Simulating Satellite Attitude Control Algorithms in Python

A.J Matty

August 2024

Abstract

This report explores the simulation and performance of two satellite control algorithms, Proportional-Integral-Derivative (PID) and Model Predictive Control (MPC), using Python for the Lunar Reconnaissance Orbiter (LRO) satellite dynamics. The simulations produced have been constructed from scratch in Python and its associated libraries. Satellite attitude control is vital for every spacecraft's mission, along with the current missions planned for modern lunar exploration, such as NASA's Artemis mission [21]. The constructed simulations use three distinct scenarios to test the attitude controllers' performance: firstly, an undisturbed rotation to model the satellite pointing its instruments in a new direction; secondly, a stationary manoeuvre with an external disturbance torque; lastly, a tumbling scenario to test the stabilisation capabilities of the controllers. The results of this study found MPC to produce the fastest results; however, this came with two caveats. Primarily, the MPC controller designed in this report did not respond to external disturbances as well in comparison to the PID controller. The MPC algorithm is also more computationally intensive; however, this can be overcome using modern technology such as FPGA [56].

Contents

List of Figures

List of Tables

1 Introduction

1.1 History and Overview

Spacecraft attitude control is a fundamental part of space exploration and satellite operations. To ensure the spacecraft maintains or advances to the desired orientation in space. This ability is vital for a wide array of functions. From the precise directing of scientific instruments to maintaining solar panel orientation. Communications antennas on spacecraft are essential for transmitting and receiving signals. This is crucial for several services around the world such as television and internet connectivity. Directional antennas are often used for large data communications. These require precise attitude control to ensure the antenna aligns with their intended target. making sure a spacecraft has a robust and stable attitude control system is therefore mission critical. Without any attitude control spacecrafts and satellites would be unable to perform their intended mission. The history of spacecraft attitude control reflects significant advancements in technology and computing. These developments have enhanced our capacity for human space exploration.

The early developments of spacecraft attitude control came about along with the progression of the first space missions. Beginning in the early 20^{th} century as the United States and the Soviet Union entered into a space race, the need for precise control of spacecraft orientation became evident. The first artificial satellite, Sputnik 1, launched by the Soviet Union in 1957 did not require complex attitude control systems since their primary mission was to broadcast radio signals and demonstrate the feasibility of space travel. The first examples of attitude stabilisation and controller are spin stabilisation and magnetic stabilisation. The spin stabilisation technique uses the conservation of angular

momentum to provide stability for satellites, the body spins around one its principal axis and maintains the direction of this axis unless acted on by an external torque [1]. The disadvantage of this method is the limited ability to change its orientation quickly and accurately while spinning. The other passive technique is magnetic stabilisation that uses permanent magnets within the vehicle to align it with the magnetic field of the earth [2]. When the magnet onboard the satellite interacts with the Earth's B-field it experiences a torque that passively controls the attitude of the spacecraft.

In the almost seventy years since the first artificial satellite, there have been significant advancements in systems used to control the attitude of satellites. Modern systems have transitioned from passive to sophisticated active control techniques. These advancements have been enabled by the development of accurate sensors and devices which record the attitude and angular velocity of the spacecraft. One device is a miniature inertial measurement unit (MIMU). The MIMU uses a combination of accelerometers and gyroscopes to determine the angular velocity of a satellite [3]. These attitude determination systems are combined with modern control methods such as reaction wheels, control moment gyroscopes and magnetorquers to form a complete control system for the spacecraft. Reaction wheels and control moment gyroscopes both use the same principle of conservation of angular momentum as the passive spin stabilisation technique. However, they both leverage this principle to actively control the attitude of a spacecraft by spinning weighted wheels to induce a torque perpendicular to the plane of rotation. Reaction wheel systems use multiple wheels aligned in different directions. By adjusting the speed of these wheels a spacecraft achieves precise control over its orientation. The other technique using the conservation of angular momentum, control moment gyroscopes, only uses a single wheel mounted on gimbals that allow the wheel's axis to be rotated. Through the control of

these gimbals the wheel's resulting torque can be directed to manage the vehicle's orientation. The final commonly used control method is the use of magnetorquers. These are electromagnetic coils which create a magnetic dipole moment. This magnetic moment interacts with the Earth's B-field, inducing a control torque on the spacecraft similar to the passive magnetic stabilisation. Magnetorquers are strategically placed and actuated to actively control a satellite's attitude, the orbital parameters have an important effect on the torques produced [4]. These magnetorquers are most commonly used for satellites in low earth orbit due to the Earth's magnetic field strength. The final piece to a complete control system is a robust control algorithm. This determines the proper torques necessary to govern a spacecraft's attitude.

At the time of writing the industry of space is growing rapidly. Most recently SpaceX has been tasked with decommissioning the International Space Station (ISS) [5]. With this $843 million contract to build a vehicle to deorbit the ISS comes many challenges. In which robust attitude control of the entire operation will be vital. Along with this mission will come equally complex ones. Including the human race returning to the moon. Hence efficient control algorithms will be crucial.

1.2 Controllers

Control algorithms are the computational procedures designed to produce the desired outputs for the dynamics of a system. By mathematically manipulating system inputs based on measured and desired outputs, control algorithms enable precise and efficient operation of many kinds of systems. The first formalised Proportional Integral Derivative (PID) control law was published in 1922 by Nicolas Minorsky [6]. His article titled Directional Stability of Automatically Steered Bodies studied and designed an automatic

ship steering system for the United States navy. The Model Predictive Control (MPC) algorithms beginning can be traced back to the 1960s with work produced by Zadeh and Whalen (1962) and Propoi (1963) [7] [8]. Zahed and Whalen recognized the link between the minimum optimum control problem and Linear Programming. They identified that certain control problems, which aim to find the best control action to minimise a cost function over time, could be solved using Linear Programming techniques. Propoi proposed the idea that is at the core of the Model Predictive Control algorithm, a moving horizon approach. MPC gained popularity in the 1980s with its extensive use in the petrochemical industry. Using the algorithm to optimise many processes to make it as profitable as possible and is still used in the industry today [9].

1.2.1 Proportional Integral Derivative Control

PID controllers are among the oldest and most used control algorithms. It is known for their simplicity and ease of implementation making it versatile for many applications. The controller operates by calculating an error value as the difference between a desired outcome and a measured process variable that is changing. It then produces a correctional output based on proportional, integral, and derivative terms, which are tuned to achieve optimal system performance using constants known as PID gains.

- Proportional (P): This component of the controller produces an output that is proportional to the current error value.
- Integral (I): This component accounts for the accumulation of past errors, it aims to eliminate residual steady-state errors that a proportional controller cannot address.
- Derivative (D): This component uses the rate of the change in the errors to dampen

the effects of the controller.

1.2.2 Model Predictive Control

The more sophisticated approach in contrast to the PID controller is model predictive control. MPC uses a dynamic model of the system to predict the future behaviour of the system over a specific horizon as written about in Propoi (1963) [8]. Using this information of the future system to produce system outputs to reach the desired goal. The key features of MPC include:

- Prediction Modelling: The controller predicts the future behaviour of the system using a variable model.
- Optimisation Problem: At each time step the controller solves an optimisation function to minimise a cost function which is subject to constraints.
- Result: The control signal that best minimises the cost function is taken as the output.

These features combined create an advanced control algorithm that is used in many industries today. Specifically, MPC's are used in optimising chemical processes in the petrochemical industry and enhancing trajectory planning and stability control in autonomous vehicles [10]. In comparison the PID controller does not have the predictive ability for the system only relying on the system errors and their rate of change to anticipate the future behaviour

1.3 Thesis Objectives

This report aims to simulate and investigate the attitude control capabilities of two control algorithms, PID control and MPC, through python simulations. The spacecraft model used in this report will be one based on the Lunar Reconnaissance Orbiter (LRO) satellite. The objectives of this project are outlines as:

1. Develop a valid spacecraft attitude dynamics model.
    - Mathematically model the LRO's mission parameters and dynamics.
    - Approximate and model the spacecraft's external disturbances.
    - Employ quaternions to characterise the spacecraft's attitude to avoid the issues involved in the use of Euler angles (Gimbal lock).
2. Construct a Python program to simulate the PID and MPC algorithms to control the spacecraft's attitude.
    - Implement the spacecraft dynamics models with the control algorithms in python.
    - Study the effectiveness of the two controllers through numerical simulations.
3. Analyse the results of the simulations to compare the two controllers.
4. Test the computational complexity and efficiency of each algorithm.

The LRO was launched in 2009 by NASA, its primary mission is to map the moon's surface in detail [11]. The mission was the first mission to be sent to the moon by NASA in the 21st century and through the use of its scientific instruments has produced many insights and discoveries. The LRO is equipped many instruments such as the Lunar

Reconnaissance Orbiter Camera (LROC) for high resolution imaging and the Cosmic Ray Telescope for the Effects of Radiation (CRaTER) to study lunar radiation [12].The LRO identified ice deposits in shaded regions of the lunar poles, which may be crucial for supporting future human lunar exploration, as published in 2018 [13]. The Lunar Reconnaissance Orbiter Camera has played a key role in selecting suitable landing sites on the lunar surface for future missions [14]. The need for the scientific instruments to point at precise coordinates justifies the need for a strong control algorithm to handle the spacecraft manoeuvres and disturbances.

Figure 1: Artist concept of NASA's Lunar Reconnaissance Orbiter (Image credit: NASA).

The LRO uses Reaction Wheels and Thrusters as control techniques, it does not use Magnetorquers as the moon's magnetic field is too weak and irregular to produce suitable control torques for the satellite [15]. The LRO has a control mode known as 'Observing Mode' where the satellite points at lunar nadir and can reorient the craft to capture images from various angles and perspectives [16]. Satellites can experience different external disturbances such as gravitational torque and solar radiation pressure which can exert a small force on the craft affecting its attitude. The Lunar orbital environment has a higher flux of small meteorites known as micrometeoroids compared to earth's orbit [17]. These

micrometeoroids can impact the LRO and affect the attitude and angular velocity of the satellite, where a control algorithm is required to stabilise the spacecraft in order to regain control. The project will use these cases to model the dynamics of the satellites and test the control algorithms. By simulating these control algorithms for the model of the LRO this dissertation aims to provide insight on control algorithms for future missions.

1.4 Literature Review

This literature review aims to provide context of the current and future lunar missions along with the potential disturbances these missions may experience. This subsection will also discuss research on MPC and PID controllers and their optimisation techniques.

1.4.1 Lunar Mission Landscape

Lunar exploration has seen a revival in the recent decade with the operation of several active lunar orbiters and ambitious plans for missions in the future [18]. Along with NASA's LRO there are other notable orbiters such as the CNSA's Tiandu-1 and Tiandu-2 satellites which were launched in 2024 as part of the development of the International Lunar Research Station (ILRS) [19][20]. These two satellites are in circumlunar orbit with the design to test communication technology for earth to moon communication. These tests are important for the design of China's proposed Queqiao lunar navigation and communication constellation which will provide the infrastructure for the future ILRS.

These future plans for Moon missions are propelled by the aim of establishing a sustainable human presence on the Moon. The lunar environment can provide a testing ground for the technologies and procedures required for martian exploration. Therefore

current and planned lunar missions are important steps towards more human cosmic exploration. NASA's Artemis program aims to return humans to the lunar surface by mid 2020s and has a long term vision of creating a lunar base on the surface [21]. The outcome of these missions rely greatly on a precise and dependable attitude control system, to make sure spacecrafts can navigate and conduct scientific experiments and successfully deliver payloads to the lunar surface. Looking back at the Apollo missions the lunar lander modules used a digital autopilot system that combined inertial guidance and radar data to control the descent and landing [22]. Famously on the first Apollo mission 11 Commander Neil Armstrong had to take manual control from the automated control system due to an unsafe landing area [23]. This event justified the need for accurate mapping of the lunar surface that the LRO has conducted to find the future Artemis missions landing sites. More modern control systems for lunar descent use more advanced control algorithms to change the spacecraft's attitude and thrust compensating for environmental disturbances to achieve a controlled landing on the Moon's surface [24].

1.4.2 External Disturbances

During their operations spacecraft are subject to different external disturbances that can affect their attitude and therefore consequently their mission performance. An example of these disturbances the LRO may experience include solar radiation pressure, gravitational gradient and physical collisions. Solar radiation pressure describes the force acted on a spacecraft's surface by incident and reflected photons emitted by the Sun [25]. The pressure exerted on a spacecraft is in the order of magnitude 1×10^{-6} pascals, the force experienced by a spacecraft depends on its orientation and surface area relative to the sun. Other factors can affect the force on a spacecraft such as the distance from the

sun and the reflective properties of the vehicle's surfaces. The more reflective the surface the larger the force experienced from solar radiation pressure [26]. The James Webb Space Telescope has a large and highly reflective sunshield that is designed to protect the telescope's instruments from the thermal effects of the sun [27]. Research has also been conducted by Li and Williams to investigate whether the sunshield can be utilised for reformations of a spacecraft taking advantage of the solar radiation pressure acting on the sunshield [28]. Solar radiation pressure depending on solar activity was discussed by Pratiwi and Herdiwijaya where the effect of an extreme geomagnetic storm on a satellite was investigated, they concluded that the change in solar radiation pressure was insignificant and the pressure remains constant [29]. Similar to SRP the lunar albedo can cause an external torque on satellites, this is due to the solar energy reflected off the lunar surface [30]. This force is lower than the SRP due to the reflectivity of the lunar surface. The second type of external forces that act on spacecraft is caused by gravity, in the case of the LRO there are three main celestial objects which affect the spacecraft; the Sun, Earth and the Moon. The gravitational forces of these three objects induce torques on the spacecraft which can change its attitude [31].

1.4.3 MPC and PID Controllers

As mentioned previously the PID controller is a simple and easy to implement control algorithm. One of the first studies on different control algorithms using control moment gyroscopes by Cochran et al, 1975, a PID controller was compared in this study as we aim to do in ours [32]. Cochran et al simulated a "Large Space Telescope" type spacecraft and results indicated that an adaptive control law is desirable [32]. More recent research conducted by Dong ,2023, compared an adaptive PID to a conventional PID controller for

a quadrotor Unmanned Aerial Vehicle [33]. The adaptive PID controller investigated in the research was a Fuzzy PID controller, Dong concluded that the controller was heavily dependent on the set of rules decided to control actions. This can be a disadvantage making the controller complex and unscalable to larger control problems although it produces faster results than the traditional PID algorithm.

The second control algorithm being simulated is traditional Model Predictive Control which has become a more advanced alternative to PID. MPC's has attracted interest in applications in the aerospace industry such as orbit tracking and attitude manoeuvres [34][35]. MPC has to model the dynamics of a system of the future finite horizon this model is either linear or nonlinear. Golzari et al,2020, designed a linear time varying MPC algorithm for quaternion based satellite attitude control, the simulation of this controller showed the method has a small computational cost and performed accurately [36]. In comparison nonlinear MPC has a high computational cost due to the more complex dynamics of the model as discussed in Maciej Ławryńczuk and Robert Nebeluk,2021, work on making nonlinear MPC more efficient using an alternative cost function [37].

Historically, the application of using MPC has been limited by its computational complexity used in simpler open loop systems such as the controllers used in the petrochemical industry [38]. The need to solve the optimisation problems in the controller as fast as possible is vital in satellite attitude controllers for the most accurate results. At the beginning of this century Bemporad et al ,2000, stated that MPC was only applicable to smaller or slower systems due to the computational power needed to solve the optimisation problem [39]. The development since the turn of the century of computational technology and its increase of power has allowed MPC to control systems that it would be previously unable to in real time [40]. In 2020 Zhang et al preposed an MPC with integration to offset a

large steady state error recorded by the satellite VELOX-II, launched in 2015, operations data [41]. As can be seen from the literature mention above MPC is becoming a more prominent and capable controller due to the advancements in computer processing power.

1.4.4 Controller Tuning

Optimising control algorithms is important to enhance the performance and efficiency of the system. In the case of the PID control algorithm parameter tuning of the gains is essential to achieve optimal results. The first method of tuning is the most intuitive and is known as manual tuning and uses a trial-and-error based procedure. This method works by manually changing and approximating the gains to produce an optimal PID loop to the tuner's desire. The oldest formalised tuning method is a heuristic one known as the Ziegler-Nichols method, introduced in 1942, it provides a more systematic approach when compared to the manual tuning method [42]. A more advanced technique is the Åström–Hägglund method, also known as the Relay Method which was first designed in 1984 [43]. The method involves estimating two parameters known as critical gain and critical period which then inform the normal PID gains for the controller. There are other types of tuning methods including variations of previous methods which are not mentioned in this report to optimise a PID controller.

Optimisation of a Model Predictive Controller involves more parameters than the PID controller which only needs three values to be tuned for an optimised controller. An optimised MPC involves selecting the appropriate model, prediction horizon, control horizon and solving the underlying optimisation problem efficiently. The MPC algorithm solves a convex optimisation problem, typical solvers are based on linear programming (LP) or quadratic programming (QP) [44]. Research has been conducted investigating a

faster and more efficient solving method as written by Milman and Davison, 2008, where they presented a new active set method to solve the QP problem associated with MPC which decreased the computation time [45]. Decreasing the solving time and choosing the correct parameters will optimise the MPC control to work quickly and effectively making it more available to faster changing systems.

