Satellite Attitude Estimation and Control via LQG

Michael Wang
ECE 5555
Date:12/11/17

1. Introduction

The Linear Quadratic Gaussian (LQG) controller has proved to be a useful optimal controller for regulating the state of a dynamical system as well as reference tracking. A LQG system consists of an optimal estimator and a Linear Quadratic Regulator (LQR). Due to the Separation Principle of controller and estimator, the LQG system can be optimal if the estimator and controller are separately optimal. This allows for the independent designs and implementations for both the estimator and the controller. This projects attempts to linearize satellite attitude error dynamics as well as to simulate and implement a LQG controller for attitude reference tracking.

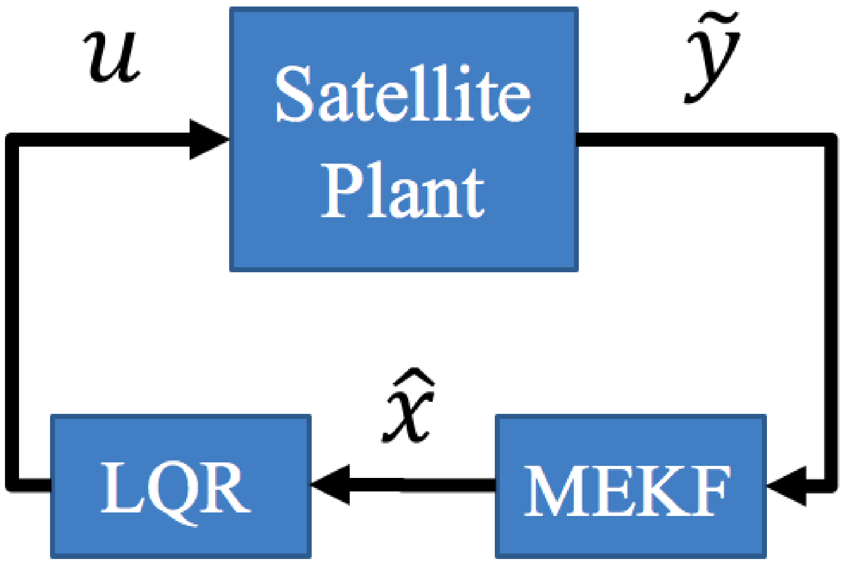

Figure 1. Linear Quadratic Gaussian Regulator Block Diagram

2. Dynamics

The dynamics of a satellite is assumed to be 3D rigid body dynamics. The equation can be derived from Euler's Second Law of Rotation:

$$J\dot{\boldsymbol{\omega}} = \boldsymbol{T} \tag{1}$$

Where $\boldsymbol{\omega}$ is the angular velocity vector of the satellite in inertial coordinates and $\boldsymbol{T}$ is the applied torque in inertial coordinates. Since gyroscope measurements and applied torques are given in body coordinates, the transport equation is used to transform equation (1) into body coordinates:

$$\dot{\boldsymbol{\omega}} = J^{-1}(-\boldsymbol{\omega} \times J\boldsymbol{\omega} + \boldsymbol{T}) \tag{2}$$

As shown, due to the cross product term, the equation of motion is highly nonlinear. Therefore, the equation of motion need to be linearized before the LQG controller can be applied.

3. Quaternions

Euler angles are commonly used in aircrafts to track their yaw, pitch, and roll. However, Euler angles can potentially experience gimbal lock, in which one rotation axis is accidentally aligned with another axis. This potential singularity is fine for aircrafts, since their motion are often very limited. However, a satellite rigid body motion has no limits. As a result, quaternions are used to track the attitude, or orientation, of the satellite with respect to the inertial frame. Quaternions, unlike Euler angles, have the benefit of being singularity-free. A quaternion is defined as:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\rho} \\ q_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}\sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{3}$$

Where **e** is the axis of rotation and θ is the angle of rotation about that axis. A quaternion in nature is a three-dimensional complex number with the form:

$$\boldsymbol{q} = q_1 i + q_2 j + q_3 k + q_4 \tag{4}$$

Just as one-dimensional complex number multiplication represents rotation in 2D, quaternion multiplication represents rotation in 3D. The rules of multiplication are:

$$i^2 = j^2 = k^2 = -ijk = -1 \tag{5}$$

Quaternion multiplication is represented by the symbol $\otimes$. To rotate a vector $\boldsymbol{v}$ about an axis $\boldsymbol{e}$ with an angle θ, the following equation is used:

$$\begin{bmatrix} \boldsymbol{v_{new}} \\ 0 \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} \otimes \boldsymbol{q}^{-\boldsymbol{1}} \tag{6}$$

Where the conjugate quaternion is defined as:

$$\boldsymbol{q}^{-\boldsymbol{1}} = \begin{bmatrix} -\boldsymbol{\rho} \\ q_4 \end{bmatrix} \tag{7}$$

A quaternion essentially tracks the transformation between two different frames of reference. In the application of satellite attitude control, the quaternion represents the transformation between inertial frame and body frame, which is a frame attached to the satellite's rigid body. Another property of quaternions is the negative of a quaternion represents the same rigid body rotation:

$$\boldsymbol{q} = -\boldsymbol{q} \tag{8}$$

4. Linear Quadratic Regulator (LQR)

In this section, a Linear Quadratic Regulator (LQR) based on quaternion error is derived for the satellite attitude control problem. This is based on *Optimal Estimation of Dynamic Systems* by Crassidis and Junkins [1]. First, we can define the propagation equation for the quaternion as:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}\\ 0\end{bmatrix} \otimes \boldsymbol{q} = \frac{1}{2}\Xi(\boldsymbol{q})\boldsymbol{\omega} = \frac{1}{2}\Omega(\boldsymbol{\omega})\boldsymbol{q} \tag{9}$$

Where the matrices $\Xi(\boldsymbol{q})$ and $\Omega(\boldsymbol{w})$ are defined as:

$$\Xi(\boldsymbol{q}) = \begin{bmatrix} q_4 I_{3x3} + [\boldsymbol{\rho}\ \times] \\ -\boldsymbol{\rho}^T \end{bmatrix} \tag{10}$$

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}\ \times] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{11}$$

Where the skew symmetric matrix is defined as:

$$[\boldsymbol{b}\ \times] = \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix} \tag{12}$$

We then define an error quaternion in the form:

$$\delta\boldsymbol{q} = \widehat{\boldsymbol{q}} \otimes \boldsymbol{q}_{\boldsymbol{desired}}^{-1} = \begin{bmatrix}\delta\boldsymbol{\rho}\\ \delta q_4\end{bmatrix} = \begin{bmatrix}\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})\widehat{\boldsymbol{q}} \\ \boldsymbol{q}_{\boldsymbol{desired}}^T\widehat{\boldsymbol{q}}\end{bmatrix} \tag{13}$$

The intuition for $\delta\boldsymbol{q}$ is that it represents a quaternion rotation from the desired quaternion $\boldsymbol{q}_{\boldsymbol{desired}}$ to the estimated quaternion $\widehat{\boldsymbol{q}}$. The error cannot be defined in the conventional arithmetic way ($\boldsymbol{q}_{\boldsymbol{desired}} - \widehat{\boldsymbol{q}}$) because the quaternion norm constraint of 1 will be violated. In addition, $\delta\boldsymbol{\rho}$ conveniently represents the axis in which the satellite must rotate towards $\boldsymbol{q}_{\boldsymbol{desired}}$. This again illustrates the benefit of a quaternion parametrization over Euler angles. The quaternion error represents the "shortest path" between the current quaternion and the desired quaternion. Note that the quaternion error is zero when $\delta\boldsymbol{q} = [0 \quad 0 \quad 0 \quad 1]^T$ (unit quaternion). Therefore, the desired error dynamics that we wish to control can be written as:

$$\delta\ddot{\boldsymbol{\rho}} + L_2\delta\dot{\boldsymbol{\rho}} + L_1\delta\boldsymbol{\rho} = 0 \tag{14}$$

