Accurate Three-Axis Control of Spacecraft with Non-Uniform Mass Distribution

Master's thesis

Project Report

Group 1034

Aalborg University
Electronics and IT

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Accurate Three-Axis Control of Spacecraft with Non-Uniform Mass Distribution

Theme:
Master's thesis

Project Period:
Spring 2023

Project Group:
Group 1034

Participant(s):
Danny Dibbern
Siddharth Pathania

Supervisor(s):
Henrik Schiøler

Copies: 1

Page Numbers: 47

Date of Completion:
June 2, 2023

Abstract:

High-precision attitude control, especially for satellites with a non-uniformly distributed mass, is vital for applications in the space industry. This work studies the implementation and comparison of linear (PD, LQR, MPC) and non-linear (SMC) control strategies in an attitude control system. Tested on a simulated low earth orbit satellite, the controllers demonstrated diverse performance outcomes. Transforming linear control issues to the principal axis frame showed improved pointing accuracy, while SMC remained robust to such changes. Despite the controllers having comparable performance in the nadir-pointing scenario, LQR in the principal axis frame performed slightly better. While MPC was quickest to settle in landmark-switching scenarios, its output revealed substantial chatter, indicating a potential for improvement. SMC outperformed PD and LQR for large-angle maneuvers with equivalent control effort, reinforcing its relevance for large-angle maneuvers.

Contents

Chapter 1

Introduction

The attitude of a satellite describes it's orientation in space i.e. how it is placed in the 3D space it occupies. A satellite's attitude can also be stated as the rotation from a fixed reference coordinate system to the satellite's body coordinate system as shown in Fig 1.1. Being able to determine and control the satellite's attitude with extremely high accuracy is crucial for almost all modern day applications. In some remarkable cases such as in the James Webb Telescope, pointing accuracy of around 1 *milliarcsecond* is achieved from the attitude control system. [1] The pointing accuracy of a typical satellite equipped with star tracker is around 15 *arcsecond*.

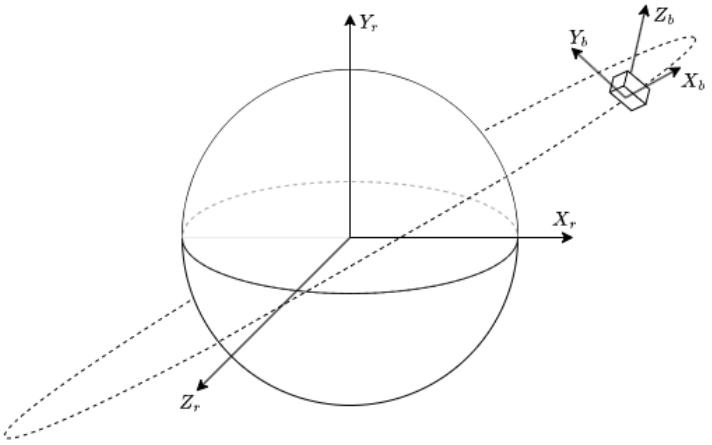

Figure 1.1: Satellite Body Coordinate and Reference Coordinate Systems

Before a satellite's attitude can be controlled the current attitude and the angular velocity has to be estimated. A satellite's attitude is estimated using the data from a variety of sensors such as gyroscopes, magnetometers, fine sun sensors and star trackers. Algorithms such as the Multiplicative Extended Kalman Filter can be used to combine the data from the sensors to estimate the satellite's attitude.[2]

However, the primary emphasis of the thesis will be on controlling the satellites' at-

titude, rather than estimation. A satellite's attitude can be controlled using various types of actuators, such as magnetorquers, reaction wheels, thrusters, and control moment gyros. Different control strategies can be applied to control the satellite's attitude such as Proportional-Integral-Derivative (PID), Linear Quadratic Regulator (LQR), Sliding Mode Control (SMC) and Model Predictive Control (MPC) depending on the application.

In this thesis, the aforementioned control strategies will be implemented for a satellite with a non-uniform mass distribution, utilizing only reaction wheels as actuators. The evaluation of these control strategies will focus on their accuracy, considering various test cases derived from common mission requirements.

The system and the controllers are simulated in MATLAB. The controllers are also implemented in C, and tested in a proprietary high-fidelity simulator from Space Inventor.[3]

Chapter 2

Satellite Modelling

In this chapter, the system model along with the model for the actuators are described.

2.1 Attitude Representation

The most basic way to represent attitude is by using Euler angles. Euler angles are a representation of the orientation of an object in 3D space, obtained by combining three elemental rotations about three different axes. But Euler angles are susceptible to singularities, which can cause ambiguities and difficulties in representing certain orientations in 3D space.[4] As such in this paper, quaternions are used to represent satellite's attitude.
Quaternions are an extension of the complex space and consists of four components, a scalar and 3 imaginary parts and is typically written as

$$q = q_x \mathrm{i} + q_y \mathrm{j} + q_z \mathrm{k} + q_w \tag{2.1}$$

Here q_x, q_y, q_z and q_w are real numbers and i, j and k are the basis vectors. Attitude and rotations are represented by using unit quaternions.

2.2 Quaternion Kinematics

This section describes the quaternion kinematics of the satellite. Kinematics describes how the attitude of the satellite changes for some angular velocity ω without taking into consideration the forces that are changing the angular velocity.
ω is a 3 component vector that represents the angular velocities in the axis of the satellite's body frame. And thus the kinematics are defined as the derivative of the quaternion as shown in Eq. 2.2.

$$\dot{q} = \frac{1}{2}\omega q \tag{2.2}$$

The vector $\omega = [\omega_x \ \omega_y \ \omega_z \ 0]$ can be interpreted as a pure quaternion, where 0 is the scalar part and $[\omega_x \ \omega_y \ \omega_z]$ forms the vector part. This structure enables us to

perform quaternion multiplication with ω and the quaternion q.
An Omega operator can be defined as follows

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{2.3}$$

As a result, we can derive the final form of the quaternion kinematics: [5]

$$\dot{q} = \frac{1}{2}\Omega q \tag{2.4}$$

2.3 Attitude Dynamics

Attitude dynamics of a satellite describes how the angular velocity of the satellite changes over time under the influence of external and internal torques. A satellite can be subject to a variety of external torques such as gravity gradient torque, aerodynamic torque, solar radiation pressure, magnetic torque, etc. [6] The satellite can also be affected by internal torques generated by actuators, such as reaction wheels.

2.3.1 Rigid Body Mechanics

The rotational behaviour of a satellite subjected to the torques depend heavily on the mass distribution of the satellite. A satellite is considered as a rigid body and the mass distribution can be represented by the moment of inertia matrix J. The moment of inertia matrix of a 3D rigid body is a symmetrical 3×3 matrix. The diagonal values represent the moment of inertia in x, y and z axes and the off-diagonal elements represent the cross-coupling between the axes. Hence, the general equation of the angular momentum of a satellite is given by:

$$H = J\omega \tag{2.5}$$

Through Euler's second law, the rate of change of angular momentum in an inertial frame is given by the net torque acting on the body.

$$\dot{H}_I = T_I \tag{2.6}$$

The equation mentioned above can be transformed into the satellite's body frame of reference, as the external torque and the satellite's moment of inertia are easier to compute in this frame. Thus the rate of change of angular momentum in the body frame is given by 2.7.

$$\dot{H}_B = T_B - \omega_B \times H_B \tag{2.7}$$

Assuming that the moment of inertia of the satellite in the body frame is constant, and by computing the derivative of Eq. 2.5 and inserting that into Eq. 2.7, the final form of Euler's rotational equation can be obtained as follows.[5]

$$\dot{\omega}_B = J^{-1}(T_B - \omega_B \times (J\omega_b)) \tag{2.8}$$

2.3.2 Non-Uniform Mass Distribution

In any rigid body, there's at least one special reference frame called the principal axes in which there's no cross-coupling between the axes and the moment of inertia matrix is a diagonal matrix.
For satellites with non-uniform mass distribution, the body frame is generally different from the principle axes frame. As the moment of inertia in the body frame is a real symmetric 3×3 matrix, it has three orthogonal eigenvectors e_B^k and corresponding real eigenvalues J_k. The eigenvalues are also known as the principle moments of inertia and thus the moment of inertia matrix in the principal axis frame is given as.

$$J_P = A_{BP}^T J_B A_{BP} = \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix} \tag{2.9}$$

Here the transformation matrix is given by

$$A_{BP} = \begin{bmatrix} e_B^1 & e_B^2 & e_B^3 \end{bmatrix} \tag{2.10}$$

Although the order and direction of the eigenvectors can be interchanged, it's crucial to verify that the resulting principal axis frame forms a right-hand coordinate system. This can be confirmed by ensuring that $e_B^1 \times e_B^2 = e_B^3$.

The transformation matrix A_{BP} serves to translate the control problem into the principal-axes coordinate system. Once the control output is computed, it can be transformed back into the body frame using the inverse of A_{BP} for implementation through the actuators.

2.4 Reaction Wheels

The reaction wheel is the primary actuator used for attitude control of satellites. It mainly consists of a rotating flywheel controlled by a motor. When the reaction wheel speed changes the spacecraft rotates in the opposite direction due to the conservation of angular momentum. One reaction wheel can only change the angular momentum of a satellite in a single axis and at least three reaction wheels are required for the attitude of the satellite to be fully controllable.

The reaction wheel is modeled as a disk with radius r, height h, and mass m_{ri}. Its moment of inertia in the wheel frame is given by:

$$J_R^W = \begin{bmatrix} m_{ri}r^2/2 & 0 & 0 \\ 0 & (m_{ri}/12)(3r^2+h^2) & 0 \\ 0 & 0 & (m_{ri}/12)(3r^2+h^2) \end{bmatrix} \tag{2.11}$$

The orientation of the reaction wheel can be represented by a vector $\hat{n}_{Ri}$ in the body frame, and thus the reaction wheel moment of inertia can be transformed to the body frame using the transformation matrix $T_{Ri}(0, \theta_{Ri}, \psi_{Ri})$ as shown in Eq. 2.12. Here the yaw (ψ_{Ri}) and pitch (θ_{Ri}) can be interpreted as the Euler angles that will rotate the vector $\hat{n}_{Ri}$ to align with the x-axis of the body frame.

$$J_R^B = T_{Ri}^T J_R^W T_{Ri} \tag{2.12}$$

The total angular momentum of the satellite is then given as:

$$H_B = J_B\omega_B + \sum_{i=1}^{n_{RW}} J_{Ri}^B \omega_{Ri}\hat{n}_{Ri} \tag{2.13}$$

and the torque applied on the satellite is given as:

$$T_{rw} = \sum_{i=1}^{n_{RW}} J_{Ri}^B \alpha_{Ri}\hat{n}_{Ri} \tag{2.14}$$

where α_{Ri} is the angular acceleration of the *ith* reaction wheel

2.4.1 Reaction Wheel Control

In many satellites, the mechanical design of the reaction wheel makes it one of the first points of failure.[7] As such for redundancy a satellite is generally equipped with more than 3 reaction wheels making it an over-actuated system. The total torque on the satellite in the body frame from the reaction wheels is given by:

$$T_{rw} = J_{rw}\alpha_{rw} \tag{2.15}$$

Here, α_{rw} is a vector of angular acceleration in the reaction wheels, and J_{rw} is a $3 \times n_{rw}$ matrix equal to:

$$J_{rw} = [J_{R1}^B\hat{n}_{R1}\ J_{R2}^B\hat{n}_{R2}\ ...\ J_{Rn}^B\hat{n}_{Rn}] \tag{2.16}$$

The controller gives the desired torque, but there's no unique solution to calculate the required reaction wheel acceleration from the desired torque, as J_{rw} is not a square matrix. Hence, the Moore-Penrose inverse of the matrix J_{rw} can be used to distribute the desired torque among the reaction wheels as shown in Eq. 2.17.[8]

$$\alpha_{rw} = J_{rw}^T(J_{rw}J_{rw}^T)^{-1}T_{des} \tag{2.17}$$

Chapter 3

Control Methods

In this chapter, the different control methods used to control the spacecraft's attitude are presented.

