model where the bias is a random walk.

As shown in Figure 4.1, the gyroscope signal is generated in the simulation by sampling the angular velocity $\boldsymbol{\omega}^b_{b/i}$. The angular velocity is calculated by the plant using the zero order hold block in Simulink before it is rotated into the Sensor Frame, where the bias and noise are added. After that, the signal is discretized using the Quantizer block to get the same discretization error as the sensor signal.

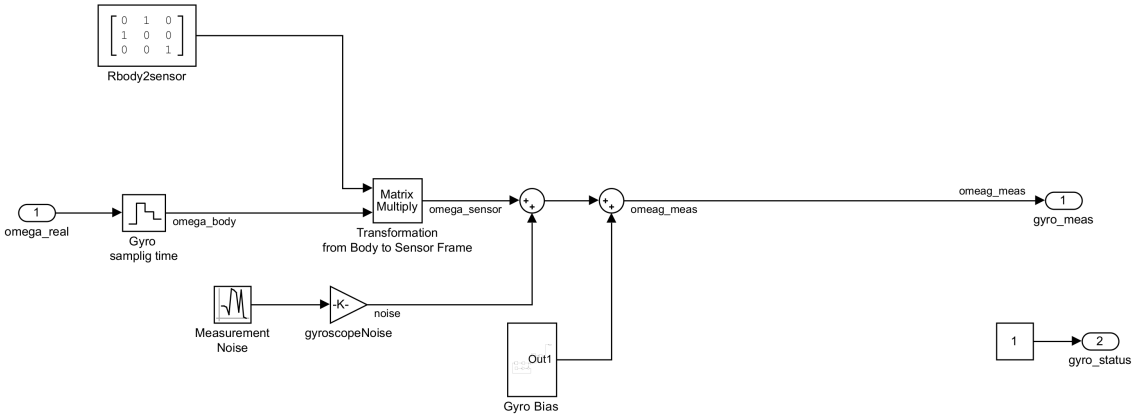
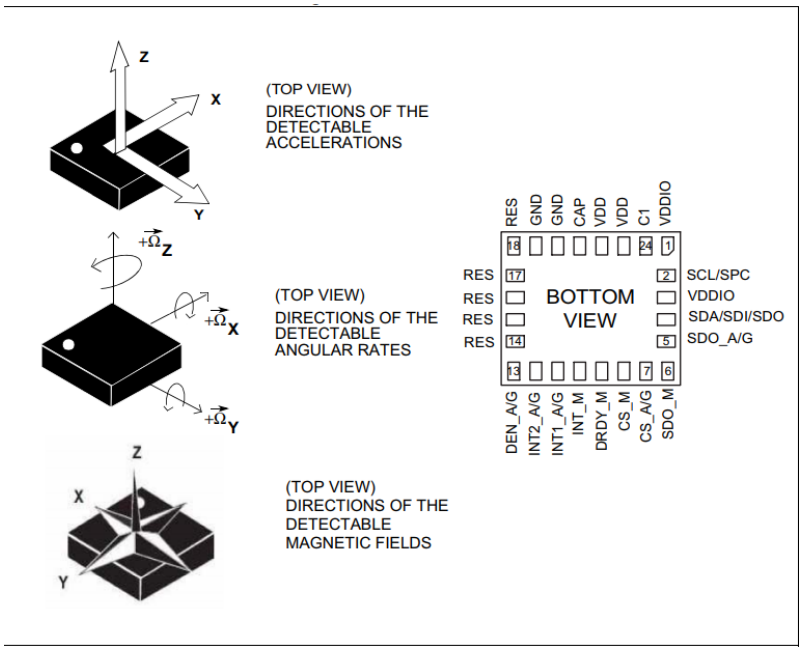
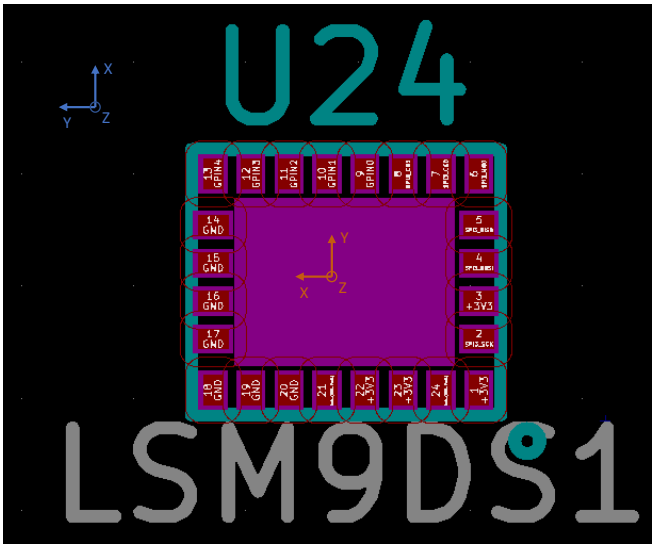


Figure 4.1: The Simulink module used to generate the gyroscope sensor values.

The transformation between the Sensor Frame and Body Frame needs to be calculated. Looking at the definition of the Sensor Frame given in the data sheet and the orientation of the IMU on the ADCS-PCB shown in Figure 4.2, one can see that the transformation can be achieved by changing the y and x-axis as expressed in Equation 4.2. Both IMUs have the same orientation, so only one transformation matrix is needed.



(a) Sensor frame for the IMU taken from the data sheet [12]



(b) Orientation of the IMU relative to the Satellite Body Frame. The blue frame shows the Body Frame and the orange frame shows the Sensor Frame.

Figure 4.2: Relationship between the Sensor Frame and Body Frame.

$$\mathbf{T}_b^m = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

There is no information in the IMU data sheet [12] about the noise characteristics. It is also considered out of scope for this thesis to do empirical investigations into the noise characteristics of the sensors. Therefore an initial estimate to get right magnitude is based on literature found about similar sensors. Three sources are used: The first source is [23], which is a technical article posted by STMicroelectronics (ST). The article contains characteristic data about their gyroscopes, but does not specify what gyroscopes they are referring to. The second source used is [3], which conducted empirical investigation on IG-500N [10]. The third source is the data sheet for the BMX055 IMU [2], used by the MOVE-II team. They all report noise density in the 0.010 - 0.025 $\frac{\text{deg/s}}{\sqrt{\text{Hz}}}$ range as shown in Table 4.1.

Table 4.1: Noise density of different gyroscopes

Source:	**Noise Density:** $[\frac{\text{deg/s}}{\sqrt{\text{Hz}}}]$
ST	0.023
IG-500N	0.012
BMX055	0.014

Assuming that the chosen samples are representative for the gyroscope on the ADCS, the noise density is chosen to be 0.020 $\frac{\text{deg/s}}{\sqrt{\text{Hz}}}$. The ADCS gyroscope will operate at $59.5\,\text{Hz}$, which leads to a standard deviation $\sigma_{gyro\_v} = 0.1543\,\text{deg/s}$.

The values for the random walk are based on two sources: The [3] as before and . They both did empirical studies of their respective gyroscope and used Allan variance to try and determine the standard deviation for the random walk process. In [3] the noise density for the random walk has a magnitude of 1×10^{-4} $\frac{\text{deg/s}}{\sqrt{\text{Hz}}}$, while ref messman master thesis has a lot more uncertainty in its analysis and reports anything from 1×10^{-2} to 1×10^{-5} $\frac{\text{deg/s}}{\sqrt{\text{Hz}}}$. Based on this, it seams reasonable to have a random walk with magnitude of 1×10^{-4} $\frac{\text{deg/s}}{\sqrt{\text{Hz}}}$.

Magnetometer The magnetometer is modeled as in [17], [6] and [14] as a system with bias and white noise. The bias is assumed to be constant for the magnetometer. The mathematical model can be seen in Equation 4.3.

$$\tilde{\mathbf{B}}^m = \mathbf{R}_i^m(\mathbf{q}_i^m)\mathbf{B}^i + \mathbf{b}_{mag}^m + \boldsymbol{\eta}_{mag}^m \tag{4.3}$$

Where $\tilde{\mathbf{B}}^m$ is the measured magnetic field in the Sensor Frame $\mathcal{F}_m$, $\mathbf{R}_i^m(\mathbf{q}_i^m)$ is the rotation matrix from the ECI Frame $\mathcal{F}_i$ to the Sensor Frame, $\mathbf{B}^i$ is the true magnetic field, $\mathbf{b}_{mag}^m$ is the bias and $\boldsymbol{\eta}_{mag}^m$ is a zero-mean white noise.