2 Theory

The model used in the simulations is formulated as a Nonlinear time-invariant (NTI) system, which accurately describes the complex dynamics of satellite attitude motion. Unlike linear models, NTI models account for the inherent nonlinearities present in the system, providing a more realistic representation of satellite behavior. To model a physical system such as a space craft we will use state-space representation. This is a mathematical model which uses input, output and state variables along with first order differential equation to describe the behavior of a system. The state-space model for a general NTI system is represented by using state vectors and nonlinear functions typically expressed as,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2}$$

The state vector, denoted as $\mathbf{x}(t)$, defines the system's conditions at a given time t and $\dot{\mathbf{x}}(t)$ is the time derivative of this vector. External influences or control inputs are represented by the input vector $\mathbf{u}(t)$. The system's observable outputs are captured by

the output vector $\mathbf{y}(t)$. The system's dynamic behavior and output relationships are characterized by nonlinear functions $f(\cdot)$ and $g(\cdot)$, respectively. The time-invarince of our model is due to the properties of the model not changing over time.

2.1 Reference Frames

To define a spacecrafts attitude and dynamics relative to its environment characterising coordinate frames is essential. The two primary reference frames employed in this dissertation are the bode-fixed and the Local-Vertical-Local-Horizontal (LVLH) reference frames.

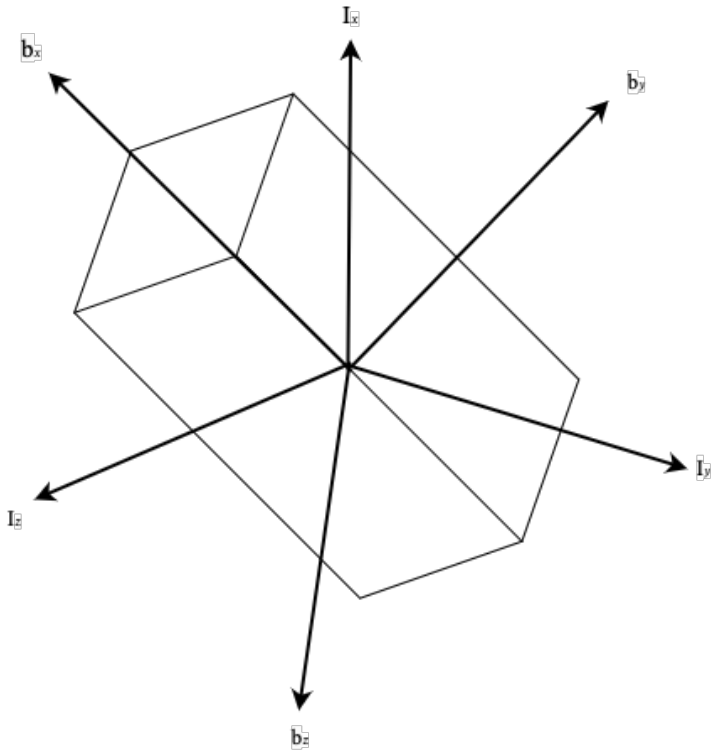

Figure 2: Orientation of Body-fixed and Local-Vertical-Local-Horizontal reference frames

2.1.1 Body-Fixed Reference Frame

The body-fixed reference frame B is a coordinate system that is strictly fixed to the centre of mass of the spacecraft. The frame moves and revolves with the satellite, the axes of the reference frame is aligned with the principal axes of inertia of the spacecraft. B has orthogonal components b_x , b_y , b_z as shown in the figure 2.

2.1.2 Local-Vertical-Local-Horizontal

The LVLH reference frame is defined relative to the satellite's orbit around a central body such as the moon as is the case for the LRO. This reference frame is important for orbit-centric operations and provides a stable base for understanding the satellite's attitude relative to the moon. The defining axes of the LVLH reference frame are Z_{LVLH}, Y_{LVLH} and X_{LVLH} that denote the local-vertical, local-horizontal and cross-track respectively (I_x,I_y,I_z as seen in figure 2). The Z_{LVLH} axis points towards the centre of the the moon whilst the Y_{LVLH} axis points perpendicular to the orbital plane. Finally X_{LVLH} completer the coordinate system aligning with the velocity vector of the satellite.

2.2 Kinematics

The kinematics of rotational motion can describe how the orientation of a spacecraft changes over times. There are a few ways to represent a vehicles attitude, one way is to use Euler angles. This is the most intuitive way of describing the orientation around three specified axis (pitch,roll and yaw) however can suffer from gimbal lock where two of the three axis align causing a loss of a degree of freedom [46]. Quaternions are a four-dimensional vectors to represent the orientation of a craft which are favourable as they

avoid the associated problems with Euler angles. In this report a quaternion number system is used to represent the LRO's kinematics.

2.2.1 Quaternion Kinematics

The quaternion is comparable to complex numbers, it consists of one scalar part and a 3-dimensional Vector (represented by i,j,k). Similar to how complex numbers can be used to describe 2-dimensional rotation, quaternions can describe 3-dimensional rotation. Quaternions are less intuitive and comes with its own challenges such as how commutative multiplication does not apply to them. We can represent the quaternion mathematically shown below.

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \mathbf{u}\sin\frac{\theta}{2} \end{bmatrix} \tag{3}$$

$$\mathbf{q} = \begin{bmatrix} \eta \\ \\ \epsilon \end{bmatrix} \tag{4}$$

Where q_0, η and $\cos\frac{\theta}{2}$ represents the scalar component and $q_{1,2,3}$, $\mathbf{u}\sin\frac{\theta}{2}$ along with ϵ denotes the vector parts of the quaternion. Such that θ is the angel of rotation and $\mathbf{u}$ is the unit vector along the axis of rotation. Furthermore the time t derivative of the quaternion kinematics can be given as,

$$\frac{d\mathbf{q}}{dt} = \frac{1}{2}\mathbf{q} \otimes \mathbf{q}_{dot} \tag{5}$$

$$\mathbf{q}_dot = \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{6}$$

where $\mathbf{q}_dot$ are the angular velocities ω, around the principle axes, expressed as a quaternion with a zero scalar part as seen in (4). The mathematical notation $\otimes$ is the Kronecker product which is used in quaternion multiplication. This multiplication is different to other multiplication as it is not commutative as stated previously. The product of two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ is given by:

$$\mathbf{q}_1 = w_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}$$

$$\mathbf{q}_2 = w_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}$$

The quaternion multiplication, using the Kronecker product, of $\mathbf{q}_1$ and $\mathbf{q}_2$ is computed as:

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{bmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2 \end{bmatrix} \tag{7}$$

2.3 Attitude Dynamics

The foundation for understanding the rotational motion of rigid bodies, such as spacecrafts, is informed by Euler's rotational equations. These equations are vital in the field of attitude dynamics as they describe how the angular velocities of a spacecraft change in response to applied torques. The following expressions in vector notation are derived from Newton's second law of motion,

$$\tau = \mathbf{I}\dot{\omega} + \omega \times (\mathbf{I}\omega) \tag{8}$$

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{9}$$

Where $\mathbf{I}$ is the moment of inertia matrix of the rigid body and ω is the angular velocities of the body. Modelling a spacecrafts mass to be symetrical around the principle axes would result in $\mathbf{I}$ being equal to a diagonal matrix with three moments of inertia as shown in (7). The final part of equation (6), τ, is the total torque vector acting on the spacecraft. This formula can be further expanded to give,

$$\tau_x = I\dot{\omega_x} + (I_z - I_y)\omega_y\omega_z$$

$$\tau_y = I\dot{\omega_y} + (I_x - I_z)\omega_z\omega_x$$

$$\tau_z = I\dot{\omega_z} + (I_y - I_x)\omega_x\omega_y$$

These equations are integral to the design of attitude control systems, allowing the prediction of how control inputs such as reaction wheel torques will affect the spacecrafts attitude.

2.4 External Disturbances

The resulting torques produced by external disturbances caused by solar radiation pressure and gravity gradients can be mathematically modelled. This section part looks at the order of magnitudes produced by the external disturbances which can be used to approximate there effect on a spacecraft.

2.4.1 Solar Radiation

The solar radiation pressure P_s and the resulting torques can be calculated from the solar luminosity $L_\odot$. The intensity of the suns energy S can be approximated at a distance d of 1Au.

$$S = \frac{L_\odot}{4\pi d^2} \tag{10}$$

$$P_s = \frac{S}{c} \tag{11}$$

The solar radiation pressure is calculated by dividing the solar intensity over the speed of light c this derived from Einstein's equations regarding mass and energy. The resulting torques caused by this pressure can be calculated as,

$$F_s = P_s \cdot A \tag{12}$$

$$\tau_s = \mathbf{r} \times \mathbf{F}_s \tag{13}$$

Where,

$$\mathbf{F}_s = F_s \cdot \hat{\mathbf{n}} \tag{14}$$

The force F_s acting on the spacecraft is directly proportional to the surface area A on which the solar radiation acts. The corresponding torque vector τ_s is determined using the cross product of the position vector $\mathbf{r}$ from the centre of mass to the point of the applied force and the vector form of the radiation force $\mathbf{F}_s$. This force vector can be computed using the unit normal vector $\hat{\mathbf{n}}$, which points in the direction the force is applied. The amplitude of these external torques due to solar radiation pressure can be approximated to be in the order of magnitude $10^{-4} N \cdot m$, which is small however not insignificant and can affect the attitude of the spacecraft.

2.4.2 Gravitational Gradients

Gravity gradient torque arises due to the variation in gravitational force acting on different parts of a satellite. This force difference occurs because gravity decreases with distance from the center of a celestial body. For an extended body like a satellite, this means the force of gravity acting on the part of the satellite closer to the Earth is slightly

stronger than the force acting on the farther side. The result is a torque that tends to align the satellite with its largest moment of inertia axis pointing towards the center of the gravitational field. This phenomenon can be sometimes used to passively control a satellites attitude called gravity gradient stabilisation [47]. The gravity gradient torque τ_{gg} acting on a spacecraft can be mathematically modelled using the equation,

$$\tau_{gg} = \frac{3\mu}{R^3}(\mathbf{r}_{cm} \times (\mathbf{I} \cdot \mathbf{r}_{cm})) \tag{15}$$

The gravity gradient torque acting on a satellite orbiting the Moon is governed by the gravitational parameter μ, which is defined as $\mu = GM$, where G represents the gravitational constant and M is the mass of the celestial body. This parameter quantifies the strength of the objects gravitational field. The torque is influenced by the distance R from the objects center to the satellites centre of mass and the position vector $\mathbf{r}_{cm}$ that points to the to the same thing as the distance. The satellites inertia matrix plays a vital role in determine the magnitude and direction of the torque based on the crafts characteristics.

2.5 Controller Mathematics

This section explains the mathematical functions used by the controller to produce control outputs which dictate the torque applied to the spacecraft. The two controllers both use the current state of the system to inform its output. The NTI model for the quaternion-based attitude dynamics system uses the state space model defined above. Where the state vector can be represented by,

$$\mathbf{x} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{16}$$

The control input vector $\mathbf{u}(t)$ described in the NTI model is the equivalent to the total torque applied to the spacecraft τ as described in the attitude dynamics section. Hence showing the nonlinear nature of the system due to the quaternion kinematics and the rotational dynamics of Euler's equations.

2.5.1 Proportional, Integral and Derivative Control

As mentioned in the first chapter PID control is a relatively simple and powerful control method. The controller calculates an error value as the difference between the desired setpoint and a measured process variable. In the case of an attitude controller it can either be the current attitude or the crafts angular velocity depending on the controllers function. This information is used in proportional, integral, and derivative terms to determine a correction output to apply to a system.

$$P(t) = K_p e(t) \tag{17}$$

$$I(t) = K_i \int_0^t e(T)dT \tag{18}$$

$$D(t) = K_d \frac{de(t)}{dt} \tag{19}$$

Equation (8) represents the proportional term of the PID controller which produces an output P_t proportional to the current error value of the system. The term $e(t)$ in all the formulae above is the error value and can be calculated as $e(t) = x_{ref} - x(t)$, where x_{ref} is the set point and $x(t)$ is the current state of the system. The response of the proportional term can be adjusted by multiplying $e(t)$ with the constant K_p known as the proportional gain. The subsequent equation describes the integral terms output $I(t)$ which is also affected by a gain constant, the integral gain K_i. The integral term is used to eliminate the steady-state error by integrating the error over time. The final equation describes the derivative term which uses the rate of change of the error, $\frac{de(t)}{dt}$, to apply a damping effect. This effect is controlled by the derivative gain K_d with the aim of preventing the system overshooting its target.

$$u(t) = K_p e(t) + K_i \int_0^t e(T)dT + K_d \frac{de(t)}{dt} \tag{20}$$

Through the combination of these three terms the control signal $u(t)$ is calculated, giving the general formula for the PID controller. This calculation can be visualised clearly as a block diagram shown in figure 3. The proportional K_p, integral K_p and derivative K_p gains play a pivotal role in the performance of the controller, making proper tuning of the gains important.

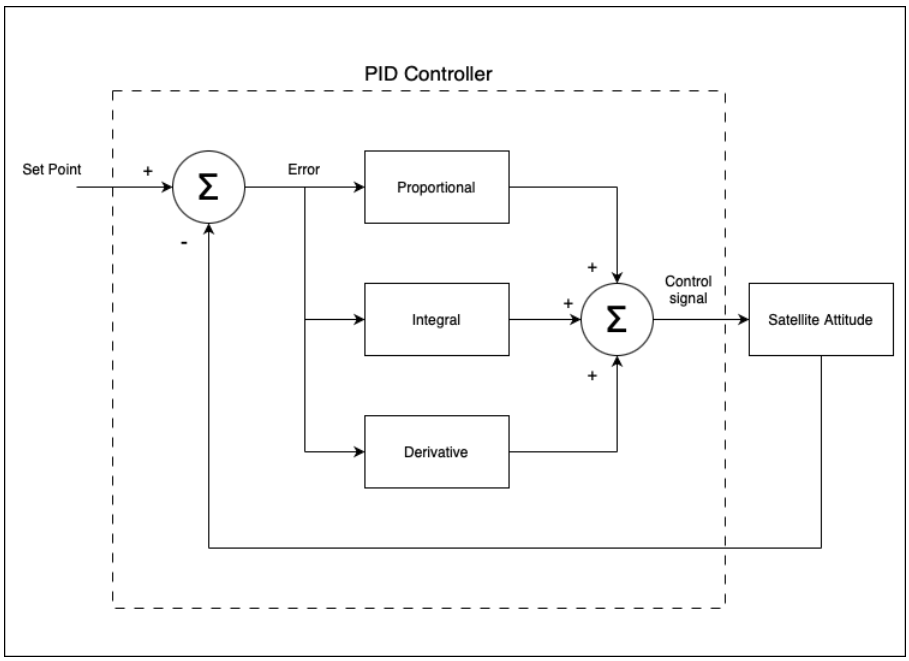

Figure 3: Satellite Attitude PID Controller Block Diagram

2.5.2 Model Predictive Control

MPC is a more sophisticated control strategy due to its predictive nature and ability to optimise control actions making it suitable for complex and dynamic systems. Unlike a PID controllers, MPC can handle multi-variable control problems with constraints on inputs and outputs. The key concepts of MPC are the prediction model along with the optimisation problem and its constraints.

The Prediction model implements a dynamics model of the system which is later used to predict the future states of the system. The model used in this report is a nonlinear time-invariant model as mentioned previously. MPC uses the dynamics described by this model over discrete-time to predict the future trajectory of the system over a finite prediction horizon N. The predicted states are denoted as $x_{k+i|k}$, where $i = 1, 2, 3, ..., N$ is the future step index and k is the discrete time step. The control inputs are denoted as u, these two terms form the state space equation for the controller.

$$x_{k+i|k} = f(x_{k+i-1}, u_{k+i-1}) \tag{21}$$

Where:

$$u_{k+i|k} = \begin{cases} u_{k+i|k}, & \text{if } i < M \\ u_{k+M-1|k}, & \text{if } i \geq M \end{cases} \tag{22}$$

Using this information the controller solves an optimisation problem with the aim of minimising a cost function J. The control inputs are only optimised for the first M steps, where M is the control horizon. The cost function used in the controller is based on a quadratic cost function which is a mathematical expression used to model systems where the cost changes at a varying rate depending on the inputs in the equation. The cost function J corresponds to,

$$J = \sum_{i=0}^{N-1} (||x_{k+i} - x_{ref}||_Q^2) + \sum_{i=0}^{M-1} (||u_{k+i}||_R^2) \tag{23}$$

In this equation $x(k)$ is the predicted output at each time step and x_{ref} is the desired set-point of the controller. The term $u(k)$ has the same definition in the prediction model above, the control inputs. N is the prediction horizon, which is the number of time steps k the controller uses to predict the future behavior of the system. The term Q is a weighting matrix to reflect the importance of the error in the system over the prediction horizon. The final coefficient R is another weighting matrix to penalise excessive control actions. Minimising this control function allows for a systematic approach to find the control inputs that allows the controller to reach its objective most efficiently. Hence minimising the cost function returns the best output for the controller u_k. This can be

represented by a block diagram of the MPC satellite attitude control system in figure 4.