With the linear feedback controller having the form:

$$\boldsymbol{u} = -[L_1 \quad L_2]\begin{bmatrix}\delta\boldsymbol{\rho} \\ \delta\dot{\boldsymbol{\rho}}\end{bmatrix} = -\boldsymbol{L}\begin{bmatrix}\delta\boldsymbol{\rho} \\ \delta\dot{\boldsymbol{\rho}}\end{bmatrix} \tag{15}$$

From equations (14) and (15), the error dynamics have the following *linear* and *time-invariant* form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{16}$$
$$A = \begin{bmatrix}0_{3x3} & I_{3x3} \\ 0_{3x3} & 0_{3x3}\end{bmatrix};\ B = \begin{bmatrix}0_{3x3} \\ I_{3x3}\end{bmatrix};\ \boldsymbol{x} = \begin{bmatrix}\delta\boldsymbol{\rho} \\ \delta\dot{\boldsymbol{\rho}}\end{bmatrix}$$

This system is shown to be reachable:

$$W_r = [B \quad AB \quad \cdots \quad A^5B] \tag{17}$$
$$Rank(W_r) = 6 = Dim(A)$$

As a result, with desired state and fuel penalties *Q* and *R*, the objective function in continuous time becomes

$$F = \frac{1}{2}\int_0^\infty (\boldsymbol{x}^T Q\boldsymbol{x} + \boldsymbol{u}^T R\boldsymbol{u})dt \tag{18}$$

With the linear feedback gain equaling:

$$\boldsymbol{L} = [L_1 \quad L_2] = R^{-1}B^T P \tag{19}$$

Where the matrix *P* is the solution to the algebraic Riccatti equation:

$$PA + A^T P + Q - PBR^{-1}B^T P = 0 \tag{20}$$

This feedback gain is constant and can be computed using MATLAB's function lqr.m offline. To transform the error dynamics control **u** back to the control torque ***T*** in body coordinates, we use equations (2), (9), and (13) with equation (19). First, the error quaternion time derivatives are calculated:

$$\delta\dot{\boldsymbol{\rho}} = \Xi^T(\dot{\boldsymbol{q}}_{\boldsymbol{desired}})\widehat{\boldsymbol{q}} + \Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})\dot{\widehat{\boldsymbol{q}}} \tag{21}$$
$$\delta\ddot{\boldsymbol{\rho}} = \Xi^T(\ddot{\boldsymbol{q}}_{\boldsymbol{desired}})\widehat{\boldsymbol{q}} + 2\Xi^T(\dot{\boldsymbol{q}}_{\boldsymbol{desired}})\dot{\widehat{\boldsymbol{q}}} + \Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})\ddot{\widehat{\boldsymbol{q}}} \tag{22}$$

Substituting equations (21) and (22) into equation (14) yields:

$$\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})\ddot{\widehat{\boldsymbol{q}}} + [2\Xi^T(\dot{\boldsymbol{q}}_{\boldsymbol{desired}}) + L_2\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})]\dot{\widehat{\boldsymbol{q}}} + [\Xi^T(\ddot{\boldsymbol{q}}_{\boldsymbol{desired}}) + L_2\Xi^T(\dot{\boldsymbol{q}}_{\boldsymbol{desired}}) + L_1\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})]\widehat{\boldsymbol{q}} = 0 \tag{23}$$

With

$$\ddot{\hat{\boldsymbol{q}}} = \frac{1}{2}\Xi(\hat{\boldsymbol{q}})\dot{\hat{\boldsymbol{\omega}}} - \frac{1}{4}(\hat{\boldsymbol{\omega}}^T\hat{\boldsymbol{\omega}})\hat{\boldsymbol{q}} \tag{24}$$

The double derivative of desired quaternion can be derived the same way:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{deisred}} = \frac{1}{2}\Xi(\boldsymbol{q}_{\boldsymbol{desired}})\dot{\boldsymbol{\omega}}_{\boldsymbol{desired}} - \frac{1}{4}(\boldsymbol{\omega}_{\boldsymbol{desired}}^T\boldsymbol{\omega}_{\boldsymbol{desired}})\boldsymbol{q}_{\boldsymbol{desired}} \tag{25}$$

Substituting equations (24) and (2) into equation (23) and solving for $\boldsymbol{T}$, we get the final form of the LQR controller:

$$\begin{gathered}\boldsymbol{T} = [\hat{\boldsymbol{\omega}}\ \times]J\hat{\boldsymbol{\omega}} + 2J[\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})\Xi(\hat{\boldsymbol{q}})]^{-1}\left\{\frac{1}{4}(\hat{\boldsymbol{\omega}}^T\hat{\boldsymbol{\omega}})\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}}) -\right. \\ \Xi^T(\dot{\boldsymbol{q}}_{\boldsymbol{desired}})\Omega(\hat{\boldsymbol{\omega}}) - \Xi^T(\ddot{\boldsymbol{q}}_{\boldsymbol{desired}}) - L_1\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}}) - \\ \left. L_2\left[\frac{1}{2}\Xi^T(\boldsymbol{q}_{\boldsymbol{desired}})\Omega(\hat{\boldsymbol{\omega}}) + \Xi^T(\dot{\boldsymbol{q}}_{\boldsymbol{desired}})\right]\right\}\hat{\boldsymbol{q}}\end{gathered} \tag{26}$$

Interestingly, the first right-hand-side term of equation (26) represents a feedforward term that attempts to counteract the nonlinear gyroscopic term ($[\boldsymbol{\omega}\ \times]J\boldsymbol{\omega}$) in the dynamics in equation (2). If the system is perfectly observable with no sensor noise, the LQR will perfectly cancel the gyroscopic term. Since there will be noise in the measurements, the estimate of angular velocity is used instead. The rest of the right-hand-side terms in equation (26) attempt to drive $\delta\boldsymbol{\rho}$ to 0 in an optimal fashion determined by the LQR gains L_1 and L_2.

5. Nonlinear Proportional-Derivative Controller

A simple nonlinear proportional-derivative (PD) controller is also derived for the sake of comparison:

$$\boldsymbol{T} = [\hat{\boldsymbol{\omega}}\ \times]J\hat{\boldsymbol{\omega}} - D(\hat{\boldsymbol{\omega}} - \boldsymbol{\omega}_{\boldsymbol{desired}}) - K\delta\boldsymbol{\rho} \tag{27}$$

Again, the first right-hand-side term is feedforward. The K and D matrices are proportional and derivative gains respectively. They are chosen for a desired time/frequency domain response.

