State Estimation for High Powered Rockets using an Unscented Kalman Filter

Govind Chari[*]
Cornell University, Ithaca, NY, 14853

In this paper I will present my physics simulation for a high powered rocket as well as my procedure for position, velocity, and attitude estimation using Unscented Kalman Filter and a typical avionics suite of an accelerometer, gyroscope, and GPS. This paper is written as the final report for MAE 6760: Model-Based Estimation.

I. Nomenclature

$\boldsymbol{r}$	=	position vector of rocket(m)
$\boldsymbol{v}$	=	velocity vector of rocket (m/s)
$\boldsymbol{q}$	=	orientation quaternion of rocket
$\boldsymbol{\omega}$	=	angular velocity vector of rocket(rad/s)
$\boldsymbol{\omega_{gyro}}$	=	gyroscope measurement (rad/s)
$\boldsymbol{a_{accel}}$	=	accelerometer measurement(m/s^2)
$\boldsymbol{r_{gps}}$	=	gps measurement(m)
$\boldsymbol{g}$	=	gravitational acceleration vector (m/s^2)
ρ	=	air density (kg/m^3)
C_L	=	lift coefficient
C_D	=	drag coefficient
α	=	angle of attack
A_{side}	=	cross sectional area of rocket when looking from the side(m^2)
A_{top}	=	cross sectional area of rocket when looking top down(m^2)
m	=	vehicle mass(kg)
$\mathbb{I}$	=	inertia tensor $(kg \cdot m^2)$
$\|\boldsymbol{x}\|$	=	ℓ_2 norm of $\boldsymbol{x}$
$\hat{\boldsymbol{x}}$	=	unit vector of $\boldsymbol{x}$
$\times$	=	vector cross product
$\boldsymbol{q}^*$	=	quaternion conjugate of $\boldsymbol{q}$
$\odot$	=	quaternion product

II. Introduction

High powered rocketry is a hobby for many people from amateurs doing it as a hobby to collegiate teams doing it competitively. In either case, collecting data and being able to reconstruct the trajectory is valuable in the iteration and redesign of the rocket or useful to determine the apogee altitude. Online state estimation is not necessary since these rockets are not actively controlled, so the state estimation can be done to post-process sensor logs offline.

In practice, there are many challenges the most prevalent of which is the so-called CoCom limits placed on consumer GPS. These limits prevent the GPS from reporting position when it is travelling faster than 1,200 mph or it is at an altitude greater than 60,000 ft. These restrictions are in place to prevent the use of commercial GPS on intercontinental ballistic missiles.

In applications where CoCom limits are exceeded, position estimation is very difficult. The most notable example in high powered rocketry is the University of Southern California's Rocket Propulsion Lab. This rocketry team successfully launched Traveller IV to space which is around 330,000ft. This rocket also hit peak speeds of around 3400 mph [1]. In these instances, analysts are forced to double integrate accelerometer readings resulting in large error bounds.

[*]Undergraduate Student, Mechanical and Aerospace Engineering

However, in this paper we assume that the rocket is operating below the CoCom limits, which is a good assumption for the rocket that the Cornell Rocketry Team (CRT) builds which has a target apogee of 10,000 ft and a peak speed below mach 1 (770 mph). In these instances, we can use GPS readings for trajectory reconstruction.

III. Simulation

A. Dynamics Model

Before we design the estimator, we need actual sensor readings or simulated sensor readings to run the estimator on. I do not have any actual flight data, so I had to write a simulation. This comes with the added advantage that I have truth readings so the performance of the estimator can be readily assessed.

This simulation is a 6DOF simulation that accounts for aerodynamic lift, drag as well as wind. However, there are a number of assumptions I made due to time constraints, however I will argue later in the estimator section that the assumptions do not affect the estimation scheme. I assume that the air density is constant through the flight, the rocket's mass stays constant, the center of gravity stays constant, the center of pressure stays constant, the lift and drag coefficients are independent of angle of attack, the motor's thrust is constant, and the gravitational field is uniform. Variable air density can be added in by using the exponential atmospheric model. In practice the mass of the rocket will decrease as the propellant is burned. A model can be made for this mass depletion, but that is a topic for another day. Similarly the changing center of gravity can be modeled once the mass depletion model is built. In reality, the center of pressure is a function of both the mach number and the angle of attack. Similarly the lift and drag coefficients are functions of both the mach number and angle of attack. These coefficients can be estimated from computational fluid dynamic simulations. The motor's exact thrust curve can also be put in simulation fairly easily as a .csv lookup table, but I modeled the thrust curve as a square wave for simplicity.