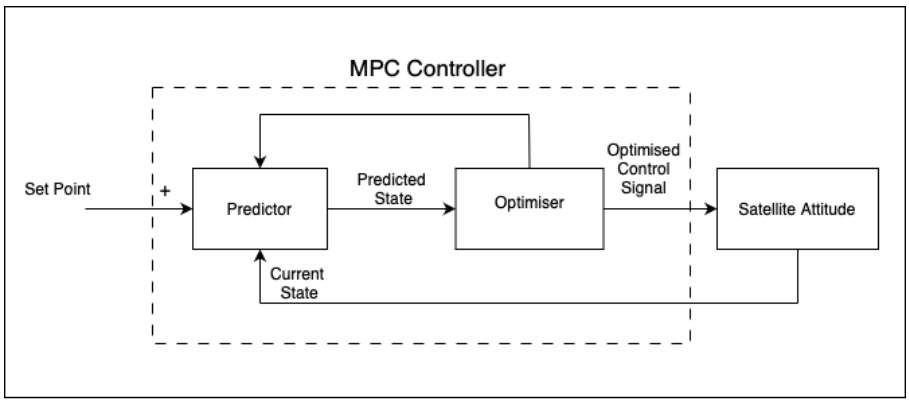

Figure 4: Satellite Attitude MPC Controller Block Diagram

3 Methodology

Python was chosen as the coding language to simulate MPC and PID attitude control this choice is an informed one. Python is highly versatile and holds a robust ecosystem of libraries available for scientific programming. Choosing python over other popular coding languages such as MATLAB has several advantages. Python's readability and object-oriented capabilities enhance code maintainability essential for complex simulations. The use of Python will provide a more in depth approach to implementing the physics, engineering and mathematics of the algorithms behind satellite attitude control simulations. The main libraries used in these simulations are Numpy for mathematical operations, SciPy for optimisation and Matplotlib to visualise the results [48][49][50]. This methodology will discuss the code to simulate the controllers. An integral function for the simulating the attitude of a satellite is the dynamics function. Utilising the previous methodology the attitude dynamics equations can be coded into python:

Listing 1: Attitude Dynamics Function in Python

```
def dynamics(state, integral_error, prev_error, dt):
    q = state[:4]  # Quaternion part
    omega = state[4:]  # Angular velocity part
    # Quaternion kinematics
    dqdt = 0.5 * quaternion_product(q, np.append([0], omega))
    # dqdt is the quaternion derivative
    control_torque = # The results of the control funtions
    # Euler's rotational equations with control torque
    omega_cross_I_omega = np.cross(omega, I @ omega)
    dwdt = inv_I @ (total_torque - omega_cross_I_omega)
    # dwdt is the angular acceleration
    return np.concatenate([dqdt, dwdt]), integral_error, prev_error
```

This function calculates the rotational dynamics of the satellite using the quaternion kinematics and Euler's rotational equations of motion. It takes the current state of the system and split up the quaternion and angular velocity parts before computing the quaternion kinematics. The control signal produced by the control algorithm is take as the control torque in this simulation. Using this torque and Euler's rotational equations (8) to calculate the angular acceleration of the system. Concatenating the kinematics and angular acceleration into a seven dimension vector which will be used by the integration method to update the satellites attitude and angular velocity.

3.1 Spacecraft Parameters

This section will determine the parameters required to simulate the dynamics of the system such as the LRO's inertia matrix and control torque limits. The LRO satellite has a maximum slew rate of 0.13 degrees per second this corresponds to: $|\boldsymbol{\omega}| = 0.002268928$rad/s [51]. To simulate the spacecraft's dynamics accurately we need to determine the inertia matrix of the LRO satellite. The inertia matrix describes the satellites mass distribution and how it will behave when subjected to torques. To calculate the inertia matrix firstly we have to determine the geometric model of the space craft.

Using the length l, width w and height h shown in the model of our approximated satellite we can calculate the inertia matrix. Approximating the center of mass to be centered at the origin we have the equation (9) for the inertia matrix. Where I_{xx}, I_{yy} and I_{zz} can be calculated using the following equations,

$$I_{xx} = \frac{1}{12}m(w^2 + h^2) \tag{24}$$

$$I_{yy} = \frac{1}{12}m(l^2 + h^2) \tag{25}$$

$$I_{zz} = \frac{1}{12}m(l^2 + w^2) \tag{26}$$

Approximating the mass m of the LRO as the dry mass of the vehicle which is equal to $1018kg$ [52]. Therefore we can calculate the inertia matrix $\mathbf{I}$ for the LRO satellite as shown below.

$$\mathbf{I} = \begin{bmatrix} 1191.9 & 0 & 0 \\ 0 & 1843.4 & 0 \\ 0 & 0 & 1798.5 \end{bmatrix} kg \cdot m^2 \tag{27}$$

Coding this in python using the Numpy library to create the inertia matrix for the satellite model.

Listing 2: Inertia Matrix Python Code

```
# Inerta matrix set up
Ix, Iy, Iz = 1191, 1844, 1798
I = np.diag([Ix, Iy, Iz])  # Resulting diagonal inertia matrix
```

3.1.1 Reaction Wheel Control

The control method to be use by the simulated Lunar Reconnaissance Orbiter is modelled using a reaction wheel assembly. In simulating the attitude control system of a satellite, modeling the reaction wheel performance is a critical component, particularly when these wheels are the primary means of controlling the spacecraft's orientation. The LRO utilises four reaction wheels to control the attitude of the satellite. To overcome the issues related to saturation momentum dumping is performed approximately every two weeks using small thruster's as part of the attitude control system [52]. Saturation occurs when the reaction wheels reach its maximum output, the reaction wheel cannot spin any faster, therefore the system can't produce any more torque even if the controller requests it. In a real implementation of a control algorithm the control torque output would be converted by a lower-level controller to a voltage to be used by the reaction wheels. The model implemented in this simulation sets the bounds of a maximum control torque output

to be $0.25Nm$. This approximates the behaviour of the LRO's dynamics used in the simulations.

3.2 Coding the Controllers

3.2.1 PID Code

As discussed in the theory section above the controller consists of three terms using the error from the satellites current state to the desired set-point. Given a system where the state is represented by a quaternion, the following Python function implements the PID controller:

Listing 3: PID Control Function for the System

```
def pid_control(error,integral_error, prev_error, dt):
    # Error is the difference between the current state and the set-point
    integral_error += error * dt
    derivative_error = (error - prev_error) / dt
    control_torque = Kp * error + Ki * integral_error + Kd * derivative_error
    # Bounds for the control torque
    control_torque = np.clip(control_torque, -0.25, 0.25)
    return control_torque, integral_error, error
```

This function takes the current error, the accumulated integral error, the previous error and the time step as inputs. It calculates the control torque using the three components described by a PID controller. The integral error is updated by accumulating the error over time and the derivative term is calculated as the rate of change of error. The function also implements the bounds of the control torques using the Numpy function

clip which returns control torques above the bounds to the maximum control output of 0.25Nm. Finally it returns the calculated control torque, the updated integral error, and the current error to be used as the previous error in the next iteration of the simulation.

3.2.2 MPC Code

The Model Predictive controller has more functions and complexity, firstly the cost function for the system needs to be defined. The following python function implements this cost function:

Listing 4: MPC Cost Function for the System

```
def cost_function(u, q_current, omega_current, error):
    total_cost = 0
    q = q_current.copy()
    omega = omega_current.copy()
    for i in range(N):
        if i < M:
            u_i = u[i*3:(i+1)*3]
        else:
            u_i = u[(M-1)*3:M*3]  # After M apply last optimized control
        state = np.concatenate([q, omega])
        state = rk4_step(state, u_i, dt) # Integration step
        q = state[:4] # Quaternion of the state vector
        q /= np.linalg.norm(q)  # Normalize quaternion
        omega = state[4:] # Angular Velocity
        error  # Error in state to set-point
```

```
        # Add penalties Q and R
        total_cost += (Q * error + R * np.sum(u_i**2))
    return total_cost
```

This function calculates the total cost of the control input 'u' applied to the satellite. The function iterates over the prediction horizon N applying the control input for each step, after the control horizon M the last optimised control is used. The satellites dynamics is updated via the RK4 integration method (Runge Kutta 4^{th} order). The total cost of the control input calculation includes the weighting coefficients Q and R to reflect the importance of the error and penalise excessive control actions. This function is then minimised in the MPC control loop to produce the true output of the controller. This is performed in an MPC control loop function defined as:

Listing 5: MPC Control loop

```
# MPC control loop
def mpc_control(q_current, omega_current):
    print("Starting MPC optimization...") # Record the start of the loop
    u_init = np.zeros(M * 3)  # Only optimize the first M control steps
    bounds = [(-0.25, 0.25)] * (M * 3)
    result = minimize(cost_function, u_init, args=(q_current, omega_current),
    |      |    method='SLSQP', bounds=bounds)
    print("MPC optimization completed.")
    return result.x[:3]
```

To begin the loop the controller makes an initial guess for the control torques 'u_init' with the three elements in the three torque axes. The control inputs are constrained

within bounds of (-0.25, 0.25) for each element to limit the control output. The function uses the minimize function from the SciPy library, which optimizes the cost function by adjusting the control inputs to minimize the total cost. The Sequential Least Squares Programming (SLSQP) method is used for the optimization. The function returns the first set of control inputs to be applied to the system in the following time step.

3.3 Simulation Scenarios

3.3.1 Attitude manoeuvre

To simulate the performance of the attitude controller we much choose a rotation movement for the satellite to perform. This example manoeuvre will test our controllers performance by simulating a point to point rotation comparable to the LRO changing its attitude to point its instruments at a specific area of the moon. This observation manoeuvre is not a large rotation, this manoeuvre is modeled as a 10 degree rotation in the y-axis, 5 degree rotation in the x-axis and a 6 degree rotation in the z-axis from the LVLH to B reference frames. The quaternion components can be computed as,

$$
\begin{aligned}
w &= \cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
x &= \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
y &= \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\
z &= \cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)
\end{aligned}
$$

Where ϕ is the rotation around the x-axis known as the roll, the rotation around the

y-axis is described as the pitch θ and finally the z-axis rotation is known as the yaw ψ. This equates to the final quaternion $\mathbf{q}_{target}$ representing the observing mode rotation. This information will be used by the controller to find the actions needed to complete this simulated manoeuvre.

$$\mathbf{q}_{target} = \begin{bmatrix} 0.9938 \\ 0.0860 \\ 0.0513 \\ 0.0608 \end{bmatrix} \tag{28}$$

To program the satellite attitude controller for this manoeuvre, $\mathbf{q}_{target}$ is taken as the set-point x_{ref} for the MPC and PID controllers. The controllers take the current states from the attitude dynamics equations and the resulting control signals corresponds to a control torque which then changes the state of the satellite completing the control loop.

$$\mathbf{q}_{error} = \mathbf{q}_{target}^{-1} \otimes \mathbf{q} \tag{29}$$

Equation (29) presents how the error between the current quaternion $\mathbf{q}$ and the target quaternion, this equation uses the conjugate of the target quaternion and the Kronecker product to calculate the error. The error calculation is coded in python as:

Listing 6: Error calculculation

```
    # Function to calculate quaternion conjugate
def quaternion_conjugate(q):
    return np.array([q[0], -q[1], -q[2], -q[3]])
```

```
# Function to multiply two quaternions
def quaternion_multiply(q1, q2):
    w1, x1, y1, z1 = q1
    w2, x2, y2, z2 = q2
    w = w1*w2 - x1*x2 - y1*y2 - z1*z2
    x = w1*x2 + x1*w2 + y1*z2 - z1*y2
    y = w1*y2 - x1*z2 + y1*w2 + z1*x2
    z = w1*z2 + x1*y2 - y1*x2 + z1*w2
    return np.array([w, x, y, z])

error = quaternion_multiply(quaternion_conjugate(q_target),q)
```

3.3.2 External Disturbances

To more realistically test the control algorithms we can include external disturbance torques that act on the spacecraft. The previously discussed two main types of torques experienced by spacecraft result from gravity and solar radiation. In the case of the LRO satellite the moons gravity will produce a higher torque on the space craft than the earths. Therefore the gravity gradient torque applied to the LRO can be approximated to be in the order of magnitude $10^{-6}N \cdot m$ taking the orbit of the LRO to be around $35km$ in observation mode [same ref as observation mode]. Therefore as solar radiation pressure produces a higher torque we will model the external disturbance as a constant torque in the x direction.

$$\boldsymbol{\tau}_{ext} = \begin{bmatrix} 1 \times 10^{-4} \\ 0 \\ 0 \end{bmatrix} \tag{30}$$

$$\boldsymbol{\tau}_{total} = \boldsymbol{\tau}_{ext} + \boldsymbol{\tau}_c \tag{31}$$

The total torque $\boldsymbol{\tau}_{total}$ used in the satellite dynamics combines the control torque $\boldsymbol{\tau}_c$ produced by the controller and the external disturbance torque $\boldsymbol{\tau}_{ext}$. This is coded into the dynamics function in python.

Listing 7: External Disturbance Torque Array Implemented into the Dynamics function

```
External_torque = np.array([0.0001, 0.0, 0.0])
Total_torque = control_torque + External_torque
```

3.3.3 Spacecraft Tumbling

The destabilisation of a spacecraft can significantly disrupt the operations of its missions and can be attributed to several factors. The two main causes are micrometeoroids impacts and system failures on the cosmic craft. Micrometeoroids are very small fast moving objects and collisions with a spacecraft can impart a momentum transfer causing the vehicle to have an unwanted change in attitude and angular velocity. These micrometeoroids can also strike the electronics on board causing the spacecraft to behaviour abnormally. This abnormal behaviour may also cause the attitude control system to destabilise the craft. System failures may also be cause by other external factors such as cosmic radiation and electromagnetic interference which cant interfere with the electrical systems on

a satellite [53]. These system failures can cause reaction wheels to produce unnecessary torques or a miss firing of a satellites thruster control system. Spacecrafts have many redundancy systems built into their design however unexpected failures can still occur. The threshold to consider an attitude to be tumbling varies from one paper to another for example F.Reichel takes it to be 0.005 rad/s, this is what is used to inform the angular velocity for the out of control satellite simulation [54].

$$\omega = \begin{bmatrix} 0.1 \\ 0.08 \\ 0.15 \end{bmatrix} rad \cdot s^{-1} \tag{32}$$

To simulate this the LRO satellite is modelled as having an initial angular velocity ω, this models a rapid tumbling unstable satellite. The MPC and PID controllers are then adapted so the set point of the controller is a zero angular velocity to simulate the spacecraft stabilisation. After this stabilisation the satellite could return to normal attitude control. The initial angular velocity chosen to approximate a satellite tumbling at several degrees per second. This models an approximate moderate disturbance caused by a system failure. To program this in python we define the initial angular velocity and adapt the controllers so the desired set point is a zero angular velocity vector.

Listing 8: Tumbiling Satellite Simulation

```
omega0 = np.array([0.1, 0.08, 0.15])    # Initial angular velocity
Target_omega = np.array([0, 0, 0]) # Target angular velocity


# Following error calculation to be included in the control algorithm
error = target_omega - omega  # Where omega is the current omega
```

3.4 Tuning Methodology

The two controllers need to be tuned to produce the most efficient results, this section describes three tuning methods. Firstly and most simple is manual trail and error tuning, this involves taking informed initial conditions for the tuning parameters and then changing them manually to observe the best results. The second and third methods are expanded on in this section, these methods are a systematic tuner for the PID controller and a systematic cost optimiser tuner for the MPC controller.

3.4.1 PID Tuning Algorithm

The goal of this optimisation method is to determine the optimal Proportional,Integral and Derivative gains (K_p,K_i,K_d) that minimises the stabilisation time of the control system. This method uses the simulation to test the gains and determine a stability time. The optimisation process then uses a grid search to find the best gains by testing various combinations, search ranges are set for the algorithm to test between. These ranges are decided from the manual tuning results and have 100 steps each, this results in a total of 1×10^6 combinations. For each combination, the satellite attitude control is simulated to monitor if it stabilises to the desired set-point within a certain time. The stabilisation time is recorded and then the best gains are updated if a faster time is achieved compared to the previous best gains time. The PID gain combination with the quickest stabilisation time is taken to be the optimal set of gains for the simulation.

3.4.2 MPC Parameter Tuning Algorithm

To properly tune the model predictive controller we need to find the optimal values for the five defining parameters N, M, dt, Q, and R of the controller. Which correspond to the prediction horizon, the control horizon, the time step of the simulation, the weighting of the state error and the weighting of the control signal respectively. The optimisation strategy to tune the controller uses the Sequential Least Squares Quadratic Programming (SLSQP) method whilst running the MPC simulation to find the optimal parameters. Using the SciPy libraray minimize in python we can use SLSQP to solve a optimisation subproblem of the parameters. After solving the subproblem SLSQP updates the parameters to new more optimum values, this processes iterates over many parameters until the controller is tuned. The function satisfies a set of constraints set for the parameters and uses an initial estimate for the variables. This method adds computational complexity to the already costly MPC controller however if it produces better results it is more useful to have an optimally tuned controller than an inefficient one.

3.5 Simulation Methods

To simulate the performance of the attitude control algorithms the mathematical models are implemented into python. These simulations are then run to compare the two different control methods with the three scenarios, the undisturbed rotation, the effect of external disturbances and the control of a destabilised model of the LRO.

3.5.1 Integration Method

To simulate the attitude dynamics and the control algorithms an integration method is required to accurately predict the behaviour of the satellites dynamics. The models involve complex nonlinear differential equations that describe the evolving inputs and orientation of the satellite. The analytical solutions to these equations are often not practical numerical integration is required to approximate the satellites state at discrete time intervals dt. The fourth order Runge-Kutta method is chosen to solve these equations in the simulations. The ordinary differential equation for our simulation takes the form:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{33}$$

where $\mathbf{x}$ again is the state vector and $\mathbf{u}$ denotes the control signal. The method involves the following steps to calculate intermediate slopes,

$$k_1 = dt \cdot \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n) \tag{34}$$

$$k_2 = dt \cdot \mathbf{f}(\mathbf{x}_n + \frac{1}{2}k_1, \mathbf{u}_n) \tag{35}$$

$$k_2 = dt \cdot \mathbf{f}(\mathbf{x}_n + \frac{1}{2}k_2, \mathbf{u}_n) \tag{36}$$

$$k_2 = dt \cdot \mathbf{f}(\mathbf{x}_n + k_3, \mathbf{u}_n) \tag{37}$$

These equations calculate the derivative at a 4 different time intervals where k_1 is the current time and k_4 is the end of the interval. The subscript n describes the n_{th} time step for the state and control signal vectors at each step. The Runge-Kutta method then updates the state vector therefore $\mathbf{x}_{n+1}$ is computed by combining these slopes as seen in equation (39). Through these equations the RK4 method ensures accurate simulation of the satellite attitude dynamics and give the controllers the optimal information to produce their control signals.