6. Sensor Models

Common sensors for satellite attitude control include gyroscopes for measuring angular velocity vectors in body coordinates and star sensors for measuring attitude. However, all real-world sensors suffer from noise and drift, which need to be accounted for with a Kalman filter. The gyroscope can be modeled as:

$$\begin{gathered}\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\beta} + \boldsymbol{\eta_v} \\ \dot{\boldsymbol{\beta}} = \boldsymbol{\eta_u} \\ \boldsymbol{\eta_v} \sim N(0_{3x1}, \sigma_v^2 I_{3x3}) \\ \boldsymbol{\eta_u} \sim N(0_{3x1}, \sigma_u^2 I_{3x3})\end{gathered} \tag{28}$$

Where the truth $\boldsymbol{\omega}$ is corrupted by gyro bias drift $\boldsymbol{\beta}$, which is modeled as a random walk process with white Gaussian noise $\boldsymbol{\eta_u}$, and an additional white Gaussian noise $\boldsymbol{\eta_v}$. The star sensor measurements can be modeled as:

$$\begin{gathered}\tilde{\boldsymbol{y}}_{\boldsymbol{k}} \equiv h_k(\hat{\boldsymbol{x}}_{\boldsymbol{k}}) + \boldsymbol{n_k} = \begin{bmatrix} A(\boldsymbol{q})\boldsymbol{r_1} \\ A(\boldsymbol{q})\boldsymbol{r_2} \\ \vdots \\ A(\boldsymbol{q})\boldsymbol{r_n} \end{bmatrix} + \begin{bmatrix} \boldsymbol{n_1} \\ \boldsymbol{n_2} \\ \vdots \\ \boldsymbol{n_n} \end{bmatrix} \\ \boldsymbol{n_i} \sim N(0_{3x1}, \sigma_{n_i}^2 I_{3x3})\end{gathered} \tag{29}$$

Where **r** represents the inertial position of a particular star and $A(\boldsymbol{q})$ represents the rotation matrix from inertial to body frame coordinates:

$$A(\boldsymbol{q}) = \Xi^T(\boldsymbol{q})\Psi(\boldsymbol{q}) \tag{30}$$

These body frame vectors are corrupted by white Gaussian noise $\boldsymbol{n}$. The number of rows of the h matrix is a function of the number of stars that can be observed, which depends on the orientation of the satellite and the star catalog in the star sensor. The more measurements there are, the better the MEKF performance.

7. Multiplicative Extended Kalman Filter (MEKF)

A Multiplicative Quaternion Extended Kalman Filter is implemented to provide optimal estimates of quaternion, angular velocity, and gyro bias drift. This is based on *Optimal Estimation of Dynamic Systems* by Crassidis and Junkins [1]. It is "Multiplicative" due to how the error is defined, similar to equation (13):

$$\delta\hat{\boldsymbol{q}} = \boldsymbol{q} \otimes \hat{\boldsymbol{q}}^{-1} = \begin{bmatrix} \delta\hat{\boldsymbol{\rho}} \\ \delta\hat{q}_4 \end{bmatrix} \tag{31}$$

Where $\delta\hat{\boldsymbol{q}}$ represents the quaternion rotation from the estimate $\hat{\boldsymbol{q}}$ to the truth $\boldsymbol{q}$**.** To form the error dynamics, we need to find the time derivative of the quaternion error:

$$\delta\dot{\hat{\boldsymbol{q}}} = \dot{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}}^{-\boldsymbol{1}} + \boldsymbol{q} \otimes \dot{\hat{\boldsymbol{q}}}^{-\boldsymbol{1}} \tag{32}$$

Where $\dot{\boldsymbol{q}}$ is defined by equation (9). To find $\dot{\hat{\boldsymbol{q}}}^{-1}$, we can take time derivative of a unit quaternion:

$$\dot{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}} = 0 = \dot{\hat{\boldsymbol{q}}} \otimes \hat{\boldsymbol{q}}^{-\boldsymbol{1}} + \hat{\boldsymbol{q}} \otimes \dot{\hat{\boldsymbol{q}}}^{-\boldsymbol{1}} \tag{33}$$

With $\dot{\hat{\boldsymbol{q}}}$ defined by equation (9), equation (33) simplifies to:

$$0 = \frac{1}{2}\begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \otimes \hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}}^{-1} + \hat{\boldsymbol{q}} \otimes \dot{\hat{\boldsymbol{q}}}^{-1} = \frac{1}{2}\begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} + \hat{\boldsymbol{q}} \otimes \dot{\hat{\boldsymbol{q}}}^{-1} \tag{34}$$

From equation (34), we can finally get $\dot{\hat{\boldsymbol{q}}}^{-\boldsymbol{1}}$:

$$\dot{\hat{\boldsymbol{q}}}^{-\boldsymbol{1}} = -\frac{1}{2}\hat{\boldsymbol{q}}^{-1} \otimes \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ \boldsymbol{0} \end{bmatrix} \tag{35}$$

Substituting equation (35) into equation (32) yields:

$$\delta\dot{\boldsymbol{q}} = \tfrac{1}{2}\left\{\begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \otimes \delta\boldsymbol{q} - \delta\boldsymbol{q} \otimes \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix}\right\} \tag{36}$$

If we define angular velocity error to be $\delta\boldsymbol{\omega} \equiv \boldsymbol{\omega} - \hat{\boldsymbol{\omega}}$, equation (36) then becomes:

$$\delta\dot{\boldsymbol{q}} = \tfrac{1}{2}\left\{\begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \otimes \delta\boldsymbol{q} - \delta\boldsymbol{q} \otimes \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix}\right\} + \tfrac{1}{2}\begin{bmatrix} \delta\boldsymbol{\omega} \\ 0 \end{bmatrix} \otimes \delta\boldsymbol{q} \tag{37}$$

After some algebraic manipulations, equation (37) can be simplified to:

$$\delta\dot{\boldsymbol{q}} = \begin{bmatrix}\delta\dot{\boldsymbol{\rho}} \\ \delta\dot{q}_4\end{bmatrix} = -\begin{bmatrix}[\widehat{\boldsymbol{\omega}}\ \times]\delta\boldsymbol{\rho} \\ 0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta\boldsymbol{\omega} \\ 0\end{bmatrix} \otimes \delta\boldsymbol{q} \tag{38}$$

By doing first-order approximation on equation (38) such that $\delta\boldsymbol{q} \ \approx [0 \quad 0 \quad 0 \quad 1]^T$, the error dynamics become:

$$\delta\dot{\boldsymbol{\rho}} = -[\widehat{\boldsymbol{\omega}}\ \times]\delta\boldsymbol{\rho} + \frac{1}{2}\delta\boldsymbol{\omega} \tag{39}$$

We can then define gyro bias error and angular velocity error as:

$$\delta\boldsymbol{\omega} \equiv \boldsymbol{\omega} - \widehat{\boldsymbol{\omega}} \tag{40}$$
$$\delta\boldsymbol{\beta} = \boldsymbol{\beta} - \widehat{\boldsymbol{\beta}} \tag{41}$$

Where the angular velocity and gyro bias time derivative estimates can be found by taking expectations:

$$\widehat{\boldsymbol{\omega}} = E\{\boldsymbol{\omega}\} = E\{\widetilde{\boldsymbol{\omega}} - \boldsymbol{\beta} - \boldsymbol{\eta_v}\} = \widetilde{\boldsymbol{\omega}} - \boldsymbol{\beta} \tag{42}$$
$$\dot{\widehat{\boldsymbol{\beta}}} = E\{\dot{\boldsymbol{\beta}}\} = E\{\boldsymbol{\eta_u}\} = 0 \tag{43}$$

Combining equations (40) - (43) with equation (39), we get:

$$\delta\dot{\boldsymbol{\rho}} = -[\widehat{\boldsymbol{\omega}}\ \times]\delta\boldsymbol{\rho} - \frac{1}{2}(\delta\boldsymbol{\beta} + \boldsymbol{\eta_v}) \tag{44}$$

We then make a small angle approximation $\delta\boldsymbol{\rho} \ \approx \frac{\delta\boldsymbol{\alpha}}{2}$, which assumes the estimation error is small. Last but not least, we can the write the error dynamics for MEKF:

$$\Delta\dot{\widetilde{\boldsymbol{x}}}(t) = F(t)\Delta\widetilde{\boldsymbol{x}} + G(t)\boldsymbol{w}(t) \tag{45}$$
$$F(t) = \begin{bmatrix}-[\widehat{\boldsymbol{\omega}}(t)\times] & -I_{3x3} \\ 0_{3x3} & 0_{3x3}\end{bmatrix}; G(t) = \begin{bmatrix}-I_{3x3} & 0_{3x3} \\ 0_{3x3} & I_{3x3}\end{bmatrix}; \Delta\widetilde{\boldsymbol{x}} = \begin{bmatrix}\delta\boldsymbol{\alpha}(t) \\ \delta\boldsymbol{\beta}(t)\end{bmatrix}$$
$$\boldsymbol{w}(t) = \begin{bmatrix}\boldsymbol{\eta_v} \\ \boldsymbol{\eta_u}\end{bmatrix}; Q(t) = E\{\boldsymbol{w}(t)\boldsymbol{w}^T(t)\} = \begin{bmatrix}\sigma_v^2 I_{3x3} & 0_{3x3} \\ 0_{3x3} & \sigma_u^2 I_{3x3}\end{bmatrix}$$