3.1 Linearization

As a prerequisite to designing the linear controllers, the nonlinear model consisting of Eq. 2.4 and Eq. 2.8:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \dot{\omega} \end{bmatrix} = f(x, u) \tag{3.1}$$

needs to be linearized and checked for controllability. Here u represents the torque applied to the satellite from the reaction wheels given by Eq. 2.14

The model is linearized around operating points $(\bar{x}, \bar{u})$, such that:

$$f(\bar{x}, \bar{u}) = 0 \tag{3.2}$$

which are found by selecting an operating input $\bar{u} = 0$ and solving Eq. 3.2. In this case, there are multiple equilibrium points and the unity quaternion is chosen as the operating point, as we will be utilizing the error quaternion in the linear control methods, and wish to keep this near unity:

$$\bar{x} = \begin{bmatrix} \bar{q}_1 & \bar{q}_2 & \bar{q}_3 & \bar{q}_4 & \bar{\omega}_1 & \bar{\omega}_2 & \bar{\omega}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

The linearization around the operating point is then achieved by a Taylor expansion of Eq. 3.1:

$$\dot{\bar{x}} + \dot{\tilde{x}} = f(\bar{x}, \bar{u}) + A\tilde{x} + B\tilde{u} \tag{3.4}$$

where the states and inputs are represented in a small signal model, as perturbations around the operating points:

$$x = \bar{x} + \tilde{x} \qquad\qquad u = \bar{u} + \tilde{u} \tag{3.5}$$

Using Eq. 3.2 and 3.5, Eq. 3.4 can be written as:

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u} \tag{3.6}$$

where A, the state transition matrix, and B, the input distribution matrix, are the linear approximations of Eq. 3.1:

$$A = \left[\frac{\partial f}{\partial x}\right]_{\bar{x},\,\bar{u}} \qquad B = \left[\frac{\partial f}{\partial u}\right]_{\bar{x},\,\bar{u}} \tag{3.7}$$

the Jacobians are then evaluated to obtain:

$$A = \begin{bmatrix} 0 & \frac{\bar{\omega}_3}{2} & -\frac{\bar{\omega}_2}{2} & \frac{\bar{\omega}_1}{2} & \frac{\bar{q}_4}{2} & -\frac{\bar{q}_3}{2} & \frac{\bar{q}_2}{2} \\ -\frac{\bar{\omega}_3}{2} & 0 & \frac{\bar{\omega}_1}{2} & \frac{\bar{\omega}_2}{2} & \frac{\bar{q}_3}{2} & \frac{\bar{q}_4}{2} & -\frac{\bar{q}_1}{2} \\ \frac{\bar{\omega}_2}{2} & -\frac{\bar{\omega}_1}{2} & 0 & \frac{\bar{\omega}_3}{2} & -\frac{\bar{q}_2}{2} & \frac{\bar{q}_1}{2} & \frac{\bar{q}_4}{2} \\ -\frac{\bar{\omega}_1}{2} & -\frac{\bar{\omega}_2}{2} & -\frac{\bar{\omega}_3}{2} & 0 & -\frac{\bar{q}_1}{2} & -\frac{\bar{q}_2}{2} & -\frac{\bar{q}_3}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8a}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8b}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ (J^{-1})_{11} & (J^{-1})_{12} & (J^{-1})_{13} \\ (J^{-1})_{21} & (J^{-1})_{22} & (J^{-1})_{23} \\ (J^{-1})_{31} & (J^{-1})_{32} & (J^{-1})_{33} \end{bmatrix} \tag{3.9}$$

To ensure the system can be controlled using this linear model, the controllability matrix needs to be computed and its rank checked:

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B & A^5B & A^6B \end{bmatrix} \tag{3.10}$$

resulting in a matrix with only 6 linearly independent columns, thus the controllability matrix is rank-deficient:

$$\text{rank}(\mathcal{C}) = 6 \neq n \tag{3.11}$$

Meaning this linear model is not able to control the system. Instead, a reduced-order model will be utilized. The quaternion can be split into its scalar and vector parts:

$$\dot{q}_{1:3} = -\frac{1}{2} q_{1:3} \times \omega + \frac{1}{2} q_4 \omega \tag{3.12a}$$

$$\dot{q}_4 = -\frac{1}{2}\omega^T q_{1:3} \tag{3.12b}$$

From Eqs. 2.3 and 2.4 we have:

$$\dot{q} = \frac{1}{2}\begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} q \tag{3.13}$$

which can be equivalently represented as:

$$\dot{q} = \frac{1}{2}\begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_2 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \begin{bmatrix} \omega \\ 0 \end{bmatrix} \tag{3.14}$$

where the scalar part can be computed as:

$$q_4 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2} \tag{3.15}$$

Using Eqs. 3.14 and 3.15 we get:

$$\dot{q}_{1:3} = \frac{1}{2}\begin{bmatrix} \sqrt{1 - q_1^2 - q_2^2 - q_3^2} & q_3 & -q_2 \\ -q_3 & \sqrt{1 - q_1^2 - q_2^2 - q_3^2} & q_1 \\ q_2 & -q_1 & \sqrt{1 - q_1^2 - q_2^2 - q_3^2} \end{bmatrix} \omega \tag{3.16}$$

which in combination with the rigid-body dynamics of Eq. **??**, is the reduced-order model[9] :

$$\dot{x} = \begin{bmatrix} \dot{q}_{1:3} \\ \dot{\omega} \end{bmatrix} = g(x, u) \tag{3.17}$$

This model is linearized as described previously in this section, resulting in the Jacobians:

$$A = \left[\frac{\partial g}{\partial x}\right]_{\bar{x},\,\bar{u}} \qquad B = \left[\frac{\partial g}{\partial u}\right]_{\bar{x},\,\bar{u}} \tag{3.18}$$

which with the operating points:

$$\bar{x} = \begin{bmatrix} \bar{q}_1 & \bar{q}_2 & \bar{q}_3 & \bar{\omega}_1 & \bar{\omega}_2 & \bar{\omega}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.19}$$

are evaluated to obtain:

$$A = \begin{bmatrix} -\frac{\bar{\omega}_1\,\bar{q}_1}{2\alpha} & \frac{\bar{\omega}_3}{2} + \frac{\bar{\omega}_1\bar{q}_2}{2\alpha} & \frac{\bar{\omega}_1\bar{q}_3}{2\alpha} - \frac{\bar{\omega}_2}{2} & \frac{\alpha}{2} & -\frac{\bar{q}_3}{2} & \frac{\bar{q}_2}{2} \\ -\frac{\bar{\omega}_3}{2} - \frac{\bar{\omega}_2\bar{q}_1}{2\alpha} & \frac{\bar{\omega}_2\bar{q}_2}{2\alpha} & \frac{\bar{\omega}_1}{2} + \frac{\bar{\omega}_2\bar{q}_3}{2\alpha} & \frac{\bar{q}_3}{2} & \frac{\alpha}{2} & -\frac{\bar{q}_1}{2} \\ \frac{\bar{\omega}_2}{2} - \frac{\bar{\omega}_3\bar{q}_1}{2\alpha} & \frac{\bar{\omega}_3\bar{q}_2}{2\alpha} - \frac{\bar{\omega}_1}{2} & \frac{\bar{\omega}_3\bar{q}_3}{2\alpha} & -\frac{\bar{q}_2}{2} & \frac{\bar{q}_1}{2} & \frac{\alpha}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.20a}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.20b}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ (J^{-1})_{11} & (J^{-1})_{12} & (J^{-1})_{13} \\ (J^{-1})_{21} & (J^{-1})_{22} & (J^{-1})_{23} \\ (J^{-1})_{31} & (J^{-1})_{32} & (J^{-1})_{33} \end{bmatrix} \tag{3.20c}$$

where $\alpha = \sqrt{-\bar{q}_1^2 + \bar{q}_2^2 + \bar{q}_3^2 + 1}$

Computing the controllability matrix with the reduced-order model;

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B & A^5B \end{bmatrix} \tag{3.21}$$

results in the controllability matrix being full rank:

$$\text{rank}(\mathcal{C}) = 6 = n \tag{3.22}$$

Thus we have a controllable linear model of Eqs. 2.4 and 2.8

Discretization

As the controllers will be implemented, to be able to execute on an onboard computer, it is necessary to discretize the model. This can be achieved using zero-order hold discretization and further approximated with sufficient accuracy by:

$$A_d = e^{At_s} \approx I + At_s \tag{3.23a}$$

$$B_d = A^{-1}(e^{At_s} - I)B \approx Bt_s \tag{3.23b}$$

$$C_d = C \tag{3.23c}$$

where I is an $n \times n$ identity matrix, and t_s is the sample time.

Now with a controllable model and discrete-time linear time-invariant system, described by the set of difference equations:

$$\tilde{x}_{k+1} = A_d\tilde{x}_k + B_d\tilde{u}_k \tag{3.24a}$$

$$y_k = C_d\tilde{x}_k \tag{3.24b}$$

It is possible to begin the linear controller designs.

3.2 Proportional-Derivative Control

The PD controller is one of the simplest and most widely used controllers in the industry. It's a state feedback controller with the control output calculated as the sum of the proportional, integral and derivative term of the error value.
For satellite attitude control the quaternion set point of the attitude is q_{ref} and the set point for the angular velocity is ω_{ref}. Hence the error can be calculated as

$$\delta q = q \otimes q_{ref}^{-1} \tag{3.25}$$

$$\delta \omega = \omega - \omega_{ref} \tag{3.26}$$

Then a simple control law can be given as:

$$T_c = -k_p \delta q_{1:3} - k_d \delta \omega \tag{3.27}$$

Here, T_c describes the desired torque and k_p and k_d are positive scalars and $\delta q_{1:3}$ is the vector part of the error quaternion.

But this control will not necessarily take the shortest path to the desired set points when the quaternion error scalar δq_4 is negative. This can be overcome by slightly modifying the control law as shown in Eq 3.28. [10]

$$T_c = -k_p sign(\delta q_4) \delta q_{1:3} - k_d \delta \omega \tag{3.28}$$

3.3 Linear Quadratic Regulator

Linear Quadratic Regulator is an optimal state feedback controller for linear systems such as the one described in Eq. 3.24. The state feedback control law is given as:

$$u_k = -K x_k \tag{3.29}$$

Here, the optimal gain matrix K, is the one that minimizes the infinite horizon discrete quadratic cost function:

$$J = \sum_{k=1}^{\infty} x_k^T Q x_k + u_k^T R u_k + 2 x_k^T N u_k \tag{3.30}$$

where Q, R and N are weighing matrices for the states, inputs and an optional cross-term. These can be seen as tuning parameters for the controller, providing a trade-off between actuator effort and controller performance. As such, the gain matrix K can be computed by:

$$K = (R + B^T P B)^{-1} (B^T P A + N)^T \tag{3.31}$$

Where P is the solution to the discrete-time algebraic Riccati equation:

$$P = A^T PA - (A^T PB + N)(R + B^T PB)^{-1}(B^T PA + N^T) + Q \tag{3.32}$$

Similar to PID, it is not necessary that the LQR controller will take the shortest path to the set point. This can be overcome by multiplying the gain matrix by a 6×6 diagonal matrix:

$$F = \begin{bmatrix} \text{sign}(\delta q_4) & 0 & 0 & 0 & 0 & 0 \\ 0 & \text{sign}(\delta q_4) & 0 & 0 & 0 & 0 \\ 0 & 0 & \text{sign}(\delta q_4) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.33a}$$

$$K' = KF \tag{3.33b}$$

But as we want to drive the system toward a changing reference quaternion, with the current state vector $x = \begin{bmatrix} q_{1:3} \\ w_{1:3} \end{bmatrix}$, integral action and/or relinearization around the changing reference quaternion would have to be added. Instead, the state vector is augmented to include the error states: $x = \begin{bmatrix} \delta q_{1:3} \\ \delta w_{1:3} \end{bmatrix}$, as such a single linearization around the unity quaternion (no error) is sufficient. The control law then becomes:

$$u_k = -K' \begin{bmatrix} \delta q_{1:3} \\ \delta w_{1:3} \end{bmatrix} \tag{3.34}$$

Comparing this to the PD control law in Eq. 3.28, it can be seen that this LQR is similar to the PD controller but with optimally derived gains.