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{38}$$

Coding this method into Python completes the simulations to update the state of the satellite for the next time step. Where the previous functions are repeated with the new state to calculate the next control input to reach the desired set-point. The following code defines the RK4 method in Python:

Listing 9: Runge Kutta 4^{th} Order in Python

```
def rk4_step(state, integral_error, prev_error, dt):
    k1, integral_error, prev_error = dynamics(state,
    |   |   integral_error, prev_error, dt)
    k2,_,_ = dynamics(state+0.5*dt*k1, integral_error, prev_error, dt)
    k3,_,_ = dynamics(state+0.5*dt*k2, integral_error, prev_error, dt)
    k4,_,_ = dynamics(state+dt*k3, integral_error, prev_error, dt)

    new_state = state + (dt / 6) * (k1 + 2*k2 + 2*k3 + k4)
    return new_state, integral_error, prev_error
```

3.6 Stability Time Method

To asses the capability of the control methods and the simulations the stability time of the satellite is calculated as when the controller completes any of the three scenarios the satellite should be returned to a stationary state. After storing the results of the simulation in a CSV file this data is read by the controller evaluation algorithm to find the stability time. The method loads the CSV data into a pandas DataFrame and sets key parameters such as the stability threshold, window size for moving averages and confidence level. The moving averages of the angular velocities are then calculated to smooth the data, then the standard error of the mean is calculated for each angular velocity. Utilising this to determine confidence intervals, using this information the code checks if the moving averages and confidence intervals lie below the stability threshold. This indicates the satellite is stable and the corresponding time step is the stability time.

4 Attitude Manoeuvre Simulation

This result section presents the results for both controllers with a manual tuning method and a systematic tuning method for comparison. These simulations model the satellite in an undisturbed environment to compare the performances of the two controllers.

Parameter	Value
$\boldsymbol{\omega}_0$	$[0\ 0\ 0]^T$
$\mathbf{q}_0$	$[1\ 0\ 0]^T$
$\lvert\boldsymbol{\tau}_{max}\rvert$	$[0.25\ 0.25\ 0.25]^T\ Nm$

Table 1: Constant Parameters for Attitude Manoeuvre

The observing mode simulation has three constant parameters, firstly the satellite has an initial stationary angular velocity vector $\boldsymbol{\omega}_0$. The initial quaternion $\mathbf{q}_0$ represents the satellite aligned with the LVLH reference frame, the final constant parameter is the maximum control torque the controller can produce.

4.1 Manual Tuning Results

The following results of the two controllers use manual tuning to decide the parameters to produce the simulation results.

4.1.1 PID Simulation

The PID gains were manually tuned through trail and error until the best results were produced. As the satellite in this simulation is sensitive to control actions due to the crafts dynamics, a large proportional gain would destabilise the satellite leading to large oscillations. This is presented in the $\mathbf{K}_p$ values where a low gain value is used to control the satellite the following large derivative gain is used to prevent these oscillations. The large moments of inertia equally contribute to the low proportional gain as the satellite is slow moving and a large proportional response would cause over corrections especially as the satellite takes time to decelerate.

Parameter	Value
$\mathbf{K}_p$	$[0.3\ 0.3\ 0.3]^T$
$\mathbf{K}_i$	$[1 \times 10^{-6}\ 1 \times 10^{-6}\ 1 \times 10^{-6}]^T$
$\mathbf{K}_d$	$[100\ 100\ 100]^T$
dt	1

Table 2: Manual Tuned PID Simulation Parameter Values

Table 2 provides the individual parameters used for the simulation of the PID control algorithm, the following gains were found through manual tuning. The high moments of inertia also dictate the choice of time step dt as the satellite takes many seconds to rotate a few degrees. This slow movement means that small time steps of milliseconds would be unnecessary and a waste computational resources.

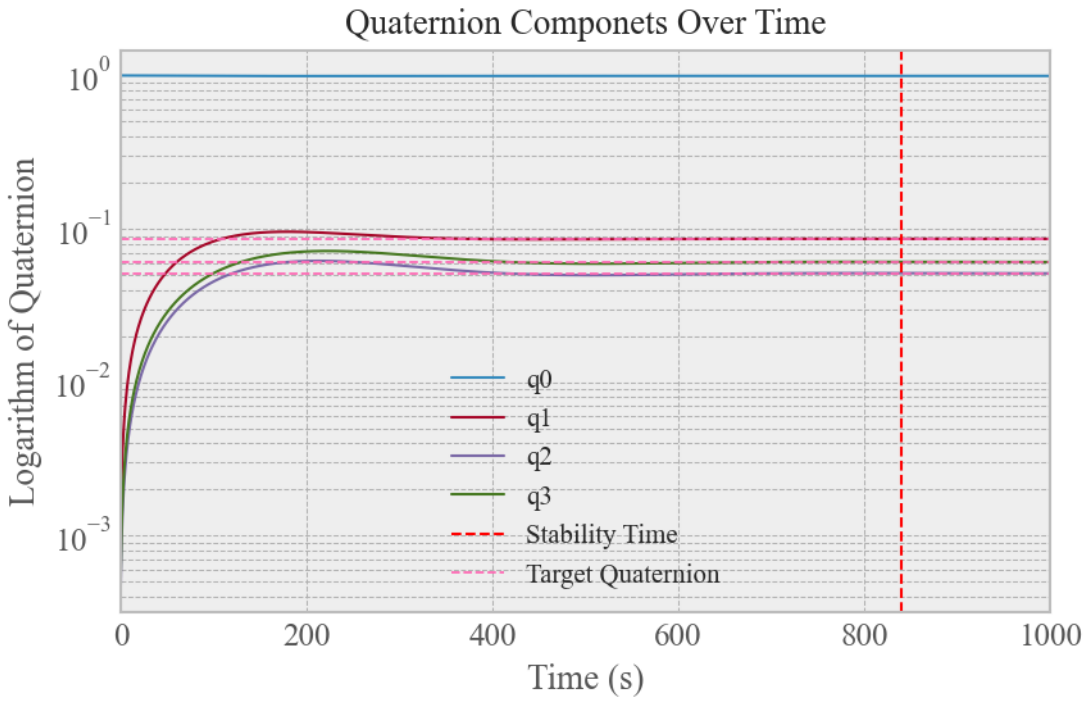

Figure 5: Attitude Control Performance of PID Controller with Manually Tuned Gains

Through the controller evaluations the completion time for the attitude manoeuvre was found to be 839.42 seconds using the manually tuned PID controller. The resulting quaternions from the simulations over time is presented in figure 5, the logarithmic scale is used to easier show the quaternions. The overshoot can be seen in figure one as the vector part of the quaternion transports to the desired coordinates. The manually tuned controller shows some overshoot of the quaternion components, these gains exhibit an average overshoot of 1.04% on the vector components. It can observed in figure one the stability time looks to be almost two-hundred seconds after the quaternion components

reach their targets. However this can be explained by figure 6 below as the angular velocity is minute but still present hence the longer seeming stability time.

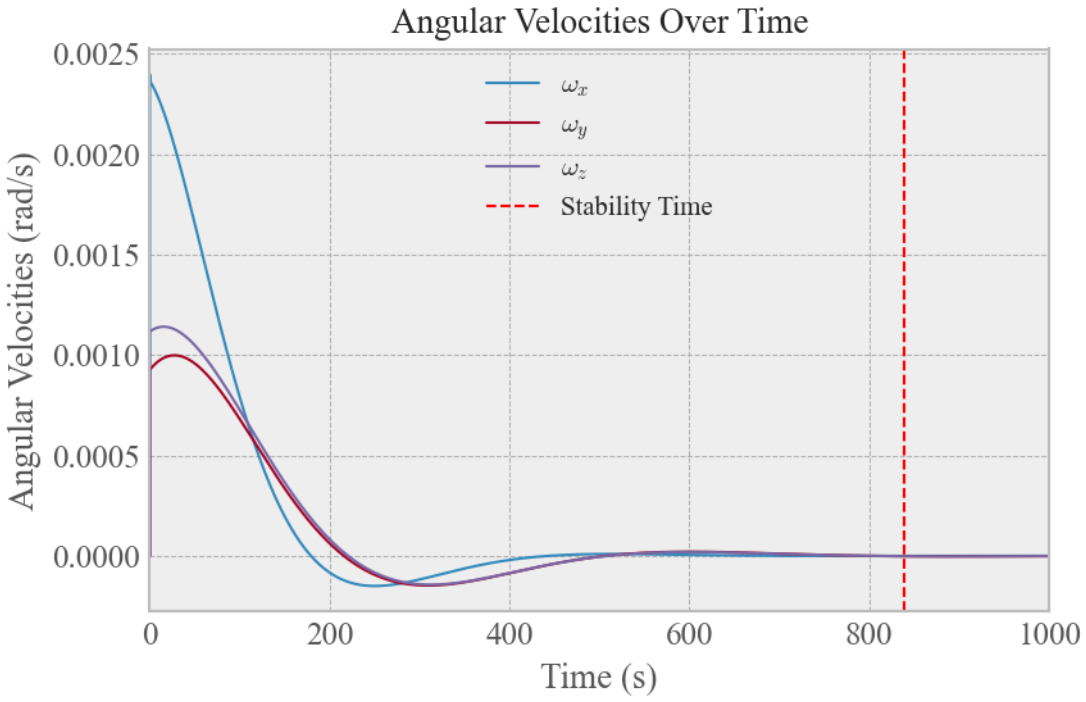

Figure 6: Angular Velocity Results of the PID Controller (Manually Tuned)

4.1.2 MPC Simulation

The parameters manually tuned were the prediction horizon, the control horizon and the two weighting coefficients as shown in table 3. The selection of the initial values of the controller was based of literature. The prediction horizon N is typically in the range 30-120 as mentioned in Seborg et al when tuning a petrochemical controller [55]. The control horizon M follows a general rule setting it to around a quarter of the prediction horizon. The weighting coefficients Q and R penalise the output errors and the rates of input changes respectively.

Parameter	Value
N	50
M	10
Q	1
R	0.00001
dt	1

Table 3: MPC Manually Tuned Controller Parameters

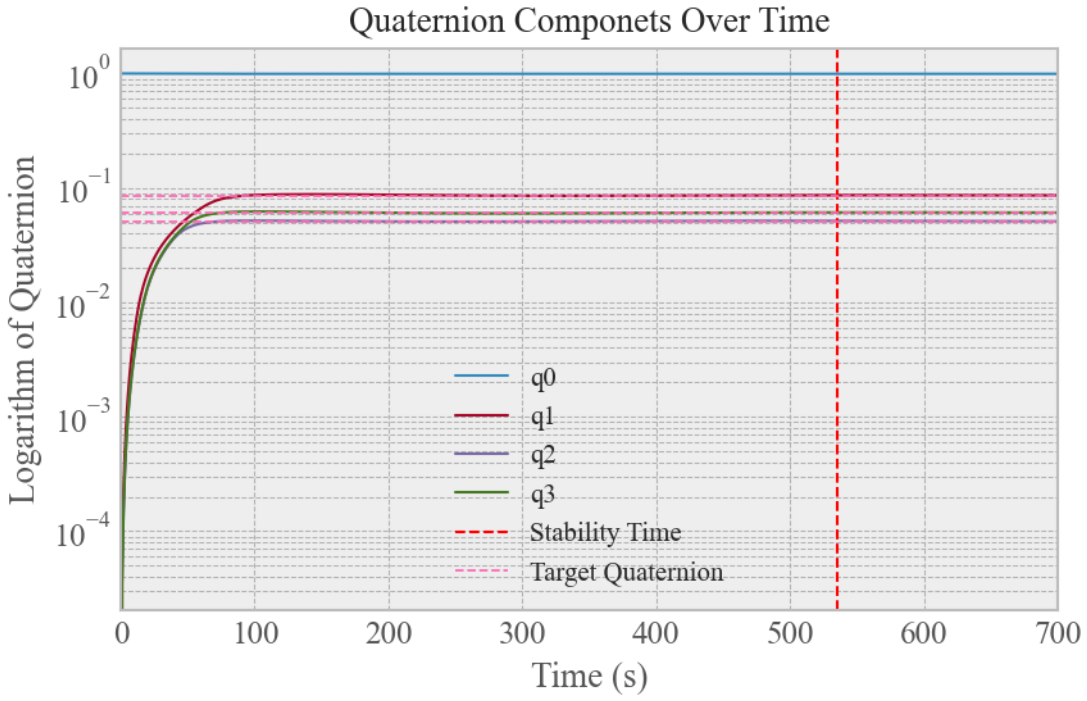

Figure 7: Attitude Control Performance of MPC Controller with Manually Tuned Gains

The manually tuned MPC controller produced a completion time of 535.0 seconds, approximately 63% faster than the manually tuned PID controller. Similar to the results above the controller takes extra few hundred seconds to fully stabilise the angular velocities of the satellites, this presents the weaknesses in the manual tuning approach. The average overshoot of vector quaternion components in the MPC results was found to be 0.13%, much lower than the manually tuned PID controller. These results show the superior results of the MPC controller for an attitude manoeuvre.

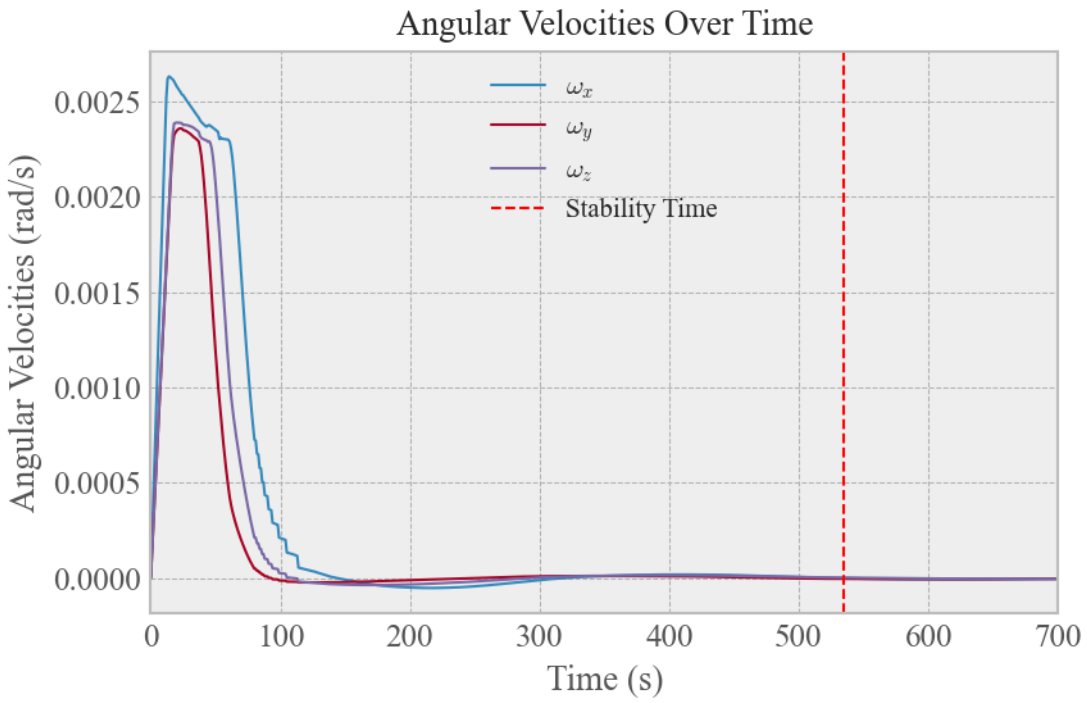

Figure 8: Angular Velocity Results of the MPC Controller (Manually Tuned)

Figure 8 presents the angular velocities of the satellite in the attitude manoeuvre using the MPC controller. In the beginning of this graph the results have a jagged nature, this indicates the controller is making aggressive adjustments. This maybe due to the weighting parameters effects on the controller, a smaller time step may smooth the controller input however this would increase the computational cost and would effect the performance of the controller. Using a lower time step would mean the prediction and control horizons would also be shorter in time.