Note the error dynamics for MEKF are driven by “process” noise while the error dynamics for LQR are driven by control input $\boldsymbol{u}$. Next, we must linearize the measurement function in equation (29). We first use the error definition in equation (31) and equation (30) as well as combined rotation transformations:

$$A(\boldsymbol{q}) = A(\delta\boldsymbol{q})A(\widehat{\boldsymbol{q}}^{-1}) \tag{46}$$

Using small angle approximation, the rotation matrix from $\widehat{\boldsymbol{q}}$ to $\boldsymbol{q}$ can be represented as:

$$A(\widehat{\boldsymbol{q}}^{-1}) \approx I_{3x3} - [\delta\boldsymbol{\alpha}\ \times] \tag{47}$$

The sensor output error can be defined as $\delta\boldsymbol{b}$ as:

$$\delta\boldsymbol{b} \equiv \boldsymbol{b} - \widehat{\boldsymbol{b}}^{-} = A(\boldsymbol{q})\boldsymbol{r} - A(\widehat{\boldsymbol{q}}^{-})\boldsymbol{r} = [A(\widehat{\boldsymbol{q}}^{-})\boldsymbol{r}\ \times]\delta\boldsymbol{\alpha} \tag{48}$$

The linearization of equation (29) is then:

$$\delta\boldsymbol{b} = H_k(\widehat{\boldsymbol{x}}_{\boldsymbol{k}}^{-})\Delta\widetilde{\boldsymbol{x}} \tag{49}$$

$$H_k(\widehat{\boldsymbol{x}}_k^{-}) = \begin{bmatrix} [A(\widehat{\boldsymbol{q}}^{-})\boldsymbol{r}_{\boldsymbol{1}}\ \times] & 0_{3x3} \\ [A(\widehat{\boldsymbol{q}}^{-})\boldsymbol{r}_{\boldsymbol{2}}\ \times] & 0_{3x3} \\ \vdots & \vdots \\ [A(\widehat{\boldsymbol{q}}^{-})\boldsymbol{r}_{\boldsymbol{n}}\ \times] & 0_{3x3} \end{bmatrix}$$

The sensor noise covariance can also be derived:

$$R = E\{\boldsymbol{n}(t)\boldsymbol{n}^T(t)\} = \begin{bmatrix} \sigma_1^2 I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & \sigma_2^2 I_{3x3} & 0_{3x3} & 0_{3x3} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{3x3} & 0_{3x3} & \dots & \sigma_n^2 I_{3x3} \end{bmatrix} \tag{50}$$

The MEKF algorithm is the following:

<u>Propagation Step:</u>
Using Runge-Kutta 4th order integrator, the gyro bias, quaternion, and error state covariance are propagated forwards in time for a specified time step. Then the angular velocity is estimated by subtracting gyro measurement by the estimated bias. Note the "+" superscript is the *a posteriori* estimate and the "-" superscript is the *a priori* estimate:

$$\dot{\widehat{\boldsymbol{\beta}}}(t) = 0 \quad => \quad \widehat{\boldsymbol{\beta}}^{+}(t) = \widehat{\boldsymbol{\beta}}^{-}(t) \tag{51}$$

$$\widehat{\boldsymbol{\omega}}(t) = \widetilde{\boldsymbol{\omega}}(t) - \widehat{\boldsymbol{\beta}}(t) \tag{52}$$

$$\dot{\widehat{\boldsymbol{q}}}(t) = \frac{1}{2}\Xi\big(\widehat{\boldsymbol{q}}(t)\big)\widehat{\boldsymbol{\omega}} \tag{53}$$

$$\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t) \tag{54}$$

Update Step:

The update step incorporates new sensor measurements (that are corrupted by white Gaussian noise) to the *a priori* error state and covariance via the Kalman gain, which is a function of linearized measurement model, *a priori* error covariance, and noise covariance.

$$\Delta\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{+} = \begin{bmatrix} \delta\hat{\boldsymbol{\alpha}}_{\boldsymbol{k}}^{+} \\ \delta\hat{\boldsymbol{\beta}}_{\boldsymbol{k}}^{+} \end{bmatrix} = K_k \left[\tilde{\boldsymbol{y}}_k - \begin{bmatrix} A(\hat{\boldsymbol{q}}^{-})\boldsymbol{r}_{\boldsymbol{1}} \\ A(\hat{\boldsymbol{q}}^{-})\boldsymbol{r}_{\boldsymbol{2}} \\ \vdots \\ A(\hat{\boldsymbol{q}}^{-})\boldsymbol{r}_{\boldsymbol{n}} \end{bmatrix} \right] \tag{55}$$

$$P_k^{+} = [I - K_k H_k(\hat{\boldsymbol{x}}_k^{-})]P_k^{-} \tag{56}$$

The Kalman gain is given by:

$$K_k = P_k^{-} H_k^{T}(\hat{\boldsymbol{x}}_k^{-})[H_k(\hat{\boldsymbol{x}}_k^{-})P_k^{-} H_k^{T}(\hat{\boldsymbol{x}}_k^{-}) + R]^{-1} \tag{57}$$

Using $\Delta\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{+}$, we can then update the quaternion and gyro bias estimates:

$$\hat{\boldsymbol{q}}_{\boldsymbol{k}}^{+} = \hat{\boldsymbol{q}}_{\boldsymbol{k}}^{-} + \frac{1}{2}\Xi(\hat{\boldsymbol{q}}_k^{-})\delta\hat{\boldsymbol{\alpha}}_{\boldsymbol{k}}^{+} \tag{58}$$

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{k}}^{+} = \hat{\boldsymbol{\beta}}_{\boldsymbol{k}}^{-} + \delta\hat{\boldsymbol{\beta}}_{\boldsymbol{k}}^{+} \tag{59}$$

The quaternion in equation (58) is normalized after each update to ensure the quaternion norm constraint is satisfied.

8. Simulation Results

Simulation of satellite rigid body dynamics is done using Runge-Kutta 4th order integrator with LQG controller in the loop. First, a reference tracking simulation was conducted with the following parameters:

Mass [kg]	6.26688
Moment of Inertia [kg • m^2]	$\begin{bmatrix} 0.0204 & 0 & 0 \\ 0 & 0.0602 & 0 \\ 0 & 0 & 0.0664 \end{bmatrix}$
State Penalty Q	$\begin{bmatrix} 1e-3 & 0 & 0 \\ 0 & 1e-3 & 0 \\ 0 & 0 & 1e-3 \end{bmatrix}$
Fuel Penalty R	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