3.4 Sliding Mode Control

Various satellite applications require large-angle maneuvers. However, the conventional linear controllers we've discussed are not suitable for these maneuvers due to the high non-linearity in the system dynamics. Moreover, the controller should be able to ensure system stability during these maneuvers, even if the system is not perfectly modeled. The robust sliding mode has been considered a good control design choice, as it ensures accurate tracking even during complex maneuvers and when faced with model variations. [11]
The sliding surface vector for the controller is chosen as follows: [12]

$$s = (\omega - \omega_{ref}) + k\delta q_{1:3} \tag{3.35}$$

where k is a positive scalar parameter. Here by setting the angular velocity set point, w_{ref}, the quaternion set point q_{ref} is then governed by the kinematics in Eq. 2.2, and as such the controller will have driven the system to the set points if $s = 0$, which is the sliding manifold.
Thus the derivative of the sliding variable is:

$$\dot{s} = (\dot{\omega} - \dot{\omega}_{ref}) + k\delta\dot{q}_{1:3} \tag{3.36}$$

This equation can be expanded by the system Eqs. 2.2 and 2.8 to get:

$$\dot{s} = -J^{-1}\omega \times (J\omega) + J^{-1}T_e - \dot{\omega}_{ref} + \frac{k}{2}[\delta q_4(\omega - \omega_{ref}) + \delta q_{1:3} \times (\omega + \omega_{ref})] \tag{3.37}$$

In this equation T_e is the equivalent control law and can be computed from the equation $\dot{s} = 0$ to get:

$$T_e = J(\frac{k}{2}[\delta q_4(\omega_{ref} - \omega) - \delta q_{1:3} \times (\omega + \omega_{ref})] - \dot{\omega}_{ref}) + \omega \times (J\omega) \tag{3.38}$$

To account for model uncertainties an additional discontinuous term can be added to the control law to ensure that the system states reach the sliding manifold:

$$T = J(\frac{k}{2}[\delta q_4(\omega_{ref} - \omega) - \delta q_{1:3} \times (\omega + \omega_{ref})] - \dot{\omega}_{ref} - G\bar{s}) + \omega \times (J\omega) \tag{3.39}$$

Here G is a positive definite matrix and $\bar{s}$ is given as the discontinous function:

$$\bar{s}_i = \text{sign}(s_i) \tag{3.40}$$

To reduce chattering in the control signal, the discontinuous function can be replaced with a boundary layer given by the saturation function:

$$\bar{s}_i = \text{sat}(s_i/\epsilon_i) \tag{3.41}$$

with ϵ_i a positive constant defining the width of the boundary layer.

The sliding vector in Eq. 3.35 can also be slightly modified to ensure that the controller takes the shortest path to the reference:

$$s = (\omega - \omega_{ref}) + k\text{sign}(\delta q_4)\delta q_{1:3} \tag{3.42}$$

Resulting in the final control law: [12]

$$T = J(\frac{k}{2}[\delta q_4(\omega_{ref} - \omega) - \text{sign}(\delta q_4)\delta q_{1:3} \times (\omega + \omega_{ref})] - \dot{\omega}_{ref} - G\bar{s}) + \omega \times (J\omega) \tag{3.43}$$

3.5 Model Predictive Control

Model Predictive Control is an optimal control strategy. But contrary to LQR, the optimization is performed online, over a finite horizon instead of the infinite horizon in LQR, while also complying with state and input constraints. The finite horizon is defined by: The prediction horizon, H_p, which defines how many future states should be predicted and included in the cost function, this should be large enough to reach the control objective at the end of the horizon. The control horizon, H_u, defines how many future control inputs should be computed, usually only the first input is applied to the real system, while the remaining are crucial in the cost function, to optimally reach the objective at the end of the prediction horizon. The difference between H_p and H_u defines an interval where the control input is constant.

This allows for MPC to better deal with changing dynamics, shifting objectives and external disturbances, compared to the static control law derived from optimizing the systems' long-term behavior in Eq. 3.30.

The following cost function was used:[13]

$$J_k = \frac{1}{2} x_{k+H_p} P x_{k+H_p} + \frac{1}{2} \sum_{i=0}^{H_p-1} x_{k+i}^T Q x_{k+i} + \sum_{i=0}^{H_u} u_{k+i}^T R u_{k+i} \tag{3.44}$$

which looks similar to the cost function used in the LQR design, except for the finite horizon and that the terminal state has its own weighing matrix, P. This is to ensure stability, by putting more weight on reaching the terminal state, which as mentioned earlier should have reached the control objective by having a large enough H_p.
There is no reference trajectory in the cost function, as seen in many other MPC implementations, for example, the cost function:

$$J_k = \frac{1}{2} \sum_{i=0}^{H_p} (x_{k+i} - r_{k+i})^T Q (x_{k+i} - r_{k+i}) + \sum_{i=0}^{H_u} \Delta u_{k+i}^T R \Delta u_{k+i} \tag{3.45}$$

where r is a vector of reference states, and Δu is the change in input. This is because we are dealing with quaternions, and thus cannot simply subtract the reference quaternion from the quaternion state. Instead, the error is computed by quaternion multiplication as in Eq. 3.25. But as we are dealing with a reduced-order quaternion model, with only the vector part in the state vector, the full predicted and reference quaternions would have to be reconstructed using Eq. 3.15 before multiplication. The square root in the cost function would complicate the quadratic programming (QP) problem to potentially no longer be convex.

To introduce reference tracking using the cost function in Eq. 3.44, the state vector is augmented to include the error states as in the LQR design:

$$x = \begin{bmatrix} \delta q \\ \delta \omega \end{bmatrix} \tag{3.46}$$

Quadratic Programming Problem

The optimization variable u, can be formulated in vector form as:

$$\mathcal{U}_k = \begin{bmatrix} u_k^T & u_{k+1}^T & \dots & u_{k+H_u}^T \end{bmatrix}^T \tag{3.47}$$

The cost function in Eq. 3.44 can thus be written in a more compact form as:[13]

$$\min_{\mathcal{U}_k} J_k = \frac{1}{2}\mathcal{U}_k^T H \mathcal{U}_k + c^T \mathcal{U}_k \tag{3.48}$$

where

$$H = (\phi_{H_p}^T P \phi_{H_p} + \sum_{i=1}^{H_p-1} Q_i + R_{H_p}) \tag{3.49a}$$

$$\phi_{H_p} = \begin{bmatrix} A_{H_p-1}B & A_{H_p-2}B & \dots & A_0 B \end{bmatrix} \tag{3.49b}$$

$$A_{H_p} = A^{H_p} \tag{3.49c}$$

$$Q_i = \begin{bmatrix} \phi_i^T Q \phi_i \end{bmatrix} \tag{3.49d}$$

$$R_{H_p} = \begin{bmatrix} R & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & R \end{bmatrix} \tag{3.49e}$$

and

$$c^T = x_k^T (A_{H_p}^T P \phi_{H_p} + \sum_{i=1}^{H_p-1} S_i) \tag{3.50a}$$

$$S_i = \begin{bmatrix} A_i^T Q \phi_i & 0 \end{bmatrix} \tag{3.50b}$$

Constraints

As mentioned, MPC is unique in the way it is able to take state and input constraints into consideration when optimizing the cost function. In this project, there are no physical constraints on the states of the system, but the reaction wheels do have physical limits on the amount of angular momentum they can store. To avoid actuator saturation, input constraints will be utilized:

$$u_{min} \leq u \leq u_{max} \tag{3.51}$$

The constraints can also be written as two inequalities:

$$-u + u_{min} \leq 0 \tag{3.52a}$$

$$u - u_{max} \leq 0 \tag{3.52b}$$

which are then formulated in matrix-vector form:

$$F \begin{bmatrix} \mathcal{U} \\ 1 \end{bmatrix} \leq 0 \tag{3.53a}$$

$$\begin{bmatrix} -1 & \cdots & 0 & u_{min} \\ 1 & \cdots & 0 & -u_{max} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & -1 & u_{min} \\ 0 & \cdots & 1 & -u_{max} \end{bmatrix} \begin{bmatrix} u_k \\ \vdots \\ u_{k+H_u} \\ 1 \end{bmatrix} \leq 0 \tag{3.53b}$$

F has the form $\begin{bmatrix} F_1 & \dots & F_{H_u} & f \end{bmatrix}$, by defining $\mathcal{F} = \begin{bmatrix} F_1 & \dots & F_{H_u} \end{bmatrix}$ the constraints can be formulated in terms of the optimization variable $\mathcal{U}$:

$$\mathcal{F}\mathcal{U} + f \leq 0 \tag{3.54}$$

Now that both constraints and the cost function in Eq. 3.44 have been formulated in terms of $\mathcal{U}$, the Quadratic Programming (QP) problem can be stated as:

$$\underset{\mathcal{U}_k}{\text{minimize}} \qquad J_k = \frac{1}{2}\mathcal{U}_k^T H \mathcal{U}_k + c^T \mathcal{U}_k \tag{3.55a}$$

$$\text{subject to} \qquad \mathcal{F}\mathcal{U}_k \leq -f \tag{3.55b}$$

Active Set Method for Solving Convex QP

In this project, the Active Set Method (ASM) is utilized to solve the QP problem stated in Eqs. 3.55, for the implementation in C.

A standard QP problem can be represented as follows [14]:

$$\min_x \quad J(x) = \frac{1}{2}x^T Q x + x^T c \tag{3.56a}$$

$$\text{subject to} \quad a_i^T x = b_i, \quad i \in \mathcal{E}, \tag{3.56b}$$

$$a_j^T x \geq b_j, \quad j \in \mathcal{I}, \quad j > i \ \forall \ i, j \tag{3.56c}$$

Here, the QP problem consists of a quadratic cost function to be minimized, subject to a combination of equality and inequality constraints, where $\mathcal{E}$ is the set of indices for equality constraints, and $\mathcal{I}$ the set for inequality constraints.

For the convenience of further manipulation, the problem can be reformulated with only inequality constraints by transforming each equality constraint into two opposite inequality constraints [15]:

$$\underset{x}{\text{minimize}} \quad J(x) = \frac{1}{2}x^T Q x + x^T c \tag{3.57a}$$

$$\text{subject to} \quad Ax \geq b, \tag{3.57b}$$

with A now defined as:

$$A = \begin{bmatrix} a_1^T \\ -a_1^T \\ \vdots \\ a_i^T \\ -a_i^T \\ a_{i+1}^T \\ \vdots \\ a_j^T \\ a_{j+1}^T \\ \vdots \end{bmatrix} \quad \forall \ i, j \in \mathcal{E} \cup \mathcal{I} \tag{3.58}$$

Following this transformation, we can redefine $\mathcal{I}$ such that the equality constraints are included in the set of inequality constraints.

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_i^T \\ \vdots \\ a_n^T \end{bmatrix} \quad \forall \ i \in \mathcal{I}, \quad \mathcal{I} = \{i \mid i = 1, 2, \ldots, n\} \tag{3.59}$$

If the Hessian of the cost function, $\nabla^2 J(x) = Q$, is positive semi-definite, the cost function is convex and hence, the QP problem is convex. For convex problems,

the Karush-Kuhn-Tucker (KKT) conditions serve as the necessary and sufficient conditions for optimality of a solution, x^* [14]. The Lagrangian of the QP problem is then defined as:

$$\mathcal{L}(x, \lambda) = q(x) - \lambda^T(Ax - b) = \frac{1}{2}x^T Qx + x^T c - \lambda^T(Ax - b) \tag{3.60}$$

with λ being the vector of Lagrange multipliers. Then the KKT-conditions are given by: [14]

$$Qx^* - A^T\lambda^* = c \tag{3.61a}$$
$$\lambda_i({a_i}^T x^* - b_i) = 0 \quad \forall\, i \in \mathcal{I} \tag{3.61b}$$
$$Ax^* \geq b \tag{3.61c}$$
$$\lambda^* \geq 0 \tag{3.61d}$$

At the optimal point, it is likely that not all constraints are active. Thus, the active set is introduced:

$$\mathcal{A}(x) = \left\{ i \in \mathcal{I} \mid a_i^T x = b_i \right\} \tag{3.62}$$

which is the set of all constraint indices for which, at a given x, the corresponding optimization variable equals its boundary value [15]
The problem can be further simplified by considering only the active constraints at the optimal point, $\mathcal{A}(x^*)$, reducing the dimensions of A, b, and λ to $\bar{A}$, $\bar{b}$ and $\bar{\lambda}^*$, respectively. Then, the KKT-conditions can be written as follows:

$$Qx^* - \bar{A}^T\bar{\lambda}^* = c \tag{3.63a}$$
$$\bar{A}x^* = \bar{b} \tag{3.63b}$$

or equivalently:

$$\begin{bmatrix} Q & -\bar{A}^T \\ \bar{A} & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \bar{\lambda}^* \end{bmatrix} = \begin{bmatrix} c \\ \bar{b} \end{bmatrix} \tag{3.64}$$

This indicates that the original inequality constrained QP problem can be solved by a sequence of equality constrained subproblems, which forms the basis of the ASM [14].