The dynamics can be split into two main parts: translational dynamics and attitude dynamics. We also have two reference frame: a North, East, Down inertial frame and a body frame that is aligned with the inertial frame before launch (the $\hat{b}$ vector in figure 1 is aligned with the down unit vector). The quaternion $\boldsymbol{q}$ will allow us to convert between these two frames. The equations of motion can be derived by looking at the following free body diagram. The FBD is drawn in two dimensions, but the same concepts apply to the three dimensional problem. The simulation is implemented in three dimensions. For reference, the center of pressure is the point on the vehicle where all aerodynamic forces can be modeled to act upon. This is similar to how the center of gravity is the point on the vehicle where we can model all inertial forces to act at.

From vector addition, we can firstly see that

$$\mathbf{v_{ra}} = \mathbf{v} - \mathbf{v_w} \tag{1}$$

we can now write out the translational dynamics equations of motion using Newton's second law

$$m\ddot{\boldsymbol{r}} = \boldsymbol{F_G} + \boldsymbol{F_T} + \boldsymbol{F_L} + \boldsymbol{F_D} \tag{2}$$

For attitude dynamics, we will consider applied moments in the body frame, which is centered at the vehicle's center of mass and whose axes are the principle axes of the vehicle. Considering moments in the body frame makes it easier to turn the aerodynamic forces into applied moments.

$$\mathbb{I}\dot{\omega} + \omega \times (\mathbb{I}\omega) = \boldsymbol{r_{gp}} \times (\boldsymbol{F_D} + \boldsymbol{F_L}) \tag{3}$$

Replacing F_G, F_L and F_D with their full representation we can derive our final equations of motion:

$$m\ddot{\boldsymbol{r}} = mg\hat{\boldsymbol{k}} - F_T\hat{\boldsymbol{b}} - \frac{1}{2}\rho C_D A_{proj}\|\boldsymbol{v}_{ra}\|^2\hat{\boldsymbol{v}}_{ra} - \frac{1}{2}\rho C_L A_{proj}\|\boldsymbol{v}_{ra}\|^2((\hat{\boldsymbol{v}}_{ra} \times \hat{\boldsymbol{b}}) \times \hat{\boldsymbol{v}}_{ra}) \tag{4}$$

$$\mathbb{I}\dot{\omega} + \omega \times (\mathbb{I}\omega) = -\boldsymbol{r_{gp}} \times \left(\frac{1}{2}\rho C_D A_{proj}\|\boldsymbol{v}_{ra}\|^2\hat{\boldsymbol{v}}_{ra} + \frac{1}{2}\rho C_L A_{proj}\|\boldsymbol{v}_{ra}\|^2((\hat{\boldsymbol{v}}_{ra} \times \hat{\boldsymbol{b}}) \times \hat{\boldsymbol{v}}_{ra})\right) \tag{5}$$

we also have an approximation to the projected area of the rocket onto the vector $\boldsymbol{v}_{ra}$