As seen in Figure 4.3, the input to the model is the magnetic field in ECI Frame calculated by the IGRF model, which is part of the plant and the attitude, which is the rotation matrix from ECI to Body Frame. The magnetic field is sampled and transformed into the Sensor Frame in two steps. First it is transformed into the Body Frame using the attitude matrix, then it is transformed into the Sensor Frame using a static transformation matrix, which is dependent on the orientation between the Body Frame and the Sensor Frame.

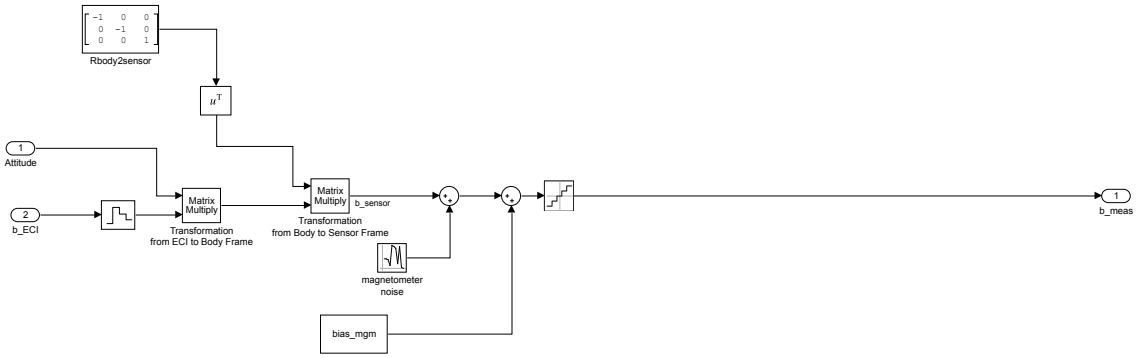


Figure 4.3: The Simulink model used to generate the magnetometer sensor values.

As it can be seen in Figure 4.2a, the magnetometer and gyroscope have different Sensor Frames so a new transformation matrix is needed for the magnetometer. It can be seen from Figure 4.4, that the rotation from Body Frame to Sensor Frame can be done by rotating 180 degrees around the z-axis. This gives rise to the transformation matrix in Equation 4.4.

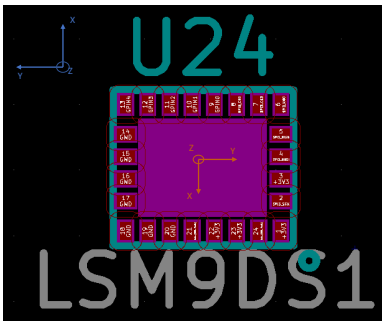


Figure 4.4: Orientation of the IMU relative to the Satellite Body Frame. The blue frame shows the Body Frame and the orange frame shows the magnetometer Sensor Frame.

$$\mathbf{T}_m^b = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

Sun Sensor The Sun Sensor model shown in Figure 4.5 works as in [14]. The input to the Sun Sensor model is the Sun vector in Body Frame. The Sun vector is created by the Sun position model in the plant. The vector is then transformed into the Sensor Frame. Noise and bias are added to the measurement afterwards. The noise and bias in the sensor model should behave as similar to the actual sensor noise and bias as possible. To archive that, the Cartesian representation of the vector is transformed into a spherical coordinate system. Where noise and bias are added to the angles θ and ϕ. This resembles more how the inaccuracies will be on the sensor as it actually calculates the angels θ and ϕ [19]. The

signal is then transformed back to a Cartesian representation and normalized as this is the form the measurements will be given from the sensor.

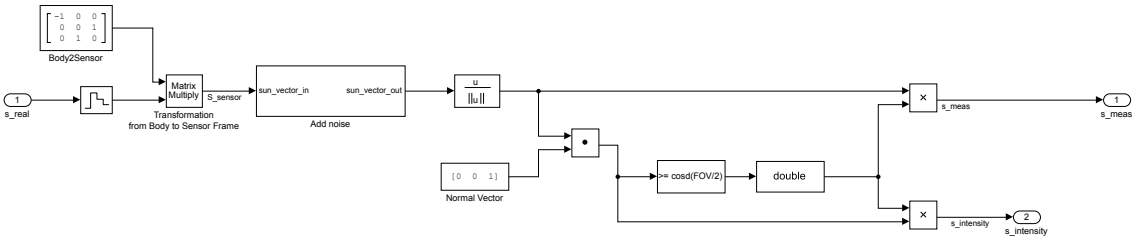


Figure 4.5: The Simulink model used to generate the Sun Sensor values for the sensor on the y- side panel.

To determine if the Sun is in the field of view of the Sun Sensor, a cross product between the Sun vector and the normal to the x-y plane in the Sensor Frame is compared to $\cos FOV$, where FOV is the field of view of the sensor. This cross product is also used to determine how directly a given sensor is facing the Sun. This is later used to select which Sun Sensor to use as only the one facing the most directly towards the Sun will be used. This is done for two reasons: First, according to the data sheet, the Sun Sensor is more accurate if the Sun is within $45\,\mathrm{deg}$ field of view [19]. The second reason is to mitigate the effect of the albedo of the Earth, as the sensor facing directly towards the Sun faces the least towards the Earth.

The Sensor Frame for the Sun Sensors is not stated in the data sheet. So it is simply assumed that the z-axis is normal to the surface that the Sun Sensor is mounted on. The y-axis is always pointing in the negative z direction of the Body Frame as shown in Figure 4.6, except for the Sun Sensor on the Top-panel where the Sensor Frame is equal to the Body Frame.

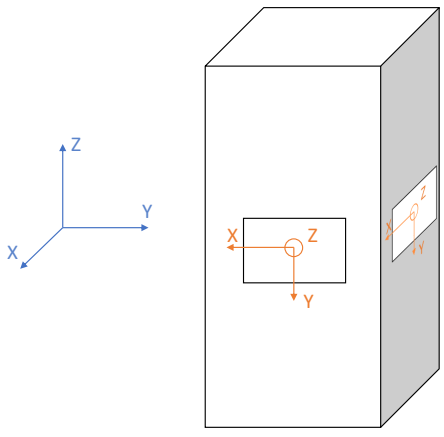


Figure 4.6: Illustration of the different Sun Sensor Frames and the Body Frame. The Body Frame is blue and the different Sun Sensor Frames are orange.

This leads to the transformation matrices shown in Equation 4.5. Where $\mathbf{T}_{x+}$ denotes the transformation for the sensor on the $x+$ side of the satellite, $\mathbf{T}_{x-}$ for the sensor on the $x-$ of the satellite and so forth.

$$\mathbf{T}_{x+} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \qquad \mathbf{T}_{x-} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{4.5a}$$

$$\mathbf{T}_{y+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \mathbf{T}_{y-} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.5b}$$

$$\mathbf{T}_{z+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5c}$$

4.1.2 Actuator Model

For the most part, the actuator will be treated as a black box, as a complete magnetorque system is bought. The system also has its own micro-controller, which controls the coils and compensates for their dynamics, their dependency on temperature and so forth [11]. Therefore the mangetorquer model takes the cross product between the desired magnetic moment and the magnetic field to produce the control torque. A small error is added to the magnetic moment before the torque is calculated to represent the inaccuracies in the magnetorquer. There is no information about what kind of error this is or how it affects the produced magnetic moment. The desired magnetic moment from the controller is also discretized.

4.1.3 Disturbance models

The disturbance models are not changed from what is described in [14] and [18], only the parameters are changed as the satellite is not the same. They are all based on the formulas presented in subsection 2.1.4.

Gravity Gradient Torque For the gravity gradient torque a way of changing the inertia matrix during the simulation was needed, as the satellite has two different inertia matrices dependent on whether the boom is out or not. When the boom is out, the differences between the elements of the inertia matrix and the gravity gradient torque increases. This functionality was realized by a switch block in Simulink, which switches the inertia matrix based on the control signal "Ctrl_Arm_out" as shown in Figure 4.7.