The ASM iterates over the subproblems by initially selecting a feasible point x_k, determining the active set of constraints $\mathcal{A}_k$, and defining a working set at each iteration:

$$\mathcal{W}_k = \left\{ x \mid a_i^T x = b_i \;\forall\, i \in \mathcal{A}_k \right\} \tag{3.65}$$

The working set is a feasible set for the EQP subproblem, and at every iteration k, the optimization of x_k is performed only within $\mathcal{W}_k$. Given an iterate x_k and the

corresponding working set $\mathcal{W}_k$, the procedure checks whether x_k minimizes $J(x)$ and, if not, computes a step Δx by solving an equality constrained problem, like that of Eq. 3.64. [14]

The equality constrained subproblems can be presented in this form:

$$\underset{x_k+\Delta x}{\text{minimize}} \quad J(x_k+\Delta x) = \frac{1}{2}(x_k+\Delta x)^T Q(x_k+\Delta x) + (x_k+\Delta x)^T c, \tag{3.66a}$$

$$\text{subject to} \quad A_{\mathcal{W}_k}(x_k+\Delta x) = b_k, \tag{3.66b}$$

where $\Delta x = x - x_k$ is the step at the kth iteration, $A_{\mathcal{W}_k}$ is a submatrix of A, with rows corresponding to the indices in the set $\mathcal{A}_k$:

$$A_{\mathcal{W}_k} = \begin{bmatrix} a_i^T \\ \vdots \end{bmatrix} \quad \forall\, i \in \mathcal{A}_k \tag{3.67}$$

Similarly, b_k is a subvector of b, again with entries corresponding to the indices in the set $\mathcal{A}_k$:

$$b_k = \begin{bmatrix} b_i \\ \vdots \end{bmatrix} \quad \forall\, i \in \mathcal{A}_k \tag{3.68}$$

As x_k is constant at each iteration, this can be further simplified into:

$$\underset{\Delta x}{\text{minimize}} \quad J(\Delta x) = \frac{1}{2}\Delta x^T Q \Delta x + (Qx_k + c)^T \Delta x, \tag{3.69a}$$

$$\text{subject to} \quad A_{\mathcal{W}_k}\Delta x = 0, \tag{3.69b}$$

This leads to the following linear KKT system:

$$\begin{bmatrix} Q & {A_{\mathcal{W}_k}}^T \\ A_{\mathcal{W}_k} & 0 \end{bmatrix} \begin{bmatrix} \Delta x^* \\ \lambda^* \end{bmatrix} = - \begin{bmatrix} Qx_k - c \\ 0 \end{bmatrix} \tag{3.70}$$

If the sum $x_k + \Delta x_k$ results in a feasible point, we can directly proceed to update our current iterate:

$$x_{k+1} = x_k + \Delta x_k \tag{3.71}$$

However, if the update renders x_{k+1} infeasible, we have to adjust our step direction Δx to maintain feasibility. In this case, we consider Δx as a direction of search and scale it using the scalar α_k to derive a feasible solution:

$$x_{k+1} = x_k + \alpha_k \Delta x_k \tag{3.72}$$

The step size α_k is chosen to be the largest value in the interval $[0, 1]$ that ensures all constraints remain valid:

$$\alpha_k = \min\left(1, \min_{i \notin \mathcal{A}_k,\, a_i^T \Delta x_k < 0} \frac{b_i - a_i^T x_k}{a_i^T \Delta x_k}\right) \tag{3.73}$$

Essentially, we take the largest possible step that does not violate any of the inactive constraints at the kth iteration.
If $\alpha_k < 1$, it signifies that the step has encountered a constraint, not in $\mathcal{A}_k$, which is limiting the algorithm from taking a larger step. In such a scenario, we include the blocking constraint to the active set:

$$\mathcal{A}_{k+1} = \mathcal{A}_k \cup \{i\} \tag{3.74}$$

Here, i refers to the index of the blocking constraint. Consequently, we construct a new working set $\mathcal{W}_{k+1}$ as per Eq. 3.65, using $\mathcal{A}_{k+1}$, and continue with the iterations. The ASM continues to iterate until it arrives at a point $\hat{x}$ that minimizes the objective function within the current working set. This point is determined when $\Delta x_k = 0$. It's worth mentioning here that at this point, the first three KKT conditions (Eqs. 3.61a through 3.61c) are automatically satisfied. The only condition left to verify is the non-negativity of the Lagrange multipliers for inequality constraints, given by Eq. 3.61d.

If all the Lagrange multipliers in $\hat{\lambda}$ (corresponding to $\hat{x}$) are non-negative, all the KKT conditions are satisfied, thus indicating that $\hat{x}$ is indeed the optimal solution x^*. Consequently, the ASM terminates.
However, if negative Lagrange multipliers exist in $\hat{\lambda}$, the constraints associated with these multipliers can be excluded from the active set. In this case, we update the active set as:

$$\mathcal{A}_{k+1} = \mathcal{A}_k \setminus \{j\} \tag{3.75}$$

Here, j corresponds to the index of the smallest Lagrange multiplier:

$$j = \arg\min_{j \in \mathcal{W}_k} \hat{\lambda}_j \tag{3.76}$$

The algorithm then proceeds to the next iteration with the updated working set and the process continues until the optimal solution is obtained.

Schur Complement and Cholesky Decomposition for Faster Computation

As shown previously in this section, the equality constrained subproblem is solvable via a linear KKT system as in Eq. 3.70, which in combination with the QP problem in Eq. 3.55, is shown as:

$$\begin{bmatrix} H & {A_{\mathcal{W}}}^T \\ A_{\mathcal{W}} & 0 \end{bmatrix} \begin{bmatrix} \Delta x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -(Hx_k + c) \\ 0 \end{bmatrix} \tag{3.77}$$

A straightforward method of solving Eq. 3.77 would be to carry out a matrix inversion on the left-hand side:

$$\begin{bmatrix} \Delta x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} H & A_{\mathcal{W}}{}^T \\ A_{\mathcal{W}} & 0 \end{bmatrix}^{-1} \begin{bmatrix} -(Hx_k + c) \\ 0 \end{bmatrix} \tag{3.78}$$

This is a rather naive implementation, as several things can be done to speed up the computation.

Supposing that the weight matrices P, Q and R in Eq. 3.49a are symmetric and positive definite, then H, is also symmetric and positive definite, and can be used as a pivot for block Gaussian elimination of Eq. 3.77. [14] The resulting system becomes:

$$\begin{bmatrix} H & A_{\mathcal{W}}{}^T \\ 0 & -(A_{\mathcal{W}} H^{-1} A_{\mathcal{W}}{}^T) \end{bmatrix} \begin{bmatrix} \Delta x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -(Hx_k + c) \\ A_{\mathcal{W}} H^{-1}(Hx_k + c) \end{bmatrix} \tag{3.79}$$

Here, $-(A_{\mathcal{W}} H^{-1} A_{\mathcal{W}}{}^T)$ is the Schur complement of $\begin{bmatrix} H & A_{\mathcal{W}}{}^T \\ A_{\mathcal{W}} & 0 \end{bmatrix}$ and the system in Eq. 3.79 can be solved using the two equations: [14]:

$$(A_{\mathcal{W}} H^{-1} A_{\mathcal{W}}{}^T)\lambda^* = -A_{\mathcal{W}} H^{-1}(Hx_k + c) \tag{3.80a}$$

$$\Delta x^* = H^{-1}(-A_{\mathcal{W}}{}^T \lambda^* - (Hx_k + c)) \tag{3.80b}$$

Since H is positive definite, Eq. 3.80a can be solved using Cholesky decomposition instead of matrix inversion. This significantly reduces computation time.
First, compute the Cholesky decomposition as $LL^T = (A_{\mathcal{W}} H^{-1} A_{\mathcal{W}}{}^T)$. Next, solve for y using forward substitution, $Ly = A_{\mathcal{W}} H^{-1}(Hx_k + c)$. Finally, solve for λ using back substitution, $L^T \lambda^* = y$. At last, use Eq. 3.80b to compute the change Δx^* in the optimization variable x_k.

Chapter 4

Implementation

In this chapter, the implementation in simulation of various control methods will be presented. The system described in Chapter 2 is simulated in MATLAB, and the control methods described in Chapter 3 are implemented and simulated in both MATLAB, and C, in a high-fidelity simulator provided by Space Inventor.

The implementation, in C, is based on confidential information, and will thus not be shared in this chapter, instead the focus will be on the MATLAB implementation. The results from the control experiments conducted in Space Inventor's simulator, which includes external disturbances such as are shared in Chapter 6.

4.1 System Simulation

The non-linear system model in Eqs. 2.2 and 2.8 is simulated with a time step of 1 sec. The Runge-Kutta 4 (RK4) numerical method is used to propagate the system states. RK4 is a 4th order fixed-step solver, in which a weighted average of the derivatives at fixed points, within the time step, is used to propagate the system as:

$$\begin{aligned} k_1 &= \dot{x}(t_k, x_k, u_k) \\ k_2 &= \dot{x}(t_k+, x_k + k_1 \Delta t/2, u_k) \\ k_3 &= \dot{x}(t_k + \Delta t/2, x_k + k_2 \Delta t/2, u_k) \\ k_4 &= \dot{x}(t_k + \Delta t, x_k + k_3 \Delta t, u_k) \\ k &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ x_{k+1} &= x_k + \Delta t k \end{aligned} \tag{4.1}$$

4.1.1 Reaction Wheels

A 4 reaction wheel NASA standard configuration is chosen, where 3 reaction wheels are aligned with the body frame axis, and the fourth wheel is mounted such that it has non-zero projections along all three axes. Thus the 3×4 orienta-

tion matrix for the reaction wheels is given as:

$$Or = \begin{bmatrix} 1 & 0 & 0 & 0.5774 \\ 0 & 1 & 0 & 0.5774 \\ 0 & 0 & 1 & 0.5774 \end{bmatrix} \tag{4.2}$$

In general, the motor control for the reaction wheels has a higher bandwidth than the attitude control system. Therefore, it is assumed that in the attitude control system, the desired acceleration of the reaction wheels is achieved instantly.
The desired acceleration is calculated using Eq. 2.17, but is also limited by the maximum speed and torque that the reaction wheel can provide.

4.2 Attitude Determination

Before the system can be controlled, the system's current states needs to be estimated. Satellite attitude determination has been thoroughly explored in [16], and is not detailed in this project. In the previous work, different variations of the Multiplicative Extended Kalman Filter (MEKF) were explored for mission deployment, along with pre-mission calibration filters. The calibration filters consist of an attitude independent Extended Kalman Filter and Unscented Kalman Filter for magnetometer bias and scaling factor calibration, and an attitude dependent MEKF for gyroscope bias and scaling factor calibration. In this project, the scaling factors are not modeled and thus the sensors are assumed calibrated. An attitude determination system for mission deployment is implemented to facilitate the development of the controllers.

The attitude and the angular velocity are estimated using a star tracker and a gyroscope. But as a gyroscope's bias drifts over time, the bias also needs to be estimated. A Sequential Murrel's formulation of the Multiplicative Extended Kalman Filter (SMEKF) is applied. It uses star tracker vector measurements and angular velocity data from the gyroscope to estimate the satellite's attitude and angular velocity.

The star tracker is implemented using the Shuster noise model with a standard deviation of 35 arcsecond.[17]

The gyroscope's bias is modeled as a random walk process driven by Gaussian white noise as shown: [18]

$$\omega = \omega^{\text{true}} + \beta^{\text{true}} + \eta_v \tag{4.3a}$$
$$\dot{\beta}^{\text{true}} = \eta_u \tag{4.3b}$$

where $\eta_v \sim \mathcal{N}(0, \sigma_v^2)$ and $\eta_u \sim \mathcal{N}(0, \sigma_u^2)$.
The bias variance and the noise variance are selected to be:

$$\sigma_v{}^2 = \text{1e-13} \qquad \sigma_u{}^2 = \text{1e-19} \tag{4.4}$$

The process noise covariance, Q, and the measurement noise covariance, R, matrices are set to the following values:

$$Q = \begin{bmatrix} 10^{-13} & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^{-13} & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-13} & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-19} & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-19} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-19} \end{bmatrix} \tag{4.5}$$

$$R = \begin{bmatrix} \text{5e-6} & 0 & 0 \\ 0 & \text{5e-6} & 0 \\ 0 & 0 & \text{5e-6} \end{bmatrix} \tag{4.6}$$

4.3 Principal Axis Transformations

As mentioned in Chapter 2, for satellites with non-uniform mass distribution the control problem can be transformed to and from the principle axis.