$$A_{proj} = A_{top}cos(\alpha) + A_{side}sin(\alpha) \tag{6}$$

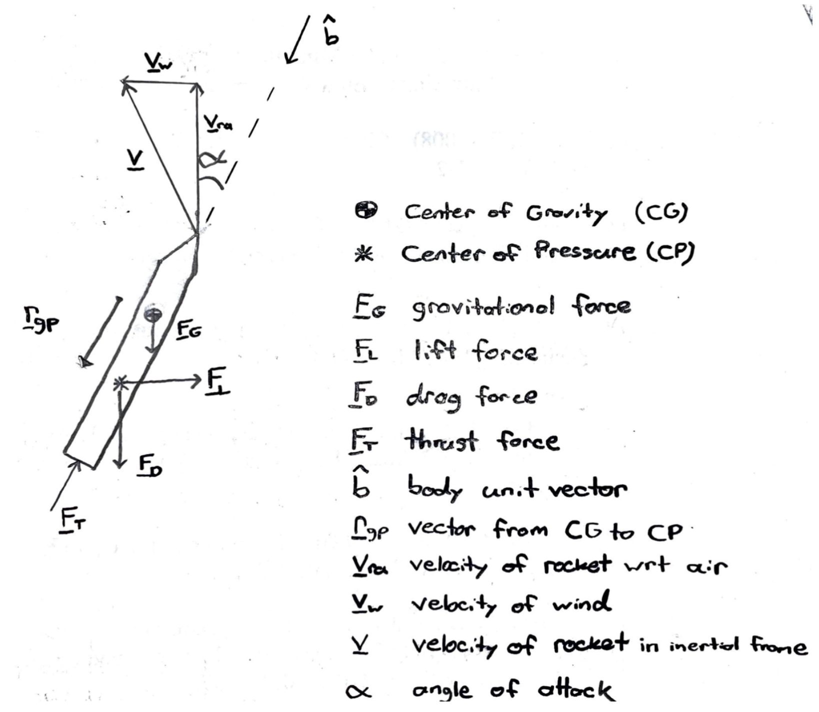

$\hat{b}$
$\underline{V}_w$
$\underline{V}_{ra}$
$\underline{V}$
α
$\underline{r}_{gp}$
$\underline{F}_G$
$\underline{F}_L$
$\underline{F}_D$
$\underline{F}_T$
Center of Gravity (CG)
Center of Pressure (CP)
$\underline{F}_G$ gravitational force
$\underline{F}_L$ lift force
$\underline{F}_D$ drag force
$\underline{F}_T$ thrust force
$\hat{b}$ body unit vector
$\underline{r}_{gp}$ vector from CG to CP
$\underline{V}_{ra}$ velocity of rocket wrt air
$\underline{V}_w$ velocity of wind
$\underline{V}$ velocity of rocket in inertial frame
α angle of attack

Fig. 1 Free Body Diagram

It is also worth noting that for an aerodynamically stable rocket, r_{gp} points along the $\hat{\boldsymbol{b}}$ vector as opposed to an aerodynamically unstable rocket which points along the $-\hat{\boldsymbol{b}}$ vector.

Since we are also using quaternions as the attitude convention, we must also present the quaternion kinematics equation, which is given below.

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \omega \end{bmatrix} \tag{7}$$

These equations of motion were programmed into Matlab for simulation. In this simulation, the rocket is simulated until apogee. Additionally, there are two phases of flight: boost and glide. The boost phase is the phase of flight where the motor is firing and after the propellant is exhausted, the glide phase starts which then ends at apogee.

B. Sensor Model

For this project, we assume the standard avionics suite of an accelerometer, gyroscope, and GPS. We assume that we have unbiased, white, Gaussian noise. In reality accelerometers and gyroscopes have biases, however these biases can be characterized while the rocket is on the pad using some type of batch estimation and can then be subtracted off the sensor readings in flight resulting in unbiased readings. This is also fine to do since the flight lasts 15 seconds and the biases will not change significantly over this timeframe. It is also assumed that the accelerometer and gyroscope are placed at the vehicle's center of gravity. If this is not the case, then the sensor frame would differ from the body frame and a transformation must be applied. We further assume that the rocket is a rigid body. If it were not a rigid body, the accelerometer and gyroscope would read some pseudo acceleration and angular rates corresponding to the rocket bending in flight.

We assume that we have an accelerometer and gyroscope that can sample at 100Hz and a GPS that can sample at 10Hz, which are reasonable rates for off the shelf sensors. The acceleration and angular rates are also reported in the body frame. Sensor readings are generated by sampling the truth readings at their respective sample rates, converting into the body frame if necessary, then adding Gaussian noise.

IV. Estimator

In this section, we will focus on how to estimate the rocket's position, velocity, and attitude from the gyroscope, accelerometer, and GPS readings. Since we have direct access to vehicle acceleration and rates we do not need to have a very good idea of the specifics of the vehicle dynamics. For example, we do not need to know the aerodynamic coefficients or center of gravity location. The acceleration of the vehicle is a combination of all of these exact parameters, so if we know the acceleration we can ignore all these parameters.

