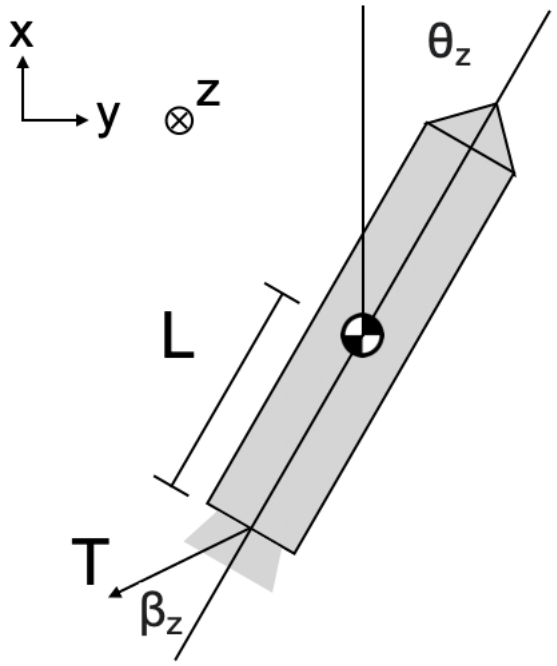
Project Sphinx Dynamics, Controls, and Estimation

Ethan Lai, ethanlai@mit.edu

February 21, 2025

1 Dynamics

A schematic of the Sphinx vehicle with inertial axes labeled is shown below.



1.1 Translational Dynamics

The rotation from the TVC is represented by two angles, β_y and β_z. Thus, we can write quaternion representations of each of the y and z rotations. Assume β_y and β_z are small.

$$\mathbf{q}_y = \begin{bmatrix} \cos(\beta_y/2) \\ 0 \\ \sin(\beta_y/2) \\ 0 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0 \\ \beta_y/2 \\ 0 \end{bmatrix}, \qquad \mathbf{q}_z = \begin{bmatrix} \cos(\beta_z/2) \\ 0 \\ 0 \\ \sin(\beta_z/2) \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ \beta_z/2 \end{bmatrix} \tag{1}$$

These can be composed to get

$$\mathbf{q}_{\text{TVC}} = \mathbf{q}_y \otimes \mathbf{q}_z = \begin{bmatrix} 1 \\ \beta_y\beta_z/4 \\ \beta_y/2 \\ \beta_z/2 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0 \\ \beta_y/2 \\ \beta_z/2 \end{bmatrix} \tag{2}$$

Note that because of the small angle approximation, this operation is commutative, which is what we expect. Also note that this rotation quaternion does not have unit norm. If we assume small angles, the error is only second order, but for long-term dynamics propagation, we must renormalize this in each time step.

The thrust generated by the propeller (in the propeller frame) is given by

$$\mathbf{T}^P = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

Thus, we can get the thrust in the body frame as

$$\mathbf{T}^B = \mathbf{q}_{\text{TVC}} \otimes \begin{bmatrix} 0 \\ T \\ 0 \\ 0 \end{bmatrix} \otimes \mathbf{q}^*_{\text{TVC}} \tag{4}$$

where we have used the P and B superscripts to denote the propeller and body reference frames respectively. If we have the body to inertial quaternion $\mathbf{q}_{\text{body}}$, we can write the inertial thrust as

$$\mathbf{T}^N = \mathbf{q}_{\text{body}} \otimes \mathbf{T}^B \otimes \mathbf{q}^*_{\text{body}} \tag{5}$$

where the N superscript denotes the inertial frame. From here, the position dynamics can be propagated as

$$\dot{\mathbf{v}} = \frac{\mathbf{T}^N}{m} + \mathbf{g}^N \tag{6}$$

$$\dot{\mathbf{x}} = \mathbf{v} \tag{7}$$

1.2 Attitude Dynamics

Define the angular velocity in the body frame to be

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{8}$$

Then the quaternion dynamics can be propagated as

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \tag{9}$$

where

$$\Omega(\omega) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{10}$$

The angular velocity dynamics can be written as

$$\dot{\boldsymbol{\omega}} = J^{-1}(\boldsymbol{\tau}^B - \boldsymbol{\omega} \times J\boldsymbol{\omega}) \tag{11}$$

where J is the inertial matrix and $\boldsymbol{\tau}^B$ is the total torque in the body frame, which can be decomposed into torque contribution from thrust and torque contribution from aerodynamics on the contrarotating propellers.