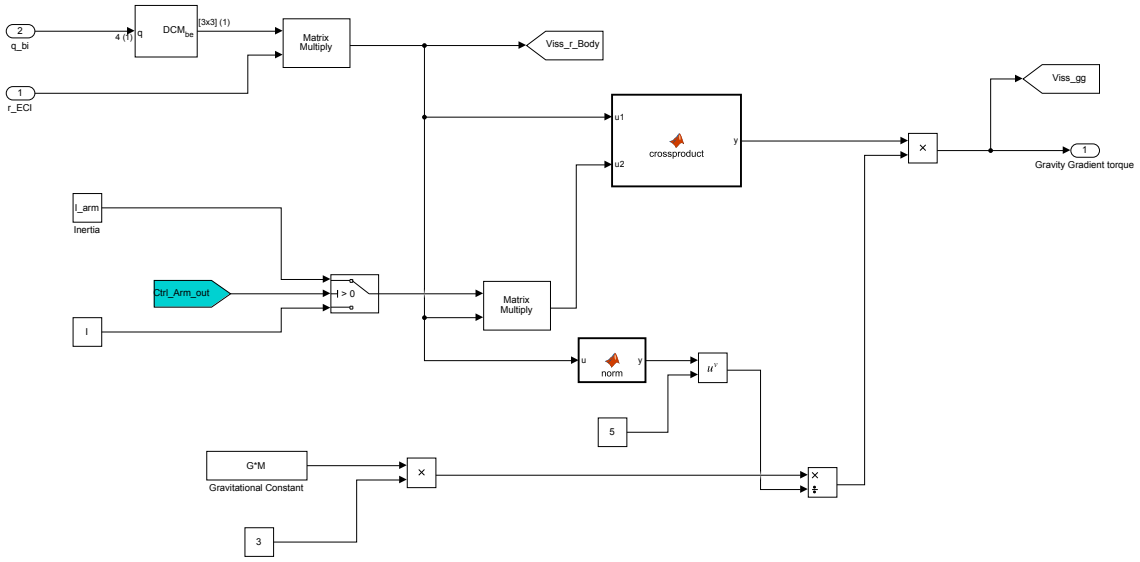


Figure 4.7: Simulink implementation of gravity gradient torque with added functionality to switch between inertia matrix dependent on whether the boom is out or not.

In this model it is assumed that the boom is instantaneously deployed and in its final position. No effort has been made to create a model of what happens while the boom is being deployed. This should not be needed, as no advanced maneuvering is taking place while the boom is being deployed. Any changes to the angular velocity caused by the deployment will be compensated by the detumbling controller once the boom has settled.

Parasitic Dipole-Moment The torque generated by the parasitic dipole-moment is modeled as described in subsection 2.1.4. The only parameter needed is an estimate for the parasitic dipole-moment. The parasitic dipole-moment is unknown, as not all the parts for the satellite are selected and the satellite is not assembled yet, so no measurements can be done. Therefore an estimate based on values found in the literature is used to get an initial estimate. Values from four sources were used. The first source[14] has an initial estimate of the MOVE-II satellites dipole-moment. The second source is the dipole-moment measured by the MOVE-II team and has been taken from internal documents that have not been published. The third source [4] is the estimated dipole-moment of the UWE-3 satellite based on data gathered when the satellite was in orbit. The fourth source is the estimate in [22] for the dipole-moment of the Alto-2 satellite gotten by measurements on ground. The different values can be seen in Table 4.2

Table 4.2: Parasitic Dipol-Moment reported by different sources

Source:	**Magnitude:** $[\mathrm{Am}^2]$	**Vector:** $[\mathrm{Am}^2]$
MOVE-II estimate	0.02	$[\,0.0115 \;\; 0.0115 \;\; 0.0115\,]$
MOVE-II measured	0.0066	$[\,0.00595 \;\; 0.00220 \;\; 0.00179\,]$
UWE-3 estimate	0.045	$[\,-0.001 \;\; 0.0012 \;\; 0.045\,]$
Alto-2 estimate	0.058	$[\,0.0145 \;\; 0.0219 \;\; 0.0523\,]$

As the Selfie-Sat is a 2U, while both the UWE-3 and MOVE-II are 1U and the Alto-2 is a 3U, it is reasonable to assume that the magnitude of the parasitic dipole-moment is in

between the Alto-2 and the 1U satellites. So the parasitic dipole-moment is estimated to $|\mathbf{m}_p| = 0.05\,\mathrm{Am}^2$. To estimate the direction of the dipole-moment, one can observe that for both UWE-3 and Alto-2 satellites, the z-axis is clearly the most dominant axis. This also makes sense if we imagine a simplified case, where all the parasitic dipole-moments are caused by current loops. On each PCB in the satellite, there is one current loop. All the current loops have the same area and current. Then there would be two loops creating a moment in the x and y-direction and five loops creating a moment in the z-direction. Therefore $\mathbf{m}_p = [\ 0.0151\ \ 0.0151\ \ 0.0452\ ]'\ \mathrm{Am}^2$.

Aerodynamic Drag For the aerodynamic drag torque an estimate of all the area and position of pressure for all the surfaces of the satellite is needed. For the Selfie-Sat this is fairly straightforward, as all the surfaces of the satellite are squares with either $A_1 = 0.01\,\mathrm{m}^2$ or $A_2 = 0.02\,\mathrm{m}^2$. The center of pressure of a square is in its center [5]. This means that the center of pressure for each panel is located on one of the axes with a distance of $0.05\,\mathrm{m}$ for panels with area A_1 or $0.1\,\mathrm{m}$ for panels with area A_2. The remaining parameters are the drag coefficient C_D and the atmospheric density ρ. The drag coefficient is set to a worst case scenario of $C_D = 2.5$ [5].

4.2 Simulation and Hardware Interface

The simulation hardware interface consists of mainly three parts. Where the first two are modified versions of the one used in the HiL setup developed by [14] and the last is developed as part of this thesis. The first part of the interface is a server written in Python that relays messages between the simulation and the beaglebone black (BBB) which is the second part. The BBB is used as a gateway connecting all the hardware components to the internet. The BBB forwards the data received from the server to a STM32-L0R8T6 micro-controller which is mounted on a STM32L053 Nucleo-64 board [26]. The STM micro-controller replaces all sensors and actuators on the ADCS-PCB and communicates with the ADCS-MCU on the same interfaces as the MCU communicates with the sensors and actuators. A full picture of the end to end communication between the simulation and the ADCS-MCU can be seen in Figure 4.8.

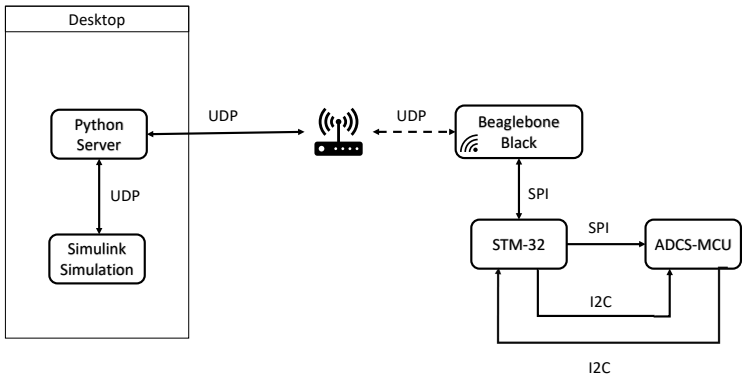


Figure 4.8: Illustrates the data flow between all the components in the HiL setup. Each arrow indicates the direction of data flow and the text what kind of protocol is used.

The Python server does nothing other than forwarding the message between the simulation and the BBB and uses a standard python WebSocket library. It will therefore not be discussed any future in this thesis. In the simulation the interface towards the Python server consists of four blocks. One block that transforms the data to be sent into binary data of the appropriate form, an UDP transmit block, an UPD receive block and lastly a block that decodes the binary data into data types that Simulink can work with.

Beaglebone Black Beaglebone Black is a small single-board computer capable of running various Linux distributions which is shipped with Debian. The BBB also has wireless capabilities. The BBB uses an open source WebSocket library called "WebSocket++" for the networking and a SPI driver developed by the MOVE-II team for the SPI communication. For the communication between the BBB and the STM-MCU, a simple communication protocol was created to specify what data the BBB is transferring or what data the BBB is requesting. The BBB has to be the SPI master, as there is no SPI slave support on any standard Linux distribution. This is one of the main reasons why the STM-MCU is needed, as both the BBB and the ADCS-MCU have to be SPI masters, so a node, that can be SPI slave to both, is needed in between.

