Reaction Wheel Attitude Control

Control & Systems Final Project

Pucciarelli Nahuel
nahuelpucciarelli@gmail.com

Control & Systems, Faculty of Engineering
National University of Cuyo
Mendoza, Argentina

December 2025

Abstract

Reaction wheel attitude control systems allow satellites to change orientation without expending fuel by using the angular momentum of spinning flywheels. This paper presents a simplified model to illustrate the working principles of such systems and examines the control strategies required to achieve and maintain a desired orientation while rejecting external disturbances.

1 Introduction

In the aerospace field, fuel consumption associated with control maneuvers is a critical concern due to its high cost and limited availability in orbit. To address this challenge, reaction wheel-based attitude control systems are commonly employed. These systems generate precise and efficient angular motion without expending propellant, making them ideal for satellite applications.

This project aims to simulate the behavior of 3 reaction wheels mounted on a CubeSat satellite. The system's evolution will be analyzed for various initial states, and a controller will be designed to accurately position and stabilize the satellite's angular orientation. The main objective is to develop a functional model that captures the fundamental dynamics and control challenges of a reaction wheel system.

Even though reaction wheel systems are widely covered in the literature, this work provides a pedagogical approach to understanding their fundamental operation. It serves as a foundation for future extensions involving more complex satellite dynamics and control.

2 Development

2.1 Single reaction wheel model

Analyzing the behavior of a satellite with reaction wheels begins with a simplified first approximation: modeling it as a rigid, uniformly distributed cube with a single reaction wheel aligned with one of its principal axes. This model considers rotation about one axis only, effectively reducing the system to one rotational degree of freedom.

The system consists of two main components:

1. The satellite structure, treated as a rigid body with moment of inertia I_{sat} about the rotation axis.
2. The reaction wheel, a motor-driven flywheel with its own moment of inertia I_{rw}, capable of generating torque.

Under the assumption of no external disturbances, the total angular momentum of the system is conserved. This yields the fundamental dynamic relationship between the angular velocity of the satellite ω and the angular velocity of the wheel ω_{rw}:

$$I_{\text{sat}}\,\omega(t) + I_{\text{rw}}\,\omega_{\text{rw}}(t) = \text{k}\,(\text{constant}), \tag{1a}$$

$$\Delta\omega = -\frac{I_{\text{rw}}}{I_{\text{sat}}}\Delta\omega_{\text{rw}} \tag{1b}$$

The latter equation illustrates that any change in the wheel's angular velocity results in an opposite change in the satellite's angular velocity. In practice, the wheel is actuated by a torque τ, which is used as the control input to drive the system toward a desired orientation.

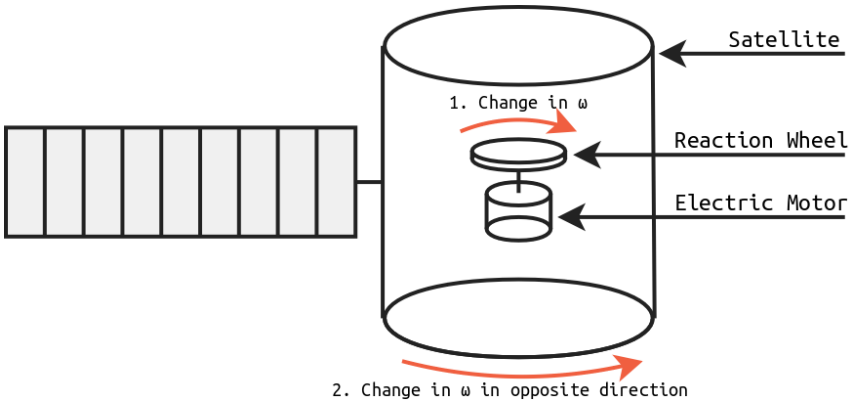


Figure 1: Diagram of a reaction wheel [1]

2.1.1 Equations of motion

Because the system only models the one axis of rotation, there is no gyroscopic coupling. This effect will be considered when additional flywheels are modeled.

Assuming torque τ is applied to the flywheel, Newton's second law is applied:

$$\tau(t) = I_{\text{rw}}\,\dot{\omega}_{\text{rw}}(t) \tag{2a}$$

$$\tau_c(t) = -\tau(t), \tag{2b}$$

where τ_c is the control torque applied to the satellite body.

Replacing the terms in the equation 1b yields:

$$\dot{\omega}(t) = \frac{1}{I_{\text{sat}}}\tau_c(t)\,, \text{ with } \omega(t) = \dot{\theta}(t) \tag{3}$$

This simple model can be expressed as:

$$\begin{cases} \dot{\theta}(t) = \omega(t) \\ \dot{\omega}(t) = \dfrac{1}{I_{\text{sat}}}\,\tau_c(t) \end{cases} \quad \Longrightarrow \quad \frac{d}{dt}\begin{bmatrix}\theta(t)\\ \omega_{\text{sat}}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta(t)\\ \omega_{\text{sat}}(t)\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{1}{I_{\text{sat}}}\end{bmatrix}\tau_c(t) \tag{4a}$$

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}}_{A}\mathbf{x}(t) + \underbrace{\begin{bmatrix}0\\ \dfrac{1}{I_{\text{sat}}}\end{bmatrix}}_{B}u(t) \tag{4b}$$

2.1.2 Satellite and wheel parameters

The satellite is assumed to be cubic with a uniform density of $1762\,\mathrm{kg/m^3}$ [2]. The mass moment of inertia can be calculated through a single variable: the mass of the satellite, equal to $2\,\mathrm{kg}$. The dimensions of a 1U CubeSat are $10\,\mathrm{cm} \times 10\,\mathrm{cm} \times 11.35\,\mathrm{cm}$ [3]. The moment of inertia is calculated for rotation about the x, y and z-axes:

$$I_{\text{sat},\,x} = \frac{1}{12} m(L_y^2 + L_z^2) = 0.00381371\,\mathrm{kg\,m^2} \tag{5a}$$

$$I_{\text{sat},\,y} = \frac{1}{12} m(L_x^2 + L_z^2) = 0.00381371\,\mathrm{kg\,m^2} \tag{5b}$$

$$I_{\text{sat},\,z} = \frac{1}{12} m(L_x^2 + L_y^2) = 0.003333\,\mathrm{kg\,m^2} \tag{5c}$$

For simplicity, only the diagonal values of the inertia matrix are considered.

The reaction wheel parameters are based on a Serenum commercial model [4]:

- mass of $40\,\mathrm{g}$,
- total momentum storage of $0.6\,\mathrm{mN\,m\,s}$,
- saturation speed of 5600 rpm in both directions,
- maximum torque of $0.2\,\mathrm{mN\,m}$.

These are typical values for a small reaction wheel designed for a 1U type CubeSat.

2.1.3 Disturbances and saturation

Even in space, a satellite is continuously subjected to internal and external torques that, if unaccounted, can degrade the pointing accuracy of cameras, sensors and antennas.

Typical disturbance sources include internal moving components, gravity-gradient effects, solar radiation pressure, atmospheric drag in low Earth orbit, and interactions with Earth's magnetic field. For the model, disturbances are represented as an external torque acting for a limited duration, simulating for example a brief physical interaction with the spacecraft.

Typical continuous environmental disturbances for a 1U CubeSat are:

- Solar pressure, $\sim 10^{-9}\,\mathrm{N\,m}$
- Gravity gradients, $\sim 10^{-8}\,\mathrm{N\,m}$
- Aerodynamic drag, $\sim 10^{-6}\,\mathrm{N\,m}$ (dominant below 500 km)

A consequence of external torques is reaction wheel saturation. Since a reaction wheel has finite angular momentum capacity, persistent or strong disturbances can drive the wheel toward its maximum speed, beyond which it can no longer generate corrective torque.

$$|I_{\text{rw}}\,\omega_{\text{rw}}(t)| \geq I_{\text{rw}}\,\omega_{\text{rw-max}} \quad \Longrightarrow \quad \tau_c(t) = 0$$

Internal actuation, such as reorienting the satellite, merely redistributes momentum between the body and the wheel. Once such a maneuver ends and the body is stabilized ($\omega = 0$), the reaction wheel speed returns to its nominal value.

The saturation limitation underscores the importance of including de-saturation mechanisms in real missions, such as magnetorquers or thrusters. The disturbance T_{ext} is modeled, in this case, only for one axis:

$$\dot{\omega}(t) = \frac{1}{I_{\text{sat}}}(\tau_c(t) + T_{\text{ext}}(t)) \tag{6}$$

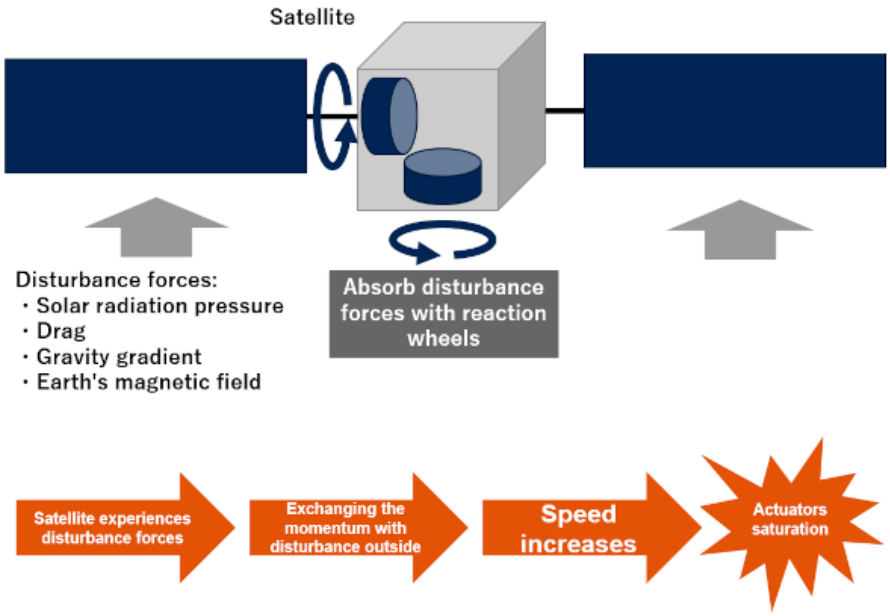


Figure 2: Disturbance forces in a satellite and the process of absorption of moment and saturation [5]

2.1.4 Simulation

The 1-DOF model derived in Equation 6 was implemented in MATLAB/Simulink to validate the fundamental principles of momentum exchange. To model the hardware limitations, two constraints were implemented in the control loop:

- Torque saturation: the commanded torque is clamped to $\pm 0.2\,\text{mN m}$ representing the peak torque capability of the motor driver.
- Momentum saturation: a logic-based switch prevents the application of torque when the wheel speed exceeds $5600\,\text{RPM}$, simulating the physical speed limit of the flywheel.

To evaluate the system's performance, a test scenario was designed containing two distinct phases:

1. Slew maneuver ($t = 5\,\text{s}$): a step command is issued to rotate the satellite from rest ($0°$) to a target attitude of $15°$. This tests the actuator's ability to generate internal torque to reorient the spacecraft.
2. Disturbance rejection ($t = 30\,\text{s}$): a disturbance torque of $0.26\,\text{mN m}$ is applied for a duration of $2\,\text{s}$, testing the controller's ability to maintain the pointing requirement. The simulation presents a significantly higher magnitude disturbance, selected to represent a worst-case event (like a micrometeoroid of $1\,\text{mg}$, traveling at $10\,\text{km/s}$).

A simple PD controller was tuned to provide adequate damping and eliminate overshoot. The following results were obtained:

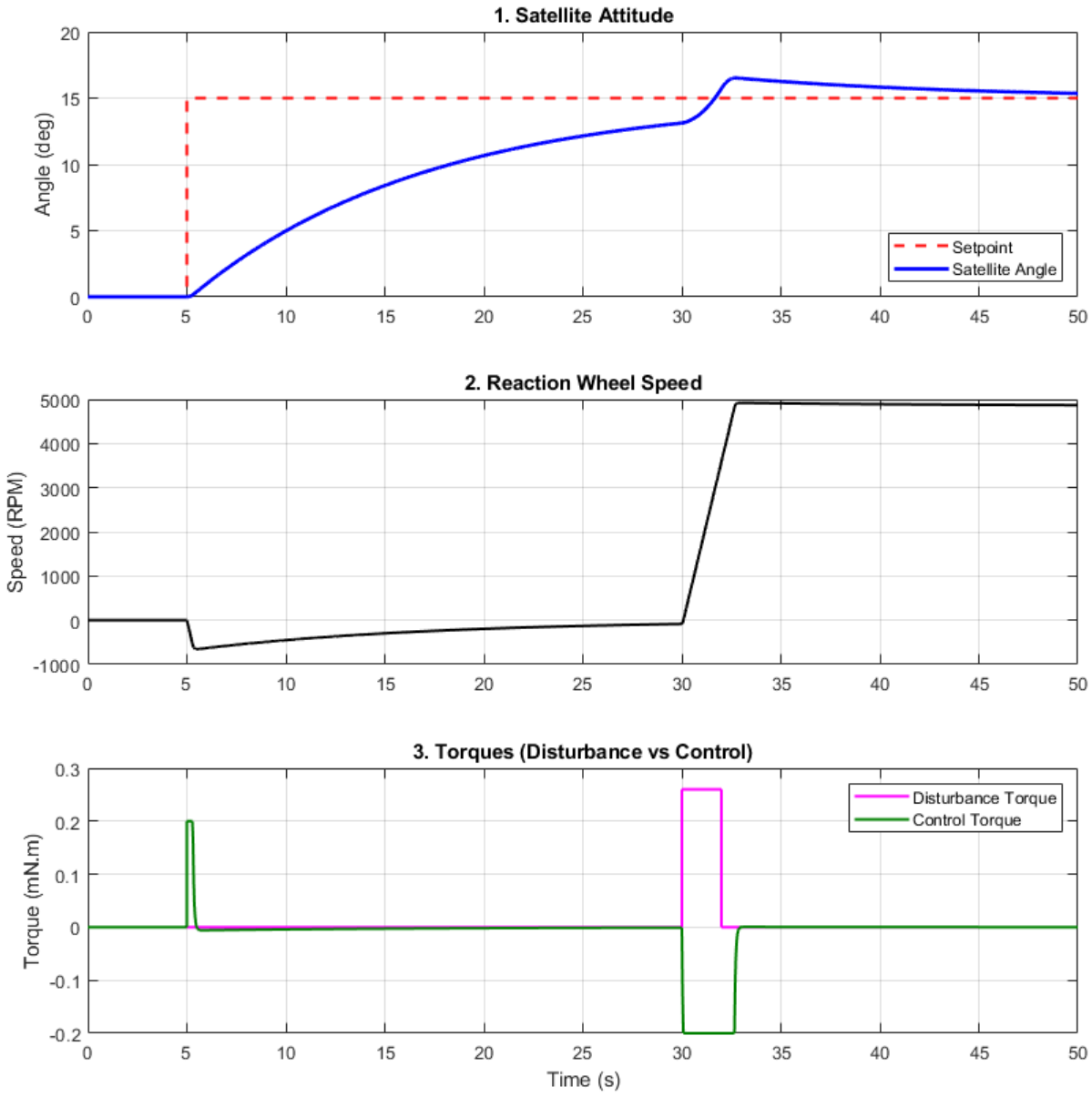


Figure 3: 1 DOF simulation

The simulation results show the reaction wheel effectively reorienting the satellite to the target attitude. The maneuver duration is non-critical in this context, prioritizing energy efficiency and control authority over agility. Upon the application of an external disturbance, the wheel accelerates to generate a counter-torque, successfully restoring the system's orientation. Notably, the wheel absorbs the injected angular momentum, driving its speed close to the saturation limit.