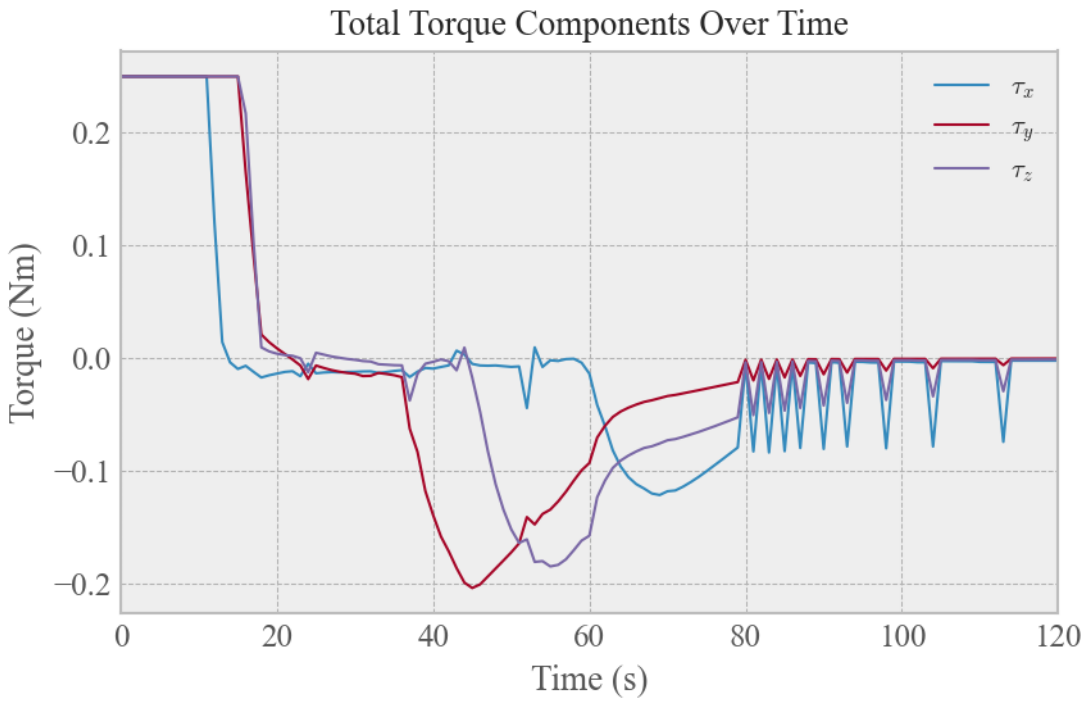

Figure 9: Control Torques Produced by the MPC Controller in the First 120s

The Control torque components acting on the satellite model reveal the dynamics of the satellite attitude control algorithm. Observed in figure 9 the controller presents a maximum initial torque of $0.25Nm$ to rapidly rotate the satellite into its desired orientation. After around twenty seconds of constant torque the controller aims to slow the angular velocities of the satellite producing negative torques to act against the rotating craft. The small sharp peaks observed after eighty seconds informs us the controller makes persistent corrections until the satellite is stable.

4.2 Systematic Tuning Results

4.2.1 PID Simulation

The systematic tuning method found the optimal results for the PID gains to produce the fastest stability time for the controller. The optimal integral gain is found again to be 1×10^{-6}, this gain is so low and suggests that the integral term in our PID controller has little to no effect on the controller.

Parameter	Value
$\mathbf{K}_p$	$[0.19795918\ 0.19795918\ 0.19795918]^T$
$\mathbf{K}_i$	$[1 \times 10^{-6}\ 1 \times 10^{-6}\ 1 \times 10^{-6}]^T$
$\mathbf{K}_d$	$[111.5151\ 111.5151\ 111.5151]^T$
dt	1

Table 4: Systematically Tuned PID Simulation Parameter Values

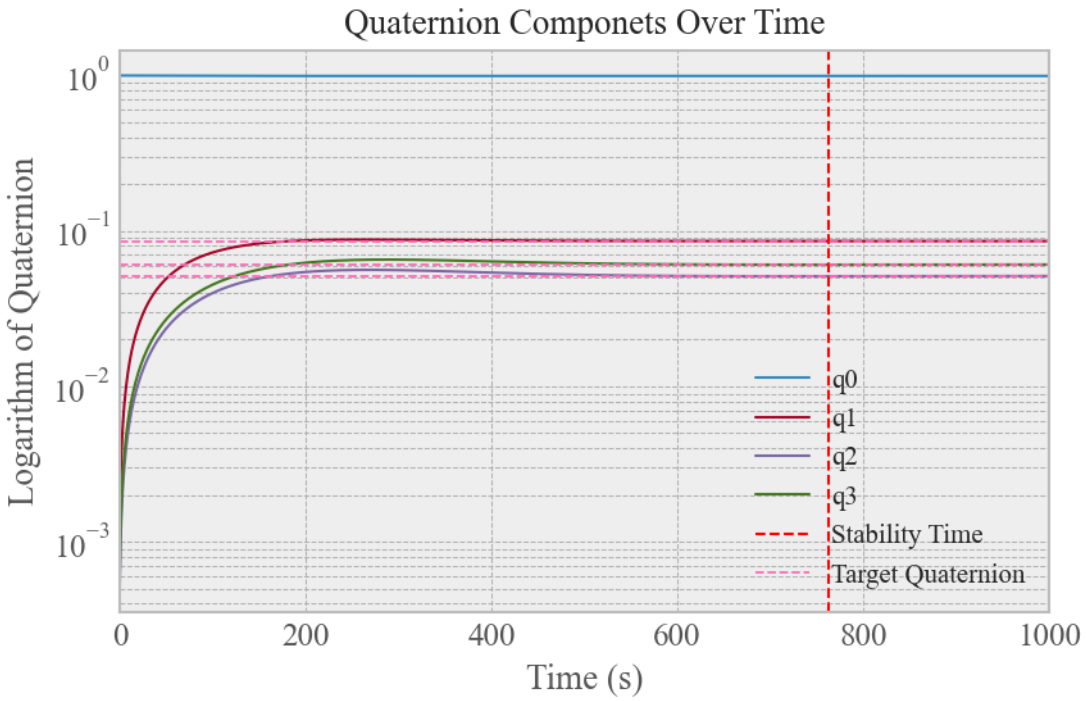

Figure 10: Optimised PID Controller Quaternion Components Over Time

The resulting simulation using these optimal PID gains produces the outcomes showing in figure 10 and 11, through the quaternions and angular velocity over time. The optimal gains produce a stability time of 762.38 seconds approximately 77 seconds faster than the manually tuned controller. These gains do a better job at stabilising the satellites angular velocities faster. This controller still produces an overshoot of the results however it is less than the manually tuned results with an average overshoot of 0.39%.

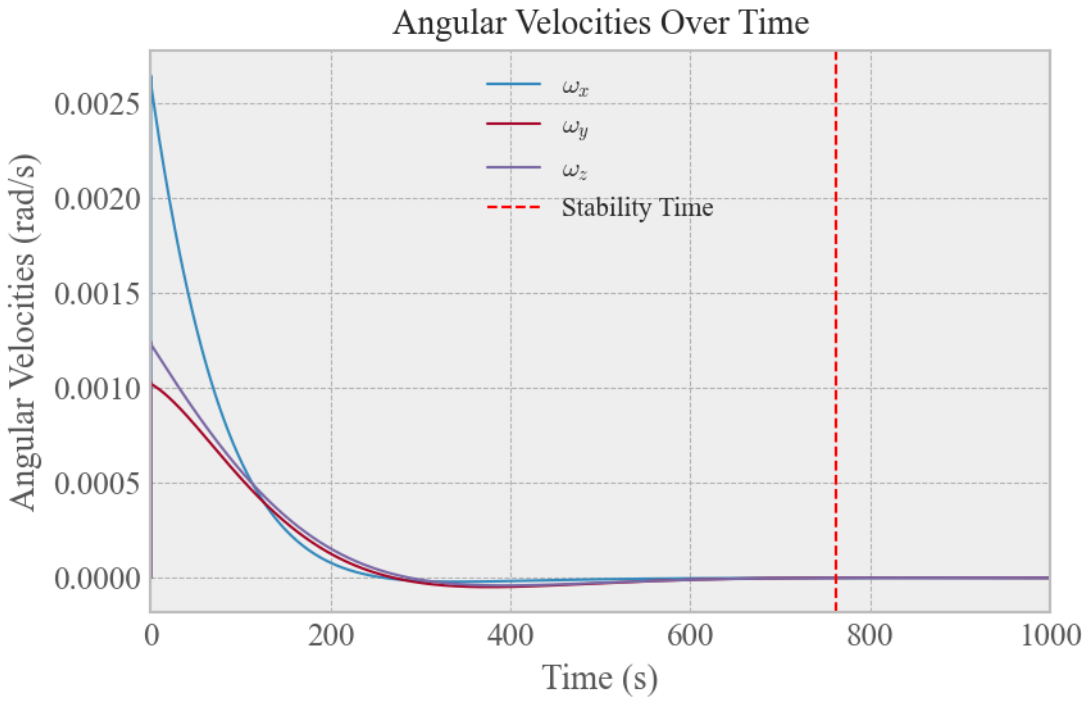

Figure 11: Angular Velocity Graph of Optimised PID Controller

4.2.2 Cost Optimised MPC Simulation

This simulation uses the parameters found by the cost optimiser previously discussed, these simulations should result in an improvements of stability time. The optimal parameters are presented in Table 5, preliminary comparison shows a lower prediction horizon therefore focusing on a shorter time window to predict the system behaviour. The larger

R coefficient produces a greater penalty on input changes this leads to a smooth resulting control signals which also effects the angular velocity. Hence aiming to reduce the sharp nature of the graphs for the manually tuned MPC results.

Parameter	Value
N	30
M	10
Q	6.0429
R	0.010537
dt	0.9696

Table 5: MPC Cost Optimised Controller Parameters

The resulting stability time for this controller with optimised parameters was calculated to be 224.947 seconds this is 42% faster than the manually tuned controller. The results of this simulation show the speed and precision of the MPC controller to much better than the PID alternative. The resulting angular velocities are confirmed to be much smoother due to the larger R penalty as observed in figure 12.

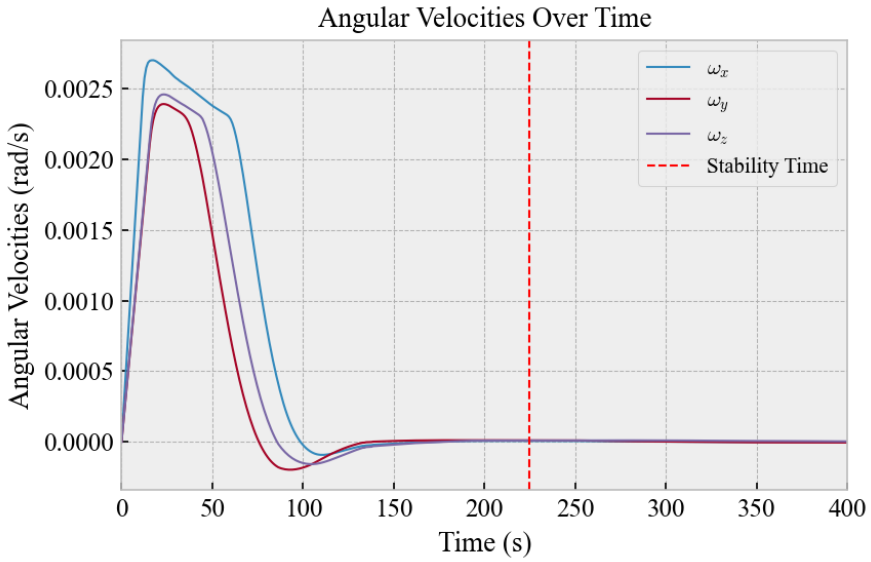

Figure 12: Optimally Tuned MPC Controller Simulation Quaternions

This optimised MPC controller uses the same large initial torque controller as shown in figure 13, however the results stabilise the satellite much faster providing a higher level of control requiring less adjustments. Based on these results the MPC is the clear choice for a satellite attitude controller to conduct an observing mode manoeuvre. The MPC controller does come a larger computational cost, this was recorded as the simulations take longer to run however with modern technology and computational optimisation it is feasible to be used by satellites.

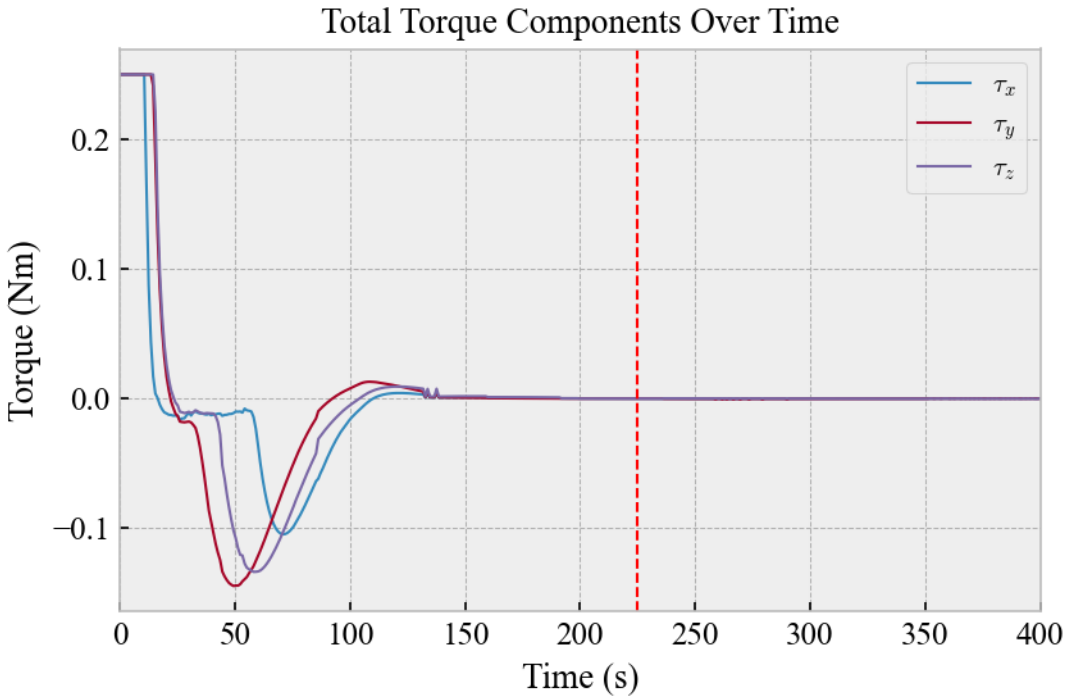

Figure 13: Optimally Tuned MPC Controller Outputs

5 External Disturbance Simulation

The aim of the controllers in this simulation is to stabilise the satellite when in the presence of a constant external torque acting on the spacecraft. As the toque acts in the x direction on the body based reference frame the main quaternion component to focus

on is the first vector one q_1. The stability time is recorded again to measure how long each controller takes to steady the satellite against the external torque. The following results demonstrate the satellite attitude control algorithms ability to handle an external disturbance torque.

5.1 PID Simulation

The PID controller stabilises against the external torque in 573.287 seconds as represented by the red dotted line on the figures below. The gains used for this simulation are the same manually tuned gains from the PID attitude manoeuvre simulation. The satellite experienced an initial deviation from the LVLH frame due to the disturbance with the satellites angular velocity reaching a peak of 2.5×10^{-6} rad/s before the satellite was stabilised. The satellite was rotated 0.03 degrees in the x axis once stable, the satellite did not return to the target orientation aligned with the LVLH frame as displayed in figure 14.

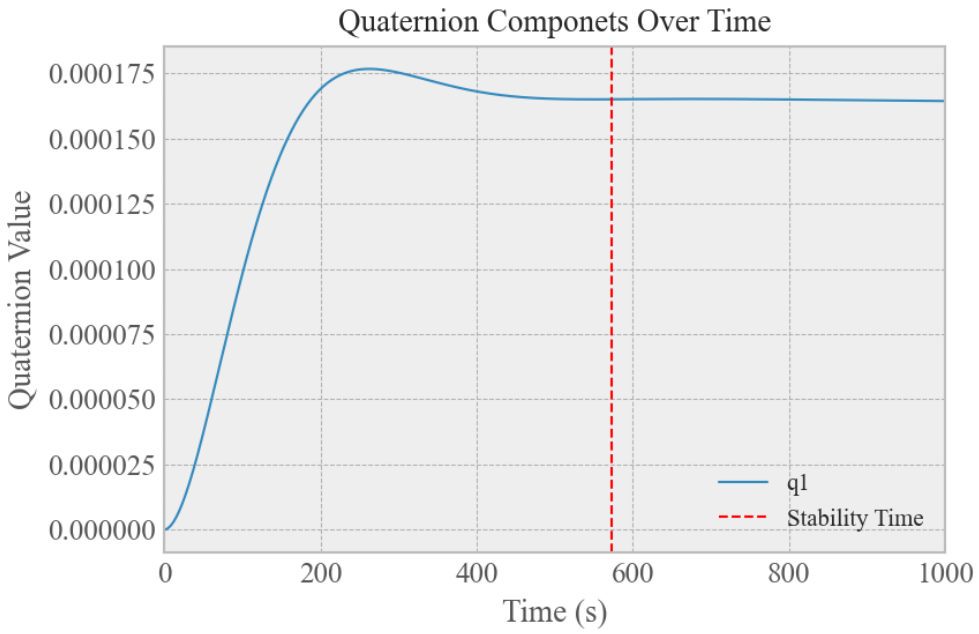

Figure 14: PID Response to External Disturbance Torque: Quaternion Component q_1

The stability control signals can be visualised in figure 15 as torque signal produced by the PID controller. The torque can be seen to settle at -0.0001 acting constantly against the external disturbance torque. Overall the effect of the external disturbance on the satellite is very small however it may cause the satellite to be misaligned. This misalignment may compound over time with more varying external disturbances and cause performance of the satellite to be reduced.