The transformation matrix (A_{BP}) given in Eq. 2.10 represents the rotation from body frame to principal axis frame. Thus the quaternion attitude in principal axis frame is calculated by multiplying it with the quaternion representing the same rotation as shown in Eq. 4.7.

$$q_P = q(A_{BP}) \otimes q_B \tag{4.7}$$

The angular velocity vector is transformed from body frame to principal axis frame using A_{BP} as shown in Eqs. 4.8. Similarly the controller output i.e. the desired torque is transformed back to the body frame using the inverse of A_{BP} which is its transpose.

$$\omega_P = A_{BP}^T \omega_B \tag{4.8a}$$

$$T_B = A_{BP} T_P \tag{4.8b}$$

All the controllers discussed in this chapter are implemented to work in both body frame as well as in the principal axis frame.

4.4 PD Control

The simple PD control law given in Eq. 3.28 is implemented. The constants K_p and K_d are chosen to be proportional to the trace of the satellites inertia matrix J.

$$K_p = \frac{\text{mean}(\text{diag}(J))}{10} \tag{4.9}$$

$$K_d = \text{mean}(\text{diag}(J)) \tag{4.10}$$

4.5 LQR

As discussed in section 3.3, LQR control law is similar to the PD control law but with optimized gains. These gains are calculated using the MATLAB command `dlqr`, which takes the discretized linear system from Eq. 3.24 and the following system and measurement covariance matrices to calculate the optimal gain.

$$Q = \begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1600 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1600 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1600 \end{bmatrix} \tag{4.11}$$

$$R = \begin{bmatrix} 40000 & 0 & 0 \\ 0 & 40000 & 0 \\ 0 & 0 & 40000 \end{bmatrix} \tag{4.12}$$

which were initially determined using Bryson's rule, in which the diagonal elements are the normalized maximum acceptable variations in states and inputs, $Q_{i,i} = \frac{1}{x_{i_{max}}^2}$, with the following values:

$$q_{max} = 0.25 \tag{4.13a}$$

$$\omega_{max} = 0.025 \tag{4.13b}$$

$$u_{max} = 0.005 \tag{4.13c}$$

and then further fine-tuned

4.6 SMC

The sliding mode controller is implemented with the sliding manifold provided in Eq. 3.42 and the control law given by Eq. 3.43. The following parameter values are chosen to ensure that the controller is stable even in the most extreme model variations.

The gain for the sliding surface is set to:

$$k = 0.15 \tag{4.14}$$

To effectively deal with uncertainties and disturbances, the gains for the switching surface $\bar{s}$ is set to:

$$G = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \tag{4.15}$$

To minimize chattering in the control signal, ϵ in the saturation function from Eq. 3.41 is set to:

$$\epsilon = \begin{bmatrix} 0.01 \\ 0.01 \\ 0.01 \end{bmatrix} \tag{4.16}$$

4.7 MPC

Model Predictive Control solves an optimization problem to get the optimal control input over a prediction horizon. The linearized state space matrices of the system can be lifted to form the QP problem with input constraints shown in Eq. 3.55. The value of various parameters used in MPC are:

With a sampling time of 1 s, a prediction horizon of 10 allows for reaching the control objective at the end of the horizon, in the slowly changing nadir-pointing reference trajectory:

$$H_p = 10 \qquad H_u = 10 \tag{4.17}$$

The constraints are set based on the maximum torque that can be applied to the satellite from the reaction wheels:

$$u_{min} = -0.005 \qquad u_{max} = 0.005 \tag{4.18}$$

The cost function weights are set to prioritize eliminating the angular velocity error

and minimizing actuator effort:

$$Q = \begin{bmatrix} 5000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16000 \end{bmatrix} \qquad P = Q \tag{4.19}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

In MATLAB, the QP problem is solved using Yalmip using the MOSEK solver.[19][20]

In C, the Active Set Method is used to solve the QP problem. The code is originally based on an MPC library for Arduino [21]. Which has been modified to:

1. Have a different cost function given by Eq. 3.44
2. Have dynamic memory allocation
3. Use Space Inventor's linear algebra functions
4. Implemented in Space Inventor's simulator environment
5. Allow different constraints on individual states and inputs
6. Allow different weights on individual states and inputs

Chapter 5

Control Experiments

In this chapter, different experiments will be presented in order to evaluate the performance of the control methods described in Chapter 3, in a variety of mission scenarios. The evaluation of the control methods will be based on the following attributes:

- Mean Squared Reference Tracking Error
- Mean Squared Actuator Effort
- Handling of Non-Uniform Mass Distribution
- Settling time of the system

As mentioned earlier, the experiments will be conducted in the high-fidelity simulator provided by Space Inventor. Here, several different reference trajectory generators are implemented (see Appendix?).

5.1 Nadir-Pointing

For the purpose of evaluating the pointing accuracy of the controllers, a nadir-pointing reference trajectory is used, which represents the satellite pointing directly below itself, perpendicular to Earth. This is a scenario with a reference quaternion that is constantly changing at a slow rate.

The nadir-pointing reference trajectory has a slew rate of about 0.001 rad/s. Periodically, the quaternion reference changes faster, with ω_{ref} increasing up to 0.015 rad/s as shown in Fig. 5.1.

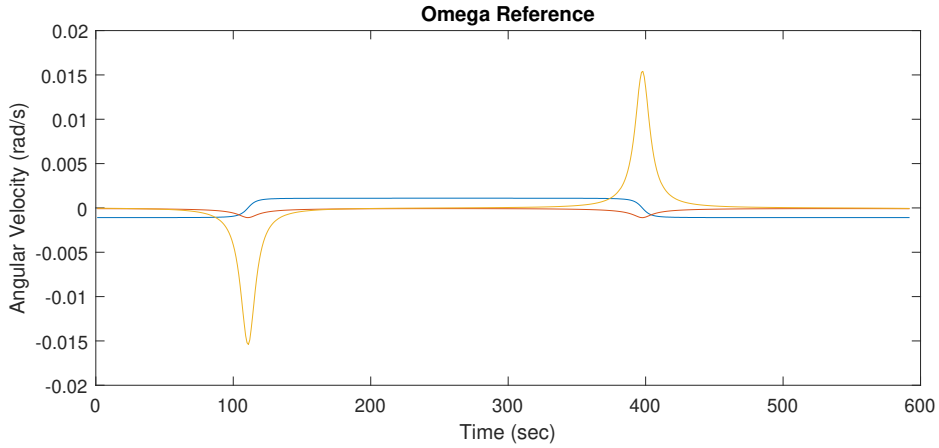

Figure 5.1: Changing ω_{ref} during nadir pointing maneuver

The corresponding reference quaternion is seen in Fig. 5.2

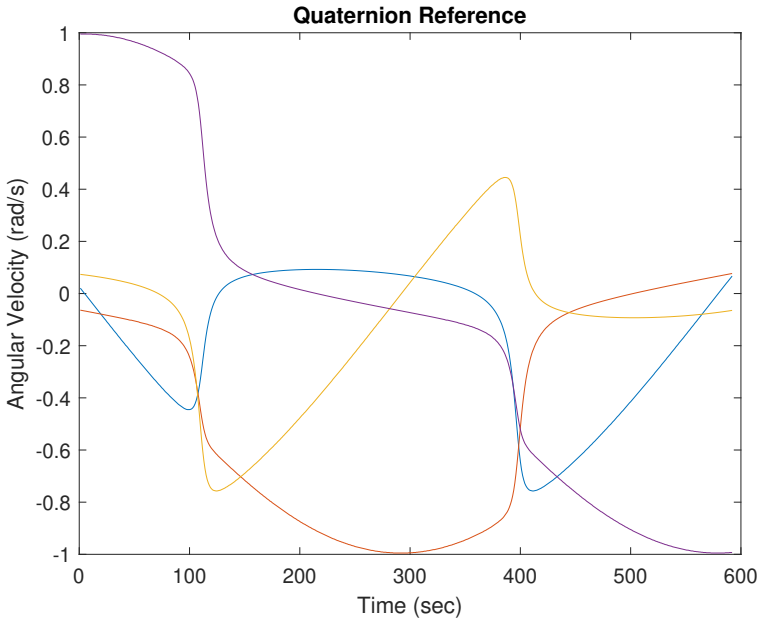

Figure 5.2: Changing q_{ref} during nadir pointing maneuver

5.2 Landmark Pointing

To also evaluate the performance of the controllers, when the reference quaternion changes quickly and discontinuously (step reference), the reference trajectory is constructed to emulate the maneuver of tracking landmarks on Earth, which will

periodically become unavailable due to the satellite's orbit, and then cause a step reference change to an available landmark.

The reference switching between two landmarks is modeled by a step reference change as seen in Fig. 5.3

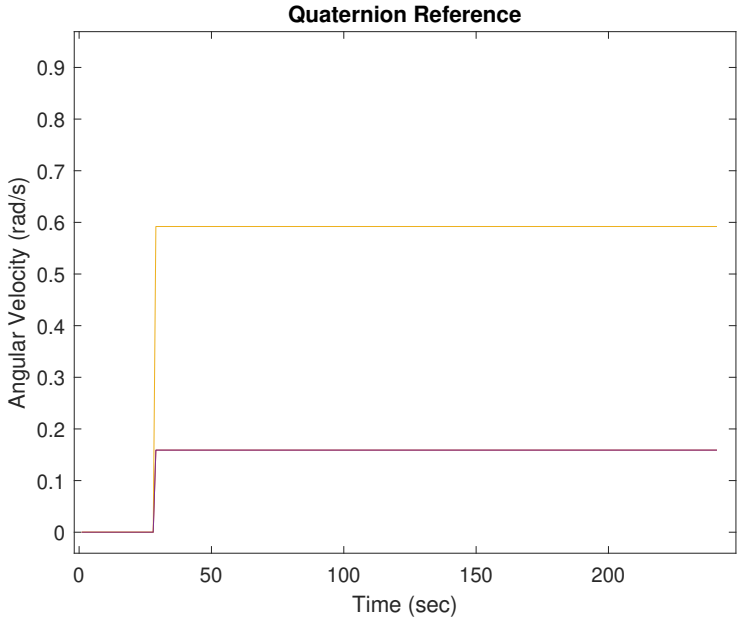

Figure 5.3: Changing q_{ref} during landmark-switching maneuver (step reference)

The corresponding angular velocity reference, ω_{ref}, is equal to 0 for the duration of the test.

5.3 Simulation Setup

During the project period, there was no physical satellite or test setup available, thus as mentioned earlier, the control experiments are conducted in a high-fidelity simulator provided by Space Inventor.

5.3.1 Parameters

The tuning parameters of the different controllers, are implemented with the values mentioned in Chapter 4.

The satellite in the nadir-pointing, landmark-pointing and model uncertainty experiment, is modeled as a micro-satellite with a mass of 12 kg, with the following

moment of inertia matrix:

$$J = \begin{bmatrix} 0.167184 & -0.035088 & 0.001204 \\ -0.035088 & 0.218268 & -0.0000516 \\ 0.001204 & -0.0000516 & 0.125216 \end{bmatrix} \tag{5.1}$$

Chapter 6

Results

In this chapter, the results obtained from the experiments described in Chapter 5 are presented. They are further discussed in Chapter 7.

6.1 Nadir-Pointing

In Table 6.1, the results of the experiments conducted with the nadir-pointing reference trajectory are shown. Here, the maximum Euler error, and the RMSE of the steady state Euler error and control effort are shown. The maximum error is recorded during the peaks of w_{ref} in Fig. 5.1, while the steady state error is computed over the slow periods.