With the fundamental mechanics of a 1-DOF system validated, the analysis can now advance to a more complex model.

2.2 Multiple reaction wheel model

This section focuses on the effects that appear when more than one reaction wheel is present. The core principles remain unchanged. However, the system uses 3 mechanically identical flywheels, aligned with the main axes of the satellite body. Disturbance torques can be applied in one or multiple axis.

2.2.1 Coupling, gyroscopic effects and quaternions kinematics

The attitude dynamics require tracking not only angular velocity ω but also the spacecraft orientation. For non-linear attitude dynamics, quaternions (q) are preferred over Euler angles as they are non-singular for all rotations. A quaternion is a four-element vector composed of a scalar part (q_0) and a vector part (q_v):

$$\mathrm{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ q_v \end{pmatrix} \tag{7}$$

Quaternions relate the satellite's angular velocity vector ω to the time derivative of the orientation $\dot{\mathrm{q}}$ through a kinematic transformation:

$$\dot{\mathrm{q}} = \frac{1}{2}\Omega(\omega)\mathrm{q} \tag{8}$$

$\Omega(\omega)$ represents a 4x4 skew-symmetric matrix:

$$\Omega(\omega) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix} \tag{9}$$

$\omega_{x,y,z}$ are the angular velocity components of the satellite. The kinematic relationship is non-linear but non singular, and requires four variables.

The vector equation governing the satellite's angular acceleration ($\dot{\omega}$) in the body frame is obtained by applying the Newton-Euler rotational law to the total angular momentum of the satellite and its reaction wheels:

$$I_{\text{sat}}\dot{\omega} + \omega \times (I_{\text{sat}}\omega + h_{\text{rw}}) + \dot{h}_{\text{rw}} = T_{\text{ext}} \tag{10}$$

I_{sat} is the inertia matrix (diagonal). h_{rw} is the total angular momentum vector of the reaction wheels, where each row is the momentum stored in the x, y and z-axis wheel:

$$h_{\text{rw}}(t) = \begin{pmatrix} I_{\text{rw},\,x}\,\omega_{\text{rw},\,x} \\ I_{\text{rw},\,y}\,\omega_{\text{rw},\,y} \\ I_{\text{rw},\,z}\,\omega_{\text{rw},\,z} \end{pmatrix}$$

By rearranging to solve for the satellite's angular acceleration:

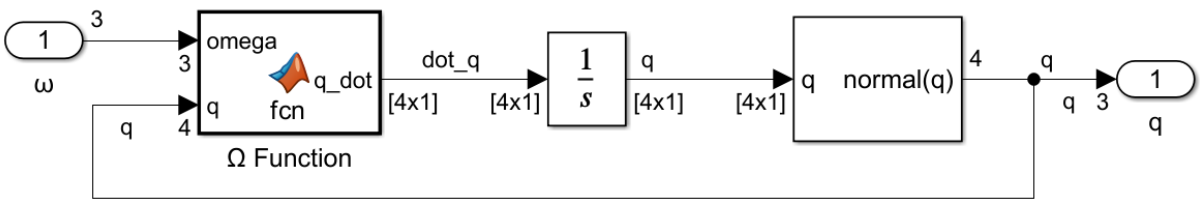
$$I_{\text{sat}}\dot{\omega} = T_{\text{ext}} + \tau_c - \omega \times (I_{\text{sat}}\omega + h_{\text{rw}}) \tag{11}$$

$\tau_c = -\dot{h}_{\text{rw}}$ is the control torque. The last term is the gyroscopic coupling torque, which is an internal and non-linear disturbance: $\omega \times (I_{\text{sat}}\,\omega)$ is the body inertia coupling and $\omega \times h_{rw}$ is the wheel momentum coupling.

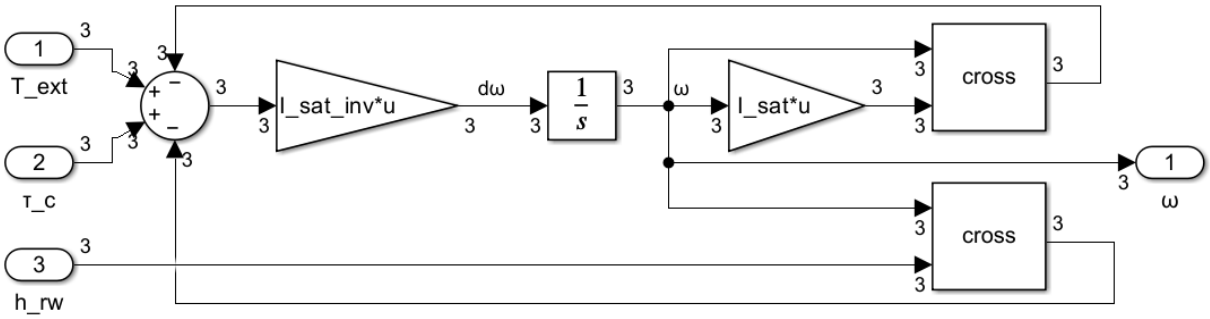
2.2.2 Complete non-linear equations and Simulink model

The non-linear equations $\mathbf{x} = f(\mathbf{x}, \mathbf{u})$ are modeled in Simulink as follows:

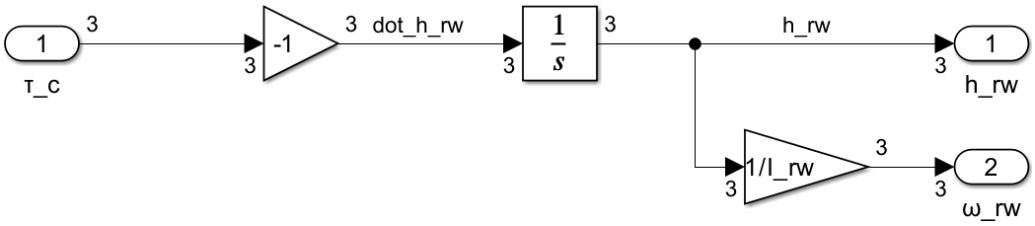
1. Attitude kinematics: $\dot{\mathrm{q}} = \frac{1}{2}\Omega(\omega)\mathrm{q}$



2. Satellite dynamics: $\dot{\omega} = I_{\text{sat}}^{-1}[T_{\text{ext}} + \tau_c - \omega \times (I_{\text{sat}}\omega) - \omega \times h_{\text{rw}}]$



3. Flywheel dynamics: $\dot{h}_{\text{rw}} = -\tau_c$



The state vector has therefore 10 elements (quaternion, rate and wheel momentum):

$$\mathbf{x}(t) = \begin{pmatrix} \mathrm{q} \\ \omega \\ h_{\text{rw}} \end{pmatrix} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & \omega_x & \omega_y & \omega_z & h_{\text{rw},x} & h_{\text{rw},y} & h_{\text{rw},z} \end{bmatrix}^{\mathsf{T}} \tag{12}$$

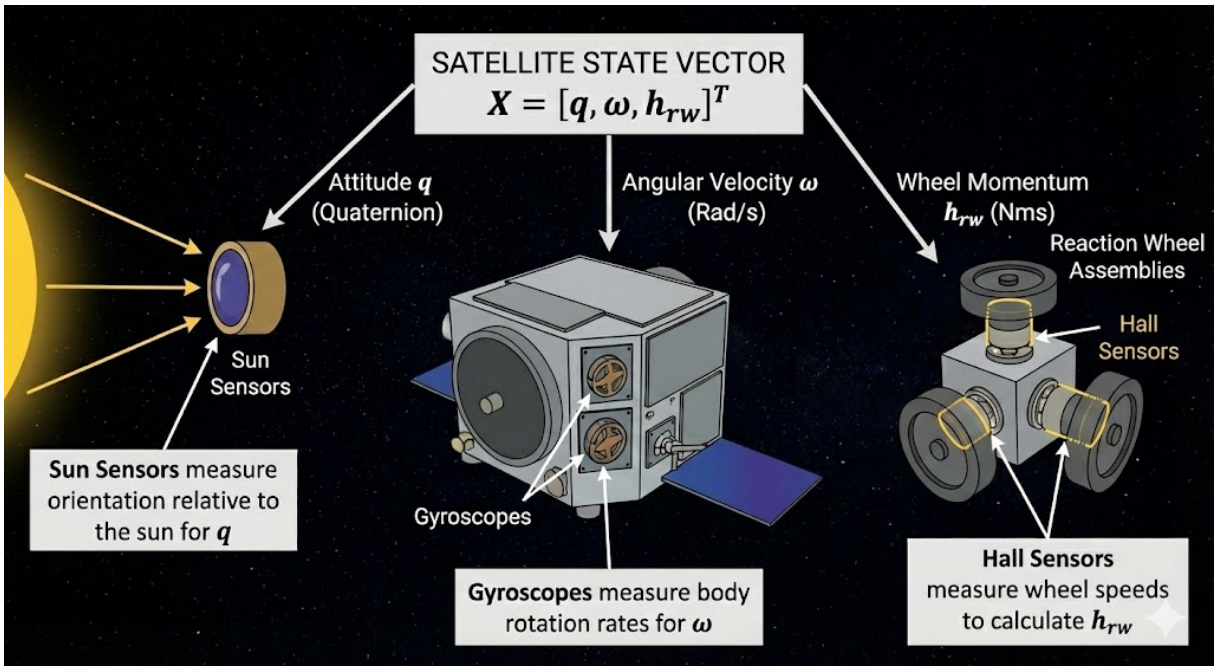


Figure 4: State vector components and their associated sensors

2.2.3 Open-loop model

The interconnected model in Simulink has the following considerations for its parameters:

- The reaction wheels are centered on the main axes of rotation of the satellite, but distanced 9 cm from the center. This offset increases the total moment of inertia of the system (parallel axis theorem). The resulting inertia matrix remains diagonal, but its values are higher than those of the bare structure.
- The model implements the same hardware limitations as before [2.1.4], with the same two constraints (extended to three axes): restricted control torque and a logic-based switch for the wheel momentum.
- No disturbance torques are applied for this simulation.