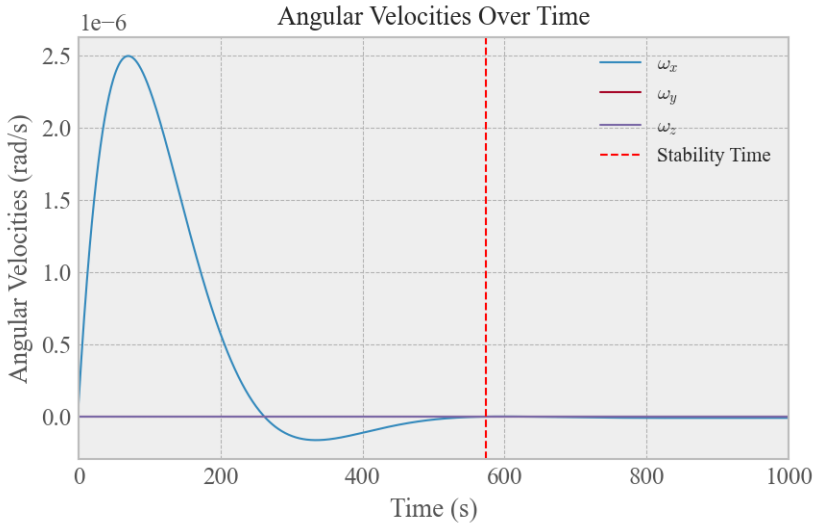

Figure 15: Angular Velocity of Satellite with an External Disturbance, PID controlled

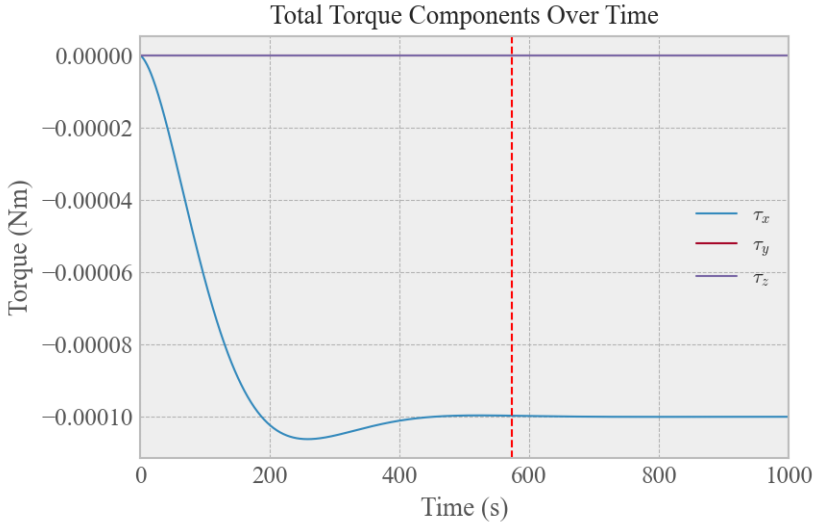

Figure 16: Control Torques Produced by the PID controller Against External Disturbance

5.2 MPC Simulation

The performance of the MPC controller was evaluated for stabilising the satellites attitude in the presence of an external disturbance torque. As in the previous simulation the satellite was initially stationary and the target attitude is aligned with the LVLH reference frame.

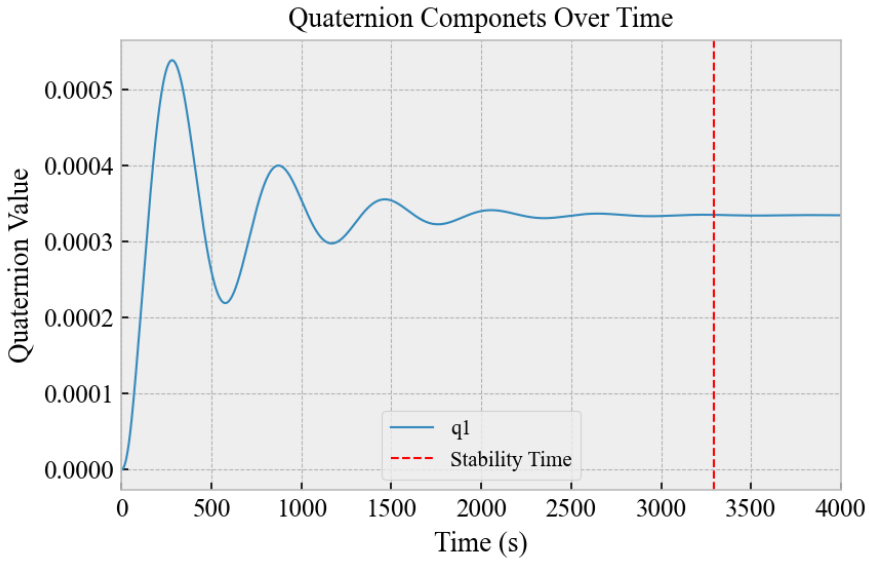

Figure 17: MPC Response to External Disturbance Torque: Quaternion Component q_1

As shown in figure 17, the quaternion component q_1 oscillates with decreasing amplitude in response to the external disturbance torques. The results present a stability time of 3291 seconds almost six times longer than the PID simulation with the same control torque. The resulting final quaternion after the satellite is stabilised describes a 0.038 degree rotation in the x axis. Therefore this control algorithm not only takes longer to stabilise but also produces a larger error from the desired orientation that the PID controller.

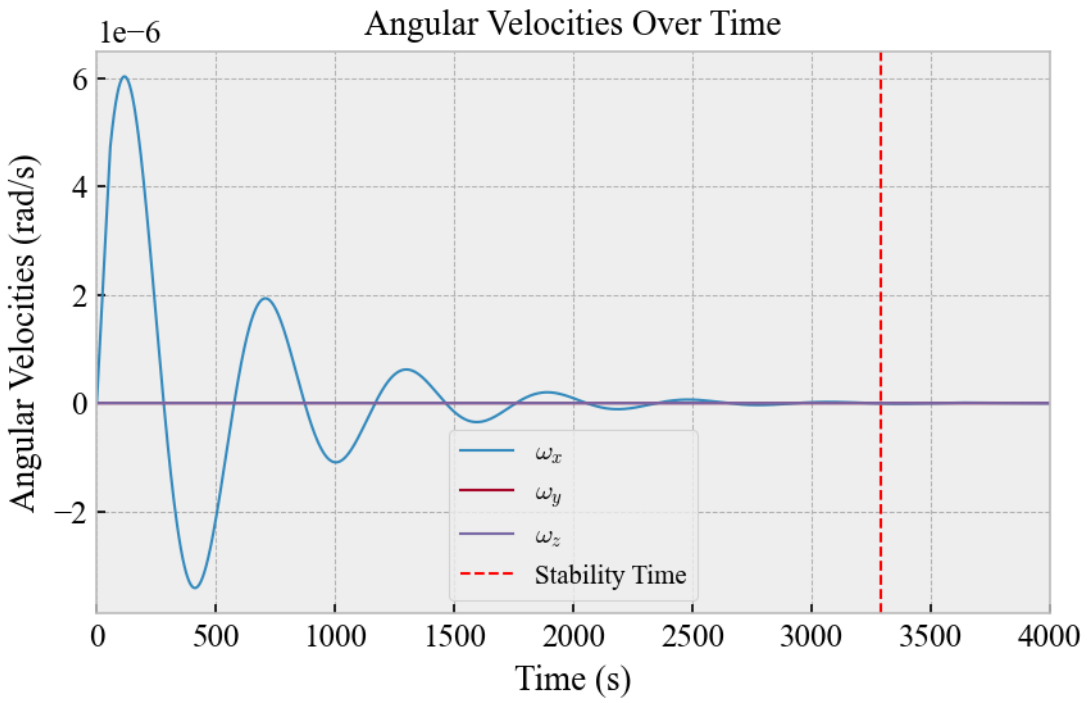

Figure 18: Angular Velocity of Satellite with an External Disturbance, MPC controlled

Figure 18 depicts the satellites angular velocity components, the disturbance torque causes the angular velocity in the x axis to reach a peak of 6×10^{-6}rad/s. The following oscillations shows the controllers inability to handle these control torques, this maybe due to the predictive nature of the model predictive controller. The MPC controller produces a much larger overshoot than the PID controller causing the oscillations, eventually the controller does stabilise the satellite, exerting a constant reactionary torque against the disturbance torque as seen in figure 19. The characteristics of these figures can be described as an exponentially decaying harmonic oscillation.

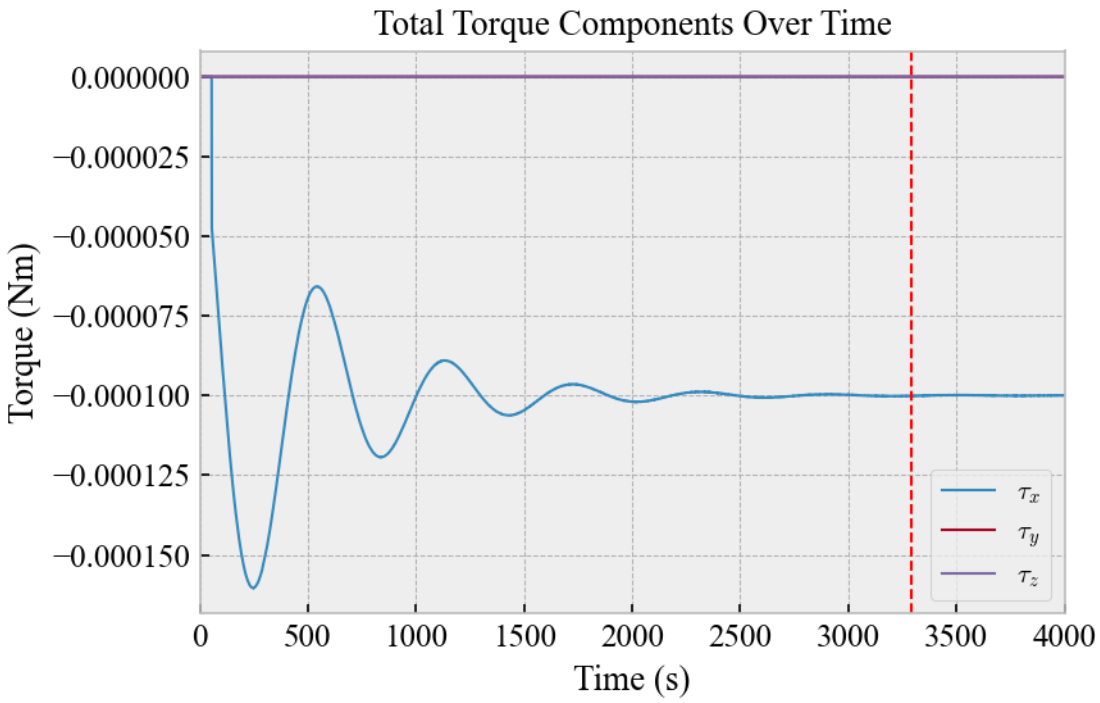

Figure 19: Control Torques Produced by the MPC controller Against External Disturbance

6 Detumbling Simulation

The following section compares the results of the two control methods with their algorithms being adjusted so the set point of the both is a zero angular velocity. The initial angular velocity of the tumbling satellite is fast as mentioned in the methodology above. In a practical application a satellite would implement a detumbling algorithm to stabilise a satellite and then use an attitude control algorithm to re-orientate the satellite back to its desired position. The following PID controller was optimised using the same optimiser methods as the attitude manoeuvre simulations. However the MPC controller in this simulation uses the same parameters as the attitude manoeuvre simulation this was due the large computational cost of the MPC optimiser and would crash after taking too long to find the optimal parameters.

6.1 PID Simulation

The resulting gains from the optimisation method were found to be: $K_p = 30$ $K_i = 1 \times 10^{-6}$ $K_d = 10$. The large proportional gain is required as the controller need to be more aggressive as the satellite has a large inertia matrix to slow the satellite down a strong response is required. The PID detumbling simulation resulted in a final stability time of 1424 seconds as shown in figure 20. The rotation of the satellite can be seen to slow down as the rotation oscillation periods gets larger in the quaternion figure.

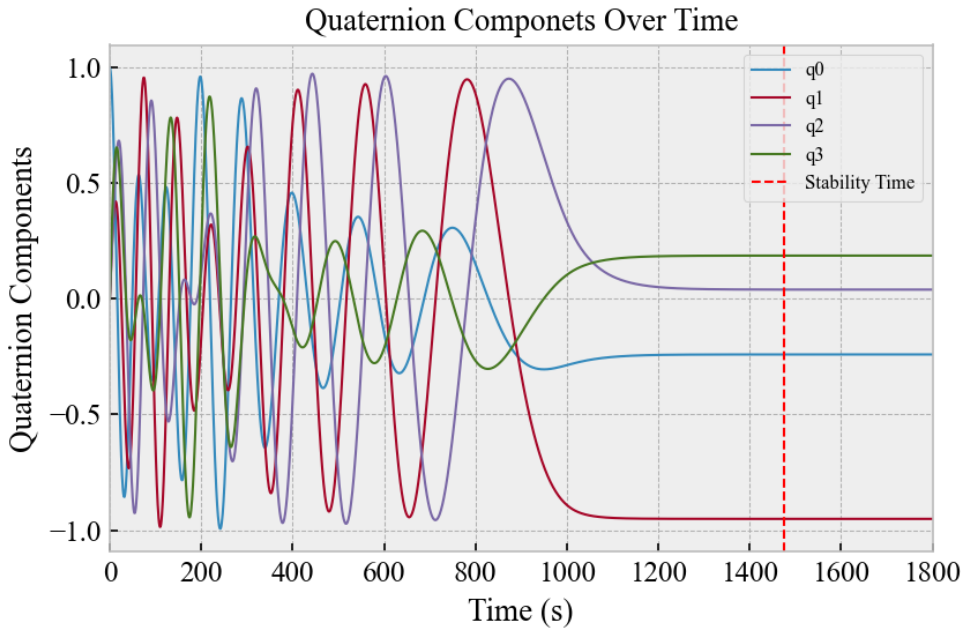

Figure 20: Quaternion Components of the Detumbling Satellite, PID Controlled

The torques produced by the controller are shown in figure 21, the maximum control torques are used in all directions in the first 300 seconds, this is to reduce the amplitude of the angular velocities as observed in figure 22. The control torques eventually tend to zero with τ_z being the final to settle this is expected as the largest initial angular velocity component was ω_z. Overall these results show the PID controllers ability to stabilise a rapidly tumbling satellite.

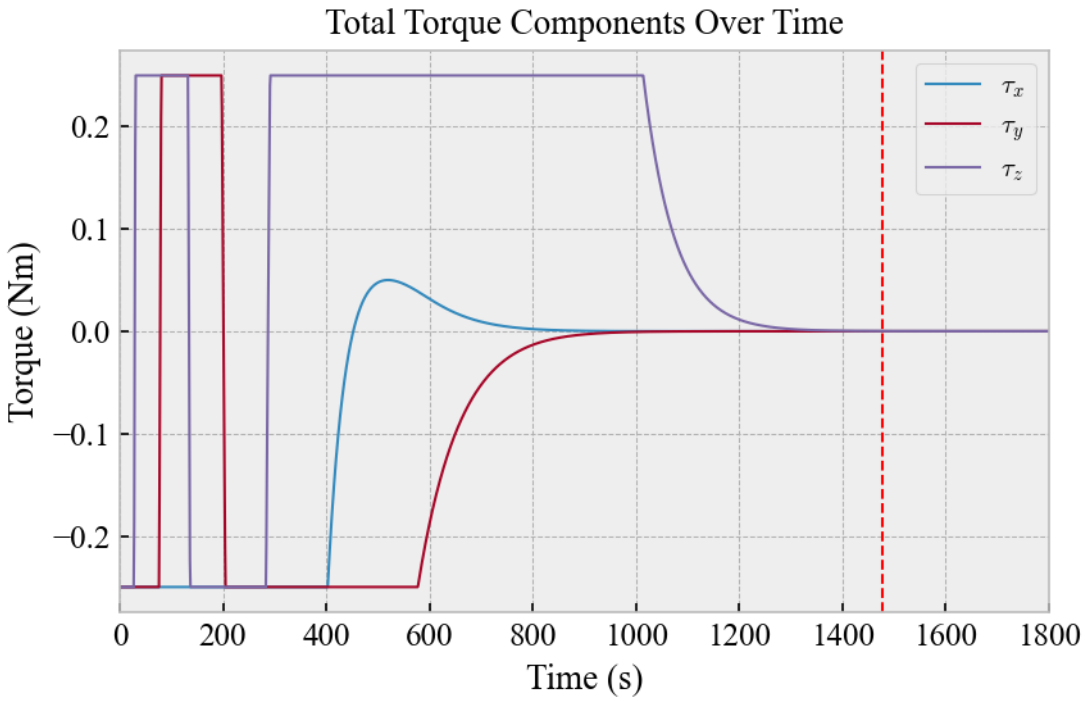

Figure 21: PID Detumbling Simualtion Control Torques

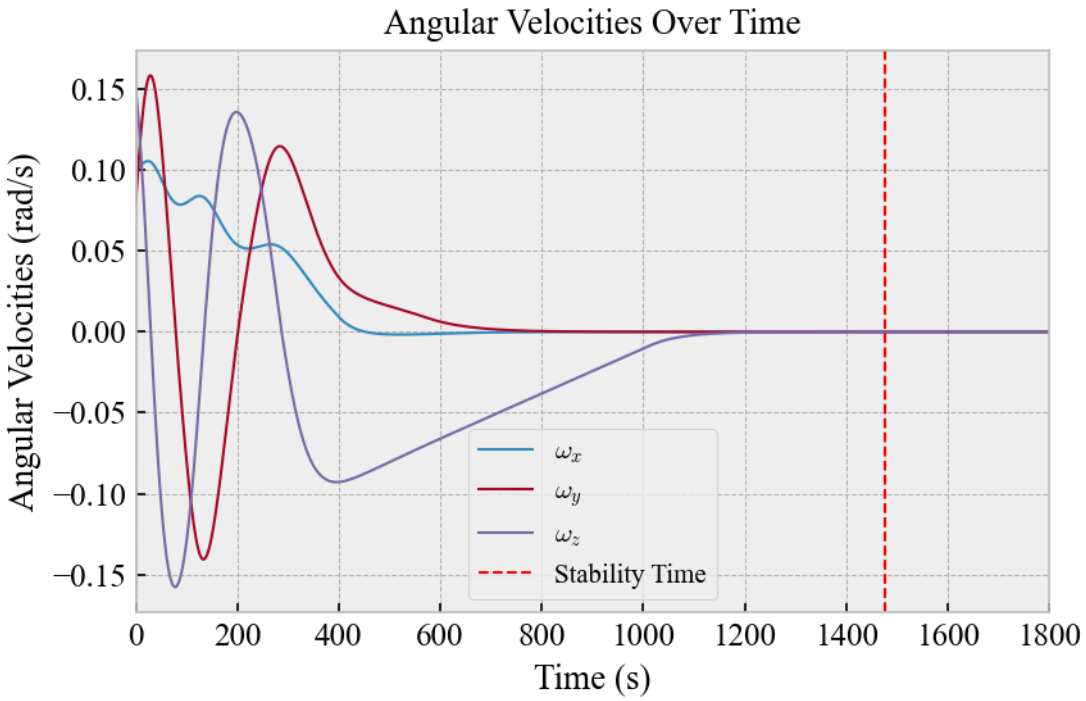

Figure 22: Angular Velocity of the PID Controlled Tumbling Satellite

6.2 MPC Simulation

The quaternion over time graph for the MPC detumble simulation is very similar to the PID controller as can be observed in figure 23. The stabilisation time of the satellite resulted in a time of 1184 seconds, this is 240s faster than the PID controller with its optimised parameters. This indicates that MPC is the superior controller to detumble a satellite working faster than the PID controller.