LQR Gain L	$\begin{bmatrix} 0.0316 & 0 & 0 & 0.2535 & 0 & 0 \\ 0 & 0.0316 & 0 & 0 & 0.2535 & 0 \\ 0 & 0 & 0.0316 & 0 & 0 & 0.2535 \end{bmatrix}$
$\boldsymbol{q_{desired}}$	$[0.2632 \quad -0.7896 \quad 0.5264 \quad 0.1736]^T$
$\boldsymbol{\omega_{desired}}\ [\frac{rad}{s}]$	$[0 \quad 0 \quad 0]^T$
$\boldsymbol{q_0}$	$[0 \quad 0 \quad 0 \quad 1]^T$
$\boldsymbol{\omega_0}\ [\frac{rad}{s}]$	$[0 \quad 0 \quad 0]^T$
$\boldsymbol{\sigma_u}\ [\frac{rad}{s^{1.5}}]$	$3.1623e-09$
$\boldsymbol{\sigma_v}\ [\frac{rad}{s^{0.5}}]$	$3.1623e-07$
$\boldsymbol{\sigma_n}\ [rad^2]$	$5.8178e-04$
$\boldsymbol{\hat{P}_0}$	$diag([0.3046 \quad 0.3046 \quad 0.3046 \quad 0.0009 \quad 0.0009 \quad 0.0009]) * 1e-4$
$\boldsymbol{\hat{q}_0}$	$[0 \quad 0 \quad 0 \quad 1]^T$
$\boldsymbol{\hat{\beta}_0}[\frac{rad}{s}]$	$[0 \quad 0 \quad 0]^T$

Table 1. Case Study 1: Stationary Reference Tracking

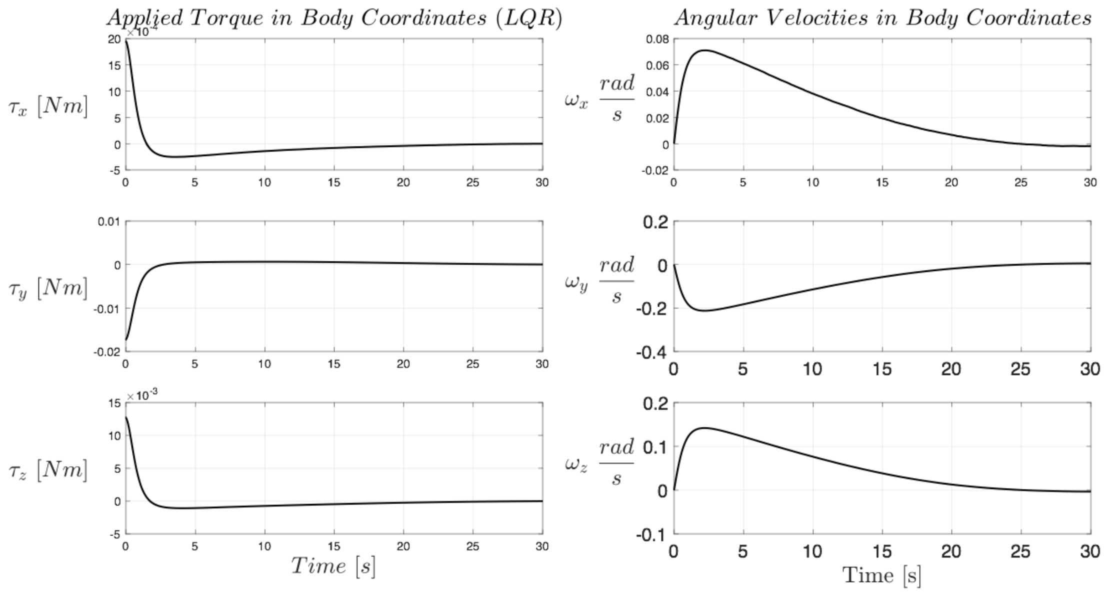

Figure 2. Case Study 1: Applied Torque and Angular Velocity

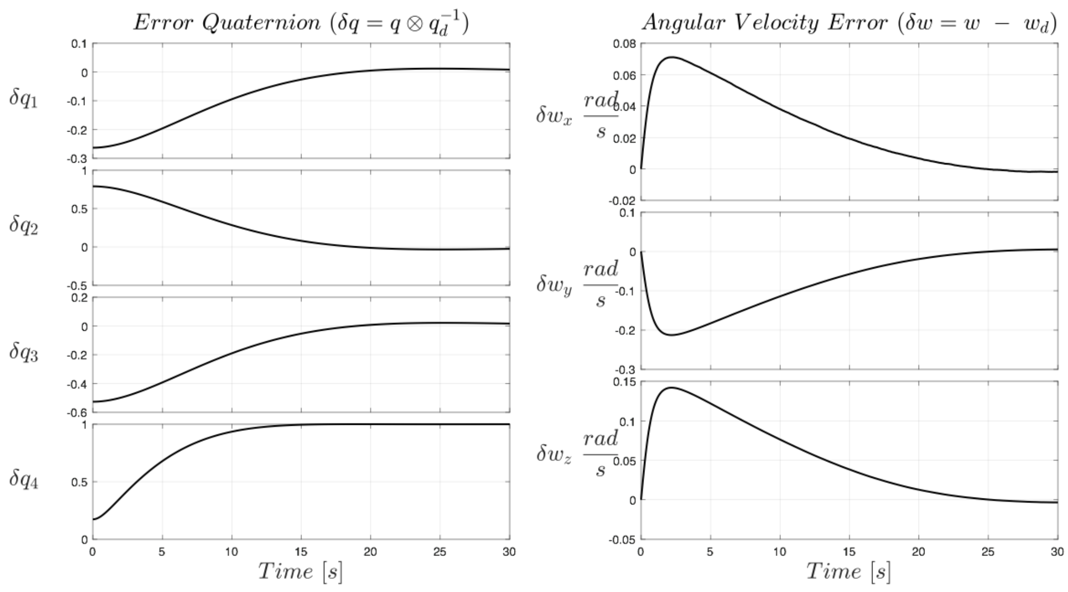

Figure 3. Case Study 1: Quaternion and Angular Velocity Tracking Errors

As shown above, the state of the system converges to the desired quaternion and angular velocity. Note again the quaternion error of 0 corresponds to $[0 \quad 0 \quad 0 \quad 1]^T$.

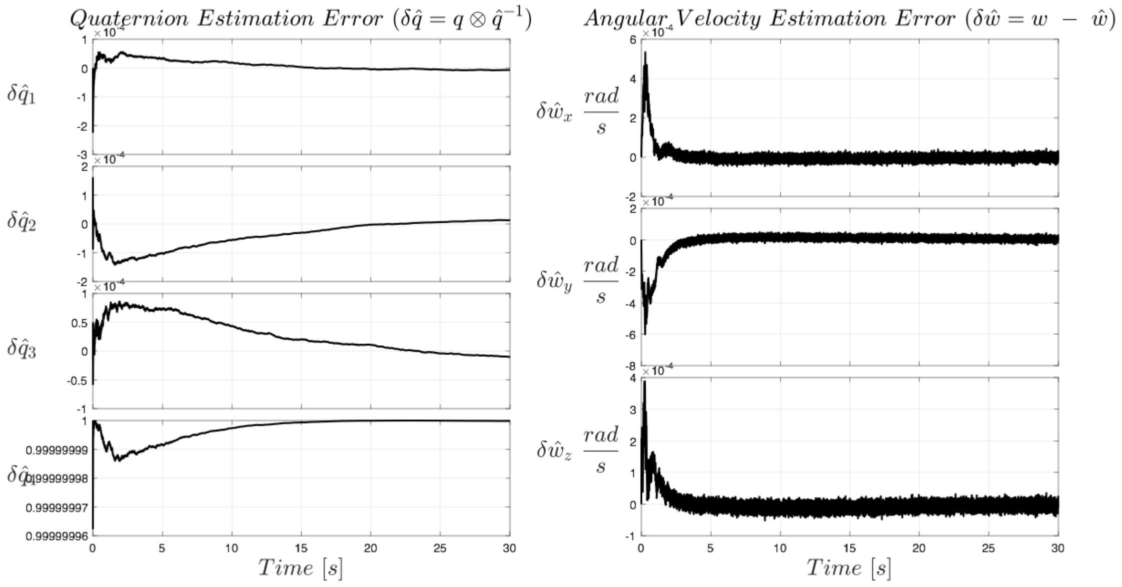

Figure 4. Case Study 1: Quaternion and Angular Velocity Estimation Errors

As shown above, the estimation errors also converge to 0.