$$\boldsymbol{\tau}^B = \boldsymbol{\tau}^B_{\text{thrust}} + \boldsymbol{\tau}^B_{\text{prop}} \tag{12}$$

The thrust component can be written as

$$\boldsymbol{\tau}^B_{\text{thrust}} = \mathbf{r}_{\text{TVC}} \times \mathbf{T}^B = \begin{bmatrix} -L \\ 0 \\ 0 \end{bmatrix} \times \mathbf{T}^B \tag{13}$$

Note that here we used $\mathbf{T}^B$ to represent a vector, whereas previously it was used to represent a pure vector quaternion. The vector representation can be recovered by taking the vector part of the pure vector quaternion. Notation for pure vector quaternions and vectors will be used somewhat interchangeably.

The torque in the body frame due to the contrarotating propellers can be written very similarly to the thrust in the body frame. The torque generated by the propeller in its own frame is

$$\boldsymbol{\tau}^P_{\text{prop}} = \begin{bmatrix} M \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

and the torque in the body frame is

$$\boldsymbol{\tau}^B_{\text{prop}} = \mathbf{q}_{\text{TVC}} \otimes \boldsymbol{\tau}^P_{\text{prop}} \otimes \mathbf{q}^*_{\text{TVC}} \tag{15}$$

2 Controls

2.1 Position Tracking

Suppose we want to track a certain state, typically some position with zero velocity, identity attitude, and zero angular velocity. Throughout this section, we will use the PID{} function signature to indicate PID with three separate gains, integrator anti-windup logic ($\tau = 0.1$) and a filtered derivative ($\tau_d = 0.0005$). Any deviations from this standard PID controller will be explicitly mentioned.

2.1.1 Altitude Controller

The altitude controller controller is given by

$$T_x = \text{PID}\{x_e\} \tag{16}$$

where x_e is the error in the vertical position and T_x is the desired thrust in the vertical direction. Then, the thrust needs to be compensated for the attitude of the vehicle. If we define the vertical efficiency η_x as

$$\begin{bmatrix} \eta_x \\ \cdot \\ \cdot \end{bmatrix} = \mathbf{q}_{\text{body}} \otimes \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \otimes \mathbf{q}^*_{\text{body}} \tag{17}$$

then (ignoring the TVC gimbal angle), the thrust level is

$$T = \frac{T_x}{\eta_x} \tag{18}$$

Note that this logic must happen inside the PID function, before the anti-windup logic to guarantee the commanded thrust level T does not exceed the saturation limit of the propellers, and that the anti-windup is handled properly.

2.1.2 Lateral Position Controller

The y and z positions are governed by the lateral position controller, which takes the error on each of these axes and generates a desired corresponding tilt angle. These are run as two independent PID controllers, where

$$\theta_z = \text{PID}\{y_e\}$$
$$\theta_y = -\text{PID}\{z_e\}$$

It is **extremely important** that θ_z is associated with y_e and θ_y is associated with z_e. Additionally, the minus sign is **critical**.

2.1.3 Tilt Controller

From here, the controller forms the desired attitude

$$\mathbf{q}_{\text{des}} = \begin{bmatrix} 1 \\ 0 \\ \theta_y/2 \\ \theta_z/2 \end{bmatrix} \tag{19}$$

and normalizes $\mathbf{q}_{\text{des}}$. The error quaternion is then generated as

$$\mathbf{q}_e = \mathbf{q}_{\text{des}} \otimes \mathbf{q}^*_{\text{body}} \tag{20}$$

Because of the double-cover property of quaternions, we want to ensure the rotation encoded by $\mathbf{q}_e$ is less than 360°, which we can do by checking if the scalar part of $\mathbf{q}_e$ is negative. If the scalar part is negative, we take the error quaternion to be $-\mathbf{q}_e$. We can write the error quaternion in component form as

$$\mathbf{q}_e = \begin{bmatrix} s_e \\ v_{1,e}/2 \\ v_{2,e}/2 \\ v_{3,e}/2 \end{bmatrix} \tag{21}$$

The "tilt error" is then

$$\text{tilt error} = \begin{bmatrix} v_{2,e} \\ v_{3,e} \end{bmatrix} \tag{22}$$