Controller	Controller Frame	Max Deviation	Steady State Euler Error (MSE)	Control Effort (MSE)
PID	Body Frame	$\begin{bmatrix} 0.0975 \\ 0.1150 \\ 0.2990 \end{bmatrix}$	$\begin{bmatrix} 0.0009 \\ 0.0044 \\ 0.0016 \end{bmatrix}$	$\begin{bmatrix} 3.68e\text{-}7 \\ 2.33e\text{-}6 \\ 2.65e\text{-}7 \\ 2.38e\text{-}6 \end{bmatrix}$
	Principal Axis	$\begin{bmatrix} 0.0879 \\ 0.1260 \\ 0.2220 \end{bmatrix}$	$\begin{bmatrix} 0.0016 \\ 0.0014 \\ 0.0012 \end{bmatrix}$	$\begin{bmatrix} 3.65e\text{-}7 \\ 2.29e\text{-}6 \\ 2.47e\text{-}7 \\ 2.32e\text{-}6 \end{bmatrix}$
LQR	Body Frame	$\begin{bmatrix} 0.0747 \\ 0.1710 \\ 0.2060 \end{bmatrix}$	$\begin{bmatrix} 0.0019 \\ 0.0024 \\ 0.0016 \end{bmatrix}$	$\begin{bmatrix} 3.85e\text{-}7 \\ 2.26e\text{-}6 \\ 2.54e\text{-}7 \\ 2.28e\text{-}6 \end{bmatrix}$
	Principal Axis	$\begin{bmatrix} 0.0786 \\ 0.1730 \\ 0.2080 \end{bmatrix}$	$\begin{bmatrix} 0.0018 \\ 0.0019 \\ 0.0009 \end{bmatrix}$	$\begin{bmatrix} 3.40e\text{-}7 \\ 2.26e\text{-}6 \\ 2.59e\text{-}7 \\ 2.28e\text{-}6 \end{bmatrix}$
SMC	Body Frame	$\begin{bmatrix} 0.0451 \\ 0.1190 \\ 0.1930 \end{bmatrix}$	$\begin{bmatrix} 0.0011 \\ 0.0019 \\ 0.0022 \end{bmatrix}$	$\begin{bmatrix} 4.91e\text{-}7 \\ 2.34e\text{-}6 \\ 5.74e\text{-}7 \\ 2.30e\text{-}6 \end{bmatrix}$
	Principal Axis	$\begin{bmatrix} 0.0780 \\ 0.1110 \\ 0.1870 \end{bmatrix}$	$\begin{bmatrix} 0.0013 \\ 0.0013 \\ 0.0011 \end{bmatrix}$	$\begin{bmatrix} 3.47e\text{-}7 \\ 2.25e\text{-}6 \\ 2.97e\text{-}7 \\ 2.27e\text{-}6 \end{bmatrix}$
MPC	Body Frame	$\begin{bmatrix} 0.0456 \\ 0.0479 \\ 0.0705 \end{bmatrix}$	$\begin{bmatrix} 0.0040 \\ 0.0035 \\ 0.0011 \end{bmatrix}$	$\begin{bmatrix} 1.03e\text{-}6 \\ 3.05e\text{-}6 \\ 3.71e\text{-}6 \\ 2.91e\text{-}5 \end{bmatrix}$
	Principal Axis	$\begin{bmatrix} 0.0477 \\ 0.0161 \\ 0.0194 \end{bmatrix}$	$\begin{bmatrix} 0.0014 \\ 0.0016 \\ 0.0027 \end{bmatrix}$	$\begin{bmatrix} 1.09e\text{-}5 \\ 1.47e\text{-}5 \\ 2.36e\text{-}5 \\ 1.66e\text{-}5 \end{bmatrix}$

Table 6.1: Results from various controllers following a Nadir-Pointing reference trajectory

In the following figures, the Euler error and torque provided by the reaction wheels over part of the experiments are shown.

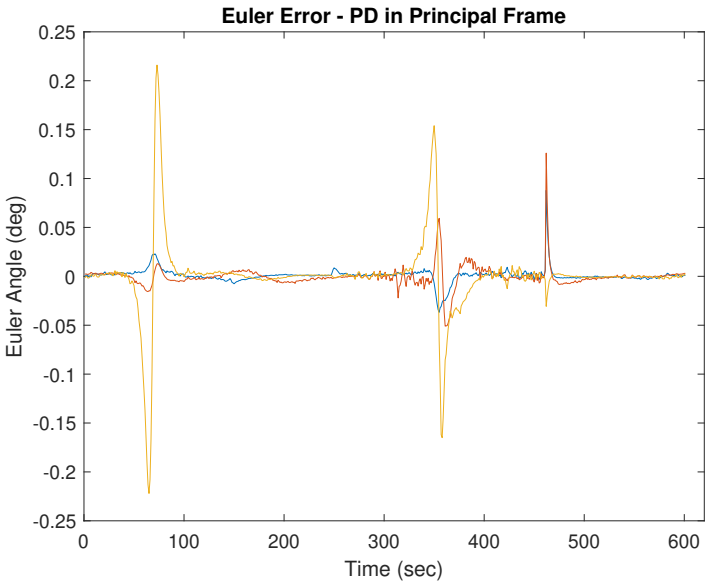

Figure 6.1: Euler angle error during nadir pointing maneuver using the PD controller in principal axis frame

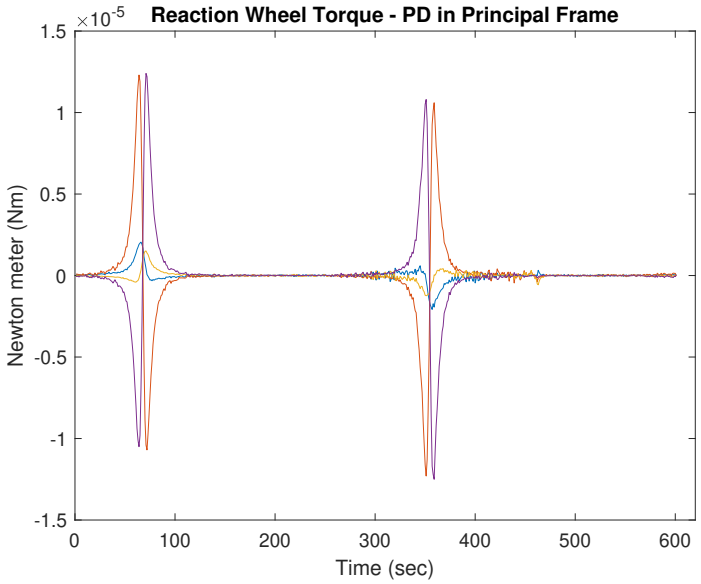

Figure 6.2: Reaction wheel torque during nadir pointing maneuver using the PD controller in principal axis frame

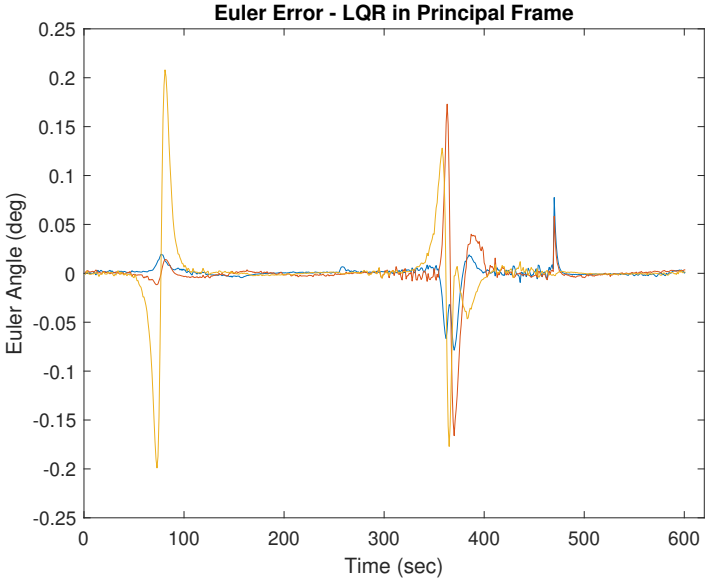

Figure 6.3: Euler angle error during nadir pointing maneuver using the LQR controller in principal axis frame

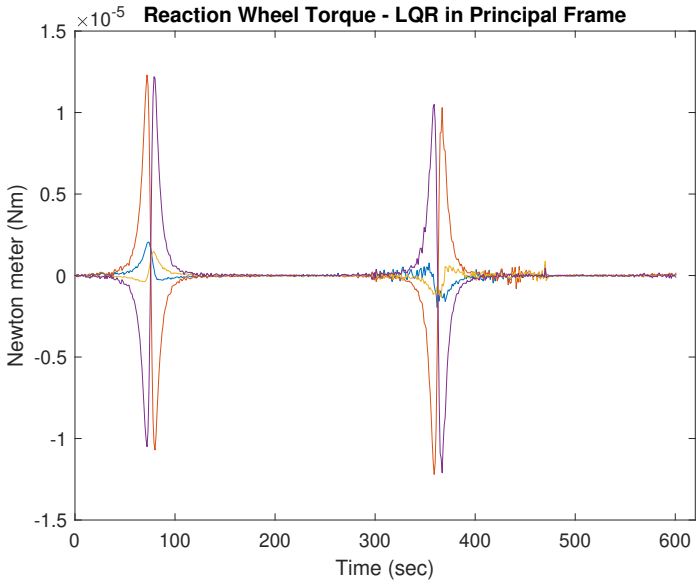

Figure 6.4: Reaction wheel torque during nadir pointing maneuver using the LQR controller in principal axis frame

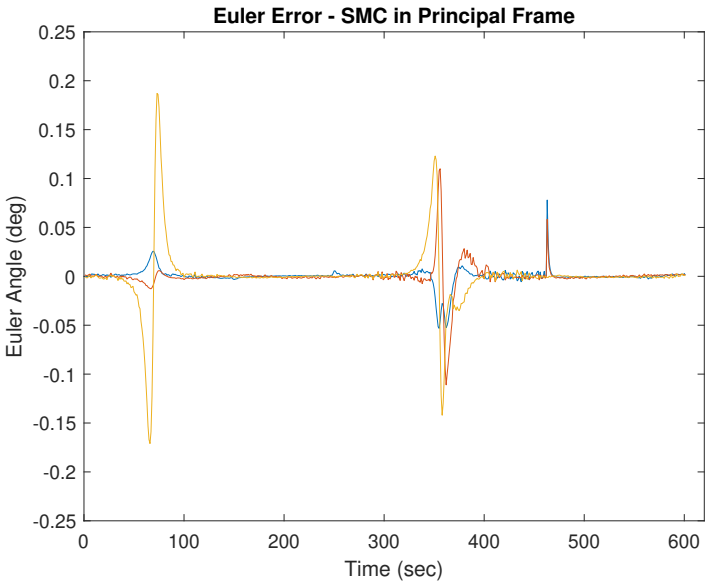

Figure 6.5: Euler angle error during nadir pointing maneuver using the SMC controller in principal axis frame

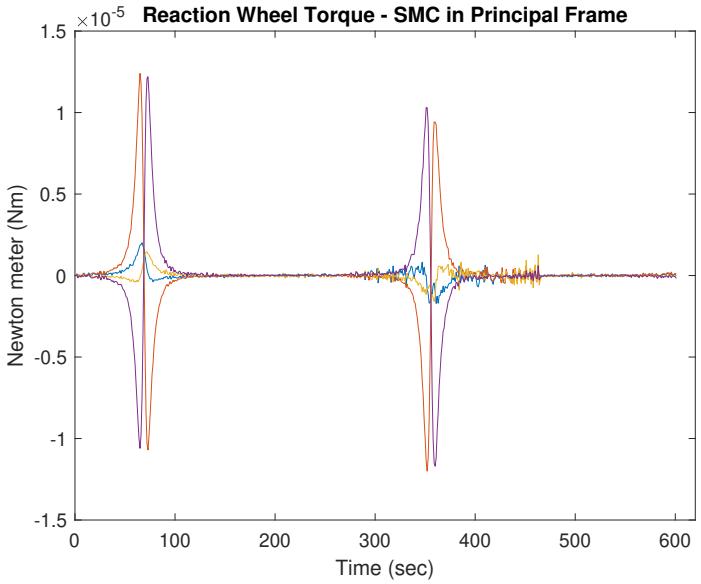

Figure 6.6: Reaction wheel torque during nadir pointing maneuver using the SMC controller in principal axis frame

Note that the y-axis limits are different in Fig. 6.8.