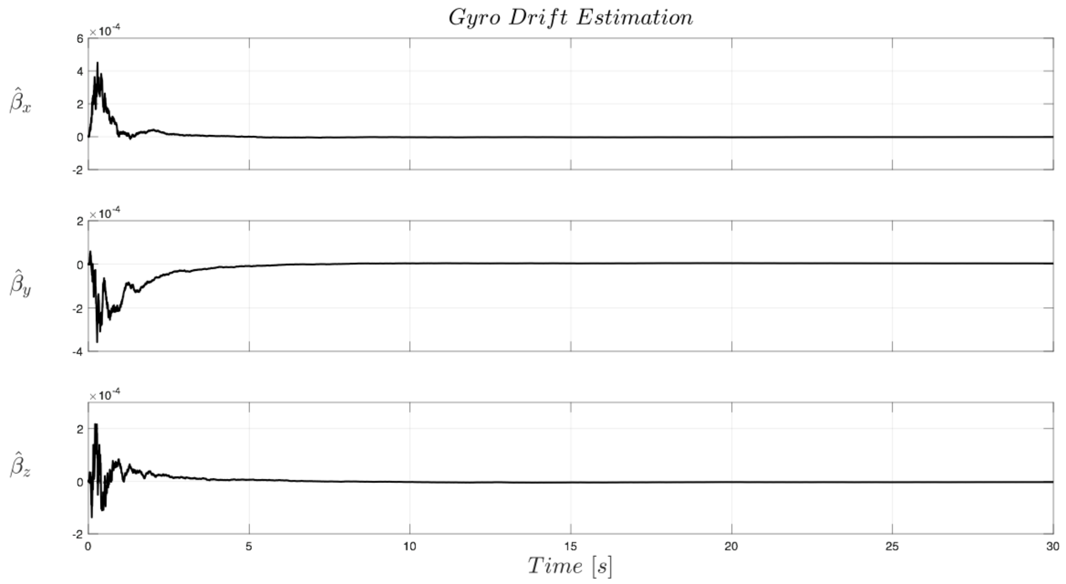

Figure 5. Case Study 1: Gyro Drift Estimation

As shown above, the gyro drift estimate converges to an approximate steady state value, which is expected.

For comparison, the nonlinear proportional-derivative controller is also simulated in place of LQR, with $K = diag([1e-1 \quad 1e-1 \quad 1e-1])$ and $D = diag([5e-2 \quad 5e-2 \quad 5e-2])$.