The tilt controller takes this tilt error as an input, and outputs the commanded gimbal angles for the TVC. First, PID is used to generate the desired torque in each direction. The controller runs two separate PID loops, so it can be thought of as a y-tilt controller and a z-tilt controller.

$$\begin{bmatrix} \tau_y \\ \tau_z \end{bmatrix} = \text{PID}\left\{ \begin{bmatrix} v_{2,e} \\ v_{3,e} \end{bmatrix} \right\} \tag{23}$$

Then, we compute the angles as

$$\beta_y = \frac{\tau_y}{TL} \tag{24}$$

$$\beta_z = \frac{\tau_z}{TL} \tag{25}$$

where T is the thrust output from the altitude controller. As with the altitude controller, there is the nuance where this compensation must happen before the anti-windup logic. The tilt controller also uses ω_y and ω_z as measured directly from the sensor, rather than a derivative.

2.1.4 Roll Controller

The "roll error" is $v_{1,e}$, taken from the error quaternion from before. The roll command is then

$$M = \text{PID}\{v_{1,e}\} \tag{26}$$

2.1.5 Gains and Summary

Controller	Input Variable	Unit	Output Variable	Output Min	Output Max	Unit
Altitude	x_e	m	T	7.848	12.753	N
y-Lateral Position	y_e	m	θ_z	-0.1745	0.1745	rad
z-Lateral Position	z_e	m	$-\theta_y$	-0.1745	0.1745	rad
y-Tilt	$v_{2,e}$	·	β_y	-0.1745	0.1745	rad
z-Tilt	$v_{3,e}$	·	β_z	-0.1745	0.1745	rad
Roll	$v_{1,e}$	·	M	-0.0056	0.0056	Nm

Table 1: Controller Overview

Controller	K_p	K_i	K_d
Altitude	10	0.4494	25
y-Lateral Position	0.015	0.005	0.07
z-Lateral Position	0.015	0.005	0.07
y-Tilt	-0.7	-0.5	-0.2
z-Tilt	-0.7	-0.5	-0.2
Roll	0.00056	0.0056	0.056

Table 2: Controller Gains

Note that I am aware the Tilt gains could be made positive and then the output would be the negative gimbal angle. Alternatively the z-Lateral Position gains could be negative, which would remove the confusing negative sign in the Lateral Position control law. Oh no.

3 State Estimation

The state estimation module will use a Multiplicative Extended Kalman Filter with the following state

$$x = \begin{bmatrix} p \\ v \\ q \end{bmatrix} \tag{27}$$

The model currently accounts for the accelerometer, gyroscope, magnetometer, GPS, and altimeter. It will soon be updated to account for the time-of-flight proximity sensor.

3.1 Sensor Models

For each sensor, we have a sensor model in the form

$$\mathbf{y}_k = h(x) \tag{28}$$

where h is a (in general) nonlinear measurement model. For the "extended" portion of the Extended Kalman Filter, we need to generate a linearized version of this about the state, i.e.

$$C_{k+1} = \left.\frac{\partial y}{\partial x}\right|_{x_{k+1}} = \left.\begin{bmatrix}\frac{\partial y}{\partial p} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial q}\end{bmatrix}\right|_{x_{k+1}} \tag{29}$$

An extra complication here is that taking the derivative with respect to $\mathbf{q}$ is weird and complicated, as is taking the derivative of $\mathbf{q}$ with respect to something else. In general, taking the derivative of something with respect to $\mathbf{q}$ will generate require you to "naively" differentiate, then multiply the result by $G(\mathbf{q})$ on the right. Similarly, to take the derivative of $\mathbf{q}$ with respect to something else will require you to "naively" differentiate, then multiply the result by $G^T(\mathbf{q})$ on the left. In both cases, taking the $d\mathbf{q}$ should result in dimension 3 along the corresponding axis, not dimension 4. This should be guaranteed by the attitude Jacobian $G(\mathbf{q})$.

In addition to this, each sensor has some inherent noise, which can be assumed to be a multivariate zero-mean Gaussian with some covariance matrix W. The covariance can be measured for each sensor empirically. This covariance is used in the update step of the Kalman filter.