The general structure of the estimator will be first rotating the accelerometer reading about the estimated attitude quaternion to get an estimate of the acceleration in the global frame. Then propagate the position and velocity using constant acceleration kinematics. We will also propagate the quaternion according using the gyroscope measurement. We will then perform the update step using the GPS reading. In this framework, the accelerometer and gyroscope readings are treated as control inputs and the GPS reading is the only measurement. The state and the state transition equations are as follows:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \\ \boldsymbol{q} \end{bmatrix} \tag{8}$$

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{9}$$

$$\dot{\boldsymbol{v}} = \Im\mathfrak{m}\left(\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \boldsymbol{a}_{accel} \end{bmatrix} \odot \boldsymbol{q}^*\right) \tag{10}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \omega_{gyro} \end{bmatrix} \tag{11}$$

The measurement equation is then

$$z = H\boldsymbol{x} \tag{12}$$

where

$$H = \begin{bmatrix} I_{3\times3} & 0_{3\times7} \end{bmatrix} \tag{13}$$

It can easily be seen that these state transition equations are highly nonlinear as there is a quaternion product between the control input and one of the states (the attitude quaternion) for the velocity and attitude quaternion transition equations. Thus we must turn to either an Extended Kalman Filter (EKF) or an Unscented Kalman Filter (UKF). For this, we will use a UKF which comes with the added benefit of not having to take any Jacobians. This UKF is implemented in Matlab, so I used the ode45 function to do the propagation of the sigma points through the state transition equation which is another benefit over the EKF. With the EKF, the state transition equations would have to be discretized first, so with a UKF, this can be avoided.

As a brief explanation, a UKF has the same predict-update structure as a regular Kalman Filter, but it is adapted to work for nonlinear dynamics. With a regular Kalman Filter, the Gaussian prior distribution can be propagated through the linear state transition equation which results in another Gaussian, but when we have nonlinear dynamics the Gaussian prior becomes non-Gaussian after being propagated. An EKF solves this issue by linearizing the state transition equation about the previous estimate thus allowing the Gaussian prior to stay Gaussian after propagation. The UKF on the other hand intelligently selects a few key points from the prior distribution (called sigma points), propagates them through the nonlinear dynamics and then fits a Gaussian to these propagated sigma points.

Since we are getting gyroscope and accelerometer data at 100 Hz and GPS data at 10 Hz only the predict step is run until we have a GPS reading. The results of the simulation and the UKF performance is given in the next section.

V. Results

The simulation was run with parameters as close to those of CRT's rocket as possible. The simulated rocket is 100 pounds, 12 feet long, 6 inches in diameter, a center of gravity to center of pressure offset of 12 inches, a drag and lift coefficient of 0.5, a thrust of 790 lbf and a burn duration of 5.3 seconds. The lift and drag coefficients were guesses and the thrust of 780 lbf is the average thrust of the designed CRT motor and the burn time was chosen to result in an impulse of 18500 Ns. In this simulation the wind is blowing 5 m/s to the North and 8 m/s to the east. This is a very strong 20 mph wind, which is at the upper limits for a launch.

The sensor noises were chosen to match those of the Adafruit BNO055, so the accelerometer has a standard deviation of 0.1 m/s^2 and the gyroscope has a standard deviation of 4 m which is typical for civilian GPS.

The UKF performance and the truth plots are in the appendix. From the UKF plots we can make a number of observations. Firstly, the filter appears to be consistent the whole time. In this context, filter consistency refers to the truth value of a certain state being within 2σ of the estimated value. This can be assessed by drawing a horizontal line at zero in an error plot and seeing if this line almost always remains within the 2σ bounds. If a filter is consistent, it means that you would be able to trust the confidence level of the filter as indicated by its estimated covariance matrix.

Secondly, the state estimate errors and 2σ bounds appear to look squarewave-esque. This behavior is due to the predict step running ten times for every update step that is run. When the update step is run, the state is sharply changed.

Finally, we can see that the error bounds on the velocity estimates are much larger than the error bounds on the position estimates. This makes sense since we have direct measurements of position so we can be more confident in our position estimates, whereas our velocity estimates are more indirect and thus we are less confident in our estimates.