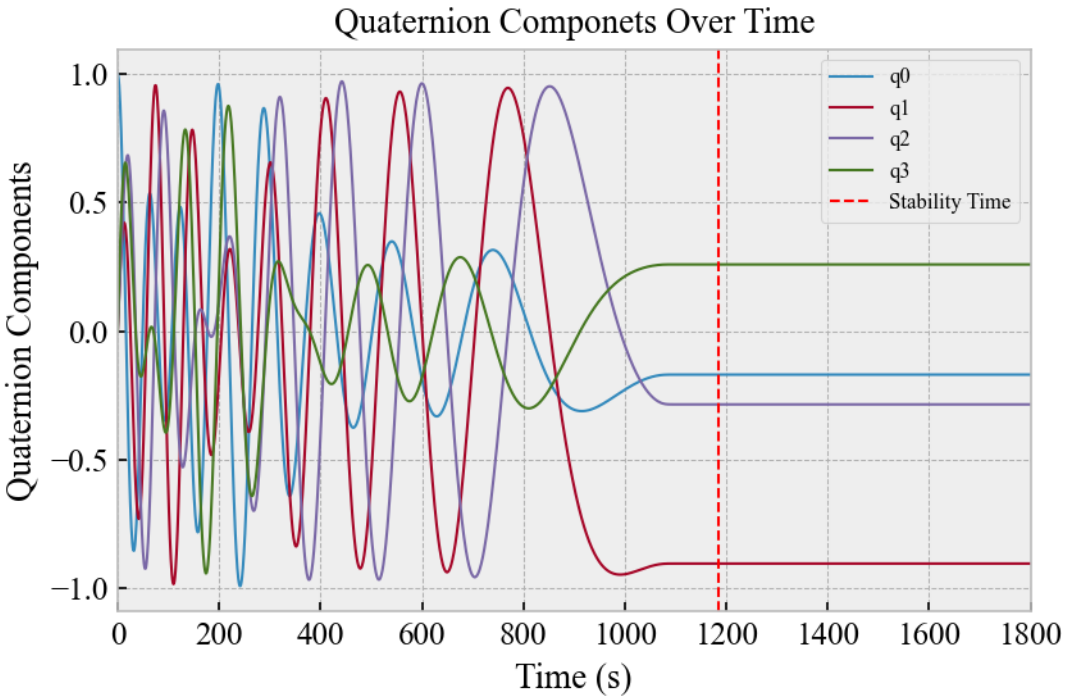

Figure 23: Quaternion Components of the Detumbling Satellite, MPC Controlled

The Torque signal produced by the controller is presented in figure 24 below, the graph is visibly less smooth at around one thousand seconds when compared to the PID control torques. This sharp cut off shown in the control torques is due to the MPC controller adjusting the controller based on the future behavior of the satellite. Hence when it predictes the system is about to settle it reduces the control torques almost instantly. This results in a faster detumbling time for the satellite.

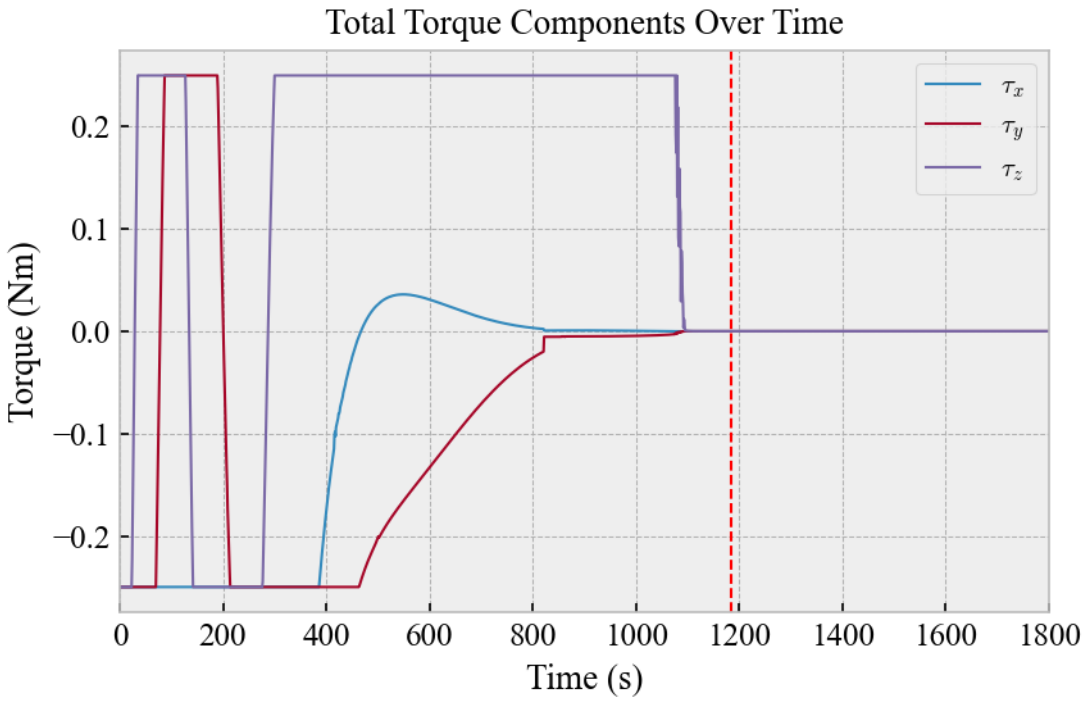

Figure 24: PID Detumbling Simualtion Control Torques

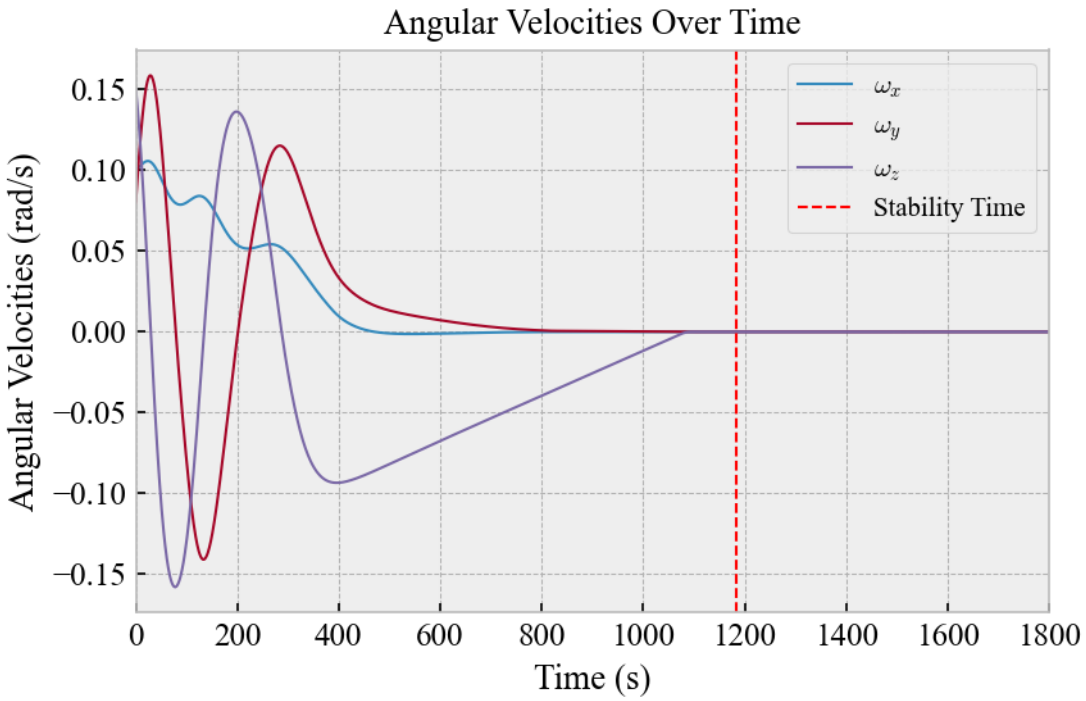

Figure 25: Angular Velocity of the MPC Controlled Tumbling Satellite

7 Summary of Results

This section summarises the results of the simulations conducted in this report, table 6 below presents all the simulations stability times. This information can be used to inform the choice between a PID or MPC controller in the usecase of satellite attitude control. Overall the MPC controller stabilises faster than the PID controller however it did not handle the exposure to external torques unlike the PID controller.

Simulation	Stability Time (s)
PID Attitude Manoeuvre	839.42
PID Attitude Manoeuvre (Optimised Gains)	762.38
MPC Attitude Manoeuvre	535.0
MPC Attitude Manoeuvre (Optimised Parameters)	242.947
PID External Torque	573.287
MPC External Torque	3291.0
PID Detumbling (Optimised Gains)	1476.0
MPC Detumbling	1184.0

Table 6: Summary of Simulation Stability Times

7.1 Attitude Manoeuvre Results Summary

The results and performance of a PID and an MPC controller for satellite attitude control in an undisturbed satellite environment, using both manual and systematic tuning method were presented. In the PID controller case, manual tuning resultted in a completion time of 839.42 seconds with an average overshoot of 1.04%. Conversely the systematically tuned PID controller reduced the stability time to 762.38 and decrease the overshoot to 0.39%.

The manually tuned MPC controller achieved a faster completion time than both PID controller of 535.0 seconds along with a lower overshoot of 0.13%. The optimally tuned MPC further improved the stability time to 224.95 seconds and produced a smoother angular velocity result. Overall, the MPC controller demonstrated superior speed and precision, but at a higher computational cost compared to the PID controller.

7.2 External Disturbance Effect Summary

Overall both controllers were able to react and stabilise against the external disturbance torque using the same controllers and parameters used in the manually tuned attitude manoeuvre simulations. In the presence of external torque the PID controller stabilised quicker and without oscillation unlike the MPC attitude controller. Each controller could be tuned to handle the external torque better however external torques are not constant in reality. Based on these simulations the PID controller proves to be a mores suitable controller for handling disturbance torques. The oscillations in the MPC simulation possibly arise dues to the model used to predict the future dynamics of the satellite being inaccurate. This model does not include a parameter for a disturbance torque, hence when the actual dynamics deviate from the predicted model this leads to the oscillations as it attempts to correct for errors between its prediction and the actual states. Further modelling and tuning would be required to produce better results however in this direct comparison of the two attitude controllers the simpler PID controller performs better in the presence of external torques.

7.3 Detumbling Results Summary

Plotting the performance of the controllers in detumbling a rapidly rotating satellite, with both methods aiming to achieve zero angular velocity. The PID controller was optimized for this task, a stability time of 1424 seconds was found. The MPC controller, using the same parameters as in the previous attitude manoeuvre, stabilized the satellite in 1184 seconds, 240 seconds faster than the PID controller. The MPC's control torques were less smooth, as it aggressively adjusted its outputs based on future behaviour of the satellite but it achieved faster detumbling.

8 Discussion

The results display the MPC controllers inability to handle unmodelled constant disturbances, such as torques resulting from solar radiation pressure. The remaining error present in both controllers was larger in the MPC controller, the error is the offset from the set-point aligned with the LVLH frame for the simulations in section 5. Further Modelling of external torques and improvements to the MPC controller would improve the controllers ability to handle external disturbances. To improve the performance of the PID controller to handle external disturbances the gains would need to be adapted to have a larger response against the external torque. However in reality external disturbances are not constant therefore the gains would not always be optimised to handle the disturbances to the satellite. Using this information an adaptive PID controller model would be more advantageous to address external torques. The results of this report also indicate that the integral term in the satellite attitude controller does not have a large effect on the simulations, as all the PID simulations used the same integral gain of 1×10^{-6}. This

gain is so small the result on the controller is almost insignificant, therefore it assumed a PD controller would be equally suitable for satellite attitude control.

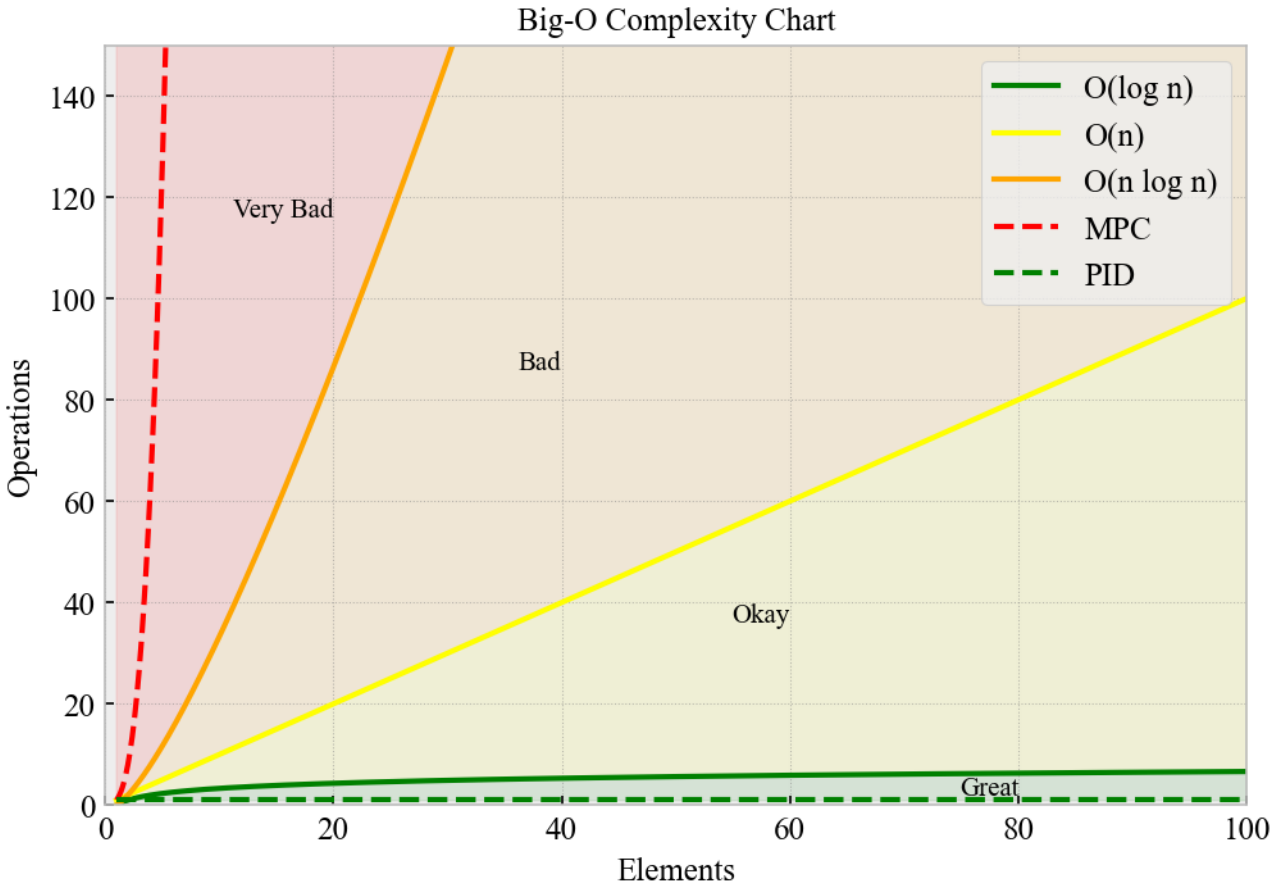

Figure 26: BigO Chart of Computational Complexity of the Control Algorithms

In the other simulations the MPC controller was more effective than the traditional PID controller for satellite attitude control. The computational costs of the two controllers differ due to their individual control strategies and numerical operations. The PID control algorithm operates with a relatively low computational cost per time step, as it involves basic feedback control based on the current quaternion state and angular velocity. The calculations used within the controller are straightforward vector operations. The overall complexity at each time step is constant as the RK4 integration involves a fixed number of operations. In big O notation this is represented as O(1) where the al-

gorithms performance doesn't change with input size. Big O notation is a mathematical way to describe the efficiency of an algorithm in terms of time as the size of the input grows. The MPC controllers predictive and optimisation nature is more computationally demanding as it predicts the behaviour of the system over N whilst optimising control inputs over the control horizon M. At every time step the controller uses the 'SLSQP' method to optimise the cost function, this requires multiple calculations of the systems dynamics through the RK4 method over the prediction horizon. The big O notation for this controller is $O(M^3 + N)$ due to the operations required to solve the optimisation problem, this is compounded by the repeated cost function evaluations over N steps. This makes the MPC method computationally expensive especially if N and M are large however this cost results in a faster controller when compared to the PID method for attitude manoeuvres and detumbling. The difference in computational complexities between the PID and MPC controller can be visualised in figure 26.