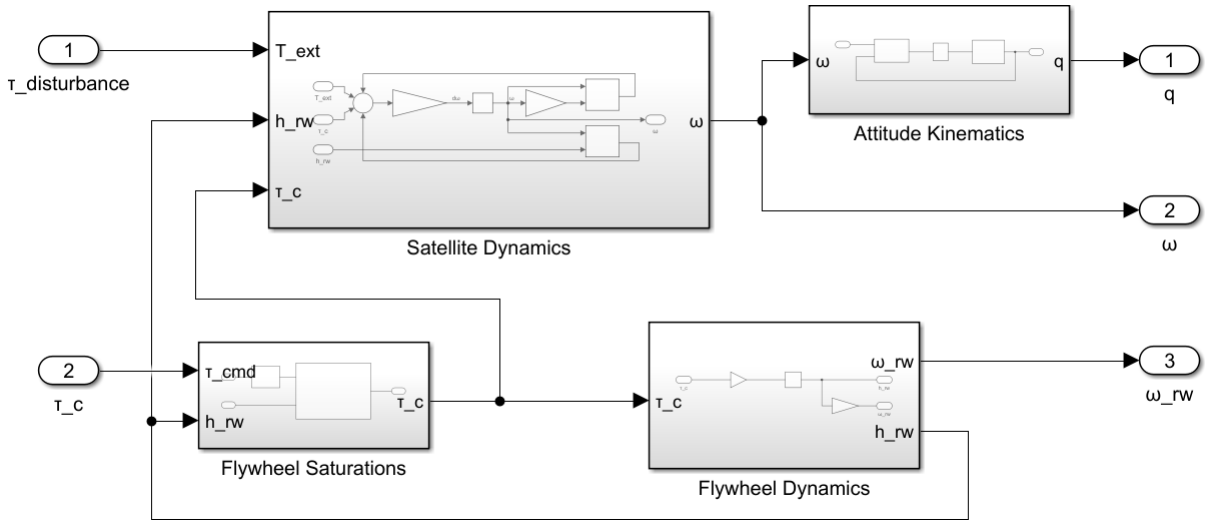
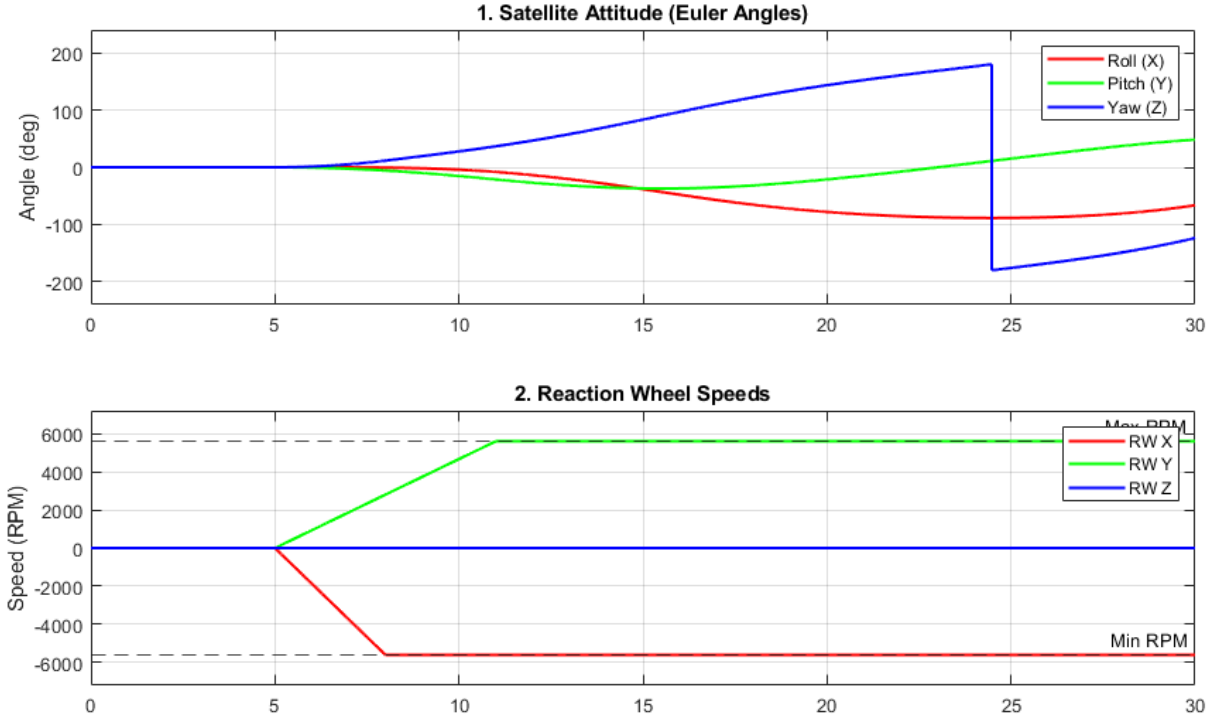


Figure 5: Non-linear model



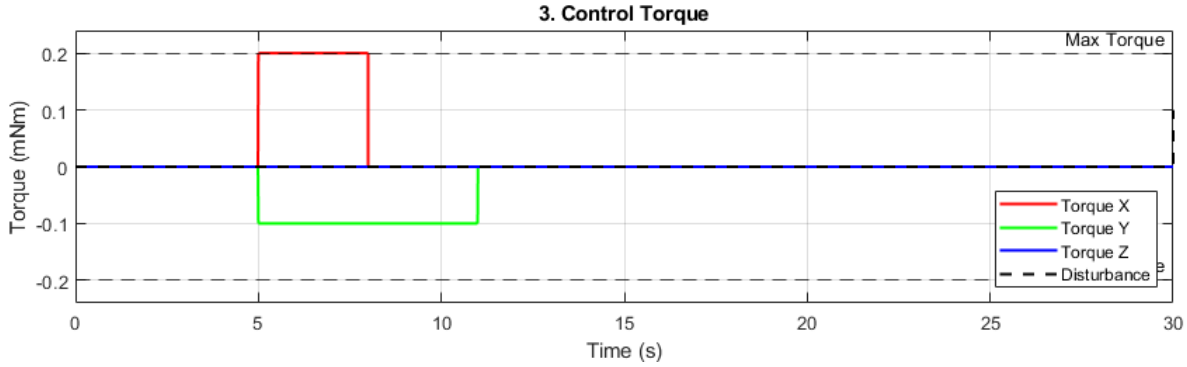


Figure 6: Open-loop results with commanded torques of $0.2\,\mathrm{mN\,m}$ in X and $-0.1\,\mathrm{mN\,m}$ in Y

The simulation integrates the equations of motion using quaternions. For visualization purposes, they are converted to Euler angles in a ZYX sequence. Consequently, the resulting graphs show instantaneous jumps between 180° and -180° (wrapping) for the roll angle, which correspond to continuous rotation and not physical discontinuities.

The model solves the full non-linear equation, implying that the axes are dynamically coupled. As observed in the simulation, applying torque to the X and Y axes generates an induced rotation in the Z axis due to the gyroscopic term $\omega \times h_{\mathrm{rw}}$, validating the interaction. It can also be noticed the torque cut-off once the wheels reach maximum speeds, validating the saturation restrictions.

2.3 LQR controller design

For the non-linear equations, the terms involving products of angular velocities and the wheel momenta act as internal disturbance torques. To design a controller (LQR), the system must be linearized.

There is a particular problem when designing a system with reaction wheels. The non-linear state vector $\mathbf{x}(t)$ is not suitable for LQR design. This is because the reaction wheel momentum h_{rw} is not a controllable state but an internal actuator variable driven by the motor torque. Linearizing the system with this component included creates an input–state structure where attitude depends on ω, and ω depends on h_{rw}, which depends on the input. This indirect actuation path yields a rank-deficient controllability matrix.

A reduced state $[\mathrm{q},\, \omega]^{\mathsf{T}}$ captures all states that determine the spacecraft's attitude dynamics, and results in a fully controllable linear model. The reaction wheel momentum is updated internally through $\dot{h}_{\mathrm{rw}} = \mathbf{u}$ in the non-linear simulation, and is constrained through saturation, but it is not regulated directly by the LQR.

This type of system actually require a second, higher-level control loop called a momentum dumping or de-saturation loop. The secondary loop would use magnetorquers or thrusters dynamics to slowly reduce the stored h_{rw} back toward zero while maintaining attitude. For simplicity, this control loop will not be modeled.

2.3.1 Jacobian linearization

To design the optimal LQR controller, the system is linearized around an equilibrium point, which corresponds to the identity quaternion and zero rates: $\mathrm{q}_0 = (1, 0, 0, 0)^{\mathsf{T}}$, $\omega_0 = [0, 0, 0]^{\mathsf{T}}$, and input $u_0 = [0, 0, 0]^{\mathsf{T}}$.

The full non-linear state vector has 7 components (by removing h_{rw}). However, since a quaternion must always satisfy the unit norm constraint ($\|\mathrm{q}\|^2 = 1$), only three of the four quaternion components are truly independent. To prevent a singular linearization, the reduced quaternion model is used [6]. In

this formulation, the attitude error state is defined by the vector part of the error quaternion, denoted as $\mathrm{q}_v = [q_1, q_2, q_3]^{\mathrm{T}}$. The math is detailed in the referenced books on attitude control [7] and [6].

For small attitude errors, the vector part relates to the small rotation angle ϕ by $\mathrm{q}_v \approx \phi/2$. The kinematics of the quaternion vector part are governed by $\dot{\mathrm{q}}_v = \frac{1}{2}(\mathrm{q}_0\omega + \mathrm{q}_v \times \omega)$. Near the equilibrium ($\mathrm{q}_0 \approx 1, \mathrm{q}_v \approx 0$), this linearizes to:

$$\dot{\mathrm{q}}_v \approx \frac{1}{2}\omega \tag{13}$$

The effective 6-state vector and the 3-input vector are defined as:

$$\mathbf{x} = \begin{pmatrix} \mathrm{q}_v \\ \omega \end{pmatrix}, \quad \mathbf{u} = \tau_c \tag{14}$$

The linearized system is expressed as $\delta\dot{\mathbf{x}} = \mathrm{A}\,\delta\mathbf{x} + \mathrm{B}\,\delta\mathbf{u}$.

The system matrix A is the Jacobian of the system dynamics $f(\mathbf{x}, \mathbf{u})$ with respect to the state $\mathbf{x}$, evaluated at the equilibrium point. Due to the choice of the origin ($\omega = 0$) and the small-angle quaternion kinematics ($\dot{\mathrm{q}}_v \approx \frac{1}{2}\omega$), all non-linear product terms vanish and simplify the matrix structure:

$$\mathrm{A} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_0, \mathbf{u}_0} = \begin{pmatrix} 0_{3\times3} & \frac{1}{2}\mathrm{I}_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{15}$$

The input matrix B is the Jacobian with respect to the control input $\mathbf{u}$, evaluated at the equilibrium point.

$$\mathrm{B} = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\mathbf{x}_0, \mathbf{u}_0} = \begin{pmatrix} 0_{3\times3} \\ I_{\text{sat}}^{-1} \end{pmatrix} \tag{16}$$

The resulting linear system is modeled in Simulink as follows:

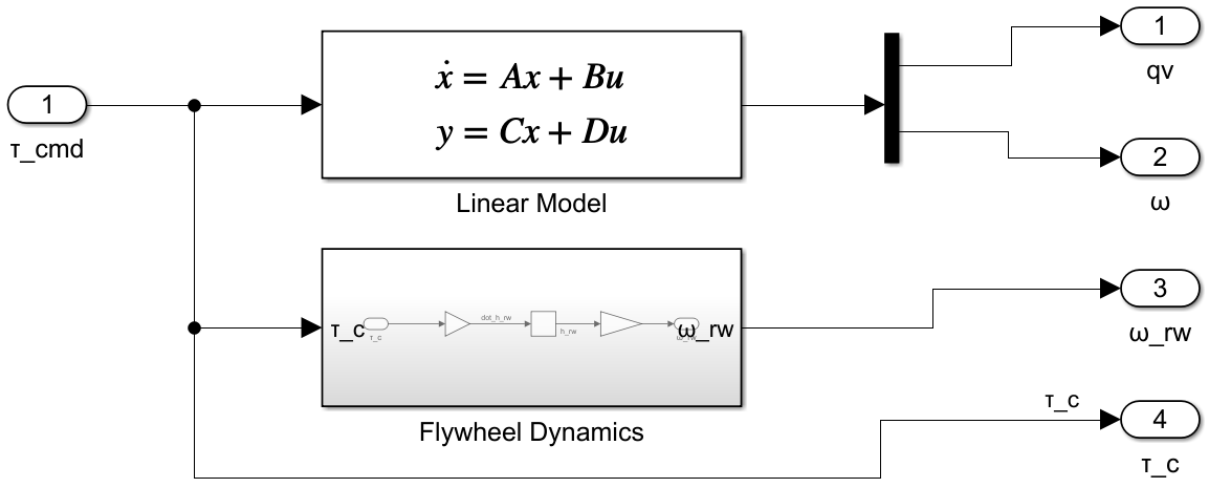


Figure 7: Linear model

2.3.2 LQR with integral action (LQI)

The system must be controllable in order to design a controller, meaning it is possible to transfer the system from any initial state to any desired final state in a finite time, using the available control inputs. For a linear time-invariant system described by $\dot{\mathbf{x}} = \mathrm{A}\,\mathbf{x} + \mathrm{B}\,\mathbf{u}$, controllability is determined by the rank of its controllability matrix, $\mathcal{C}$.

Using the MATLAB `ctrb` function, it is found that the rank of the controllability matrix is 6, indicating that the system, as represented by this linear model, is completely controllable. As pointed out before, the momentum of each wheel cannot be controlled directly, and as such it is imperative to correctly tune each penalization of the LQR, in order to maintain control of the system.

To design an LQI controller, new augmented A_{aug} and B_{aug} matrices need to be created. Let η be the vector of integral states. For attitude control, the vector part of the attitude error quaternion r_v is integrated. So, $\eta = \int \mathrm{r}_v \, dt$. Since the linear state r_v represents the deviation from the target (identity quaternion), the dynamics of the integral states are defined simply as the accumulation of this error state:

$$\dot{\eta} = \mathrm{r}_v \tag{17}$$

Since r_v is a 3-element vector, η will also be a 3-element vector.

The new state vector will have 9 states: the original 6 states plus the 3 integral states.

$$\mathbf{x}_{\text{aug}} = \begin{bmatrix} \mathrm{r}_v & \omega & \eta \end{bmatrix}^{\mathsf{T}} \tag{18}$$

The augmented system dynamics are described by:

$$\dot{\mathbf{x}}_{\text{aug}} = \mathrm{A}_{\text{aug}} \, \mathbf{x}_{\text{aug}} + \mathrm{B}_{\text{aug}} \, \mathbf{u}_{\text{aug}} \tag{19}$$

The augmented system matrices A_{aug} and B_{aug} are constructed from the original linear system matrices:

$$\mathrm{A}_{\text{aug}} = \begin{bmatrix} \mathrm{A} & 0_{6\times 3} \\ \mathrm{C}_{\mathrm{r}_v} & 0_{3\times 3} \end{bmatrix} \qquad \mathrm{B}_{\text{aug}} = \begin{bmatrix} \mathrm{B} \\ 0_{3\times 3} \end{bmatrix} \tag{20}$$

A is the original 6×6 state matrix, and C_ϕ is a 3×6 selection matrix that extracts the quaternion vector states from $\mathbf{x}$: $\mathrm{C}_{\mathrm{r}_v} = \begin{bmatrix} \mathrm{I}_{3\times 3} & 0_{3\times 3} \end{bmatrix}$. B is the original 6×3 input matrix.