The BBB is programmed to listen to any UDP message it might get from the Python server. Once the BBB receives a new message, it decodes the data into the appropriate sensor data and forwards it to the STM-MCU. Once all data has been transferred to the STM-MCU, it requests new magnetometer data from the SMT-MCU, which it then sends back to the Python Server. An illustration of the program flow of the BBB can be seen in Figure 4.9.

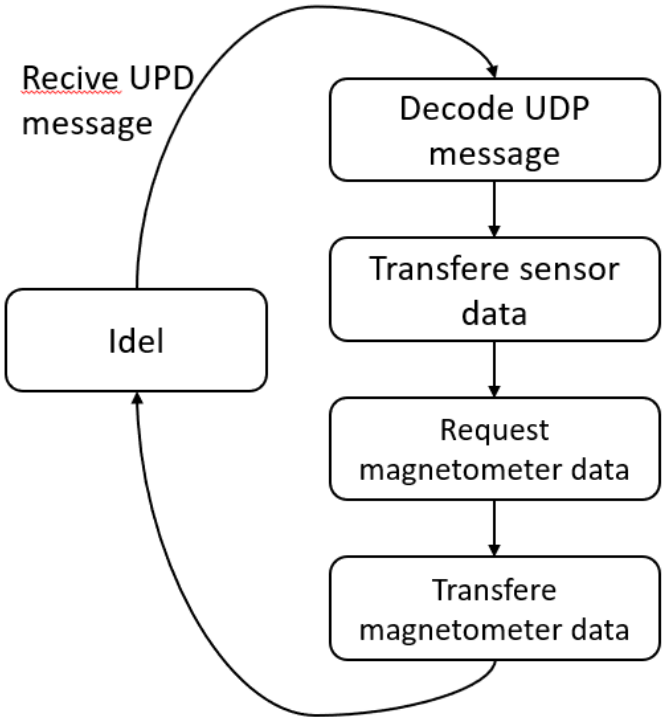


Figure 4.9: Shows the sequence of commands that the Beaglebone Black executes each time it gets a new UPD message.

STM32-L0R8T6 The STM32-L0R8T6 micro-controller is micro-controller developed by STMicroelectronics and it has a 32-bit ATM Cortex M0 core. For the development on the micro-controller the Hardware Abstraction layer (HAL) library is used for interfacing with the lower level hardware components on the micro-controller. To set up all the configuration of the pins and low level drivers, the STM32Cubemx is used to auto-generate the initialization functions. The micro-controller uses four different interfaces to talk to the BBB and the ADCS-MCU. There is one SPI interface to the BBB, one SPI and two I2C interfaces to the ADCS-MC. The SPI interface is used to replicate data that should have come from the IMUs. One of the I2C interfaces is used to replicate data that should come from the Sun Sensor and the last I2C interface is used to receive data that should have been sent to the magnetorquer board.

On each of the interfaces the STM-MCU is a slave and for each interface there is a state machine. All the state machines work in the same way: They have an idle mode where they are waiting for a command from the master. Based on the command it goes in the appropriate state preparing either to receive or transmit the appropriate data. So each transaction between the STM-MCU and one of the master is initiated by a command message telling the STM-MCU what type of transaction is about to start followed by the master initiating the transaction of the actual data.

ADCS-MCU The idea behind a HiL test setup is to use as much of the hardware and software that is going on the final product in testing. With that in mind when designing the HiL specific parts of the ADCS, no changes are done to the hardware. Everything is running on the latest iteration of the PCB design and all the interfaces replacing the sensors and actuator are the same as the ones used by the replaced sensors and actuators. The underlying low level drivers that are used for this interfaces are also used. What is different is the communication protocol used. This means that the sensor specific driver is not used but replaced by HiL specific drivers. To connect the ADCS-MCU and the STM-MCU the exposed headers on the PCB is used.

Chapter 5

Results

There are no results for the HiL tests as the system was not working well enough to give any meaningful data. Instead the results will be from normal simulation.

5.1 Camera Pointing eMPC Results

The parameters of the best performing controller for camera pointing, which is the one used to get the results in this section are listed in Table 5.1. All the tests are run without changing magnetic field but rather using a static one equal to the one used when creating the controller of $\mathbf{B}^b = [\, 5{\times}10^{-5}\; 5{\times}10^{-5}\; 5{\times}10^{-5}\,]^T$. This is mainly because the difference between having a realistic model and using a static magnetic field were negligible for the control performance, but increases the simulation time tenfold. For all tests the gravity gradient is always on as it was modeled as a part of the system. The other disturbance forces are only part of the simulations when specified.

Table 5.1: Parameters of the camera pointing controller.

Parameter:	**Value:**
Q	diag{100 100 100 100 100 10}
R	diag{10000 10000 10000}
N (Horizon)	2
State constraints	$[\, -1\; -1\; -1\; -1\; -1\; -1\,]^T \leq \mathbf{x} \leq [\, 1\; 1\; 1\; 1\; 1\; 1\,]^T$
Actuator constraints	$[\, -0.2\; -0.2\; -0.2\,]^T \leq \mathbf{u} \leq [\, 0.2\; 0.2\; 0.2\,]^T$
Scaling state	$N_x = \text{diag}\{10\ 10\ 10\ 1000\ 1000\ 1000\}$
Scaling actuator	$N_x = \text{diag}\{0.1\ 0.1\ 0.1\}$

From Figure 5.1 it can be seen that the controller works reasonably well. It is capable of holding all the angles within $20\,\text{deg}$.

If we however look at Figure 5.3 where the time has increased to one whole orbit, it can be seen that the controller has one sudden problem at the end of the orbit. It seams like this has more to with the way the rotation between the Body Frame and Orbit Frame is calculated than with the controller. Before the spike a jump can also be observed in the quaterion representation. This jump is probably caused by some kind of wrapping when

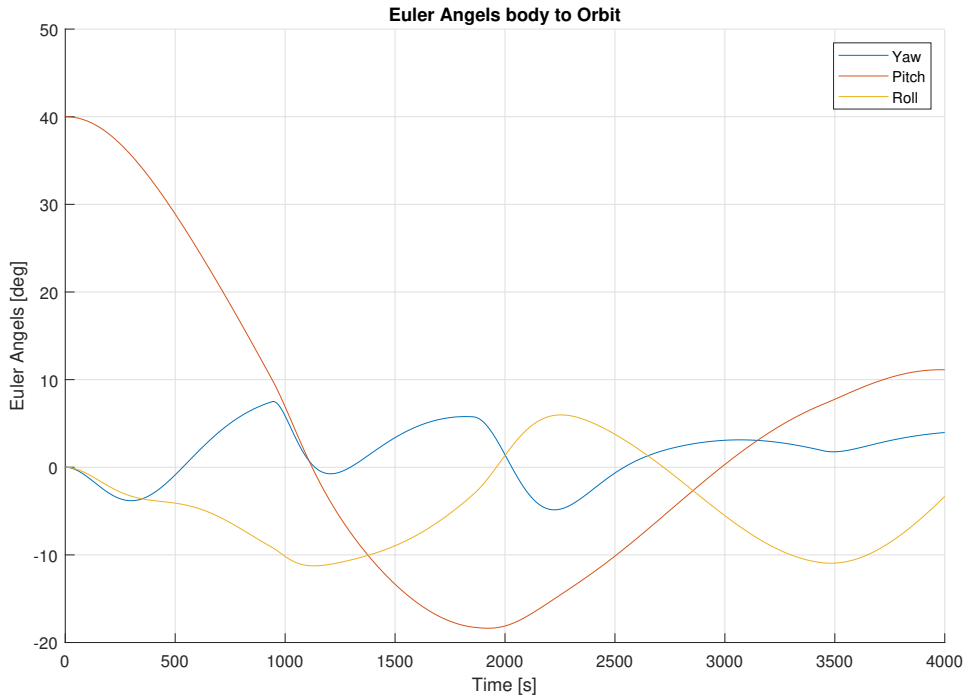


Figure 5.1: Plot of Euler angles between Body Frame and Orbit Frame. Using the camera pointing controller.

the satellite completes one orbit. It can also be seen that the angular velocity is kept very low which is good for picture taking. There is also a minimum of actuation which also is good for the power budget.