It is also interesting to think about how GPS readings affect attitude estimates. The two are coupled via the velocity transition equations. The accelerometer readings are first rotated about the attitude quaternion then double integrated to get position readings. Thus a position measurement gives some information about the attitude of the rocket.

In the simulation, the center of gravity is ahead of the center of pressure making the rocket statically stable and we can see that from the angular rate truth plot. The rates are oscillating and damping out until the rocket loses too much speed after which the rocket moves too slow to have a significant restoring moment due to aerodynamics forces.

VI. Future Work

One problem with using quaternions in the UKF is that to represent attitude using quaternions, the quaternions must be of unit length. After the predict step, if a small enough integration step is taken and a high order integrator is used, this is not a real concern as the propagated quaternion will also be of unit length, however in the update step we take the sum of this unit length quaternion and a correction due to the innovation. This can lead to the quaternion becoming quickly unnormalized. In my simulation, the norm of the estimated quaternion went up to 1.003 which is a significant amount of error. In my estimator, I normalized the quaternions after each update step. With this normalization, the largest attitude determination error that was seen was around 1 degree, but if this process is done over long enough timespans, large attitude determination error can result.

Quaternions being overdetermined makes it difficult to use them in a Kalman Filter, so if I redid the project, I would use a minimum attitude formalism like Euler Angles or Modified Rodrigues Parameters.

VII. Conclusion

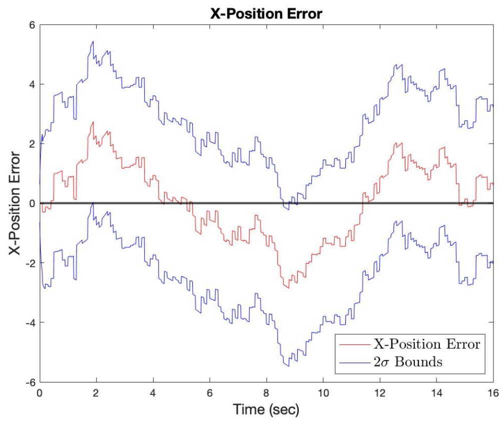
In this project I developed a simple simulation for high powered rockets and a UKF to estimate position, velocity, and attitude using accelerometer, gyroscope, and GPS readings. After working on this project I have a more intuitive understanding of how a UKF works and I grew more confident in my ability to construct simulations from first principles, as this was the first time I implemented an aerodynamics model in simulation.

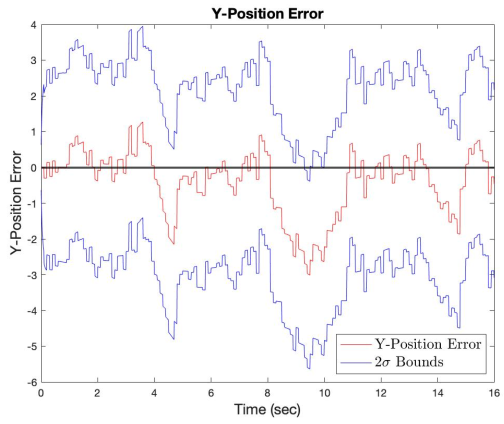
The codebase for this project can be found at `https://github.com/govindchari/Rocket-UKF`.