With the augmented system defined and its controllability verified, the LQI gain matrix K_{lqi} is computed by applying the LQR algorithm to the augmented system matrices. The optimal control law is given by $\mathbf{u} = -\mathrm{K}_{\text{lqi}} \, \mathbf{x}_{\text{aug}} = -(\mathrm{K}_{\text{q}} \, \mathrm{r}_v + \mathrm{K}_\omega \, \omega + \mathrm{K}_{\text{I}} \, \eta)$.

The penalty matrices for the computation of K_{lqi} are constructed using the inverse square of the maximum acceptable deviation for each component, scaling the cost function to align with the required physical tolerances and actuator limits.

$$\mathrm{Q}_{\text{aug}} = \mathrm{diag}\left([\underbrace{1/\phi_{\text{max}}^2}_{\phi}, \underbrace{1/\omega_{\text{max}}^2}_{\omega}, \underbrace{1/\eta_{\text{max}}^2}_{\eta}] \right) \tag{21}$$

$$\mathrm{R}_{\text{lqi}} = \mathrm{diag}\left([\underbrace{1/\tau_{\text{max}}^2}_{\tau_c}] \right) \tag{22}$$

Initial numerical settings prioritize attitude accuracy ($\phi_{\text{max}} = 0.1^\circ$), allows higher angular velocities $\omega_{\text{max}} = 2.5 \, \mathrm{rad/s}$, and highly penalize the integral error ($\eta_{\text{max}} = 0.005$). Commanded torque is highly penalized to avoid saturation ($\tau_{\text{max}} = 1\mathrm{e}^{-5} \, \mathrm{N\,m}$).

The LQI gain matrix K_{lqi} is computed as $\mathrm{K}_{\text{lqi}} = \mathrm{LQR}(\mathrm{A}_{\text{aug}}, \mathrm{B}_{\text{aug}}, \mathrm{Q}_{\text{aug}}, \mathrm{R}_{\text{lqi}})$, and the resulting 3×9 matrix is used to close the control loop in the linear Simulink model.

The controller is modeled in Simulink:

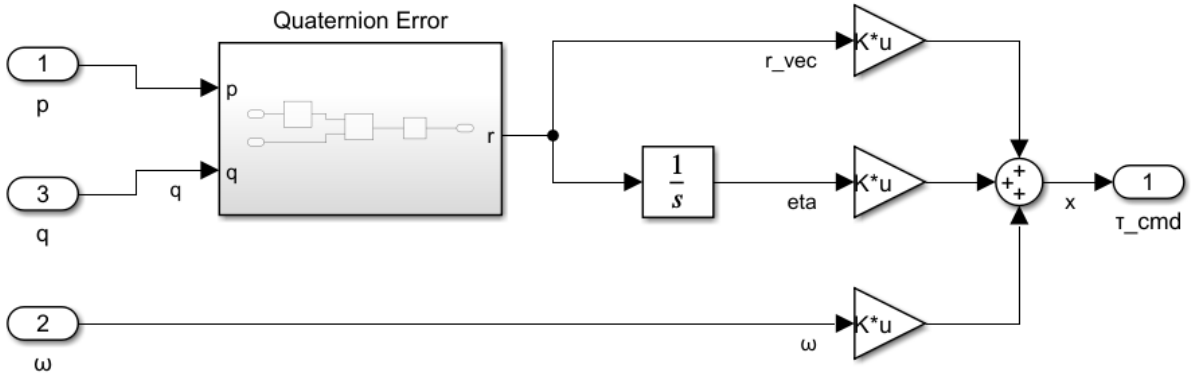


Figure 8: LQI Controller

The gain matrix is separated and used as a negative gain for each vector state. The Quaternion Error block calculates the quaternion error between a desired angular orientation and the actual orientation. It then extracts the vector components of the error to obtain $\dot{\eta}$.

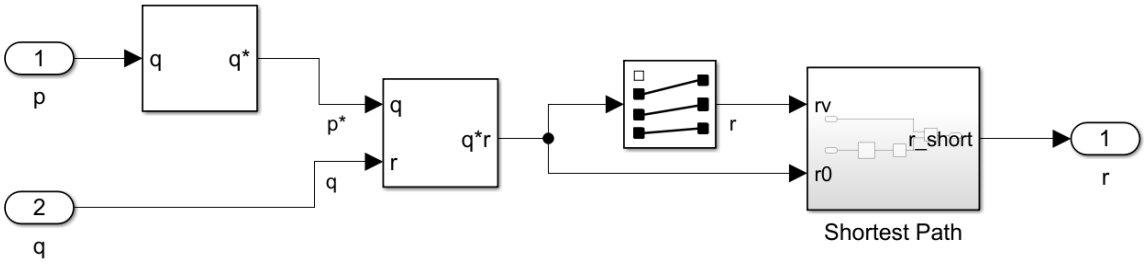


Figure 9: Calculation of the quaternion error

The $\bar{\mathrm{p}}$ input is the reference as quaternion, and $\bar{\mathrm{q}}$ is the output of the system as quaternion. The quaternion error is:

$$\bar{\mathrm{r}} = \bar{\mathrm{p}}^{-1} \otimes \bar{\mathrm{q}} = \bar{\mathrm{p}}^{*} \otimes \bar{\mathrm{q}} \tag{23}$$

For large step reference angles, the controller must be able to automatically choose the most efficient path of rotation. The Shortest Path block uses the scalar part of the quaternion error as a measure of the angle wrapping. If the scalar component is negative ($\mathrm{r}_0 < 0$), it indicates a rotation exceeding $180°$. In this instance, the vector error quaternion is negated ($\mathrm{r}_v \rightarrow -\mathrm{r}_v$), compelling the controller to execute the equivalent rotation along the shorter arc, thereby minimizing maneuver time and energy consumption.

2.4 State estimator and sensors

An LQI controller relies on the fundamental assumption of full-state feedback. It requires precise, instantaneous knowledge of the complete augmented state vector to calculate the optimal control command. However, in a practical implementation, perfect state knowledge is impossible. Real-world sensors are inherently corrupted by measurement noise, biases, and quantization errors. Feeding raw sensor data directly into feedback controllers can lead to actuator jitter, excessive power consumption, and degraded performance.

2.4.1 Extended Kalman filter

To bridge the gap between the ideal states required by the controller and the imperfect measurements available onboard, a state estimator is needed. The Kalman filter is the standard solution: it predicts the

system state using the dynamic model and then corrects the prediction using sensor data, weighting each input according to their statistical uncertainties to obtain an optimal estimate.

Because the spacecraft attitude dynamics are non-linear, a linear Kalman filter is inadequate. Instead, an Extended Kalman Filter (EKF) is used. The EKF propagates the state with the full non-linear model and linearizes the dynamics at each time step to calculate the covariance, enabling consistent and accurate state estimation.

The EKF requires a mathematical model of the non-linear system as well as the symbolic Jacobian A matrix. The models run at each step of the discrete time dt in a continuous loop, taking the state from the last instant, predicting where it should be now, and then correcting that prediction with new sensor data.

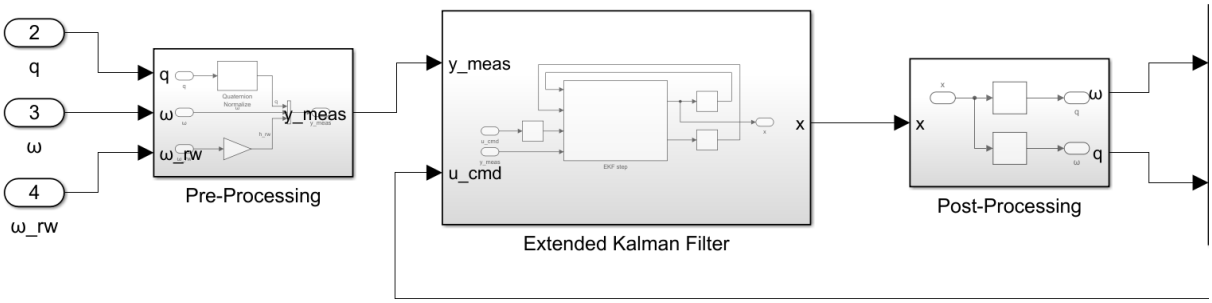


Figure 10: Extended Kalman filter

A pre-processing block assembles the measured state vector. A post-processing block selects the vector components for the LQI controller.

The Extended Kalman Filter block is a Matlab function that performs one discrete time step of the EKF. It moves the state forward using the physics model, checks how wrong it was compared to sensor data, and fixes the estimate while keeping the math stable. The function can be found in the Appendix section at the end of the document.

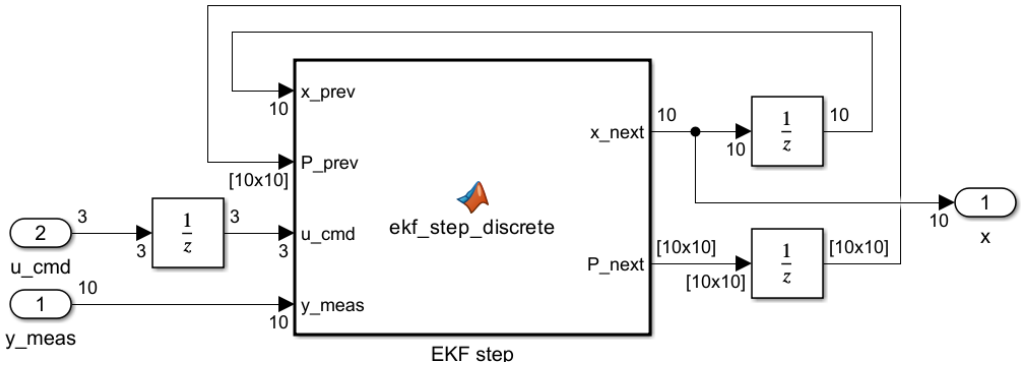


Figure 11: EKF step structure

2.4.2 Sensor suite

In a typical CubeSat, an ADCS (Attitude Determination and Control System) is used to measure the spacecraft's physical states. It includes a comprehensive sensor suite and a microcontroller [8][9]:

- Sun sensors (6×): Measure the direction of the Sun.
- Magnetometers (2×): Measure the Earth's magnetic field.
- MEMS gyroscopes (3×): Measure the body angular velocity vector.

- Hall-effect sensors (3×): Measure the rotational speed of reaction wheels.

The ADCS computer fuses the sensor measurements to calculate the attitude q and angular velocity vector ω using algorithms such as TRIAD or QUEST. For the case study, the ADCS is considered a black box that provides the attitude. The manufacturer assures a pointing precision of:

$$\text{RPE (jitter): } 0.0028^\circ \text{ during } 10\,\mathrm{ms} \;@\; 1\sigma \rightarrow \text{ Noise and noise power}$$

$$\text{APE (absolute pointing error): } 1.0^\circ \;@\; 1\sigma \rightarrow \text{ Bias}$$

Gyroscope information is not provided, so the following values are based on a standard industrial-grade MEMS (e.g., Bosch BMI088 or Analog Devices ADIS16470):

$$\text{ARW (angular random walk): } 0.007^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}} \;@\; 1\sigma \rightarrow \text{ Noise power}$$

$$\text{Bias Instability: } 10^\circ/\mathrm{hr} \approx 0.0028^\circ/\mathrm{s} \;@\; 1\sigma \rightarrow \text{ Bias}$$

For the Hall sensors the values are based on a standard CubeSat wheel (e.g., Maxon EC Flat motor with 3 Hall sensors):

$$\text{Speed Stability (jitter): } \pm 2.0, \mathrm{RPM} \approx 0.21\,\mathrm{rad/s} \;@\; 1\sigma \rightarrow \text{ Noise (Ripple)}$$

$$\text{Speed Accuracy (resolution): } \pm 0.5, \mathrm{RPM} \approx 0.05\,\mathrm{rad/s} \rightarrow \text{ Quantization/Bias}$$

The sensors are modeled in Simulink:

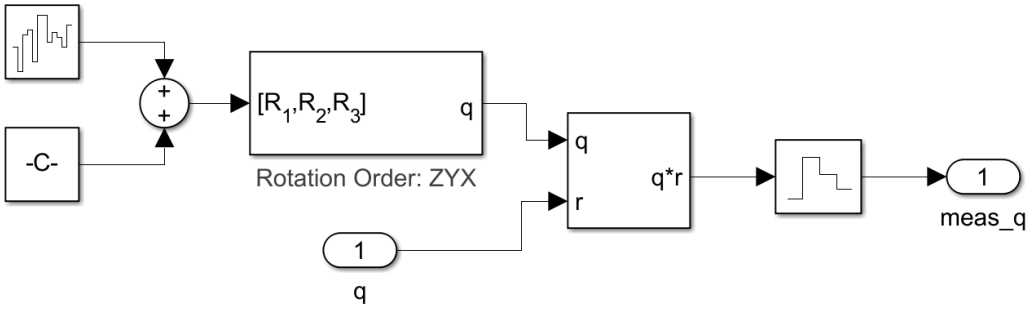


Figure 12: Attitude sensor

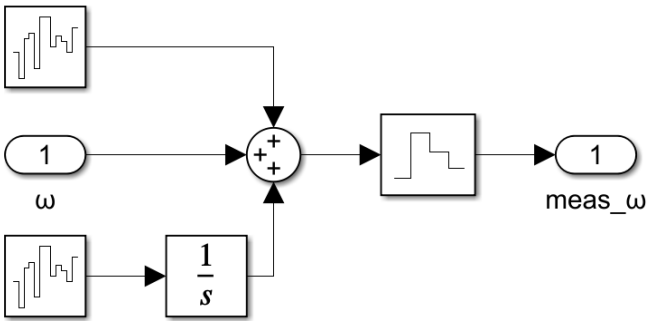


Figure 13: Gyroscope sensor

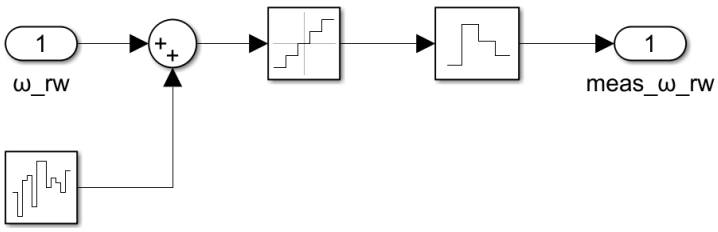


Figure 14: Hall sensor

To verify the claims of the manufacturer, a test is made to calculate the values of the attitude determination:

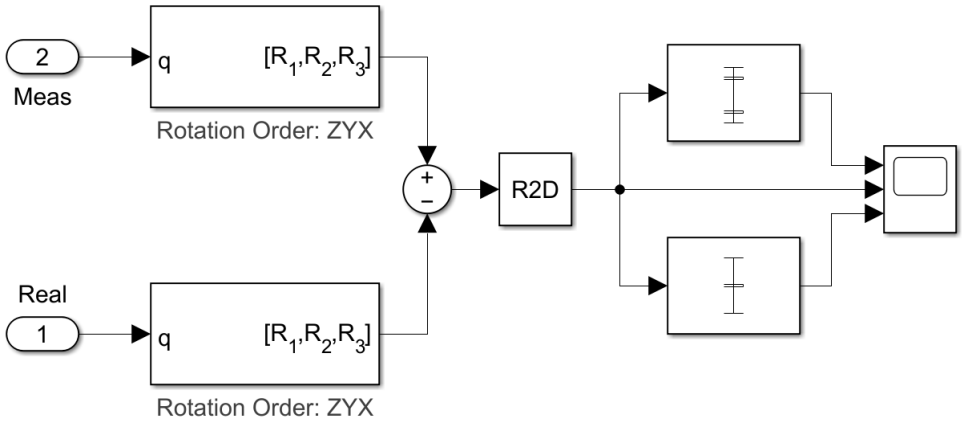


Figure 15: Attitude error test to calculate bias and jitter

The test calculates the error in attitude in order to remove the actual satellite movement (and see just the sensor imperfections). A mean and standard deviation block is then used to calculate bias and jitter.