3.1.1 Accelerometer

The accelerometer measures the true acceleration vector in the body frame summed with the gravity vector in the body frame. Thus, the measurement model is

$$\mathbf{y}_k = Q^T(\mathbf{q}_k)[\mathbf{a}^N_{\text{k, true}} + \mathbf{g}^N] \tag{30}$$

First define the reference vector (in the inertial frame) as

$$\mathbf{r}^N_k = \mathbf{a}^N_{\text{k, known}} + \mathbf{g}^N \tag{31}$$

where

$$\mathbf{a}^N_{\text{k, known}} = \frac{Q(\mathbf{q}_k)Q(\mathbf{q}_{\text{k, TVC}})T^P_k}{m} + \mathbf{g}^N \tag{32}$$

where

$$T^P = \begin{bmatrix}T\\0\\0\end{bmatrix} \tag{33}$$

Note that T here is thrust, not the conjugate matrix. Apologies for the overloaded notation but this is the only time it should be a source of confusion. This is the best guess for the body acceleration at time k based on the control input. With this in mind, we can write

$$C_{q,\,k+1} = \left.\frac{\partial \mathbf{y}}{\partial \mathbf{q}}\right|_{x_{k+1}} = H^T[L^T(\mathbf{q}_k)L(\mathbf{a}^N_{k,\text{ known}}) + R(\mathbf{q}_k)R(\mathbf{a}^N_{k,\text{ known}})T]G(\mathbf{q}_k) \tag{34}$$

It is clear that

$$\frac{\partial \mathbf{y}}{\partial \mathbf{p}} = 0,\ \frac{\partial \mathbf{y}}{\partial \mathbf{v}} = 0 \tag{35}$$

so the C matrix is

$$C_{k+1} = \begin{bmatrix}0_{3x3} & 0_{3x3} & C_{q,\,k+1}\end{bmatrix} \tag{36}$$

Also, the empirically determined covariance matrix is

$$W = \begin{bmatrix}2\times10^{-4} & 0 & 0\\ 0 & 7\times10^{-5} & 0\\ 0 & 0 & 3.6\times10^{-4}\end{bmatrix} \tag{37}$$

3.1.2 Gyroscope

The gyroscope directly measures the angular velocity.

$$\mathbf{y} = \boldsymbol{\omega} \tag{38}$$

We don't need to use the gyroscope for the "update" portion of the MEKF, only the "predict" step. This means that we don't need to worry about the measurement model. Also, we don't need to worry about gyroscope drift because the flight time is relatively short (around 90 seconds) and we can average out the initial bias and subtract it in a pre-flight calibration sequence.

However, the covariance of the gyroscope is still relevant, because in principle that can be used as a proxy for the process noise. Unfortunately due to unmodeled wind and other aerodynamic effects, the process noise will be assumed much higher. Regardless, the measured gyroscope covariance is reproduced here.

$$W = \begin{bmatrix} 8 \times 10^{8} & 0 & 0 \\ 0 & 1.5 \times 10^{-6} & 0 \\ 0 & 0 & 9 \times 10^{-7} \end{bmatrix} \tag{39}$$

3.1.3 Magnetometer

The magnetometer measures the magnetic field in the body frame, i.e.

$$\mathbf{y} = Q^{T}(\mathbf{q})\mathbf{r}^{N} \tag{40}$$

where $\mathbf{r}^{N}$ is the unit vector pointing towards magnetic north at the Earth's surface. If we start the Sphinx vehicle such that this is in the positive y direction, we have

$$\mathbf{r}^{N} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{41}$$

This measurement model is very similar to the accelerometer, but the accelerometer has a much more complication version of $\mathbf{r}^{N}$. We can write

$$C_{q,\,k+1} = H^{T}[L^{T}(\mathbf{q}_k)L(H\mathbf{r}^{N}) + R(\mathbf{q}_k)R(H\mathbf{r}^{N})T]G(q) \tag{42}$$

so the C matrix is

$$C_{k+1} = \begin{bmatrix} 0_{3x3} & 0_{3x3} & C_{q,\,k+1} \end{bmatrix} \tag{43}$$

The measured covariance matrix is

$$W = \begin{bmatrix} 7 \times 10^{-5} & 0 & 0 \\ 0 & 1.5 \times 10^{-5} & 0 \\ 0 & 0 & 1.5 \times 10^{-5} \end{bmatrix} \tag{44}$$

3.1.4 GPS

We are using GPS only for lateral displacement measurement, so the measurement is

$$\mathbf{y} = \begin{bmatrix} y \\ z \end{bmatrix} \tag{45}$$

and the C matrix is

$$C_{k+1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{46}$$

The measured covariance matrix is

$$W = \begin{bmatrix} 1.35 \times 10^{-2} & 0 \\ 0 & 1.35 \times 10^{-2} \end{bmatrix} \tag{47}$$