It can be seen a spike in the quaternion representation . It has probably less to do with the controller because

If the initial condition changes so that there is an angular velocity, the result becomes much worse. It can be seen that the controller is able to reduce the angular velocity to some reasonable value, but it is not able to control the orientation in any meaningful way. It should also be noted that despite the fact that the orientation is far away from the desired orientation, there is almost no actuation. The controller seems to emphasize reducing the amount of actuation too much.

The controller can also not handle the remaining disturbance forces very well. The plot in Figure 5.4 is completely chaotic and the controller has no control over the satellite. This is not that surprising if we look at the actuated magnetic moment. It is only in the magnitude of 10^{-2} which only results in a torque in the order of 10^{-6} which is the same level as the combined disturbance forces. The actuation just gets drowned in disturbance forces. Once again it seem like the controller is too focused on reducing the actuation to actually reach the desired orientation.

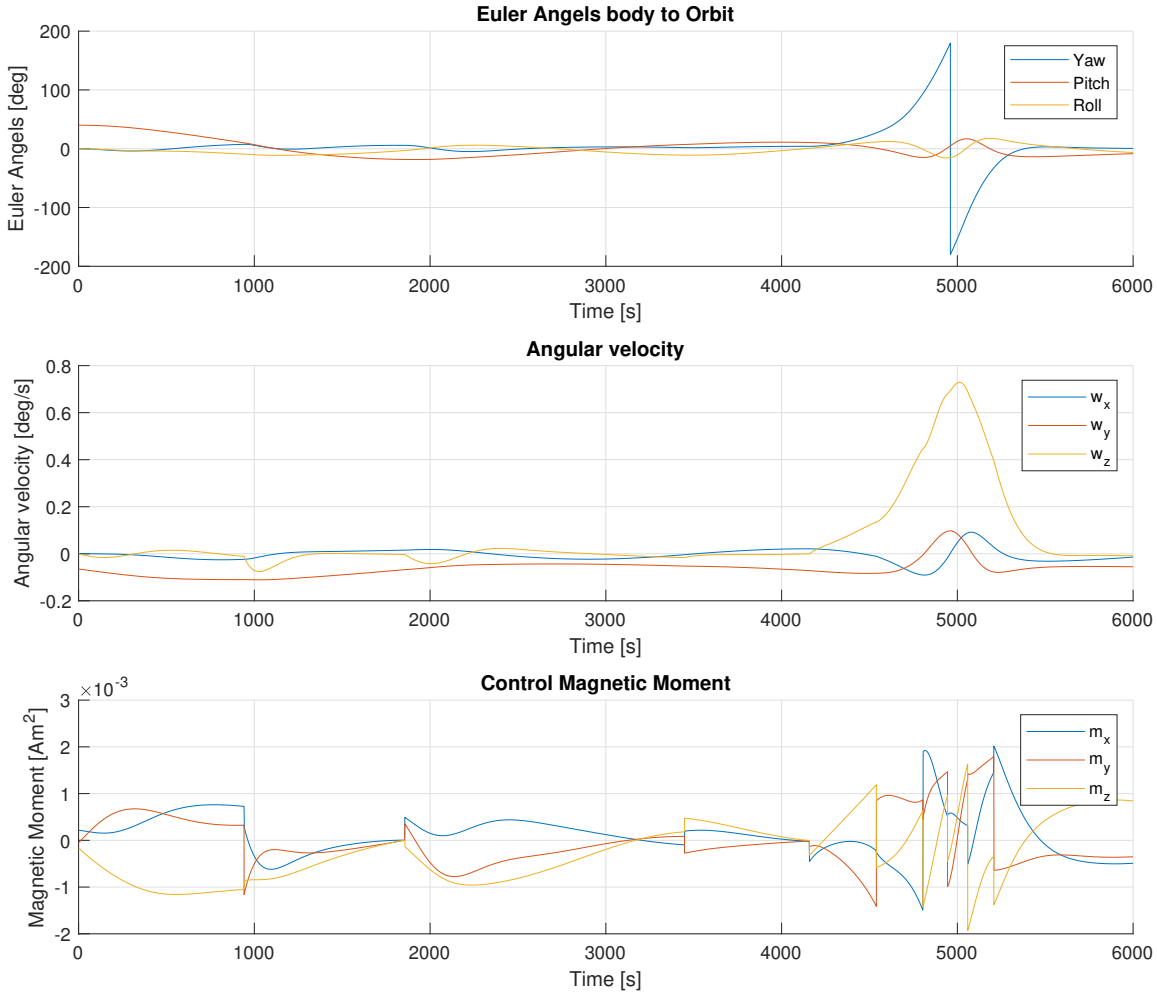


Figure 5.2: Plot of the Euler angles between Body Frame and Orbit Frame, angular velocity and actuated magnetic moment of a complete orbit. There is a strange spice in the Euler angles at the end of the orbit.

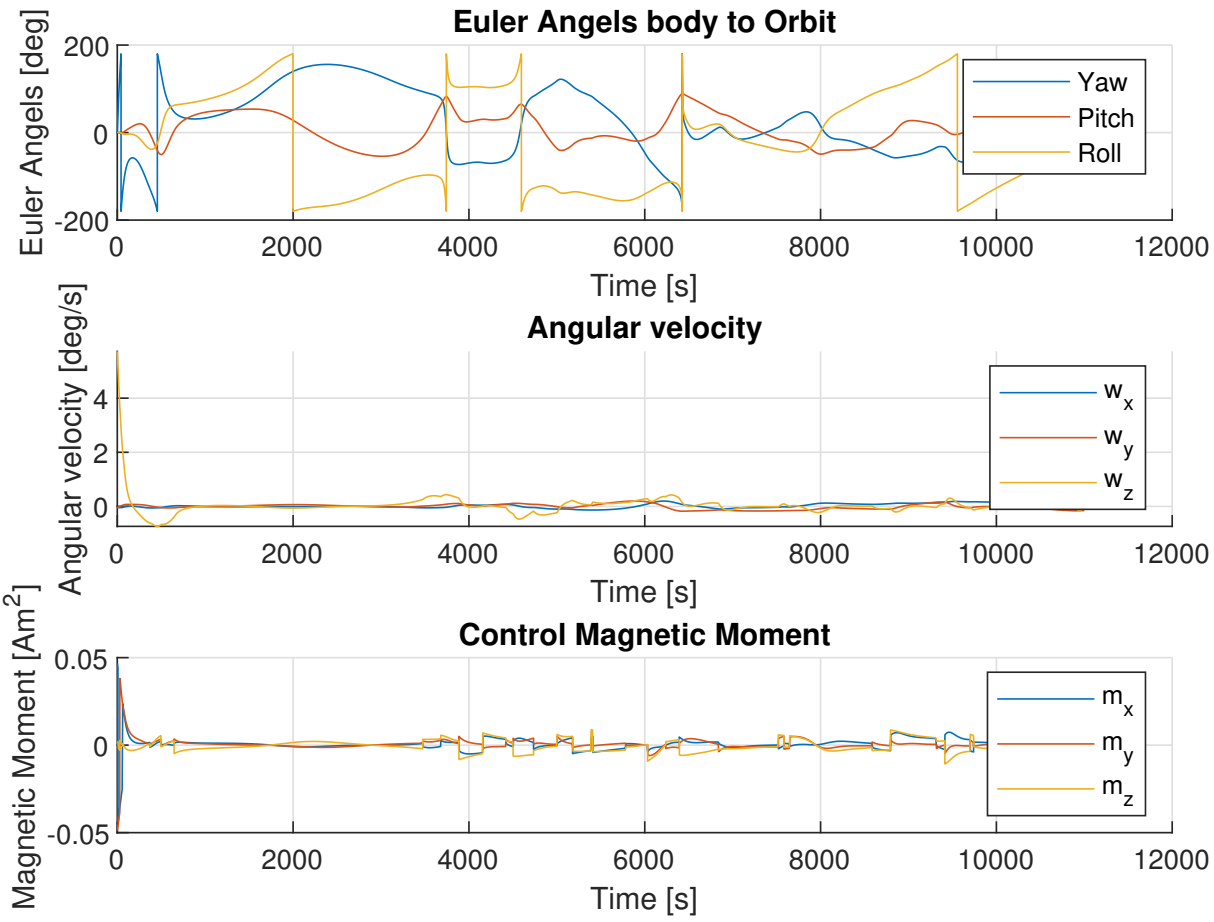


Figure 5.3: Plot of the Euler angles between Body Frame and Orbit Frame, angular velocity and actuated magnetic moment of a complete orbit. There is some initial.