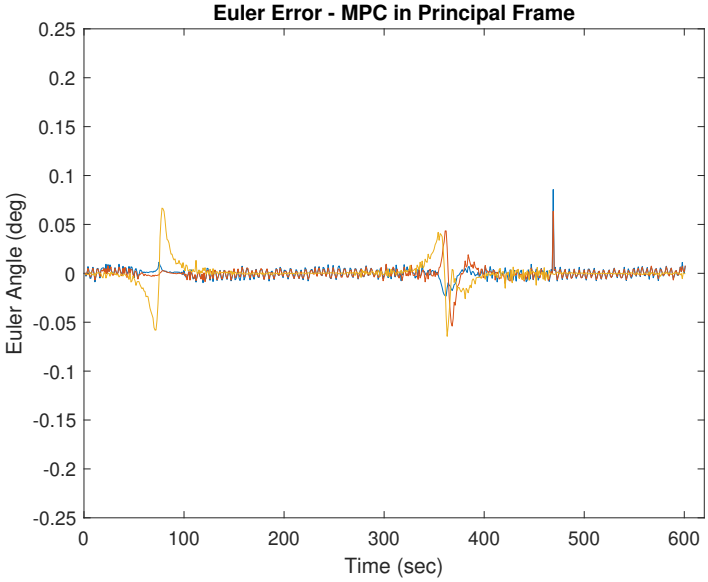

Figure 6.7: Euler angle error during nadir pointing maneuver using the MPC controller in principal axis frame

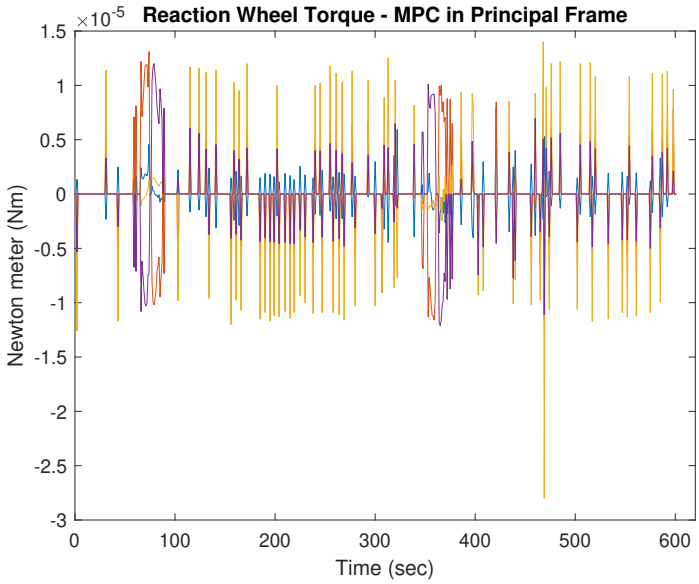

Figure 6.8: Reaction wheel torque during nadir pointing maneuver using the MPC controller in principal axis frame

6.2 Landmark-Pointing

In Table 6.2, the results of the experiments conducted in the landmark-switching scenario with a step reference are presented. Here, the metrics of interest are the settling time, steady state error, and control effort.

Controller	Controller Frame	Settling Time (sec)	Steady State Euler Error (deg)	Control Effort (MSE)
PID	Body Frame	92	$\begin{bmatrix} 0.0031 \\ 0.0073 \\ 0.0037 \end{bmatrix}$	$\begin{bmatrix} 4.21e\text{-}4 \\ 5.37e\text{-}5 \\ 4.41e\text{-}4 \\ 2.44e\text{-}6 \end{bmatrix}$
	Principal Axis	97	$\begin{bmatrix} 0.0407 \\ 0.0123 \\ 0.0050 \end{bmatrix}$	$\begin{bmatrix} 4.339e\text{-}4 \\ 2.39e\text{-}4 \\ 4.23e\text{-}4 \\ 4.82e\text{-}4 \end{bmatrix}$
LQR	Body Frame	99	$\begin{bmatrix} 0.0167 \\ 0.0349 \\ 0.0391 \end{bmatrix}$	$\begin{bmatrix} 4.48e\text{-}4 \\ 1.36e\text{-}4 \\ 4.36e\text{-}4 \\ 3.15e\text{-}4 \end{bmatrix}$
	Principal Axis	102	$\begin{bmatrix} 0.2000 \\ 0.0111 \\ 0.0532 \end{bmatrix}$	$\begin{bmatrix} 4.81e\text{-}4 \\ 3.03e\text{-}4 \\ 4.21e\text{-}4 \\ 5.02e\text{-}6 \end{bmatrix}$
SMC	Body Frame	59	$\begin{bmatrix} 0.0578 \\ 0.0006 \\ 0.0099 \end{bmatrix}$	$\begin{bmatrix} 3.69e\text{-}4 \\ 2.26e\text{-}5 \\ 3.75e\text{-}4 \\ 1.83e\text{-}4 \end{bmatrix}$
	Principal Axis	57	$\begin{bmatrix} 0.1600 \\ 0.0098 \\ 0.0204 \end{bmatrix}$	$\begin{bmatrix} 4.13e\text{-}4 \\ 2.33e\text{-}4 \\ 5.21e\text{-}4 \\ 3.95e\text{-}6 \end{bmatrix}$
MPC	Body Frame	21	$\begin{bmatrix} 0.0185 \\ 0.0047 \\ 0.0036 \end{bmatrix}$	$\begin{bmatrix} 8.34e\text{-}4 \\ 1.39e\text{-}4 \\ 9.02e\text{-}4 \\ 6.46e\text{-}4 \end{bmatrix}$
	Principal Axis	21	$\begin{bmatrix} 0.1110 \\ 0.0151 \\ 0.0474 \end{bmatrix}$	$\begin{bmatrix} 1.2e\text{-}3 \\ 8.89e\text{-}4 \\ 8.84e\text{-}4 \\ 1.1e\text{-}3 \end{bmatrix}$

Table 6.2: Results from the Step Response of the system

In the following figures, the Euler errors of all the experiments are shown.

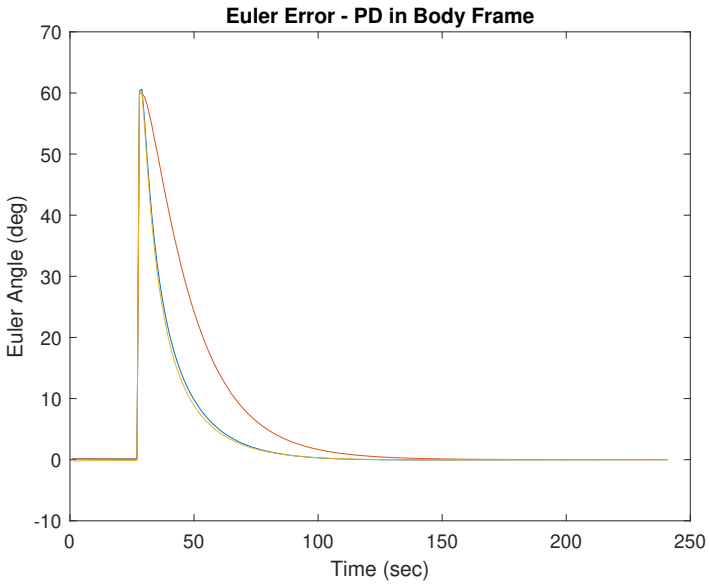

Figure 6.9: Euler angle error during step reference (landmark-switching maneuver) using the PD controller in body axis frame

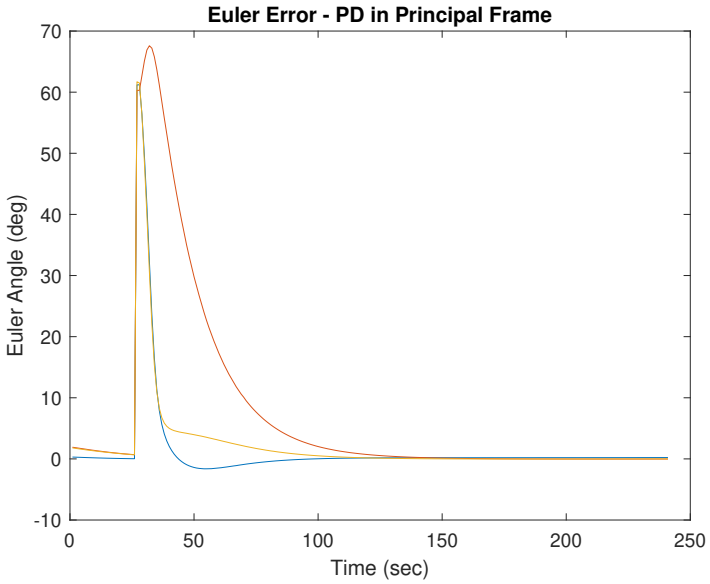

Figure 6.10: Euler angle error during step reference (landmark-switching maneuver) using the PD controller in principal axis frame

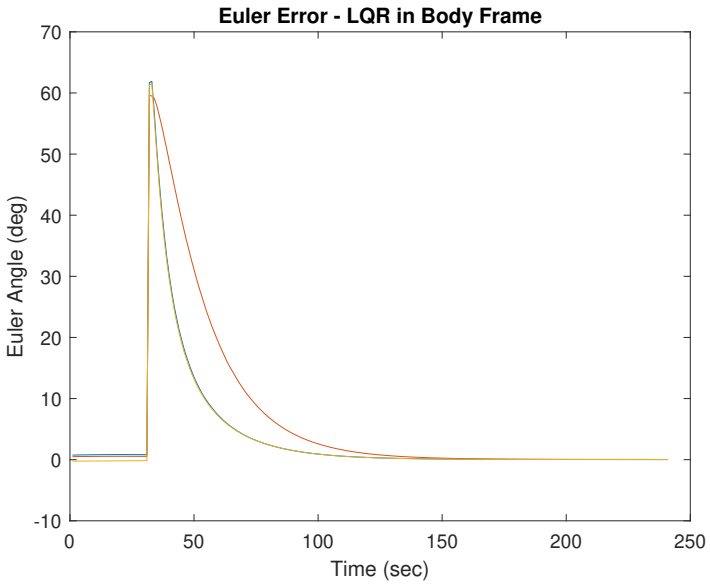

Figure 6.11: Euler angle error during step reference (landmark-switching maneuver) using the LQR controller in body frame

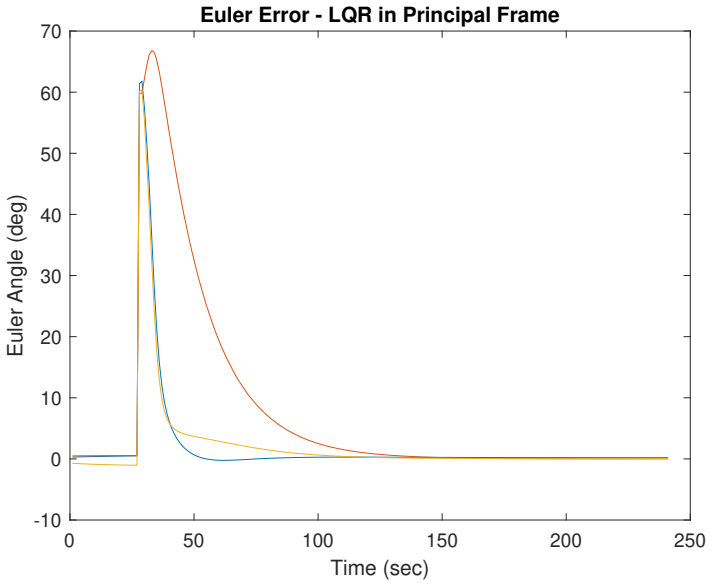

Figure 6.12: Euler angle error during step reference (landmark-switching maneuver) using the LQR controller in principle axis frame

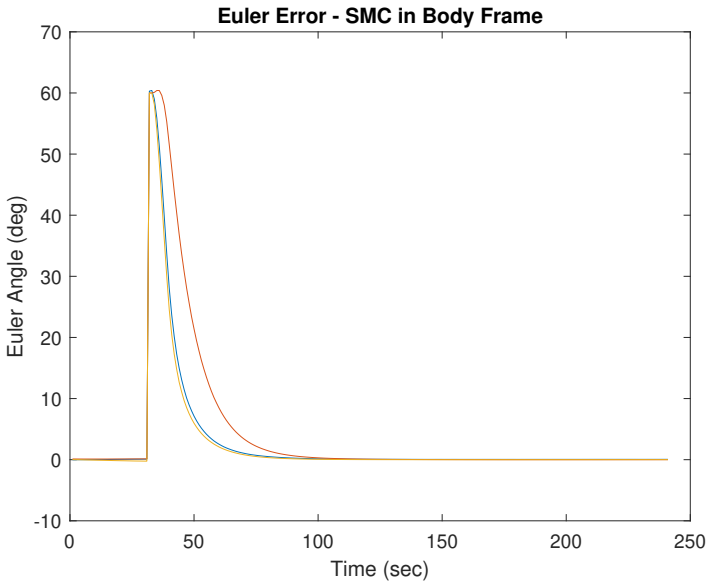

Figure 6.13: Euler angle error during step reference (landmark-switching maneuver) using the SMC controller in body frame

Note that the y-axis limits are different in Fig. 6.14.