3.1.5 Altimeter

The altimeter reads only the altitude, so the measurement is

$$\mathbf{y} = x \tag{48}$$

and the C matrix is

$$C_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{49}$$

The covariance "matrix" is

$$W = \begin{bmatrix} 1.35 \times 10^{-2} \end{bmatrix} \tag{50}$$

3.2 Predict

3.2.1 Propagate Dynamics

The update step of the Kalman filter begins with

$$x_0 = \begin{bmatrix} p \\ v \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; P = 0_{9x9} \tag{51}$$

Then, the dynamics are propagated as

$$p_{k+1} = p_k + v_k \Delta t \tag{52}$$

$$v_{k+1} = v_k + \left(\frac{Q(\mathbf{q}_k)\mathbf{T}^B}{m} + \mathbf{g}^N \right) \Delta t \tag{53}$$

$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \text{expq}\left(\frac{1}{2} \Delta t H \boldsymbol{\omega}_k \right) \tag{54}$$

where $\mathbf{T}^B$ is the thrust in the body frame defined in Equation 4. The expq function is defined as

$$\text{expq}(\boldsymbol{\phi}) = \begin{bmatrix} \cos\left(|\boldsymbol{\phi}|\right) \\ \frac{\boldsymbol{\phi}}{|\boldsymbol{\phi}|} \sin(|\boldsymbol{\phi}|) \end{bmatrix} \tag{55}$$

which converts an axis angle vector into a quaternion. Note that

$$\Delta q_k = \text{expq}\left(\frac{1}{2} \Delta t H \boldsymbol{\omega}_k \right) \tag{56}$$

so

$$\mathbf{q}_{k+1} = R(\Delta \mathbf{q}_k)\mathbf{q}_k \tag{57}$$

3.2.2 Update Covariance

To update the covariance, we need to generate a linearized model of the dynamics. The dynamics are in the form

$$\begin{bmatrix} \mathbf{p}_{k+1} \\ \mathbf{v}_{k+1} \\ \mathbf{q}_{k+1} \end{bmatrix} = f(\mathbf{p}_k, \mathbf{v}_k, \mathbf{q}_k) \tag{58}$$

We need

$$A_k = \left. \frac{\partial f}{\partial x} \right|_{x_k} = \begin{bmatrix} \frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{p}_k} & \frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{p}_{k+1}}{\partial \mathbf{q}_k} \\ \frac{\partial \mathbf{v}_{k+1}}{\partial \mathbf{p}_k} & \frac{\partial \mathbf{v}_{k+1}}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{v}_{k+1}}{\partial \mathbf{q}_k} \\ \frac{\partial \mathbf{q}_{k+1}}{\partial \mathbf{p}_k} & \frac{\partial \mathbf{q}_{k+1}}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{q}_{k+1}}{\partial \mathbf{q}_k} \end{bmatrix} = \begin{bmatrix} I_{3x3} & \Delta t I_{3x3} & 0 \\ 0 & I_{3x3} & \frac{\partial \mathbf{v}_{k+1}}{\partial \mathbf{q}_k} \\ 0 & 0 & \frac{\partial \mathbf{q}_{k+1}}{\partial \mathbf{q}_k} \end{bmatrix} \tag{59}$$

where

$$\frac{\partial \mathbf{v}_{k+1}}{\partial \mathbf{q}_k} = \frac{\Delta t}{m} H^T [L^T(\mathbf{q}_k) L(H\mathbf{T}^B) + R(\mathbf{q}) R(H\mathbf{T}^B) T] G(\mathbf{q}_k) \tag{60}$$

and

$$\frac{\partial \mathbf{q}_{k+1}}{\partial \mathbf{q}_k} = G(\mathbf{q}_{k+1})^T R(\Delta \mathbf{q}_k) G(\mathbf{q}_k) \tag{61}$$

The A matrix should have dimension 9x9. Now update the covariance via

$$P_{k+1|k} = A_k P_{k|k} A_k^T + V \tag{62}$$

where V is the process noise covariance. The numerical value will be written out later.