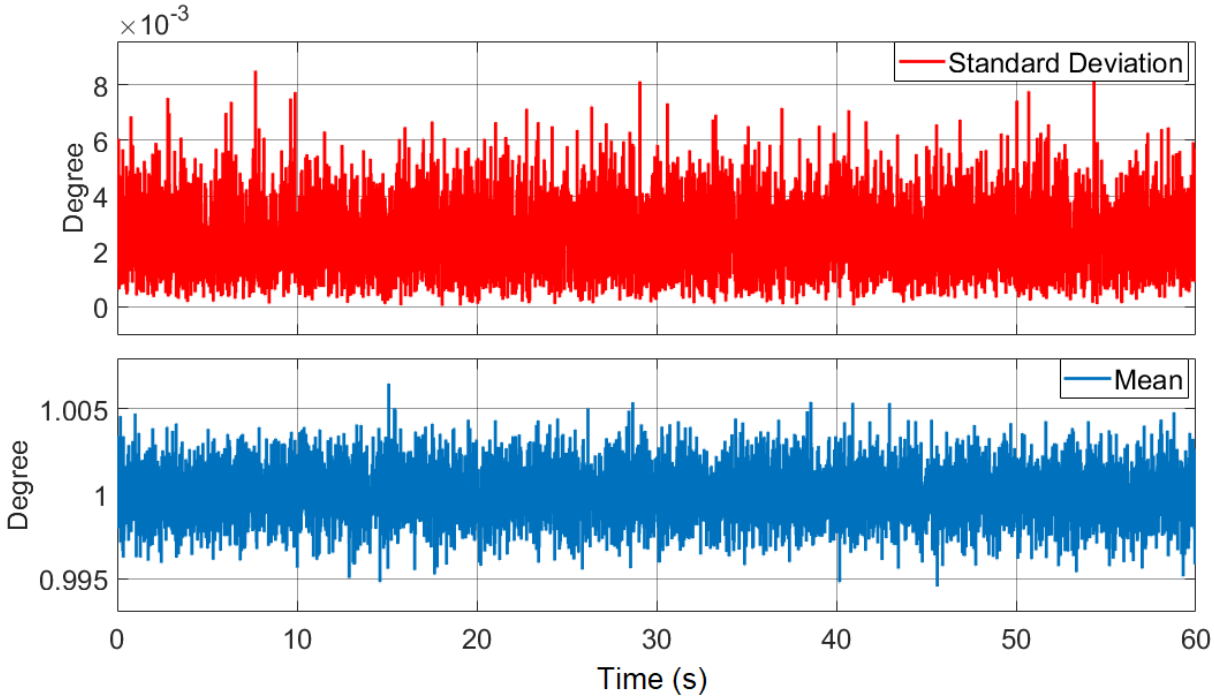


Figure 16: Mean and standard deviation results for the ADCS

The test shows that the manufacturer's claims are true, the mean value is effectively 1° on average, and the standard deviation RMS is 0.00279°.

2.4.3 Anti-windup and actuators

As briefly discussed before, reaction wheels are subject to physical constraints. When a controller commands a control effort that exceeds these bounds, the actuator saturates, creating a discrepancy between the commanded torque and the actual torque applied. During this period the attitude error persists, causing the integral state η to accumulate excessively. If unaddressed, this results in severe overshoot and

prolonged settling times, or even worst, complete loss of control. To mitigate it, a Back-Calculation Anti-Windup scheme is implemented, by feeding the difference between the saturated and commanded signals back into the integration loop, dynamically halting the error accumulation whenever the actuators are operating at their limits. In order to convert torque units into integrator state units, a gain matrix $\mathrm{K_{aw}}$ is needed. A good rule of thumb for LQI systems is to base it on the inverse of the integral gain ($\mathrm{K_{aw}} \approx \mathrm{K_I^{-1}}$). The integral state becomes:

$$\dot{\eta} = \mathrm{r}_v + \mathrm{K_{aw}} \left(\tau_{\mathrm{act}} - \tau_{\mathrm{cmd}} \right) \tag{24}$$

The Anti-Windup block is modeled in Simulink:

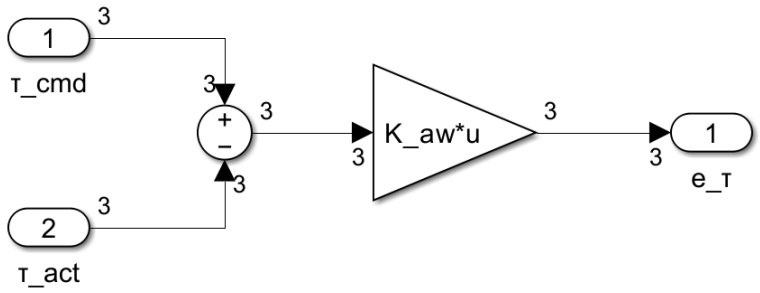


Figure 17: Anti-windup block

Additionally, realistic actuators cannot generate torque instantaneously due to the inductance of the motor coils and the finite bandwidth of the internal current control loop. To capture this physical lag, the reaction wheel dynamics are modeled as a first-order low-pass filter with a time constant $\mathrm{T_{motor}} = 20\,\mathrm{ms}$. This represents the delay between the digital command signal and the physical torque applied.

$$\dot{\tau} = \frac{1}{\mathrm{T_{motor}}} \left(\tau_{\mathrm{cmd}} - \tau_{\mathrm{act}} \right) \tag{25}$$

The Actuators block is modeled in Simulink:

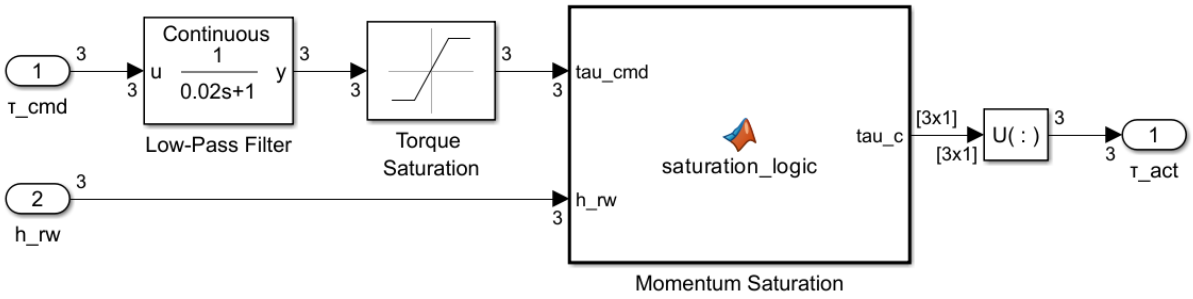


Figure 18: Actuator block

2.5 Discrete controller

LQI was design in continuous time. However, the controller implementation runs entirely in discrete time: sensor readings, estimator updates, and control commands all occur at fixed sampling intervals. The LQI controller must be discretized consistently with the rest of the system.

To achieve this, the continuous integrator block used to generate the error integral state (η) is replaced with its discrete-time equivalent, implemented using the Trapezoidal integration method (with anti-windup calculation included). The optimal feedback gain matrix was then re-calculated using the discrete implementation of the linear quadratic regulator (DLQR).

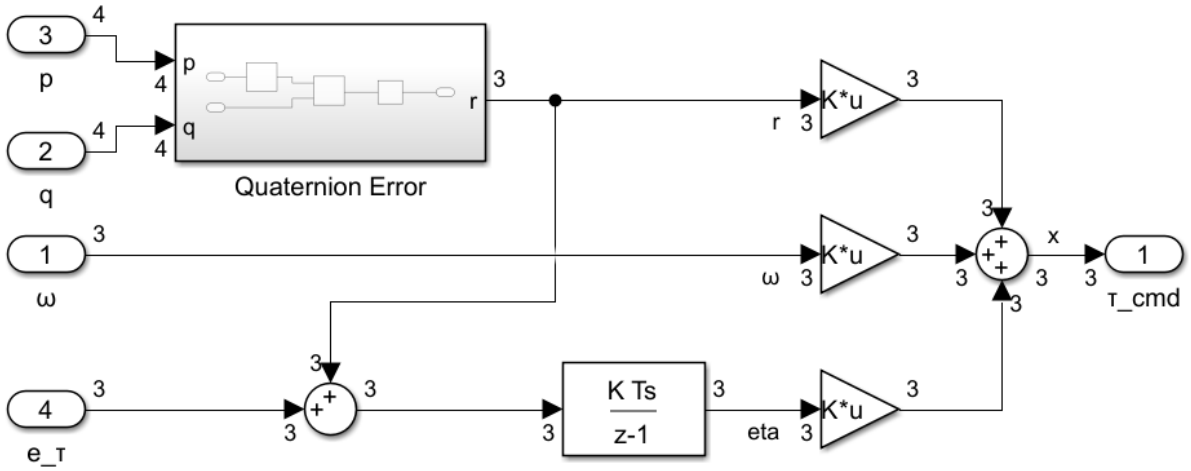


Figure 19: Discrete LQI controller

2.5.1 Sample time selection

Selecting an appropriate sample time is critical. The sample rate must be sufficiently high to capture the fastest relevant dynamics of the plant to avoid aliasing and ensure stability. For satellite attitude control, the linearized open-loop dynamics around the origin consist of integrators (poles at the origin), meaning the plant does not contain high-frequency resonant modes. In practice, the fastest dynamics arise from the reaction wheel actuators and the desired closed-loop bandwidth.

Since the current model does not include the internal motor dynamics of the reaction wheels, the closed-loop bandwidth is assumed based on the ADCS systems. For agile maneuvers, a control bandwidth in the range of 1–5 Hz is common. To satisfy standard design guidelines (sampling at least 10–20 times faster than the control bandwidth), a sampling frequency of 100 Hz was chosen:

$$\Delta t = 0.01 \text{ s} \tag{26}$$

The Controller block (LQI, EKF and anti-windup) is modeled in Simulink:

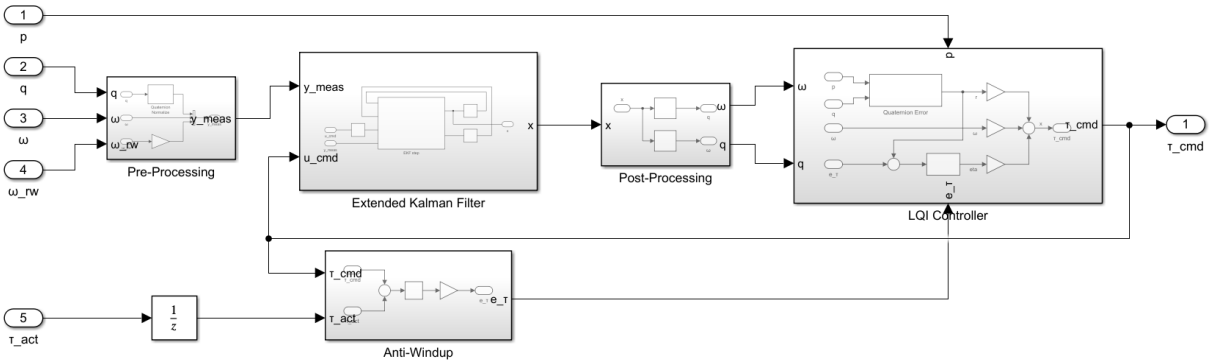


Figure 20: Integrated controller

2.6 Complete system

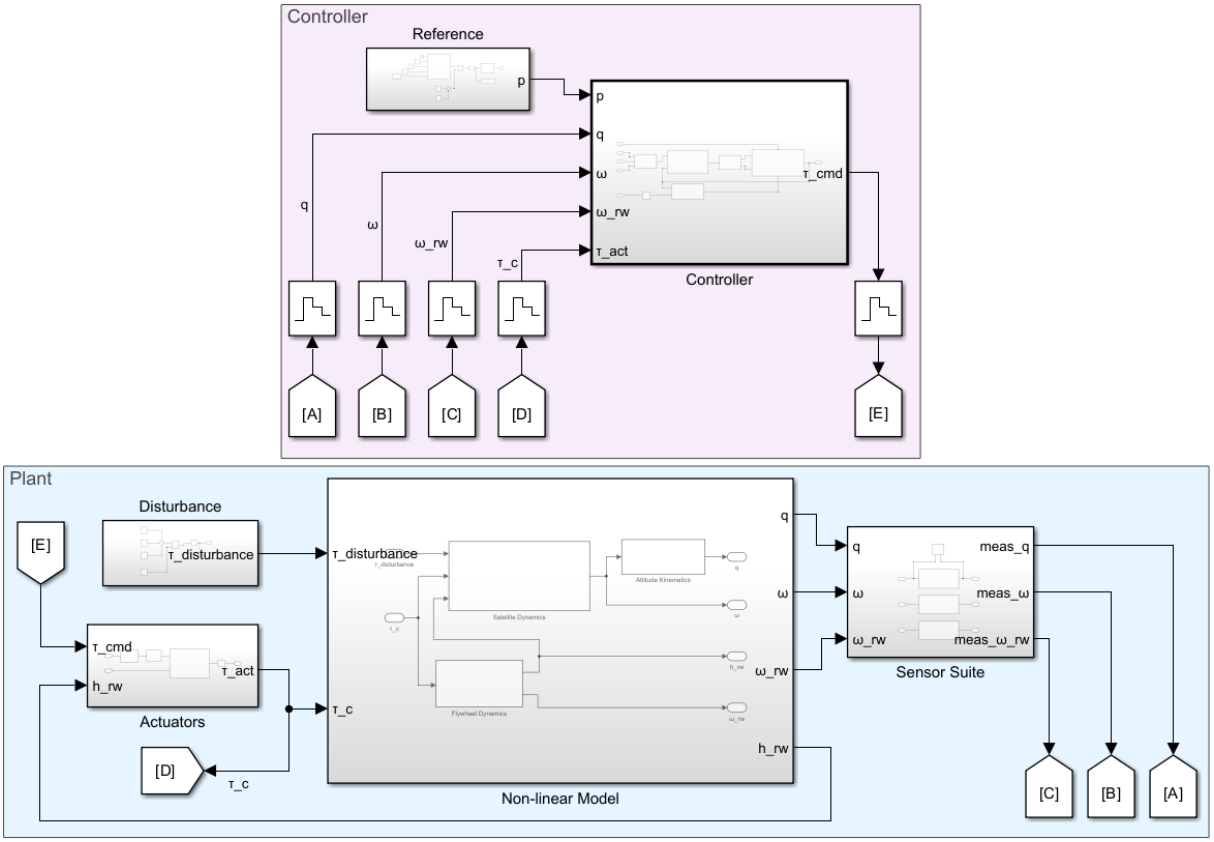


Figure 21: Non-linear model with discrete controller

The ZOH blocks represent the discrete I/O of the controller, with its discrete nature. The plant is simulated in ‘continuous’ time.

3 Results

This section provides some scenarios to test and analyze the designed controller. The results were obtained with the non-linear model and the integrated controller.

3.1 Anti-windup

The first test is done with ideal sensors, a step reference attitude $[20°, 0°, 0°]$ at $t = 5s$, and no disturbance torque. The comparison shows the effects of integral error accumulation.

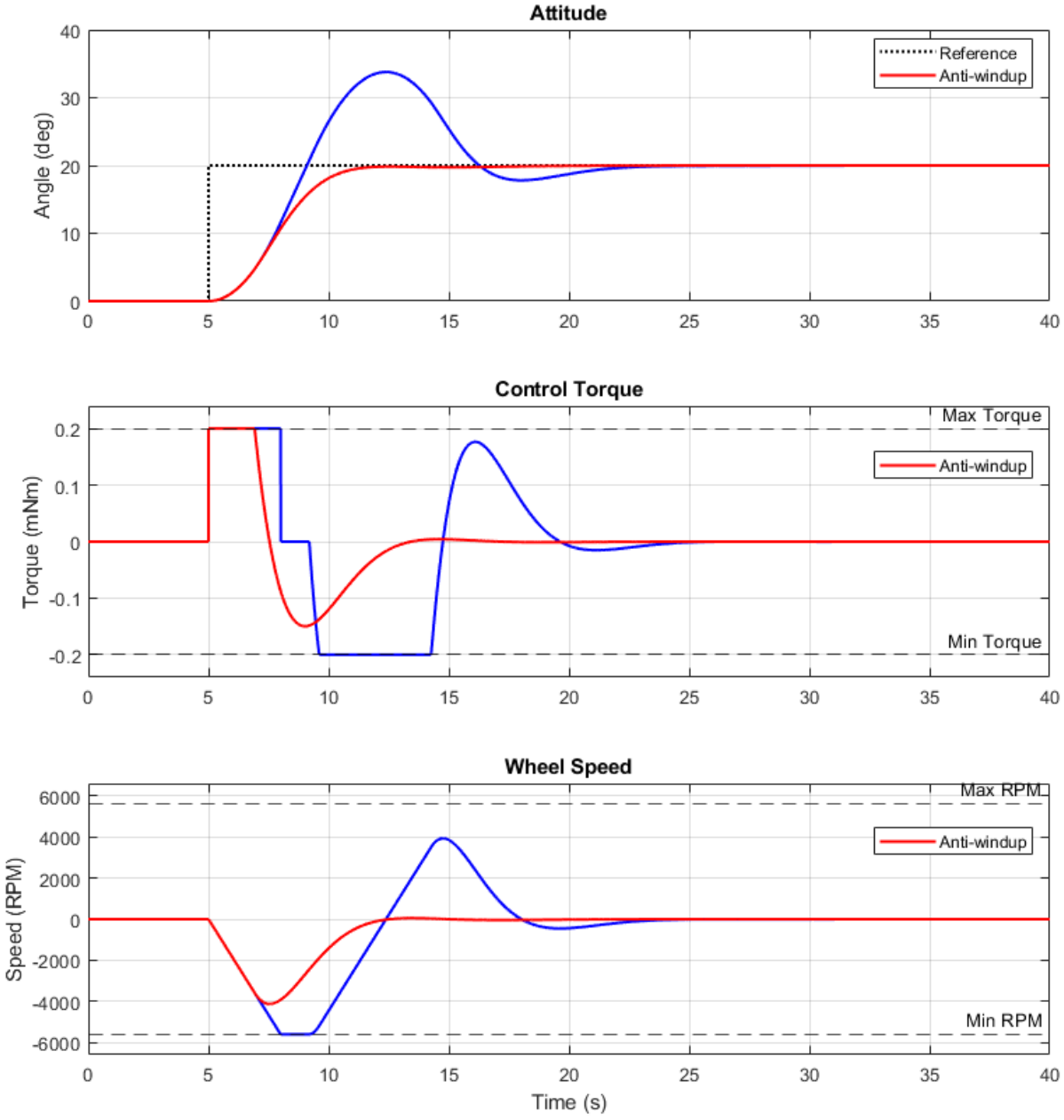


Figure 22: Anti-windup comparison

The necessity of the anti-windup block is made clear, as it not only avoids slower settling times, but also the overshoot is reduced, which translates to less energy consumed by the actuators. From now on all tests are done with active anti-windup.

3.2 Disturbance rejection and coupling

The second test is done with ideal sensors, and a smooth reference attitude trajectory, starting at $t = 5s$. The attitude objective is $[150°, 0°, 0°]$, and a disturbance torque is applied at $t = 30s$, with components $[0, 0, -0.1\mathrm{e}^{-3}]\,\mathrm{Nm}$, lasting 2 seconds.

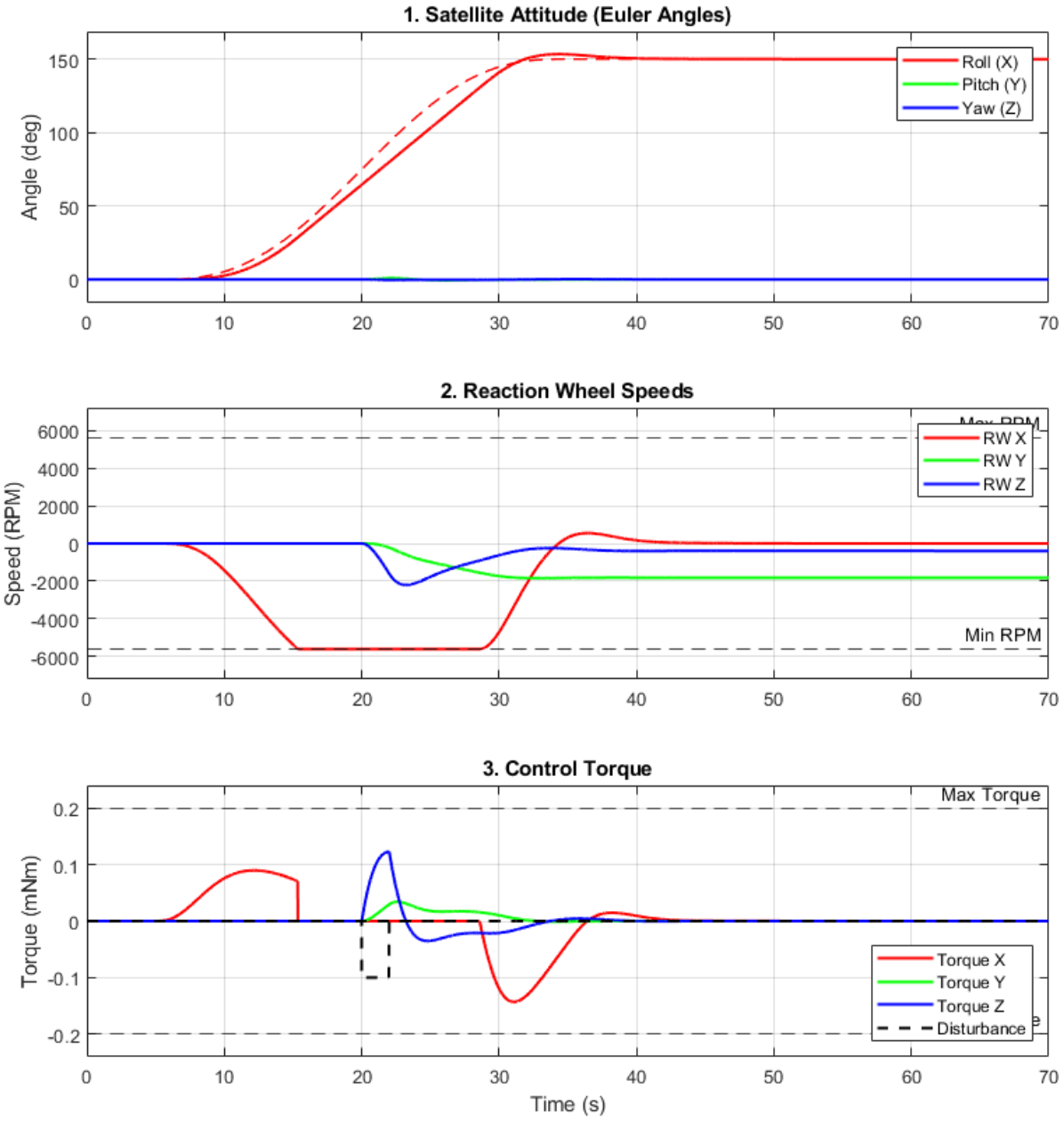


Figure 23: Disturbance rejection and coupling

The dotted line in the first graph is the references. The system is able to closely follow it, and the anti-windup block allows the controller to maintain control even if the wheel is saturated for a time. The disturbance generates less than a $1°$ offset in the Y and Z axis, which is fully corrected in under $10\,\mathrm{s}$

Notably, although the disturbance only has a Z component, the Y-axis reaction wheel (RW Y) absorbs momentum, reaching approximately -1800 RPM. This effect is due to gyroscopic coupling: the large momentum stored in RW X (from the maneuver) interacts with the Z-axis angular velocity caused by the disturbance. Mathematically, this interaction ($\omega_z\, \mathrm{h}_x$) generates a strong gyroscopic torque vector

purely along the Y-axis. Essentially, because the spacecraft holds significant angular momentum in X, any attempt to rotate it about Z results in a precession torque on Y, forcing the Y-axis reaction wheel to spin up to absorb this energy and maintain stability.

3.3 Shortest path logic

The main objective of the satellite is to perform adequately with minimum energy spent. If large angles are commanded, the system needs to be able to take the most direct route, unless intermediate points are specified (e.g., following an object).

The third test is also done with ideal sensors, and a step reference of $[340°, 60°, 160°]$ and no disturbances.