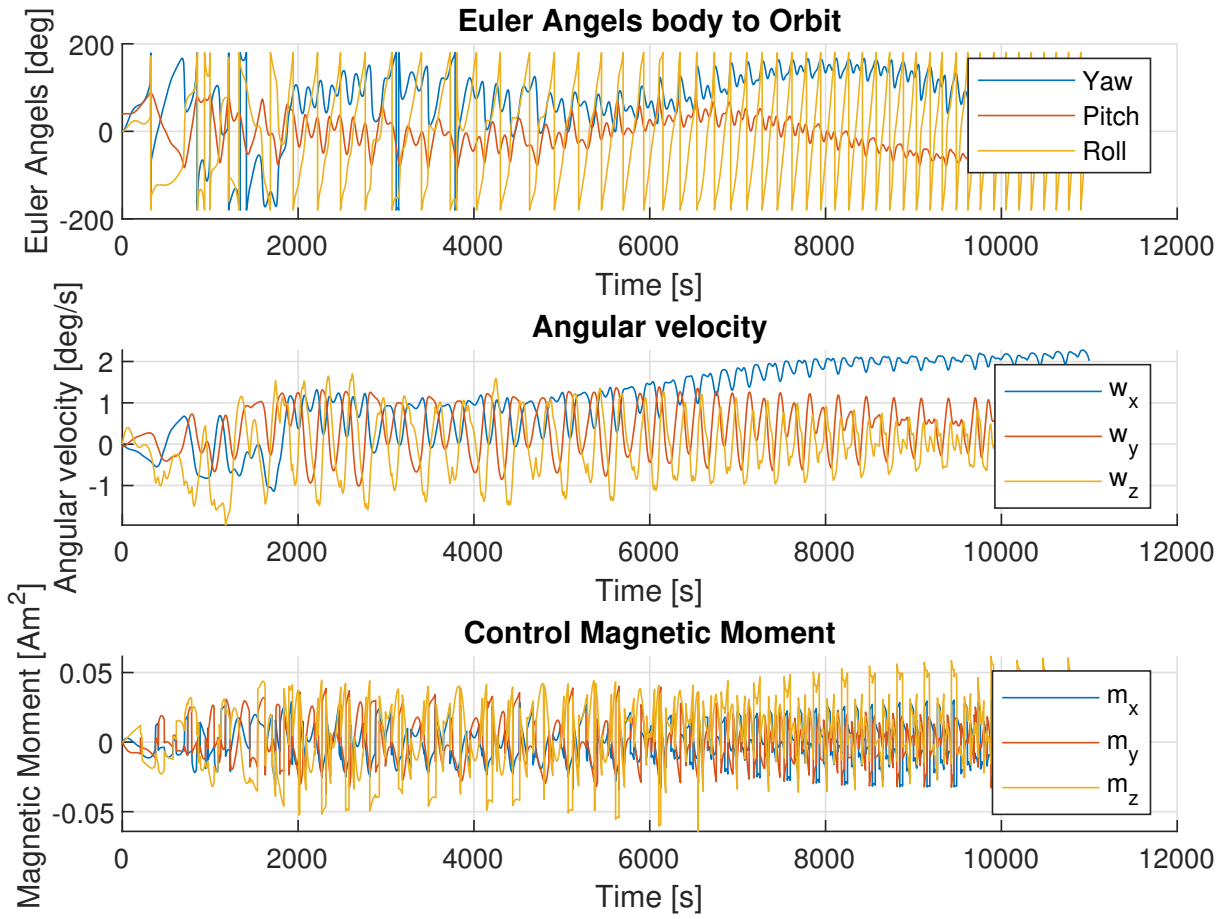


Figure 5.4: Plot of the Euler angles between Body Frame and Orbit Frame, angular velocity and actuated magnetic moment of a complete orbit. The simulation also included disturbance forces.

5.2 Charging Mode eMPC Results

The parameters for the charging mode controller were listed in Table 5.2 for all the results. For the charging mode the magnetic field will also be constant.

Table 5.2: Parameters of the camera pointing controller

Parameter:	**Value:**
Q	$\text{diag}\{10 \times 10^5\ 10 \times 10^5\ 10 \times 10^5\ 100 \times 10^5\ 100 \times 10^5\ 100 \times 10^5\}$
R	$\text{diag}\{100000\ 100000\ 100000\}$
N (Horizon)	2
State constraints	$[\,-1\ -1\ -1\ -1\ -1\ -1\,]^T \leq \mathbf{x} \leq [\,1\,1\,1\,1\,1\,1\,]^T$
Actuator constraints	$[\,-0.2\ -0.2\ -0.2\,]^T \leq \mathbf{u} \leq [\,0.2\,0.2\,0.2\,]^T$
Output constraints	$[\,-1\ -1\ -1\ -1\ -1\ -1\,]^T \leq \mathbf{y} \leq [\,1\,1\,1\,1\,1\,1\,]^T$
Scaling state	$N_x = \text{diag}\{1\,1\,1\,1\,1\,1\}$
Scaling actuator	$N_x = \text{diag}\{1\,1\,1\}$

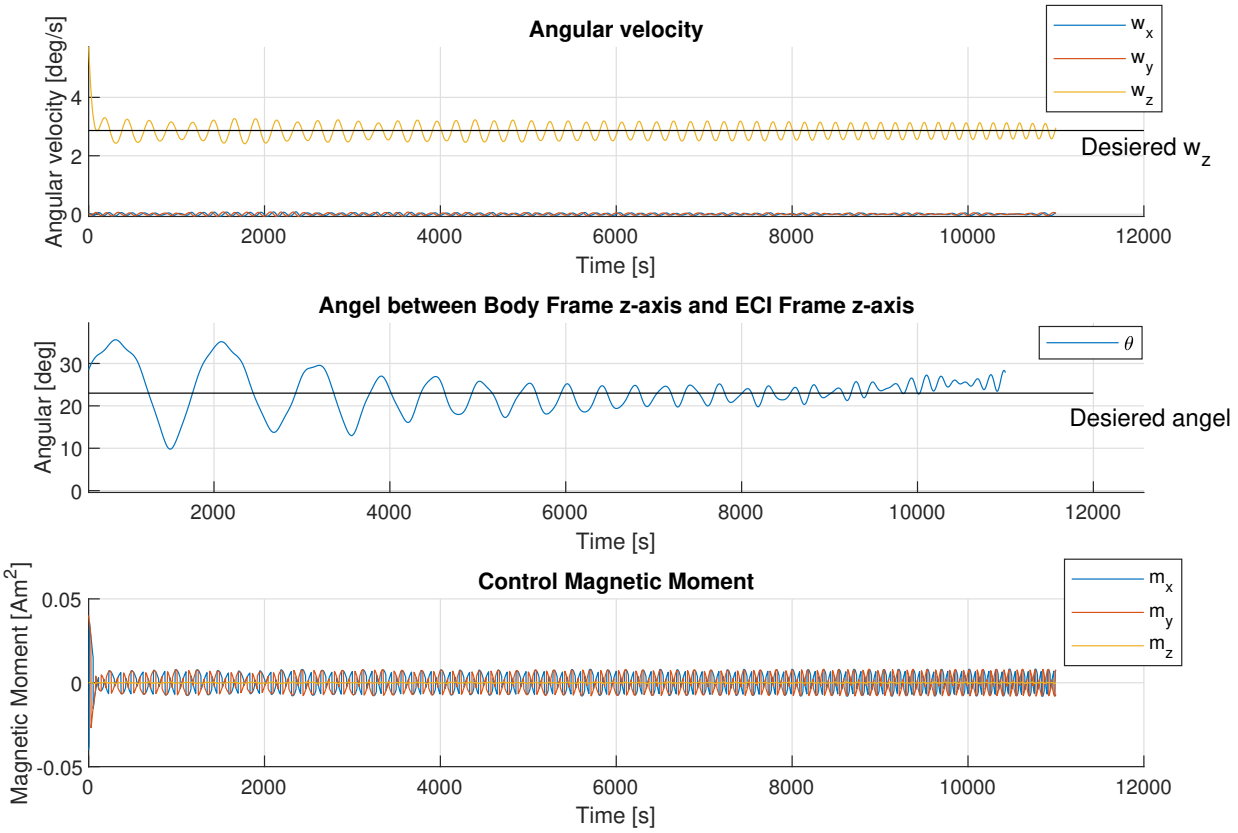


Figure 5.5: Plot of the angular velocity, angle between the Body Frame z-axis, the ECI Frame z-axis and the actuated magnetic moment. The desired angular velocity ω_z and desired angle between the z-axes is marked by a black line.