3.3 Update

The update steps can be run asynchronously from the predict step. As soon as a measurement is ready, it can be used to perform an update step, which will update the entire state and covariance matrix. The general moves for the update step are as follows:

1. Innovation
2. Innovation covariance
3. Kalman gain
4. Update

However, each sensor is sufficiently different that is it worth writing out the steps for each sensor individually. Also, the accelerometer and magnetometer measurements depend on the attitude $\mathbf{q}$, which means that we must use a multiplicative approach (hence *Multiplicative* Extended Kalman Filter).

3.3.1 Accelerometer

1. Innovation

$$\mathbf{z}_{k+1} = \mathbf{y}_k - Q^T(\mathbf{q}_k)(\mathbf{a}^N_{k,\text{ known}} + \mathbf{g}^N) \tag{63}$$

2. Innovation covariance

$$S_{k+1} = C_{k+1} P_{k+1|k} C_{k+1}^T + W \tag{64}$$

3. Kalman gain

$$K_{k+1} = P_{k+1|k} C_{k+1}^T S_{k+1}^{-1} \tag{65}$$

4. Update

$$\begin{bmatrix} \cdot \\ \cdot \\ \boldsymbol{\phi}_{3\text{x}1} \end{bmatrix} = K_{k+1} z_{k+1} \tag{66}$$

$$\mathbf{q}_{k+1|k+1} = L(\mathbf{q}_{k+1|k}) \begin{bmatrix} \sqrt{1 - \boldsymbol{\phi}^T \boldsymbol{\phi}} \\ \boldsymbol{\phi} \end{bmatrix} \tag{67}$$

$$P_{k+1|k+1} = (I - K_{k+1} C_{k+1}) P_{k+1|k} (I - K_{k+1} C_{k+1})^T + K_{k+1} W K_{k+1}^T \tag{68}$$

3.3.2 Magnetometer

1. Innovation

$$\mathbf{z}_{k+1} = \mathbf{y}_{k+1} - Q^T(\mathbf{q}_k)\mathbf{r}^N \tag{69}$$

2. Innovation covariance

$$S_{k+1} = C_{k+1}P_{k+1|k}C_{k+1}^T + W \tag{70}$$

3. Kalman gain

$$K_{k+1} = P_{k+1|k}C_{k+1}^T S_{k+1}^{-1} \tag{71}$$

4. Update

$$\begin{bmatrix} \cdot \\ \cdot \\ \boldsymbol{\phi}_{3\text{x}1} \end{bmatrix} = K_{k+1}z_{k+1} \tag{72}$$

$$\mathbf{q}_{k+1|k+1} = L(\mathbf{q}_{k+1|k}) \begin{bmatrix} \sqrt{1 - \boldsymbol{\phi}^T\boldsymbol{\phi}} \\ \boldsymbol{\phi} \end{bmatrix} \tag{73}$$

$$P_{k+1|k+1} = (I - K_{k+1}C_{k+1})P_{k+1|k}(I - K_{k+1}C_{k+1})^T + K_{k+1}WK_{k+1}^T \tag{74}$$

3.3.3 GPS

1. Innovation

$$\mathbf{z}_{k+1} = \mathbf{y}_{k+1} - C_{k+1}\mathbf{x}_{k+1|k} \tag{75}$$

2. Innovation covariance

$$S_{k+1} = C_{k+1}P_{k+1|k}C_{k+1}^T + W \tag{76}$$

3. Kalman gain

$$K_{k+1} = P_{k+1|k}C_{k+1}^T S_{k+1}^{-1} \tag{77}$$

4. Update

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k} + K_{k+1}\mathbf{z}_{k+1} \tag{78}$$

$$P_{k+1|k+1} = (I - K_{k+1}C_{k+1})P_{k+1|k}(I - K_{k+1}C_{k+1})^T + K_{k+1}WK_{k+1}^T \tag{79}$$

3.3.4 Altimeter

The altimeter is the same as the GPS, but with a different C matrix, so there is no need to reproduce all the equations. Some care needs to be taken when implementing this in software, since the altimeter measurement space is 1-dimension, this could require some extra attention in some linear algebra libraries.

3.4 MEKF Summary

In summary, there is the "predict" step and the "update" step, which can happen completely asynchronously. In implementation, this can be done by maintaining a "most up-to-date" state vector and state covariance, which will be continuously propagated by the predict step, and continuously corrected by the update step as sensor measurements come in. Additionally, the control law can always just use the most up-to-date available state from the estimator.