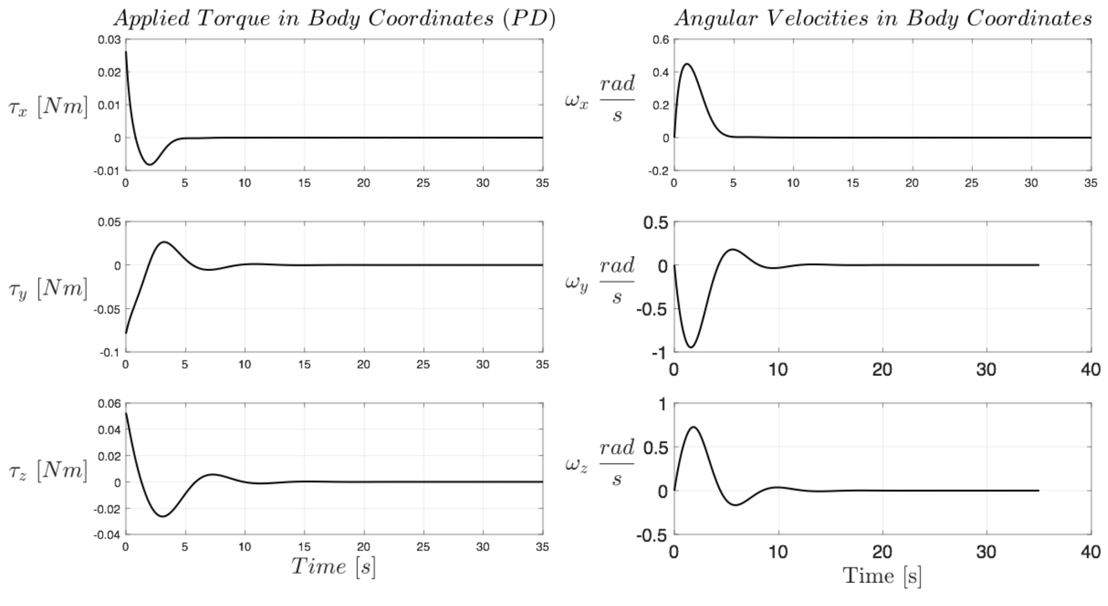

Figure 6. Case Study 2: Applied Torque and Angular Velocity

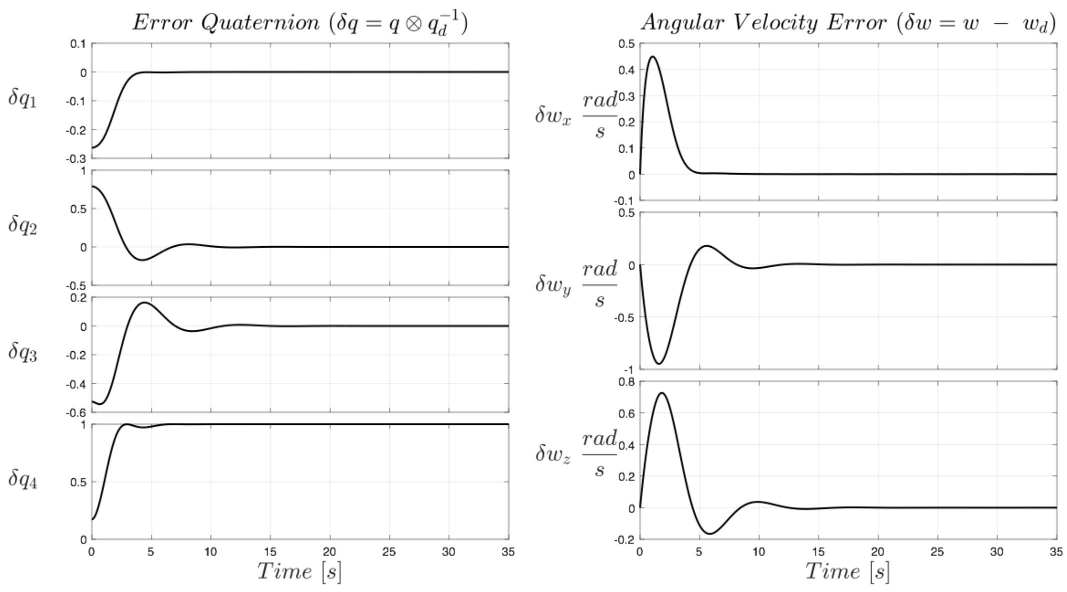

Figure 7. Case Study 2: Quaternion and Angular Velocity Tracking Error

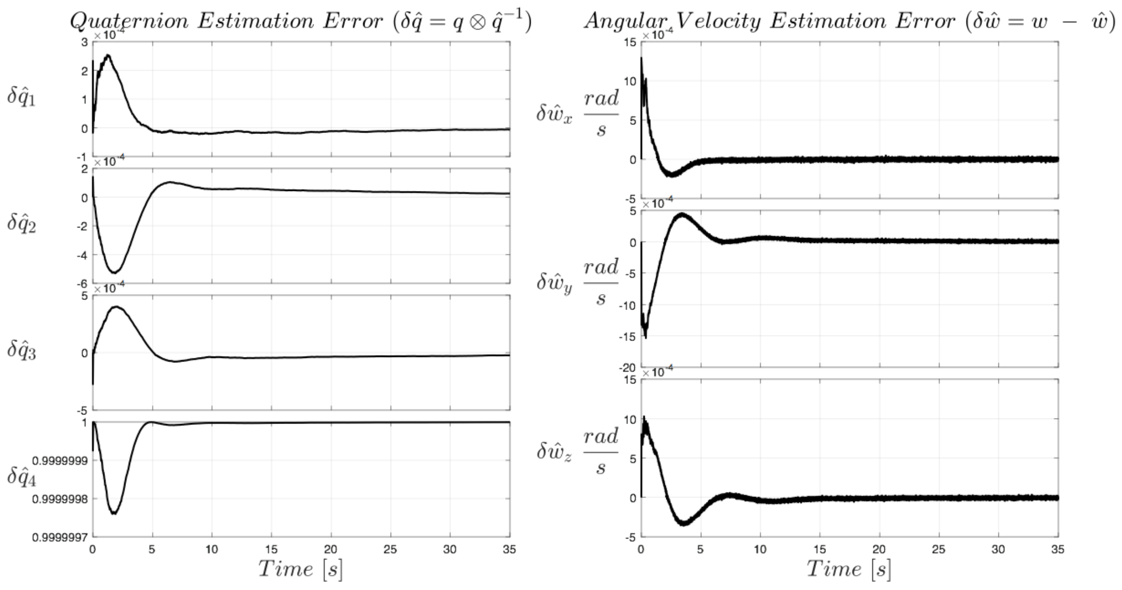

Figure 8. Case Study 2: Quaternion and Angular Velocity Estimation Error

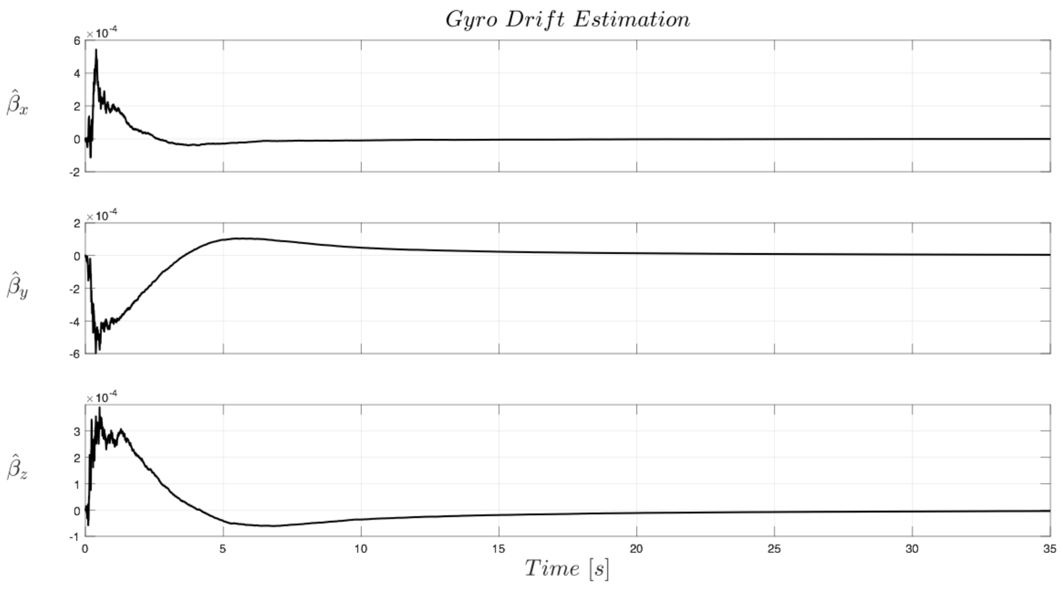

Figure 9. Case Study 2: Gyro Drift Estimation

As shown, the state tracking errors and estimation errors converge to 0 over time. Gyro drift estimates also converge over time. A third case study involving a slewing maneuver via LQG is also simulated with initial conditions:

$\boldsymbol{\omega}_{\boldsymbol{desired}_{\boldsymbol{0}}}\ [\frac{\boldsymbol{rad}}{\boldsymbol{s}}]$	$[0.1 \quad 0.1 \quad -0.1]^T$
$\boldsymbol{q}_{\boldsymbol{desired}_{\boldsymbol{0}}}$	$[0 \quad 0 \quad 0 \quad 1]^T$