For the case with only gravity gradient disturbance, the results are looking very good. The controller is able to hold the satellite in the right orientation and with the right angular

velocity while almost not actuating. There are some oscillations around the desired values but that is acceptable.

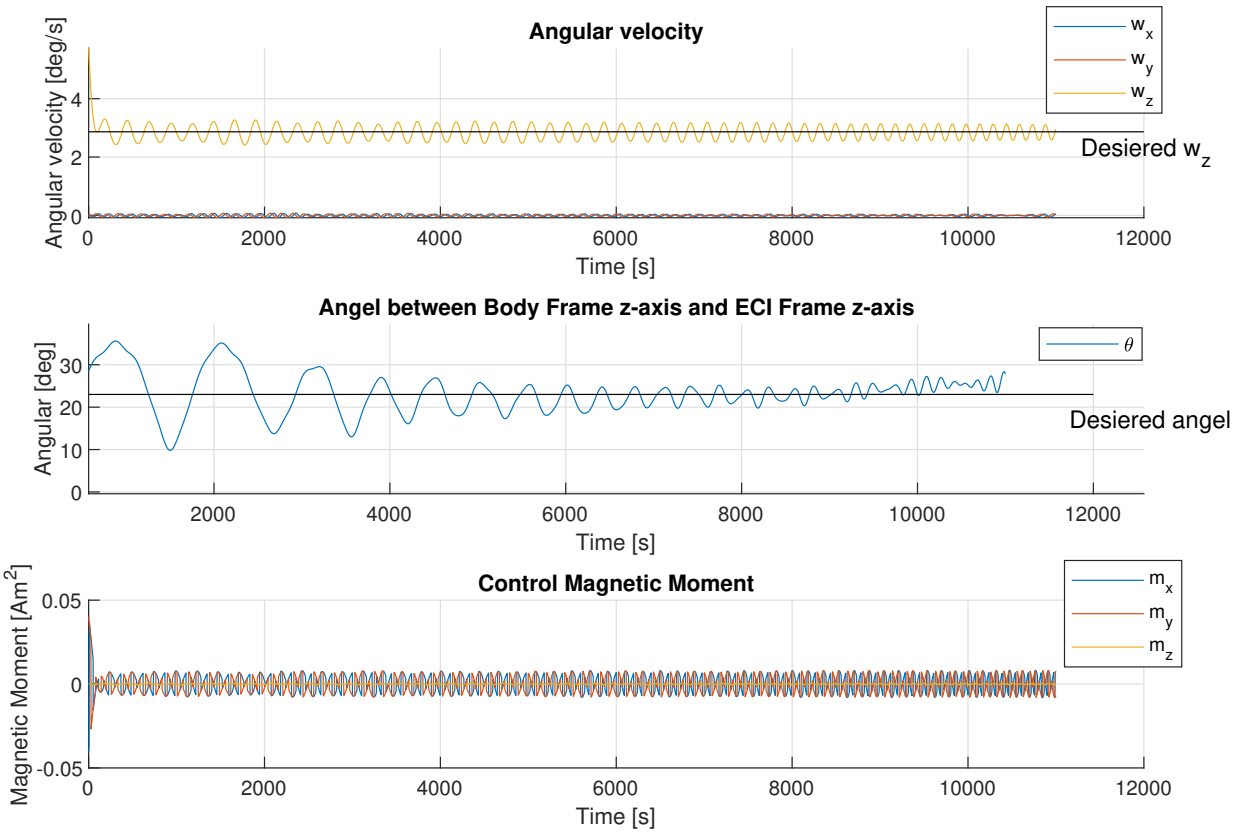


Figure 5.6: Plot of the angular velocity, angle θ between the Body Frame z-axis, the ECI Frame z-axis and the actuated magnetic moment. The desired angular velocity ω_z and desired angle between the z-axes is marked by a black line. The simulation is running with disturbances.

When all disturbances are turned on the controller is having problems again. The motion of the satellite is not as chaotic as it was for the camera pointing controller, but still the system is not stable. The oscillations of the system are much larger than what they were when there was no disturbance. In addition it can be seen that the value for both the angle θ and ω_z is slowly moving away from their desired values. It can be seen that the actuation is still minimal from the controller despite the disturbance foresees changing the satellites orientation far away from the desired orientation. It can seem like the controller is way too concerned with actuating as little as possible and not concerned enough about keeping the satellite in the desired orientation.

Chapter 6

Discussion

Both the controllers and HiL test setup did not come out satisfactory. The controller only worked in very ideal situations and the HiL setup was not able to handle the amount of data that it needed to pass between the ADCS and the simulations. Therefore, the discussion will mostly revolve around what the challenges were with the development process and what can be done to improve the final product.

6.1 Explicit Model Predictive Control Discussion

Numerical issues and approximations There are some numerical issues when using the algorithm to generate the controller. The algorithm often reports that it has encountered inaccuracies and that approximations have been used. This was especially prevalent when generating the controller for the camera pointing mode. Two attempts were made to improve this. The first was regarding the underlying solvers used by the MPT3 toolbox. The default solver for both linear and quadratic programming is the LCP solver. Using the LCP solver often resulted in the algorithm reporting inaccuracies and the need for an approximation, it also sometimes failed to find a solution. When the solver for linear programming were changed to the CDD, no more failures were reported and the need for approximations also seem to have been reduced. So changing to the CDD solver seems to improve the numerical stability even though the underlying reason is not clear.

The second attempt was changing the scaling of the system. In [15] it is mentioned that the the system was scaled to improve numerical stability. To look at the effects of the scaling, Matlabs condition number were investigated for both the $\mathbf{A}$ and $\mathbf{B}$ matrices. The conditioning number gives an indication about the inaccuracies of the matrix inversion. A low condition number means that the matrix is well conditioned for matrix inversion. The unscaled condition number for $\mathbf{A}$ is about 1.2 for both the camera pointing and charging mode controllers. The condition number for $\mathbf{B}$ is unsatisfactory in both cases as it is in the 10^{16} order of magnitude. The $\mathbf{B}$ matrix is also almost skew-symmetric in the lower half which means that it can not become well conditioned with scaling. So if the reason for the numerical issues arises from the bad conditioning of the $\mathbf{B}$ matrix the problem can not be solved with scaling.

Tuning of the Q and R Matrix Tuning the $\mathbf{Q}$ and $\mathbf{R}$ to get good results turned out to be challenging. Looking at the magnetic moment the controller produces when the satellite does not have the desired orientation it can be seen that it is very low. Relative to the maximum magnetic moment it is often hundred or thousand times smaller than the maximum magnetic moment the satellite can produce. From this one might reasonably conclude that the problem is a too big $\mathbf{R}$ matrix which to an actuation, which is too heavily punished in the MPC formulation. The problem is that there seems to be some kind of lower bound for what the elements of $\mathbf{R}$ can be before the number of regions in the solution explodes. If the elements of $\mathbf{R}$ are around 100000, the number of regions in the solution are 100 or 200 which is reasonable, but as soon as the elements of $\mathbf{R}$ become any lower than that, the number of regions in the solution become in the thousands. There also seem to be an upper limit on how high the elements of $\mathbf{Q}$ can be. If the values of $\mathbf{Q}$ get to high the problem becomes infeasible so no solution can be found. This combination of the upper bound on $\mathbf{Q}$ and lower bound on $\mathbf{R}$ means that there is not a lot of room to find good values for $\mathbf{Q}$ and $\mathbf{R}$ that give a satisfying controller.

No Slack variable The addition of slack variables is something that is often added to MPC or eMPC problem to make them a bit more robust and avoid the problem with infeasibility. Several attempts at formulating a problem with slack variables were attempted, but non of them were successful. They all either led to the number of regions going into the thousands or the complete opposite where the solution only contained one region.