Appendix

A Quaternions

A.1 Quaternion Basics

Quaternions are a 4-element parameterization of three-dimensional attitude. They represent a rotation from a set of body axes to inertial axes. A quaternion can be thought of as a unit vector $\mathbf{u}$ that serves as an axis of rotation and an angle θ about that unit vector. From these two, the corresponding quaternion is written as

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} s \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \mathbf{u}\sin(\theta/2) \end{bmatrix} \tag{80}$$

Quaternions by definition have unit norm. They can be composed (noncommutatively) as

$$\mathbf{q}_{\text{total}} = \mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{bmatrix} s_1 s_2 - \mathbf{v}_1^T \mathbf{v}_2 \\ s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} q_{10}q_{20} - q_{11}q_{21} - q_{12}q_{22} - q_{13}q_{23} \\ q_{10}q_{21} + q_{11}q_{20} + q_{12}q_{23} - q_{13}q_{22} \\ q_{10}q_{22} - q_{11}q_{23} + q_{12}q_{20} + q_{13}q_{21} \\ q_{10}q_{23} + q_{11}q_{22} - q_{12}q_{21} + q_{13}q_{20} \end{bmatrix} \tag{81}$$

which can be thought of as applying $\mathbf{q}_2$, then applying $\mathbf{q}_1$. It is important to keep in mind this operation is in general *not* commutative. Vectors can be rotated (i.e. body to inertial) via the following operation.

$$\mathbf{v}^N = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v}^B \end{bmatrix} \otimes \mathbf{q}^* \tag{82}$$

where $\mathbf{q}^*$ defines the quaternion conjugate

$$\mathbf{q}^* = \begin{bmatrix} s \\ -\mathbf{v} \end{bmatrix} \tag{83}$$

A.2 Deriving Attitude Dynamics

Recall the attitude quaternion can be written as

$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \mathbf{u}\sin(\theta/2) \end{bmatrix} \tag{84}$$

Take the time derivative via chain rule.

$$\frac{d\mathbf{q}}{dt} = \frac{d\mathbf{q}}{d\theta}\frac{d\theta}{dt} = \begin{bmatrix} -\frac{1}{2}\sin(\frac{\theta}{2})\dot{\theta} \\ \frac{1}{2}\mathbf{u}\cos(\frac{\theta}{2})\dot{\theta} \end{bmatrix} \tag{85}$$

We can write the instantaneous angular velocity as

$$\boldsymbol{\omega} = \dot{\theta}\mathbf{u} \tag{86}$$

I know that this is kind of suspicious, but I think the idea is that we can write this instantaneously without committing ourselves to a pure spin. From this, we also have

$$\boldsymbol{\omega}^T\mathbf{u} = \dot{\theta}\mathbf{u}^T\mathbf{u} = \dot{\theta} \tag{87}$$

Substitute (86) and (87) into (85) to get

$$\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} -\boldsymbol{\omega}^T\mathbf{u}\sin(\frac{\theta}{2}) \\ \boldsymbol{\omega}\cos(\frac{\theta}{2}) \end{bmatrix} \tag{88}$$

Now let's examine the expression presented previously.

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \tag{89}$$

where

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{90}$$

Before proceeding, let's introduce something called the *hat map*. Let's write the cross product of two vectors in component form.

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix} \tag{91}$$

Now, we can pull out all u coefficients into a 3x3 matrix.

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{92}$$

We define this matrix as the hat map $\hat{u}$ of $\mathbf{u}$.

$$\hat{u} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{93}$$

Such that

$$\mathbf{u} \times \mathbf{v} = \hat{u}\mathbf{v} \tag{94}$$

Now returning to quaternion dynamics, notice that $\Omega(\boldsymbol{\omega})$ can be written in block matrix form as

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\hat{\omega} \end{bmatrix} \tag{95}$$

Equation 89 becomes

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\hat{\omega} \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ \mathbf{u}\sin(\theta/2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\boldsymbol{\omega}^T \mathbf{u} \sin(\frac{\theta}{2}) \\ \boldsymbol{\omega}\cos(\frac{\theta}{2}) - \hat{\omega}\mathbf{u}\sin(\frac{\theta}{2}) \end{bmatrix} \tag{96}$$

Recall we defined

$$\boldsymbol{\omega} = \dot{\theta}\mathbf{u} \tag{97}$$

so

$$\boldsymbol{\omega} \times \mathbf{u} = \hat{\omega}\mathbf{u} = 0 \tag{98}$$

Thus, Equation 96 becomes

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} -\boldsymbol{\omega}^T \mathbf{u} \sin(\frac{\theta}{2}) \\ \boldsymbol{\omega}\cos(\frac{\theta}{2}) \end{bmatrix} \tag{99}$$

which is the same expression we obtained in Equation 88. Thus, the expression we use to propagate the dynamics is the same as the result we obtain from directly differentiating $\mathbf{q}$.