Table 2. Case Study 3: Slewing Maneuver Initial Conditions

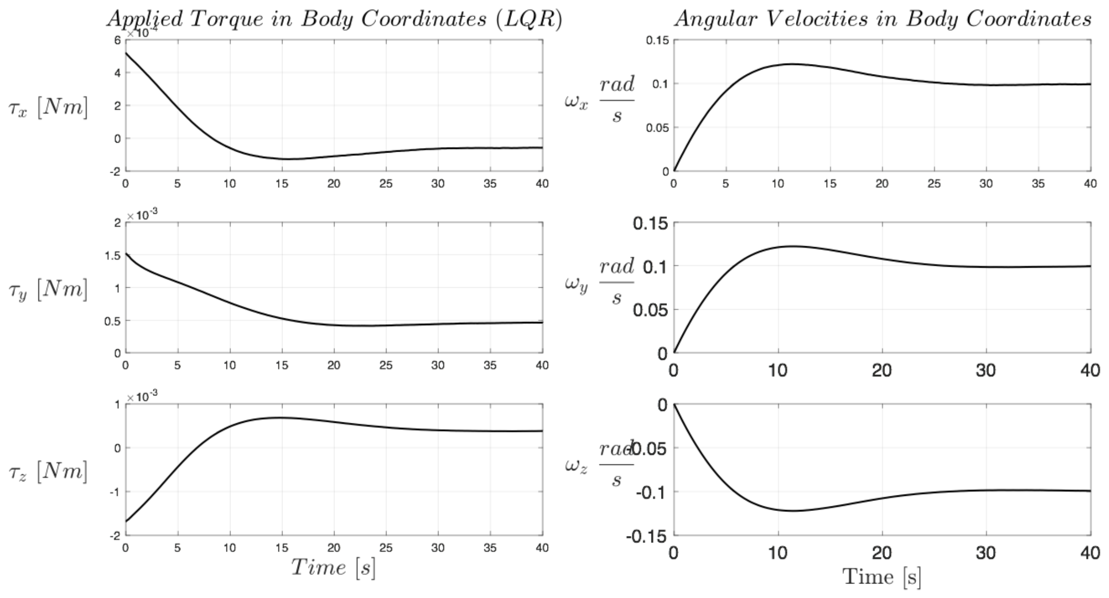

Figure 10. Case Study 3: Applied Torque and Angular Velocity

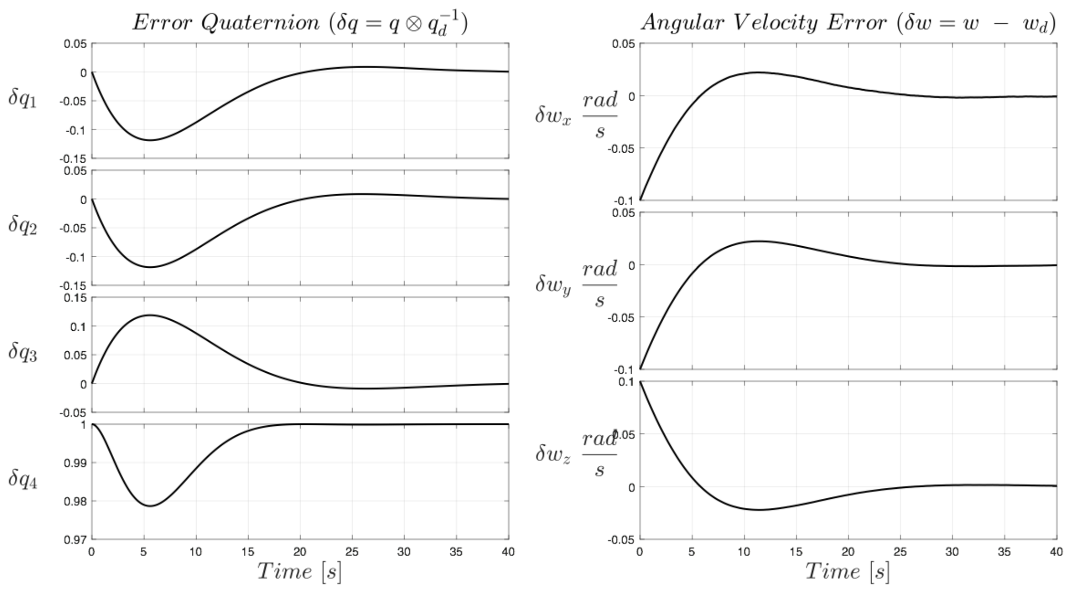

Figure 11. Case Study 3: Quaternion and Angular Velocity Tracking Error

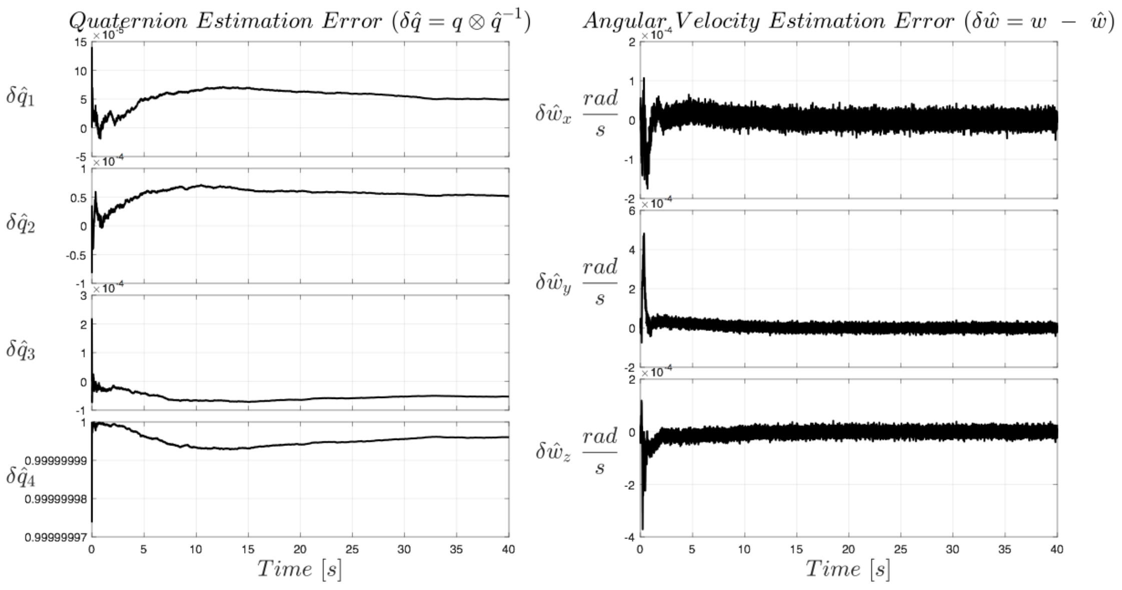

Figure 12. Case Study 3: Quaternion and Angular Velocity Estimation Error

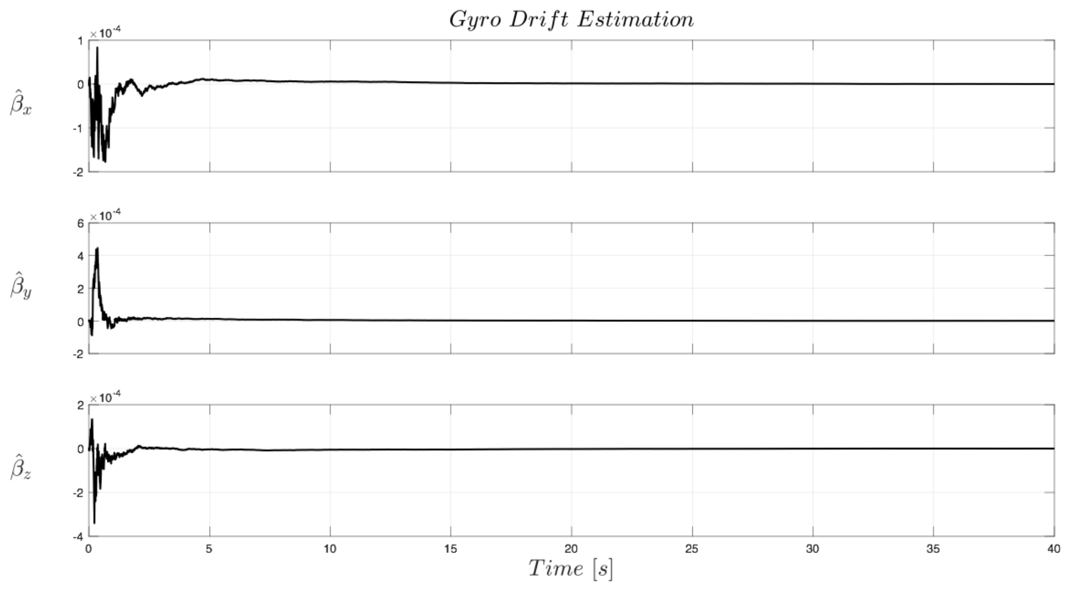

Figure 13. Case Study 3: Gyro Drift Estimation

As shown, the quaternion and angular velocity tracking errors converge to 0 for the slewing maneuver. The angular velocity estimation error also converges to 0. However, there is a small steady state quaternion estimation error on the order of $1e-4$. This is most likely due to lag in estimation, since the desired angular velocity is nonzero. The gyro drift estimate converges to a steady state value.

9. Conclusion

	Advantages	**Disadvantages**
LQR	The linear quadratic regulator can optimize over fuel consumption and state regulation.	Still need to tune Q and R matrices. There are no guaranteed margins for LQG controller.
PD	The proportional-derivative controller is simple to implement. Can easily compute stability margins in frequency domain.	Harder to tune the K and D matrices.

Table 3. LQR vs. PD Controller Comparison

In this project, a LQG controller has been simulated and implemented, which is compared to a simple nonlinear proportional-derivative controller. The advantages and disadvantages of both controllers are outlined in Table 3. In conclusion, the separation principle allows for independent designs and implementations of various controllers and estimators depending on the need and specifications of the problem.

10. References

[1] J. Crassidis and J. Junkins, *Optimal estimation of dynamic systems*. Boca Raton, Fla.: Chapman and Hall/CRC, 2012.
[2] R. Murray, *Optimal-Based Control*. Pasadena: California Institute of Technology, 2017.
[3] B. Wie, H. Weiss and A. Arapostathis, "Quaternion feedback regulator for spacecraft eigenaxis rotation", *Journal of Guidance Control and Dynamics*, 1989.
[4] R. Paielli and R. Bach, "Attitude control with realization of linear error dynamics", *Journal of Guidance Control and Dynamics*, 1993.
[5] M. Blanke and M. Larsen, "Satellite Dynamics and Control in a Quaternion Formulation (2nd edition)", *Technical University of Denmark*, 2010.
[6] L. Markley, "Attitude Error Representations for Kalman FIltering", *NASA Goddard Spaceflight Center*, 2000.
[7] J. Kuipers, *Quaternions and rotation sequences*. Princeton, NJ [u.a.]: Princeton University Press, 2007.
[8] C. Pong, "High-Precision Pointing and Attitude Estimation and Control Algorithms for Hardware-Constrained Spacecraft", PhD., MIT, 2014.