6.2 Hardware in the Loop Architecture discussion

The architecture works in the sense that it can send data from the simulation to the ADCS and back again. However it fails in actually being useful for testing the proposed controllers. As soon as the sampling rate of the HiL setup becomes too, low the STM32-MCU starts having problems and is not capable of handling all the data requests coming from the BBB and the ADCS-MCU. It can do it for some time, but eventually there will be an instance where it can not handle all the requests at once. There are several steps that could be taken to mitigate this. The first could be to add propper error handling so that the entire system does not lock up when the STM-MCU gets too many requests and simply live with some data loss. A second option is to try and speed up the code running on the STM32-MCU. The SPI module has some room for improvement as it is currently using blocking mode for SPI transfer which takes up a lot of the CPU time. A better way would be to use the direct memory access (DMA) both for transfer and receive. The third and arguably the simplest solution is to introduce one more slave node. One node for transferring data from the BBB to the ADCS and another one for transferring the data back again. By dividing the load between the two nodes, they should be able to handle all the data requests.

Chapter 7

Conclusion

The eMPC presented in this thesis where proven to work in ideal conditions will being very power efficient. They where able to achieve good enough accuracy without have to produce a lot of magnetic moment. Both controller failed as soon as they where exposed to none ideal conditions by adding disturbances. The camera pointing controller failed when it had initial conditions with some angular velocity. The failures of the controllers when exposed to disturbances seem to be a result of the controllers not producing a high enough magnetic moment. The controllers missing ability to produce sufficient magnetic moment might be traced back to the problems related to solving the eMPC problem and having a need for a very large $\mathbf{R}$ matrix to archive reasonable solutions. If this issues related to solving the eMPC problem can be solved than eMPC controllers might have potential in attitude control of satellite as shown in other sources and by the results in the ideal conditions.

The lack of result form the HiL can mostly be attributed to pore design which was not capable of scaling to the handle the needed data flow. There is nothing inherent to the problem that says it can not be solved by a better design or implementation. That under laying principles seem to be good and the design seems perfectly fixable by some improvements as adding an extra slave node.

Chapter 8

Outlook

There are several open questions that remain regarding the eMPC for attitude of satellites. The natural place to start is future investigate why there seem to be this upper and lower bound to the $\mathbf{Q}$ and $\mathbf{R}$ matrices respectively so that better tuning of the controller can be preformed. There should also be some look into the numerical stability of the problem to understand what causes it and how bad the approximations that sometime are needed affect the quality of the final controller. In regards to numerical stability some effort could also be put into trying to find LP and QP solvers that work beather with the current problem formulation as there was some small indication that the underlying solvers had different performance.

For the HiL setup there are two paths going forward. The first is to get it to actually function properly. This can probably be archived by following one or more of the suggestions in chapter 6. Once the setup works the next natural step is to test the proposed controller in ideal conditions to verify that they still work when they are implemented on the actual hardware.

The second is looking into improving the all the sensor models by adding better estimates of the sensor characteristics which are not only based on literature but on actual data from the sensors. Once Orbit has bought all it sensor it can do extensive testing on them to try and get good estimations of all sensor characteristics.

A minor improvement that can be done to the setup is also to try and make the interface between the ADCS and the STM-MCU closer to the way the interface between the ADCS and its peripherals. This way more of the driver code on the ADCS can be also used when communicating with the HiL environment and even more of the ADCS can be tested using HiL.

Bibliography

[1] Alberto Bemporad et al. “The explicit linear quadratic regulator for constrained systems”. In: *Automatica* 38.1 (Jan. 2002), pp. 3–20. DOI: 10.1016/s0005-1098(01)00174-1.

[2] *BMX055 Small, versatile 9-axis sensor module*. Bosch. Oct. 2013. URL: https://www.bosch-sensortec.com/bst/products/all_products/bmx055.

[3] Thomas Brunner et al. “Magnetometer-Augmented IMU Simulator: In-Depth Elaboration”. In: *Sensors* 15.3 (Mar. 2015), pp. 5293–5310. DOI: 10.3390/s150305293.

[4] S. Busch et al. “UWE-3, in-orbit performance and lessons learned of a modular and flexible satellite bus for future pico-satellite formations”. In: *Acta Astronautica* 117 (Dec. 2015), pp. 73–89. DOI: 10.1016/j.actaastro.2015.08.002.

[5] Yunus A. Cengel and John M. Cimbala. *Fluid Mechanics fundamentals and Applications Third edition in SI Units*. Mc Graw Hill, 2014.

[6] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, Apr. 2011. DOI: 10.1002/9781119994138.

[7] D. Harper. “The Astronomical Almanac”. In: *Astronomy & Geophysics* 39.1 (Feb. 1998), pp. 16–16. DOI: 10.1093/astrog/39.1.1.6.

[8] Øyvind Hegrenæs, Jan Tommy Gravdahl, and Petter Tøndel. “Spacecraft attitude control using explicit model predictive control”. In: *Automatica* 41.12 (Dec. 2005), pp. 2107–2114. DOI: 10.1016/j.automatica.2005.06.015.

[9] M. Herceg et al. “Multi-Parametric Toolbox 3.0”. In: *Proc. of the European Control Conference*. http://control.ee.ethz.ch/~mpt. Zürich, Switzerland, July 2013, pp. 502–510.

[10] *IG-500N GPS aided Orientation Sensor*. URL: http://www.alava-ing.es/repositorio/6f24/pdf/2609/2/unidad-inercial-gps-peq-dim.pdf?d=1.

[11] *iMTQ Inerface Control Document*. ISIS.

[12] *iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometet*. STMicroelectronics. URL: https://www.st.com/resource/en/datasheet/DM00103319.pdf.

[13] *International Geomagnetic Reference Field*. URL: https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html.

[14] Jonis Kiesbye. “Hardware-in-the-Loop Verification of the Distributed, Magnetorquer-Based Attitude Determination & Control System of the CubeSat MOVE-II”. MA thesis. Technische Universität München, 2017.

[15] T. Krogstad, J.T. Gravdahl, and P. Tondel. “Explicit Model Predictive Control of a Satellite with Magnetic Torquers”. In: *Proceedings of the 2005 International Sym-*

posium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005. IEEE. DOI: 10.1109/.2005.1467064.

[16] Thomas R. Krogstad. "Attitude control and stability analysis of satellites in earth and moon orbit". MA thesis. Norwegian University of Science and Technology, 2004.

[17] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Springer New York, 2014. DOI: 10.1007/978-1-4939-0802-8.

[18] David MESSMANN et al. "Magnetic Attitude Control for the MOVE-II Mission". Eng. In: *7th European conferance for aeronautics and aerospace sciences (EUCASS)* (2017). DOI: 10.13009/eucass2017-664.

[19] *NanoSense Fine Sun Sensor Datasheet*. GOMspace. URL: https://gomspace.com/shop/subsystems/attitude-orbit-control-systems/nanosense-fss-(1).aspx.

[20] Óscar R. Polo et al. "End-to-end validation process for the INTA-Nanosat-1B Attitude Control System". In: *Acta Astronautica* 93 (2014), pp. 94, 105.

[21] Meghan Kathleen Quadrion. "Testing the Attitude Determination and Control of A CubeSat with Hardware-in-the-Loop". MA thesis. Massachusetts Institute of Technology, 2014.

[22] Bagus Adiwiluhung Riwanto. "CubeSat Attitude System Calibration and Testing". MA thesis. Alto University School of electrical Engineering, Aug. 2015.

[23] *TA0343 Technical article Everything about STMicroelectronics' 3-axis digital MEMS gyroscopes*. STMicroelectronics. July 2011. URL: https://www.elecrow.com/download/TA0343.pdf.

[24] *TMS570LS1224 16- and 32 Bit RISC Flash Microcontroller*. Texas Instruments. Feb. 2015. URL: http://www.ti.com/product/TMS570LS1224#.

[25] Petter Tøndel. "Constrained Optimal Control via Multiparametric Quadratic Programming". PhD thesis. Department of Engineering Cybernetics, Norwegian University of Science and Technology, 2003.

[26] *UM1724 User manual STM32 Nucleo-64 boards*. STMicroelectroncis. Dec. 2017. URL: https://www.st.com/en/evaluation-tools/nucleo-l053r8.html.

[27] David Vallado and Paul Crawford. "SGP4 Orbit Determination". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2008. DOI: 10.2514/6.2008-6770.