A.3 More Quaternion Background

A.3.1 Left and Right Multiply Matrices

Let's return to our definition of quaternion composition

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{bmatrix} s_1 s_2 - \mathbf{v}_1^T \mathbf{v}_2 \\ s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 \end{bmatrix} \tag{100}$$

and take out components of each quaternion, exploiting the hat map to conveniently express the cross product. This yields

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{bmatrix} s_1 & -\mathbf{v}_1^T \\ \mathbf{v}_1 & s_1 I + \hat{v}_1 \end{bmatrix} \begin{bmatrix} s_2 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} s_2 & -\mathbf{v}_2^T \\ \mathbf{v}_2 & s_2 I - \hat{v}_2 \end{bmatrix} \begin{bmatrix} s_1 \\ \mathbf{v}_1 \end{bmatrix} \tag{101}$$

which will let us define our left and right multiply matrices as

$$L(\mathbf{q}) = \begin{bmatrix} s & -\mathbf{v}^T \\ \mathbf{v} & sI + \hat{v} \end{bmatrix}, \qquad R(\mathbf{q}) = \begin{bmatrix} s & -\mathbf{v}^T \\ \mathbf{v} & sI - \hat{v} \end{bmatrix} \tag{102}$$

so that

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = L(\mathbf{q}_1)\mathbf{q}_2 = R(\mathbf{q}_2)\mathbf{q}_1 \tag{103}$$

Note that

$$L(\mathbf{q}^*) = L^T(\mathbf{q}), \qquad R(\mathbf{q}^*) = R^T(\mathbf{q}) \tag{104}$$

Also, we can define the "conjugate matrix" as

$$T = \begin{bmatrix} 1 & 0 \\ 0 & -I_{3x3} \end{bmatrix} \tag{105}$$

such that

$$\mathbf{q}^* = T\mathbf{q} \tag{106}$$

A.3.2 Rotation Matrices

Let's return to rotating a vector. We can write the Equation 82 more carefully as

$$\begin{bmatrix} 0 \\ \mathbf{v}^N \end{bmatrix} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v}^B \end{bmatrix} \otimes \mathbf{q}^* \tag{107}$$

Define the H matrix as the 4x3 matrix that transforms a vector into a pure vector quaternion.

$$H = \begin{bmatrix} 0 \\ I_{3x3} \end{bmatrix} \tag{108}$$

Thus, our rotation equation can be written as

$$H\mathbf{v}^N = \mathbf{q} \otimes H\mathbf{v}^B \otimes \mathbf{q}^* \tag{109}$$

$$= L(\mathbf{q})R(\mathbf{q}^*)H\mathbf{v}^B \tag{110}$$

$$\longrightarrow \mathbf{v}^N = H^T L(\mathbf{q})R^T(\mathbf{q})H\mathbf{v}^B \tag{111}$$

so we can write the rotation matrix that corresponds to the quaternion as

$$Q(\mathbf{q}) = H^T L(\mathbf{q})R^T(\mathbf{q})H \tag{112}$$

which defines the body-to-inertial rotation matrix that satisfies

$$\mathbf{v}^N = Q(\mathbf{q})\mathbf{v}^B \tag{113}$$

A.3.3 Attitude Jacobian

Recall the quaternion dynamics

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \tag{114}$$

I won't repeat the derivation here, but turns out this can be refactored as

$$\dot{\mathbf{q}} = \frac{1}{2}L(\mathbf{q})H\boldsymbol{\omega} \tag{115}$$

and we can define the "attitude Jacobian" as

$$G(\mathbf{q}) = L(\mathbf{q})H \tag{116}$$

so that the dynamics become

$$\dot{\mathbf{q}} = \frac{1}{2}G(\mathbf{q})\boldsymbol{\omega} \tag{117}$$