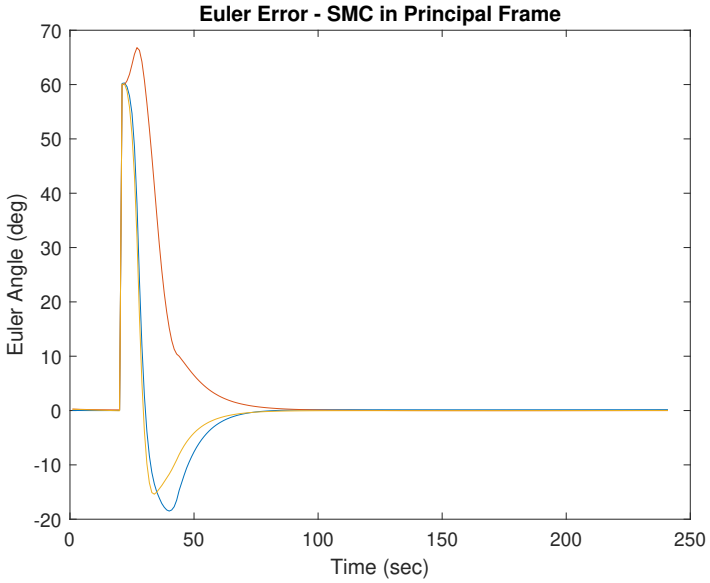

Figure 6.14: Euler angle error during step reference (landmark-switching maneuver) using the SMC controller in principal axis frame

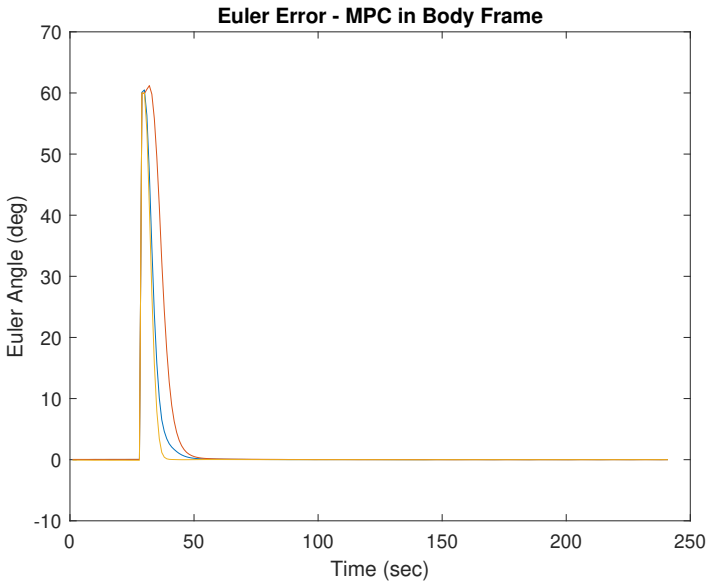

Figure 6.15: Euler angle error during step reference (landmark-switching maneuver) using the MPC controller in body frame

Note that the y-axis limits are different in Fig. 6.16.

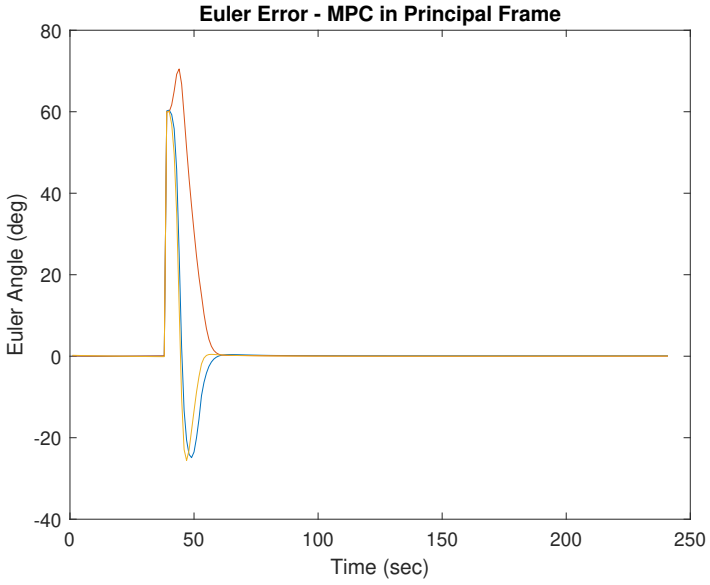

Figure 6.16: Euler angle error during step reference (landmark-switching maneuver) using the MPC controller in principal axis frame

Chapter 7

Discussion and Conclusion

In this chapter, the results from Chapter 6 are discussed and a conclusion of the work done during the project is presented.

7.1 Discussion

In this thesis, PD, LQR, SMC, and MPC were implemented and tested for their pointing accuracy and response to large-angle maneuver (step reference) in a satellite with non-uniform mass distribution. The controllers were tested in both the body frame and the principal axis frame.

It is noted that in the tracking case, there is a slight but notable improvement in the pointing accuracy of the linear controllers when the control problem is transformed to the principal axis frame, instead of running it directly in the body frame. It can be inferred that this improvement will be even more drastic as the mass distribution of the satellite becomes more non-uniform. On the other hand, the non-linear SMC experienced no such performance improvements because of the aforementioned frame transformations, which could be attributed to its robustness to uncertainties

In the nadir-pointing scenario, the LQR controller in the principal axis frame had the best pointing accuracy of around 10 arcseconds, but almost all controllers performed similarly and the minuscule differences in performance can be attributed to the individual tuning of the controllers.

In the landmark-switching scenario, MPC was the fastest to settle but again this is most probably due to the slightly more aggressive tuning of the controller which can be seen in its increased control effort. The SMC performed better than PD and LQR, with a similar control effort, showcasing its use for large-angle maneuvers.
Additionally, in MPC a lot of chattering was observed in the control signal as seen in fig 6.8 which was relected on the quaternion attitude, specifically in the nadir pointing test case. The cause of this chattering is unclear as it could not be over-

come by tuning the cost matrix. One hypothesis could be the lack of reference propagation within the prediction horizon of the controller, another could be a bug in the implementation.

In the future, it would be interesting to explore implementing a cost function for MPC, that includes the full reference trajectory, instead of only propagating the current quaternion error. This would allow the controller to take into account future set points, giving it the ability to plan maneuvers ahead of time, while optimizing the trajectory for actuator efficiency and performance. This could improve the ability of the system to deal with complex maneuvers, disturbances and slow system dynamics. Inspiration can be gathered from [22], where a full-order quaternion model is made controllable by over-actuation with magnetic torques and reaction wheels, and a cost function with the full reference trajectory included. In [23], a tube-based MPC approach with reference trajectory is used to control an under actuated system.

7.2 Conclusion

In this work, the control of a spacecraft with non-uniform mass distribution using a range of control paradigms has been investigated. Linear and non-linear control methods were implemented for an attitude control system. The experiment and implementation were done in simulation, where a satellite in low earth orbit, containing reaction wheels, was modeled. The purpose of the experiments was to evaluate the performance of the controllers in different mission scenarios. This was done using metrics such as maximum deviation, steady state error, settling time and control effort.

The results show that for satellites with non-uniform mass distribution, linear control problem should be transformed to the principal axis frame to increase the stability and performance of the attitude control system.
The results also validate the continued prominence of PID and LQR controllers within the industry. Despite their simplicity, they can match the performance of more complex control schemes, such as SMC and MPC, with the caveat that the implemented MPC is not truly predictive due to its cost function.

Bibliography

[1] Space Telescope Science Institute. *JWST Pointing Performance*. 2023. URL: https://jwst-docs.stsci.edu/jwst-observatory-characteristics/jwst-pointing-performance (visited on May 30, 2023).

[2] F Landis Markley. "Multiplicative vs. additive filtering for spacecraft attitude determination". In: *Dynamics and Control of Systems and Structures in Space* 467-474 (2004), p. 48.

[3] Space Inventor. 2023. URL: https://space-inventor.com/about/ (visited on May 30, 2023).

[4] Evan G Hemingway and Oliver M O'Reilly. "Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments". In: *Multibody System Dynamics* 44 (2018), pp. 31–56.

[5] F. Landis Markley and John L Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. eng. Vol. 33. Space Technology Library. New York, NY: Springer, 2014. ISBN: 9781493908011.

[6] Michael Tolstoj. "Analysis of disturbance torques on satellites in low-earth orbit based upon grace". PhD thesis. RWTH Aachen, 2017.

[7] NASA. *Dawn Observing Ceres; 3rd Reaction Wheel Malfunctions*. 2023. URL: https://www.nasa.gov/feature/jpl/dawn-observing-ceres-3rd-reaction-wheel-malfunctions (visited on June 1, 2023).

[8] F Landis Markley, Reid G Reynolds, Frank X Liu, et al. "Maximum torque and momentum envelopes for reaction wheel arrays". In: *Journal of Guidance, Control, and Dynamics* 33.5 (2010), pp. 1606–1614.

[9] Yaguang Yuan. "Quaternion-Based LQR Spacecraft Control Design Is a Robust Pole Assignment Design". In: *Journal of Aerospace Engineering* 27.1 (2014), pp. 282–284. DOI: 10.1061/(ASCE)AS.1943-5525.0000232. URL: https://ascelibrary.org/doi/abs/10.1061/\%28ASCE\%29AS.1943-5525.0000232.

[10] Bong Wie and Peter M. Barba. "Quaternion feedback for spacecraft large angle maneuvers". In: *Journal of Guidance, Control, and Dynamics* 3 (1985).

[11] Y-P Chen and S-C Lo. "Sliding-mode controller design for spacecraft attitude tracking maneuvers". In: *IEEE transactions on aerospace and electronic systems* 29.4 (1993), pp. 1328–1333.

[12] F. Landis Markley John L. Crassidis Srinivas R. Vadali. "Optimal Variable-Structure Control Tracking of Spacecraft Maneuvers". In: *Journal of Guidance, Control, and Dynamics* 3 (2000).

[13] Yaguang Yang. *Spacecraft Modeling, Attitude Determination, and Control - Quaternion-based Approach*. Crc Press, 2019. ISBN: 9780429822148.

[14] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. eng. Revised. Springer New York, 2006. ISBN: 0387227423.

[15] Eddie Wadbro. *Quadratic programs and Active set methods*. 2016. URL: https://people.cs.umu.se/eddiew/optpde2016/QP.pdf (visited on May 2, 2023).

[16] Rasmus Skjelsager Siddharth Pathania Danny Dibbern. "High-Precision Attitude Estimation for Spacecraft". In: *AAU Project Library* (2022).

[17] Malcolm D Shuster. "Kalman filtering of spacecraft attitude and the QUEST model". In: *Journal of the Astronautical Sciences* 38 (1990), pp. 377–393.

[18] R.L. Farrenkopf. "Analytic Steady-State Accuracy Solutions for Two Common Spacecraft Attitude Estimators". In: *Journal of Guidance and Control* 1.4 (1978), pp. 282–284. DOI: 10.2514/3.55779. eprint: https://arc.aiaa.org/doi/pdf/10.2514/3.55779. URL: https://arc.aiaa.org/doi/abs/10.2514/3.55779.

[19] *YALMIP*. 2022. URL: https://yalmip.github.io/ (visited on May 13, 2022).

[20] *MOSEK*. 2022. URL: https://www.mosek.com/ (visited on May 13, 2022).

[21] *Arduino Constrained MPC Library*. 2020. URL: https://github.com/pronenewbits/Arduino_Constrained_MPC_Library (visited on May 14, 2023).

[22] Raffaele Mozzillo Sabrina Corpino Loris Franchi Lorenzo Feruglio. "Model predictive and reallocation problem for CubeSat fault recovery and attitude control". In: *Mechanical Systems and Signal Processing* (2018).

[23] M. Khosrojerdi M. Mirshams. "Attitude control of an underactuated spacecraft using quaternion feedback regulator and tube-based MPC". In: *Acta Astronautica* (2017).