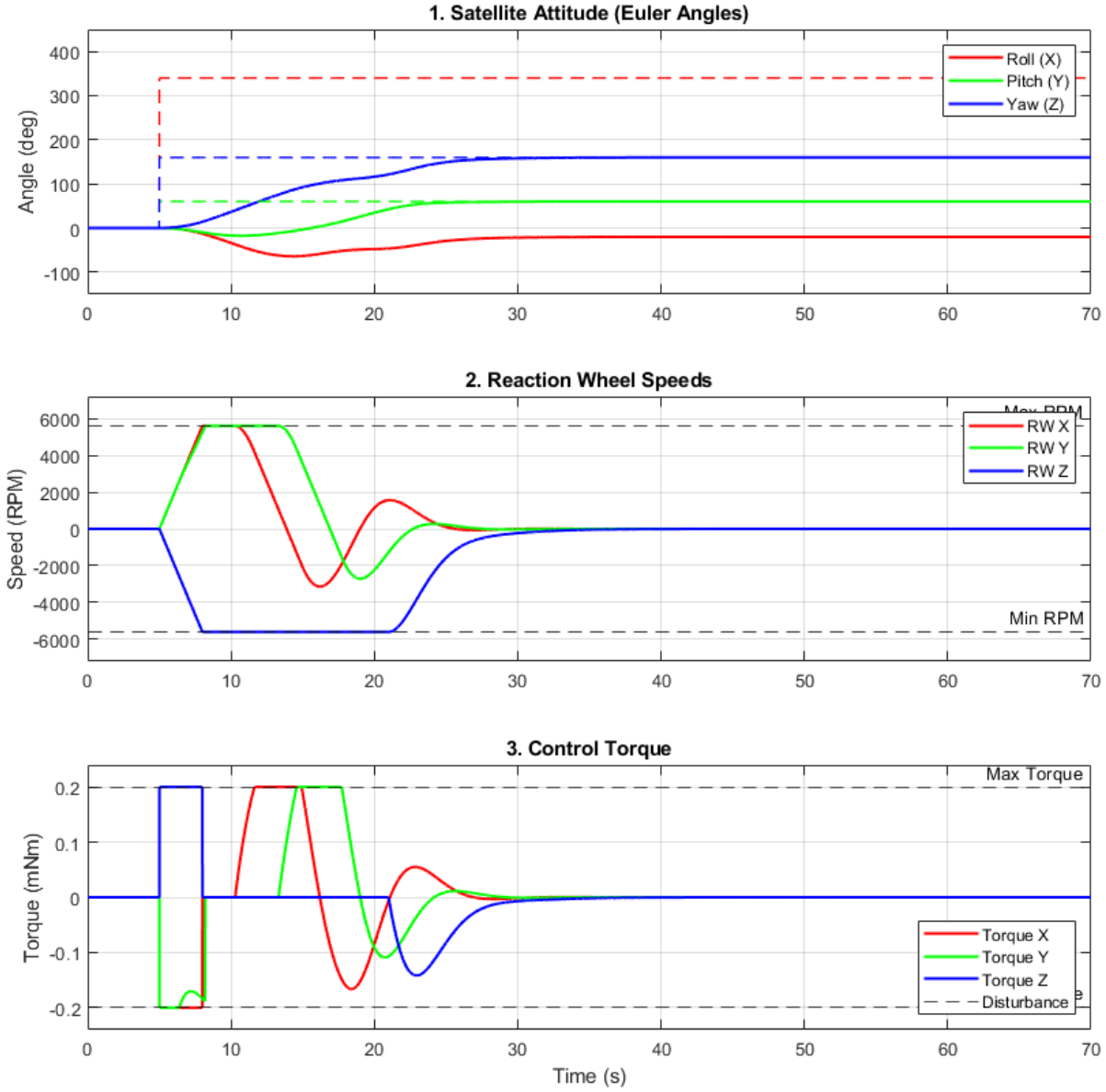


Figure 24: Shortest path

A key observation is the system's strong coupling, which the LQI controller is unable to mitigate. The small angle approximation breaks down with large angles and step references. However, the integral action with anti-windup provides reasonable performance. This approach would not be viable if precise,

decoupled rotation is required; in that case, a smooth trajectory would be necessary. Finally, the observed roll angle of $-20°$ is logical, as it represents the most efficient path.

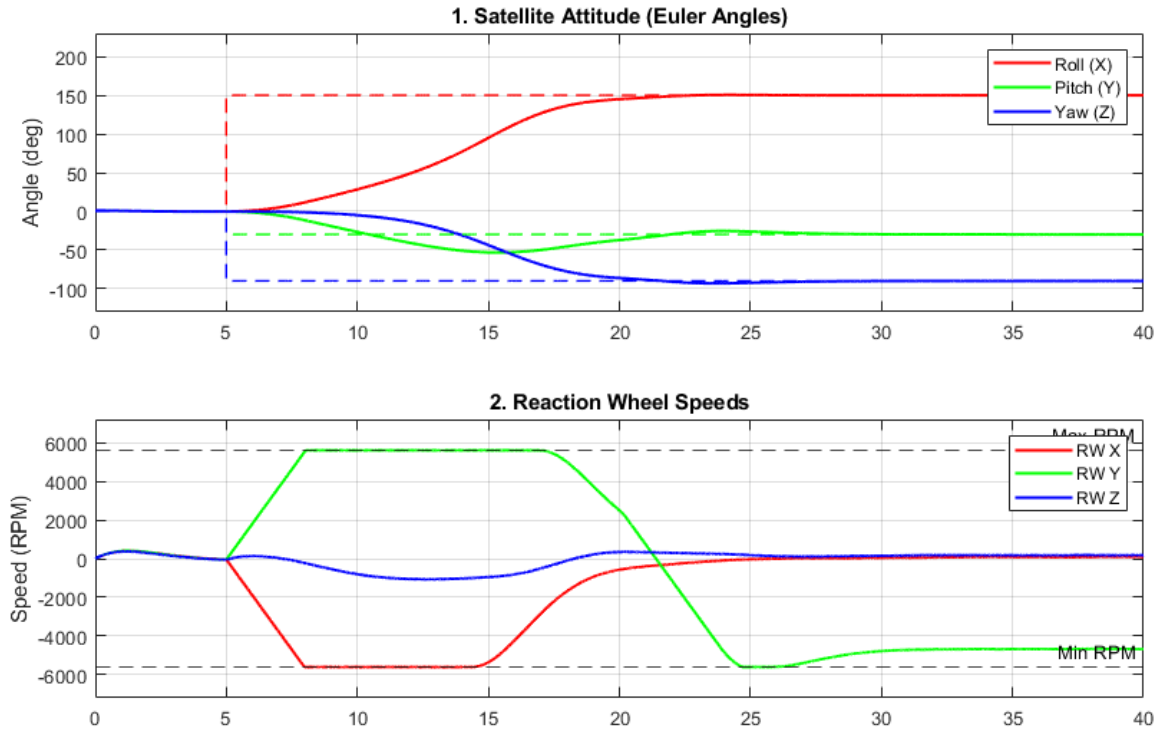
3.4 Noisy sensors and the Kalman filter

Non-ideal sensors characterized by noise and bias are introduced. The system performance must remain comparable to the ideal case. The Kalman filter needs to balance the information from its internal model with the incoming sensor data. Furthermore, since the filter effectively operates blind to external disturbances, it cannot completely trust the internal model. This means the EKF Q and R gain matrices need to be tuned accordingly. As a first approach, the following values are used, based on the sensor parameters:

	State / Sensor	Parameter Source	Covariance Value
Measurement Noise Covariance (**R**)			
R_q	Attitude (ADCS)	RPE ($0.0028°, 1\sigma$)	$2.4 \times 10^{-9}\,\text{rad}^2$
R_ω	Angular Rate (Gyro)	ARW ($0.007°/\sqrt{\text{Hz}}$)	$1.5 \times 10^{-6}\,(\text{rad/s})^2$
R_h	Wheel Speed (Hall)	Jitter (± 2 RPM, 3σ)	$4.9 \times 10^{-3}\,(\text{rad/s})^2$
Process Noise Covariance (**Q**)			
Q_q	Kinematics	Derived from Gyro Noise	$1.5 \times 10^{-10}\,\text{rad}^2$
Q_ω	Dynamics	Model Uncertainty (Tuned)	$1.0 \times 10^{-4}\,(\text{rad/s})^2$
Q_h	Momentum/Speed	Actuator Uncertainty (Tuned)	$1.0 \times 10^{-5}\,(\text{rad/s})^2$

Table 1: Kalman Filter Covariance Parameters.

The fourth test is done with a step reference of $[150°, -30°, -90°]$ at $t = 5\,\text{s}$ and a disturbance torque of $[0, -2.5\text{e}^{-4}, 0]\,\text{Nm}$ at $t = 20s$.



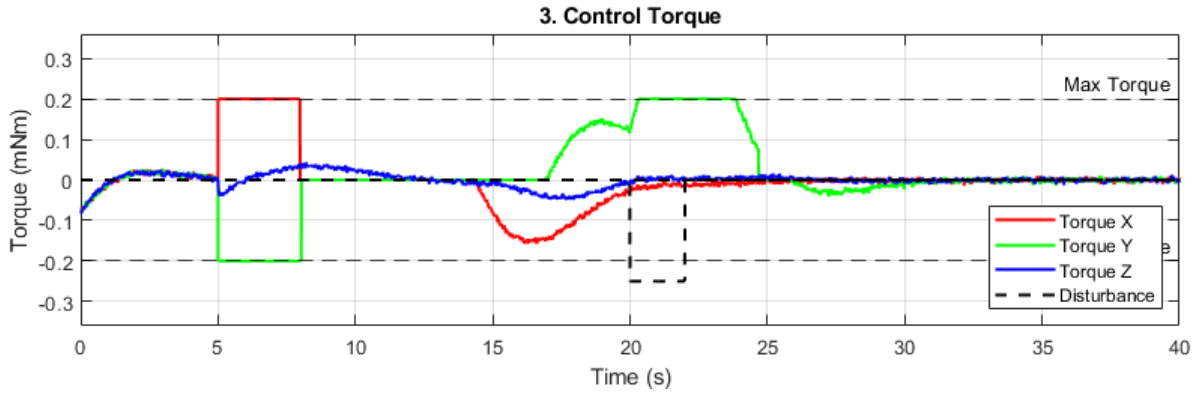


Figure 25: Noisy sensors

The control system manages to position the satellite in the correct orientations, even with large angles and noisy sensors. The disturbance is absorbed, mostly by the Y reaction wheel, almost reaching saturation (the X and Z wheels spin because of coupling).

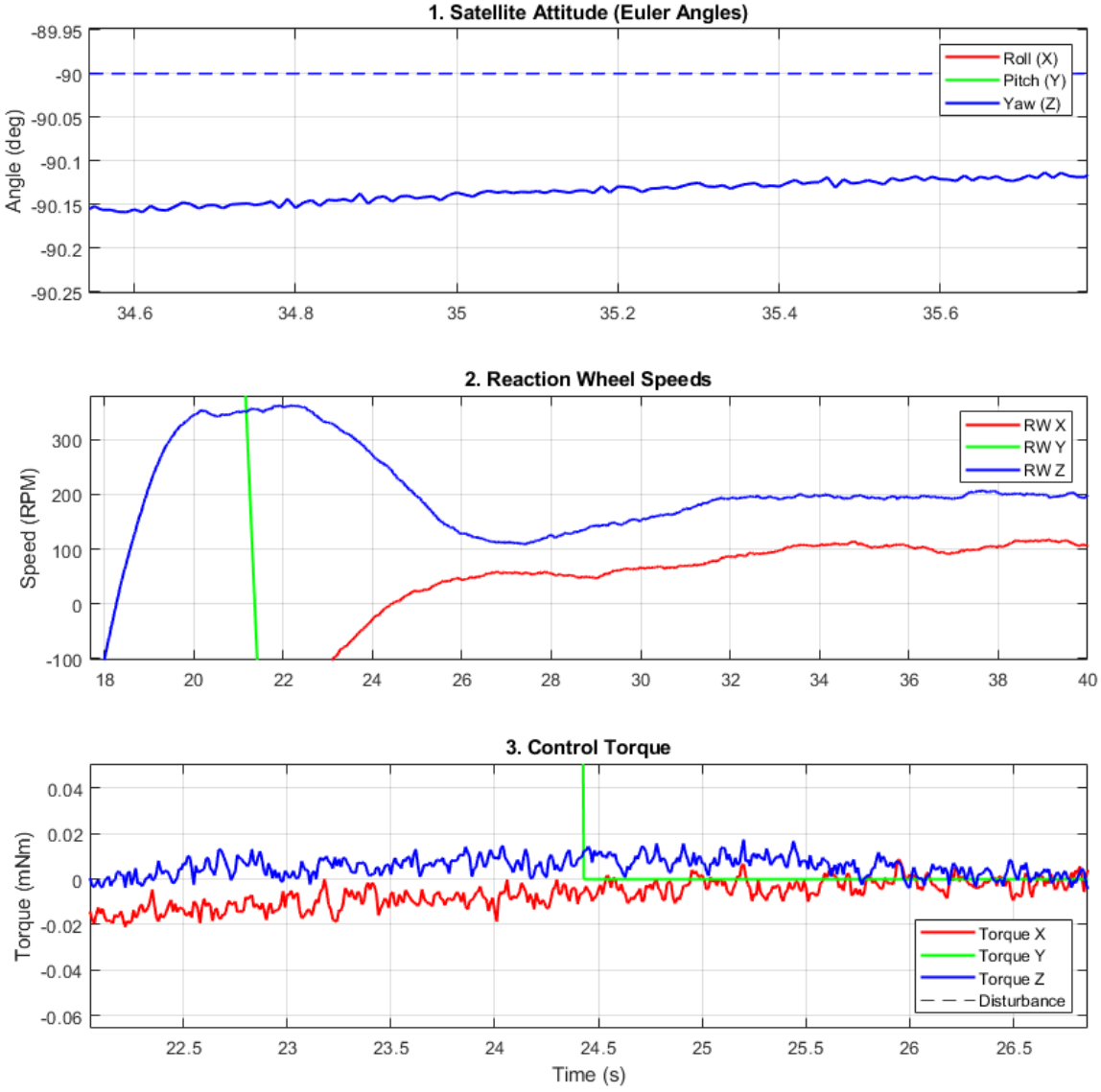


Figure 26: Zoomed-in signals

The torque signal shows more noise than the attitude and angular velocity responses because of how the actuator is modeled. The model includes a first-order lag to approximate the mechanical response, but it does not account for the motor's electrical dynamics. In a real motor, the winding inductance acts as an extra low-pass filter that would naturally smooth out high-frequency components, leading to a cleaner torque profile than what the simulation produces. The high-gain LQI controller also amplifies small estimator noise, making these fluctuations visible in the torque command. In contrast, the satellite's large moment of inertia effectively filters out these high-frequency variations, which is why the attitude and angular velocity outputs remain smooth.

3.5 Saturation and control limits

The fifth test shows the effects of reaching the wheels' maximum speed. The step reference is $[50°, 10°, -6°]$ at $t = 5\,\text{s}$, and then another step of $[-15°, 0°, 30°]$ relative to the first one at $t = 50\,\text{s}$. A constant disturbance is applied, simulating aerodynamic drag in the X-axis, equal to $10^{-6}\,\text{N m}$.