To overcome this computational cost Field-Programmable Gate Arrays (FPGAs) hardware can be used in satellite to speed up controller computation times. FPGAs enable parallel processing which speeds up the computation of the optimisation problem in the MPC controller [56]. Utilising this technology engineers can implement real-time MPC for attitude control producing precise results without taxing the satellites limited size and computing power. Therefore complimenting the advanced control attitude control system to meet the requirements of modern space missions.

9 Conclusion

This dissertation created simulations of satellite attitude dynamics from scratch in python, the satellite model was base of the Lunar Reconnaissance Orbiter mission. Two control methods, PID and MPC, where then implemented into the simulation to test their satellite attitude control capabilities. Three scenarios were chosen to model the satellites experiences in orbit while also testing the proficiency of the two control methods. The first scenarios was a small rotation to model the satellite attitude change to point its instruments in a new location. Second scenario adapted the dynamics of the simulation to include an external disturbance torque acting on that satellite that is initially stationary. The final scenario adapts the controllers and the initial angular velocity of the satellite to simulate a tumbling satellite regaining control. The effect of optimal tuning was also investigated by comparing the simulations with manual tuning to systematic tuning algorithms. Overall the results showed the superior performance of the MPC controller which worked faster and more efficiently producing less overshoot in comparison to the PID controller. However the shortcoming of MPC controller was when the satellite was simulated with an external disturbance. The PID controller proved to be simple and robust however if speed is vital for a spacecrafts mission the MPC controller would be a stronger choice.

9.1 Future Work

The results found in the simulation in this report has presented opportunities for future work and investigations. Future alternations to the simulations and models to make them more detailed could be investigated to investigate how the MPC model could be

adapted to handle external torques. Saturation and the issues surrounding it was left out of the scope of this report however future work can be conducted to learn more about the topic. Research into solar radiation pressure being actively used to control a spacecrafts angular momentum through solar sailing has been conducted by T. Ito et al [57], further work based on similar research could be conducted to study the feasibility of using solar radiation pressure do control and desaturate a satellite. This active used of solar radiation pressure would be important for further deep space exploration. To improve the simulations in future extra detail can be implemented into the reaction wheel control systems to simulate the electronics and dynamics of the reaction wheel system.

One area of research which has gained popularity in multiple areas is artificial intelligence (AI) including control systems engineering, where reinforcement learning has become more popular [58]. Applications in adaptive PID control has already been studied as AI can enable controllers to adapt to changing conditions and system dynamics [59]. The development of control algorithms requires a lot of expertise from control engineers along with the tuning of the parameters for optimal performance can also be challenging as discussed in this report. These challenges may be more approachable with the use of reinforcement learning and machine learning to autonomously learn and calculate the optimal control inputs without direct input from engineers. Through training, reinforcement learning based controllers have the potential to outperform traditional control methods. Therefore future work should be conducted into utilising artificial intelligence and machine learning methods to produce better performing attitude controllers.

References

[1] Goel, P.S., Srinivasan, P.N. and Malik, N.K. (1978). The stabilisation system. Proceedings of the Indian Academy of Sciences Section C Engineering Sciences, 1(2), pp.135–143. doi:https://doi.org/10.1007/bf02843539.

[2] Santoni, F. and Zelli, M. (2009). Passive magnetic attitude stabilization of the UNISAT-4 microsatellite. Acta Astronautica, 65(5-6), pp.792–803. doi:https://doi.org/10.1016/j.actaastro.2009.03.012.

[3] Miniature Inertial Measurement Unit. (2018). Space Microsystems and Micro/nano Satellites, pp.233–293. doi:https://doi.org/10.1016/b978-0-12-812672-1.00007-2.

[4] None Ping Wang and Y.B. Shtessel (1998). Satellite attitude control using only magnetorquers. doi:https://doi.org/10.1109/acc.1998.694663.

[5] NASA Selects International Space Station US Deorbit Vehicle - NASA. [online] Available at: https://www.nasa.gov/news-release/nasa-selects-international-space-station-us-deorbit-vehicle/.

[6] Minorsky., N. (2009). DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES. Journal of the American Society for Naval Engineers, 34(2), pp.280–309. doi:https://doi.org/10.1111/j.1559-3584.1922.tb04958.x.

[7] Zadeh, L. and Whalen, B. (1962). On optimal control and linear programming. IRE Transactions on Automatic Control, 7(4), pp.45–46. doi:https://doi.org/10.1109/tac.1962.1105469.

[8] Propoi, A.I. (1963). Use of linear programming methods for synthesizing sampled-data automatic systems. 24(7), pp.837–844.

[9] Ribeiro, C.H.P., Miyoshi, S.C., Secchi, A.R. and Bhaya, A. (2016). Model Predictive Control with quality requirements on petroleum production platforms. Journal of Petroleum Science and Engineering, 137, pp.10–21. doi:https://doi.org/10.1016/j.petrol.2015.11.004.

[10] Liu, C., Lee, S., Varnhagen, S. and Tseng, H.E. (2017). Path planning for autonomous vehicles using model predictive control. [online] IEEE Xplore. doi:https://doi.org/10.1109/IVS.2017.7995716.

[11] Archive.org. (2024). . Lunar Reconnaissance Orbiter . [online] Available at: https://web.archive.org/web/20130214124135/http://lunar.gsfc.nasa.gov/launch.html [Accessed 18 Aug. 2024].

[12] Donahue, K. (2015). CRaTER Cosmic Ray Telescope. [online] Unh.edu. Available at: https://crater.unh.edu/lro_payload.shtml [Accessed 18 Aug. 2024].

[13] Li, S., Lucey, P.G., Milliken, R.E., Hayne, P.O., Fisher, E., Williams, J.-P., Hurley, D.M. and Elphic, R.C. (2018). Direct evidence of surface exposed water ice in the lunar polar regions. Proceedings of the National Academy of Sciences, 115(36), pp.8907–8912. doi:https://doi.org/10.1073/pnas.1802345115.

[14] Kokhanov, A.A., Karachevtseva, I.P., Zubarev, A.E., Patraty, V., Rodionova, Zh.F. and Oberst, J. (2018). Mapping of potential lunar landing areas using LRO and SELENE data. Planetary and Space Science, 162, pp.179–189. doi:https://doi.org/10.1016/j.pss.2017.08.002.

[15] Dyal, P., Parkin, C.W. and Daily, W. (1974). Magnetism and the interior of the Moon. Reviews of Geophysics, 12(4), pp.568–568. doi:https://doi.org/10.1029/rg012i004p00568.

[16] Calhoun, P.C. and Garrick, J.C. (2007). Observing mode attitude controller for the Lunar Reconnaissance Orbiter.

[17] Cremonese, G., Borin, P., Lucchetti, A., Marzari, F. and Bruno, M. (2013). Micrometeoroids flux on the Moon. Astronomy & Astrophysics, 551, p.A27. doi:https://doi.org/10.1051/0004-6361/201220541.

[18] Isro.gov.in. (2023). Current Space Situation around the Moon – An assessment. [online] Available at: https://www.isro.gov.in/Current_Space_Situation_around_Moon_Assessment.html: :text=Majorityof%20the%20currentlyorbiting%20lunar%20probes%20operate%20in%20LLO.&text=As%20of%20July%202023%2C%20there [Accessed 18 Aug. 2024].

[19] Gunter's Space Page. (2024). Tiandu 1, 2. [online] Available at: https://space.skyrocket.de/doc_sdat/tiandu-1.htm [Accessed 18 Aug. 2024].

[20] XU, L., LI, H., PEI, Z., ZOU, Y. and WANG, C. (2022). A Brief Introduction to the International Lunar Research Station Program and the Interstellar Express Mission. Chinese Journal of Space Science, 42(4), p.511. doi:https://doi.org/10.11728/cjss2022.04.yg28.

[21] Smith, M., Craig, D., Herrmann, N., Mahoney, E., Krezel, J., McIntyre, N. and Goodliff, K. (2020). The Artemis Program: An Overview of NASA's Activities to Return Humans to the Moon. IEEE Xplore. doi:https://doi.org/10.1109/AERO47225.2020.9172323.

[22] W. David Woods (2011). How Apollo Flew to the Moon. Springer Science Business Media. Chapter 11.

[23] www.rmg.co.uk. (n.d.). Apollo 11 Moon landing: minute by minute. [online] Available at: https://www.rmg.co.uk/stories/topics/apollo-11-moon-landing-minute-minute.

[24] Scorsoglio, A., Furfaro, R., Linares, R. and Gaudet, B. (2020). Image-based Deep Reinforcement Learning for Autonomous Lunar Landing. AIAA Scitech 2020 Forum. doi:https://doi.org/10.2514/6.2020-1910.

[25] Vladimir Aslanov and Ledkov, A. (2023). Basics of space flight mechanics and control theory. Elsevier eBooks, pp.1–52. doi:https://doi.org/10.1016/b978-0-323-99299-2.00003-3.

[26] Newman, C., Hollister, J., Davis, D. and Zimovan-Spreen, E. (n.d.). AAS 22-728 INVESTIGATING SOLAR RADIATION PRESSURE MODELING FOR OPERATIONS IN NEAR RECTILINEAR HALO ORBIT. [online] Available at: https://ntrs.nasa.gov/api/citations/20220010382/downloads/NRHO_SRP.pdf.

[27] NASA. The Sunshield Webb/NASA. [online] webb.nasa.gov. Available at: https://webb.nasa.gov/content/observatory/sunshield.html.

[28] Li, H. and Williams, T. (2006). Reconfiguration of Sun-Earth Libration Point Formations Using Solar Radiation Pressure. Journal of Spacecraft and Rockets, 43(6), pp.1328–1339. doi:https://doi.org/10.2514/1.16348.

[29] N Pratiwi and D Herdiwijaya (2022). Solar radiation pressure on LAPAN A1 satellite due to extreme geomagnetic storm. Journal of Physics Conference Series, 2243(1), pp.012013–012013. doi:https://doi.org/10.1088/1742-6596/2243/1/012013.

[30] Floberghagen, R., Visser, P. and Weischede, F. (1999). Lunar albedo force modeling and its effect on low lunar orbit and gravity field determination. Advances in Space Research, 23(4), pp.733–738. doi:https://doi.org/10.1016/s0273-1177(99)00155-6.

[31] Capó-Lugo, P.A. and Bainum, P.M. (2011). Environmental and actuator torques. Elsevier eBooks, pp.121–154. doi:https://doi.org/10.1533/9780857093875.121.

[32] Cochran, J. and Junkinst, J. LARGE ANGLE SATELLITE ATITITUDE MANEUVERS. [online] Available at: https://ntrs.nasa.gov/api/citations/19760003092/downloads/19760003092.pdf [Accessed 18 Aug. 2024].

[33] Dong, J. (2023). Performance comparison and analysis of traditional PID and fuzzy PID control applied to UAV. Journal of Physics Conference Series, 2649(1), pp.012001–012001. doi:https://doi.org/10.1088/1742-6596/2649/1/012001.

[34] Hu, Q., Xie, J. and Wang, C. (2019). Dynamic path planning and trajectory tracking using MPC for satellite with collision avoidance. ISA Transactions, 84, pp.128–141. doi:https://doi.org/10.1016/j.isatra.2018.09.020.

[35] Zhang, X., Ling, K.V., Lu, Z., Zhang, X., Liao, W. and Lim, W.S. (2021). Piece-wise affine MPC-based attitude control for a CubeSat during orbital manoeuvres. Aerospace Science and Technology, 118, pp.106997–106997. doi:https://doi.org/10.1016/j.ast.2021.106997.

[36] Golzari, A., Nejat Pishkenari, H., Salarieh, H. and Abdollahi, T. (2020). Quaternion based linear time-varying model predictive attitude control for satellites with two reaction wheels. Aerospace Science and Technology, 98, p.105677. doi:https://doi.org/10.1016/j.ast.2019.105677.

[37] Ławryńczuk, M. and Nebeluk, R. (2021). Computationally Efficient Nonlinear Model Predictive Control Using the L1 Cost-Function. Sensors, 21(17), p.5835. doi:https://doi.org/10.3390/s21175835.

[38] Morari, M. and H. Lee, J. (1999). Model predictive control: past, present and future. Computers & Chemical Engineering, 23(4-5), pp.667–682. doi:https://doi.org/10.1016/s0098-1354(98)00301-9.

[39] Bemporad, A., Morari, M., Dua, V. and Pistikopoulos, E.N. (2000). The explicit solution of model predictive control via multiparametric quadratic programming. doi:https://doi.org/10.1109/acc.2000.876624.

[40] Schwenzer, M., Ay, M., Bergs, T. and Abel, D. (2021). Review on model predictive control: an engineering perspective. The International Journal of Advanced Manufacturing Technology, 117(5-6), pp.1327–1349. doi:https://doi.org/10.1007/s00170-021-07682-3.

[41] Zhang, X., Ling Keck Voon, Lim Wee Seng, Chong Kwang Tong, Lu, Z., Zhang, X. and Liao, W. (2020). The MPC Attitude Control with Integration for VELOX-II. doi:https://doi.org/10.23919/ccc50068.2020.9189274.

[42] Ziegler, J.G. and Nichols, N.B. (1993). Optimum Settings for Automatic Controllers. Journal of Dynamic Systems, Measurement, and Control, [online] 115(2B), pp.220–222. doi:https://doi.org/10.1115/1.2899060.

[43] Åström, K.J. and Hägglund, T. (1984). Automatic Tuning of Simple Regulators. IFAC Proceedings Volumes, [online] 17(2), pp.1867–1872. doi:https://doi.org/10.1016/S1474-6670(17)61248-5.

[44] Darby, M.L., Harmse, M. and Nikolaou, M. (2009). MPC: Current Practice and Challenges. IFAC Proceedings Volumes, 42(11), pp.86–98. doi:https://doi.org/10.3182/20090712-4-tr-2008.00014.

[45] Milman, R. and Davison, E.J. (2008). A Fast MPC Algorithm Using Nonfeasible Active Set Methods. Journal of Optimization Theory and Applications, 139(3), pp.591–616. doi:https://doi.org/10.1007/s10957-008-9413-3.

[46] Hemingway, E.G. and O'Reilly, O.M. (2018). Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. Multibody System Dynamics, 44(1), pp.31–56. doi:https://doi.org/10.1007/s11044-018-9620-0.

[47] Kershner, R.B. and Fischell, R.E. (1965). Gravity-Gradient Stabilization of Earth Satellites. IFAC Proceedings Volumes, 2(1), pp.249–266. doi:https://doi.org/10.1016/s1474-6670(17)69092-x.

[48] Numpy (2009). NumPy. [online] Numpy.org. Available at: https://numpy.org/.

[49] SciPy (2020). SciPy.org — SciPy.org. [online] Scipy.org. Available at: https://scipy.org/.

[50] Matplotlib (2012). Matplotlib: Python plotting — Matplotlib 3.1.1 documentation. [online] Matplotlib.org. Available at: https://matplotlib.org/.

[51] Karpenko, M., Lippman, T., Ross, I., Halverson, J., Mcclanahan, T., Barker, M., Mazarico, E., Besser, R., Dennehy, C., Vanzwieten, T. and Wolf, A. (n.d.). FAST ATTITUDE MANEUVERS FOR THE LUNAR RECONNAISSANCE ORIBTER. [online] Available at: https://ntrs.nasa.gov/api/citations/20190000640/downloads/20190000640.pdf [Accessed 18 Aug. 2024].

[52] LRO Spacecraft Description - PDS Geosciences Node. (2022). Available at: https://pds-geosciences.wustl.edu/missions/lro/docs/lro_insthost.txt.

[53] Drolshagen, G. (2008). Impact effects from small size meteoroids and space debris. Advances in Space Research, [online] 41(7), pp.1123–1131. doi:https://doi.org/10.1016/j.asr.2007.09.007.

[54] Reichel, F. (2012). Attitude Control System of UWE-3 : Design, Testing and Verification.

[55] Seborg, D.E., Edgar, T.F., Mellichamp, D.A. and Doyle, F.J. (2017). Process Dynamics and Control. 4th ed. Hoboken, Nj: Wiley. Chapter 20. pp.384.

[56] Stellato, B. and Goulart, P.J. (2016). Real-time FPGA implementation of direct MPC for power electronics. Oxford University Research Archive (ORA) (University of Oxford). doi:https://doi.org/10.1109/cdc.2016.7798474.

[57] Ito, T., Ikari, S., Ryu Funase, Sakai, S., Yasuhiro Kawakatsu, Atsushi Tomiki and Takaya Inamori (2018). Active use of solar radiation pressure for angular momentum control of the PROCYON micro-spacecraft. Acta Astronautica, 152, pp.299–309. doi:https://doi.org/10.1016/j.actaastro.2018.08.009.

[58] Dogru, O., Velswamy, K., Ibrahim, F., Wu, Y., Sundaramoorthy, A.S., Huang, B., Xu, S., Nixon, M. and Bell, N. (2022). Reinforcement learning approach to autonomous PID tuning. Computers & Chemical Engineering, 161, p.107760. doi:https://doi.org/10.1016/j.compchemeng.2022.107760.

[59] Lisitsyn, A.N. and Zadorozhnaya, N.M. (2019). Adaptive Wind Turbine PID Controller Tuner Algorithm with Elements of Artificial Intelligence. Procedia Computer Science, 150, pp.591–596. doi:https://doi.org/10.1016/j.procs.2019.02.098.