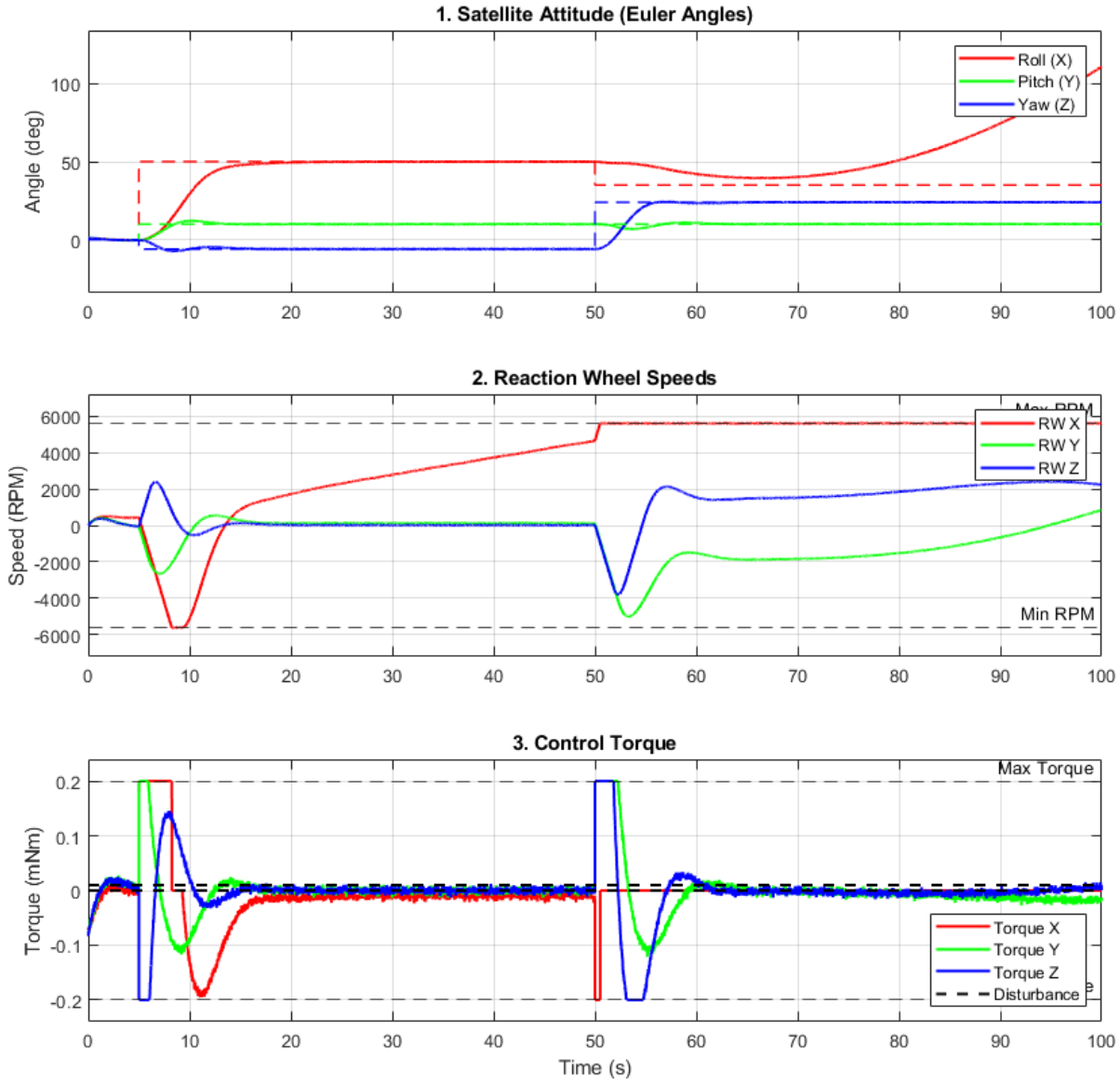


Figure 27: Continuous disturbance and X-axis saturation

Initially, the disturbance opposes the commanded attitude, but as time passes, it forces the X reaction wheel to accelerate and accumulate momentum to compensate. When a subsequent command requires

torque in the same direction, the remaining momentum margin is exhausted. The wheel reaches saturation, causing the available control torque to drop to zero. Consequently, control authority in the X-axis is lost, as the actuator can no longer impart the energy required to rotate the satellite. The system spins indefinitely in the roll axis, compromising the stability of the other two axes in the long term, because of the gyroscopic coupling, until all wheels are saturated.

4 Conclusions

This project successfully designed, implemented, and validated a simplified Attitude Determination and Control System for a 1U CubeSat stabilized by three reaction wheels. The development progressed from a fundamental 1-DOF analysis to a 3-DOF non-linear simulation, culminating in a robust control architecture capable of operating under realistic hardware constraints.

The simulation results confirm that the proposed design is not only theoretically sound but also practically viable, navigating the trade-offs between precision, power consumption, and hardware limitations. Future work would focus on implementing active momentum management (magnetorquers) to de-saturate reaction wheels and expanding the EKF to estimate disturbances.

References

[1] Charles Grassin. Reaction wheel attitude control. *https://charleslabs.fr/en/project-Reaction+Wheel+Attitude+Control*, 2019.

[2] Wikipedia. Cubesat. *https://en.wikipedia.org/wiki/CubeSat*, 2025.

[3] Cubesat. Cubesat information. *https://www.cubesat.org/cubesatinfo*, 2025.

[4] Serenum. Reaction wheels. *https://www.serenumspace.com/products/reaction-wheels*, 2025.

[5] Aspina Group. Satellite attitude control design and reaction wheel. *https://www.aspina-group.com/en/learning-zone/columns/2023041901/*, 2023.

[6] Yaguang Yang. Spacecraft modeling, attitude determination, and control: Quaternion-based approach. *https://ntrs.nasa.gov/api/citations/20240009554/downloads/Space%20Attitude%20Development%20Control.pdf*, 2007.

[7] F. Landis. *Fundamentals of Spacecraft Attitude Determination and Control*. Springer, 2014.

[8] Serenum. Adcs. *https://www.serenumspace.com/products/vac02-adcs*, 2025.

[9] Wikipedia. Spacecraft attitude determination and control. *https://en.wikipedia.org/wiki/Spacecraft_attitude_determination_and_control*, 2004.

Appendix

Extended Kalman filter

1. EKF step discrete function

```
function [x_next, P_next] = ekf_step_discrete(x_prev, P_prev, u_cmd, y_meas
    , I_sat, I_sat_inv, dt, R_noise, Q_noise)
    % Performs one discrete time step of the Extended Kalman Filter.

    % 0. Input Saturation Model
    MAX_TORQUE = 0.0002; % 0.2 mNm
    MAX_MOMENTUM = 0.0006; % 0.6 mNms

    % Clamp Torque Magnitude
    u_limited = max(min(u_cmd, MAX_TORQUE), -MAX_TORQUE);

    % Apply Momentum Speed Limit Logic
    h_est = x_prev(8:10);
    u_act = zeros(3,1);

    for i = 1:3
        % Check if adding torque would exceed momentum limits
        if (h_est(i) >= MAX_MOMENTUM) && (u_limited(i) < 0)
            u_act(i) = 0; % Wheel is full, cannot accelerate
        elseif (h_est(i) <= -MAX_MOMENTUM) && (u_limited(i) > 0)
            u_act(i) = 0; % Wheel is full negative
        else
            u_act(i) = u_limited(i);
        end
    end

    % 1. Predict Step
    % Predict state forward using non-linear dynamics
    % Heun's Method (Runge-Kutta 2) (slower but accurate)
    % Calculate initial slope (k1)
    k1 = ekf_state_transition(x_prev, u_act, I_sat, I_sat_inv);
    % Predict end state using Euler guess
    x_guess = x_prev + k1 * dt;
    % Calculate slope at the guess point (k2)
    k2 = ekf_state_transition(x_guess, u_act, I_sat, I_sat_inv);
    % Average slopes and take final step
    x_pred_raw = x_prev + (dt/2) * (k1 + k2);

    % Normalize predicted quaternion to prevent drift
    q_pred = x_pred_raw(1:4) / norm(x_pred_raw(1:4));
    x_pred = [q_pred; x_pred_raw(5:10)];

    % Calculate Jacobian at previous state for covariance prediction
    A_jac = calculate_jacobian(x_prev, I_sat);

    % Predict Covariance forward (Discrete Time Linearized Propagation)
    % F_k = I + A*dt approx for discrete transition matrix
    F_k = eye(10) + A_jac * dt;
    % Formula for propagating Gaussian uncertainty
    P_pred = F_k * P_prev * F_k' + Q_noise;

    % 2. Update Step
    % Measurement Matrix H is Identity (all states are measured directly)
```

```
    H = eye(10);

    % Calculate Measurement Residual
    y_pred = H * x_pred; % Expected measurement based on prediction
    residual = y_meas - y_pred; % Innovation (new information)

    % Calculate Kalman Gain (K)
    S = H * P_pred * H' + R_noise; % Innovation covariance
    K = P_pred * H' / S;           % Optimal gain

    % Update State Estimate
    x_next_raw = x_pred + K * residual;

    % Normalize updated quaternion again just to be safe
    q_next = x_next_raw(1:4) / norm(x_next_raw(1:4));
    x_next = [q_next; x_next_raw(5:10)];

    % Update Covariance Estimate
    % Using Joseph form for numerical stability: P = (I-KH)P(I-KH)' + KRK'
    I10 = eye(10);
    P_next = (I10 - K*H) * P_pred * (I10 - K*H)' + K*R_noise*K';
end
```

2. EKF state transition function

```
function x_dot = ekf_state_transition(x, u, I_sat, I_sat_inv)
%   Non-linear state dynamics for EKF prediction step.
%   Calculates dx/dt = f(x,u)

    % Unpack State Vector
    q = x(1:4);          % Quaternion [q0; q1; q2; q3]
    omega = x(5:7);      % Angular Velocity [rad/s]
    h_rw = x(8:10);      % Wheel Momentum [Nms]

    % Quaternion Kinematics: q_dot = 1/2 * Omega(omega) * q
    Omega_mat = [0,         -omega(1), -omega(2), -omega(3);
                 omega(1),   0,         omega(3), -omega(2);
                 omega(2), -omega(3),   0,         omega(1);
                 omega(3),   omega(2), -omega(1),  0       ];

    q_dot = 0.5 * Omega_mat * q;

    % Angular Velocity Dynamics (Euler's Equation):
    % omega_dot = I_sat_inv * (u - omega x H_total)
    % System angular momentum
    H_total = I_sat * omega + h_rw;
    % Gyroscopic coupling term
    coupling = cross(omega, H_total);
    % Angular acceleration
    omega_dot = I_sat_inv * (u - coupling);

    % Reaction Wheel Momentum Dynamics
    % h_rw_dot = -tau_wheel_on_body = -u
    h_rw_dot = -u;

    %% Pack Derivative Vector
    x_dot = [q_dot; omega_dot; h_rw_dot];
end
```

3. Jacobian calculation function

```
    function A_num = calculate_jacobian(x_current, I_sat)
%   Calculates the Jacobian matrix A = df/dx at the current state x.
%   x_current: 10x1 state vector [q; omega; h_rw]

    % Unpack current state estimate
    q0 = x_current(1); q1 = x_current(2); q2 = x_current(3); q3 = x_current
        (4);
    wx = x_current(5); wy = x_current(6); wz = x_current(7);
    h_rwx = x_current(8); h_rwy = x_current(9); h_rwz = x_current(10);

    Ixx = I_sat(1,1); Iyy = I_sat(2,2); Izz = I_sat(3,3);

    % Matrix Structure based on Symbolic Derivation from symbolic_jacobian.
       mlx
    % The top-left 4x4 block depends on omega
    A_q_omega = 0.5 * [0, -wx, -wy, -wz;
                       wx, 0,   wz, -wy;
                       wy, -wz, 0,   wx;
                       wz, wy, -wx, 0];

    % The top-right 4x3 block depends on q
    A_q_q = 0.5 * [-q1, -q2, -q3;
                    q0, -q3,  q2;
                    q3,  q0, -q1;
                   -q2,  q1,  q0];

    % The middle block depends on omega, h_rw, and Inertias (Euler terms)
    A_omega_omega = zeros(3,3);
    A_omega_omega(1,2) = (Iyy*wz - Izz*wz - h_rwz)/Ixx;
    A_omega_omega(1,3) = (Iyy*wy - Izz*wy + h_rwy)/Ixx;
    A_omega_omega(2,1) = (Ixx*wz - Izz*wz + h_rwz)/Iyy;
    A_omega_omega(2,3) = (Ixx*wx - Izz*wx - h_rwx)/Iyy;
    A_omega_omega(3,1) = (Ixx*wy - Iyy*wy - h_rwy)/Izz;
    A_omega_omega(3,2) = (Ixx*wx - Iyy*wx + h_rwx)/Izz;

    % The middle-right block relates d(omega) to d(h_rw)
    A_omega_h = zeros(3,3);
    A_omega_h(1,2) = wz/Ixx; A_omega_h(1,3) = -wy/Ixx;
    A_omega_h(2,1) = -wz/Iyy; A_omega_h(2,3) = wx/Iyy;
    A_omega_h(3,1) = wy/Izz; A_omega_h(3,2) = -wx/Izz;

    % --- Assemble Full 10x10 Jacobian ---
    A_num = zeros(10,10);
    A_num(1:4, 1:4) = A_q_omega;
    A_num(1:4, 5:7) = A_q_q;
    A_num(5:7, 5:7) = A_omega_omega;
    A_num(5:7, 8:10) = A_omega_h;

    % Rows 8-10 are all zero because d(h_rw) = -tau (no state dependence)
end
```

Matlab and Simulink files

Final Project - Google Drive