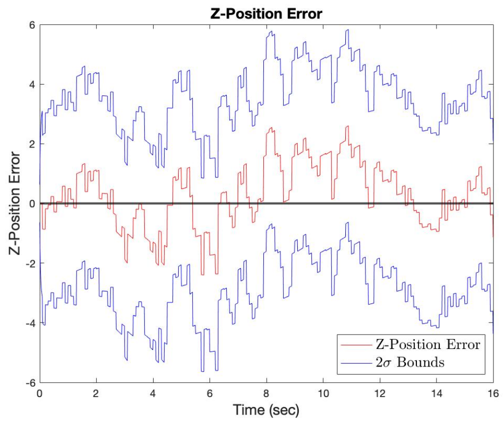
References

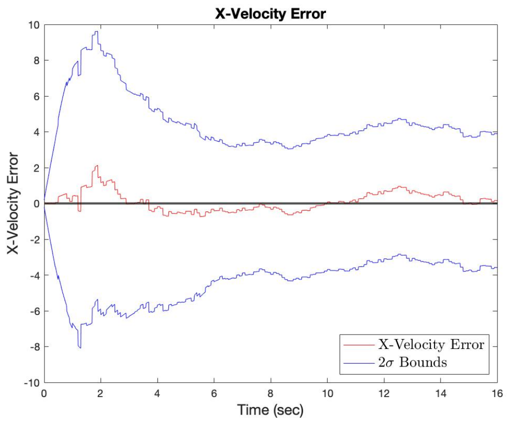
[1] Aitoumeziane, A., Eusebio, P., Hayes, C., Ramachandran, V., Smith, J., Sridharan, J., St. Regis, L., Stephenson, M., Tewksbury, N., Tran, M., and Yang, H., "Traveler IV Apogee Analysis," 2019.

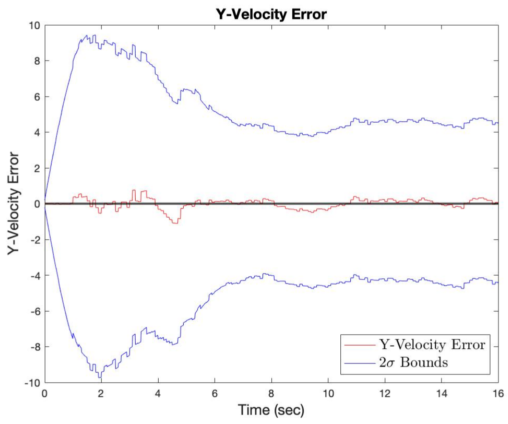
Appendix

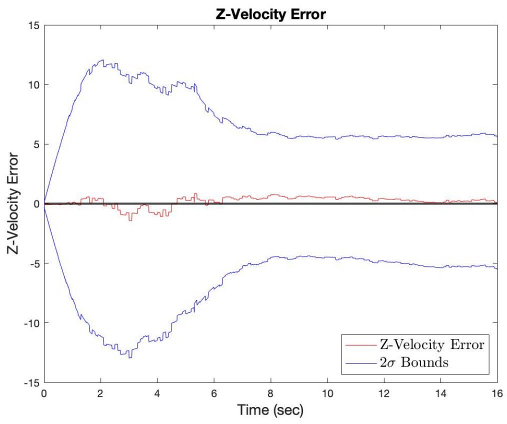

X-Position Error
X-Position Error
Time (sec)
X-Position Error
2σ Bounds

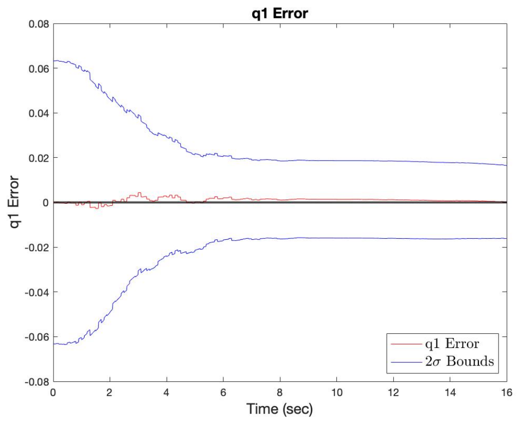

Y-Position Error
Y-Position Error
Time (sec)
Y-Position Error
2σ Bounds

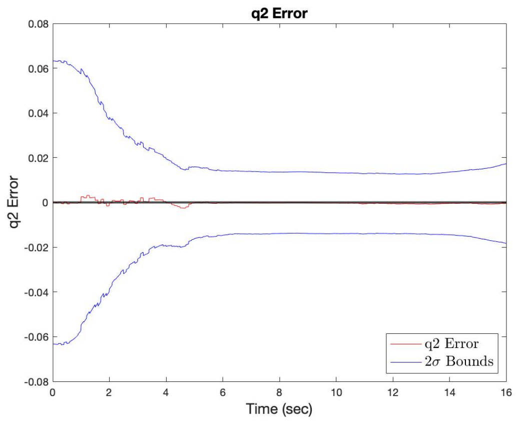

Z-Position Error
Z-Position Error
Time (sec)
Z-Position Error
2σ Bounds

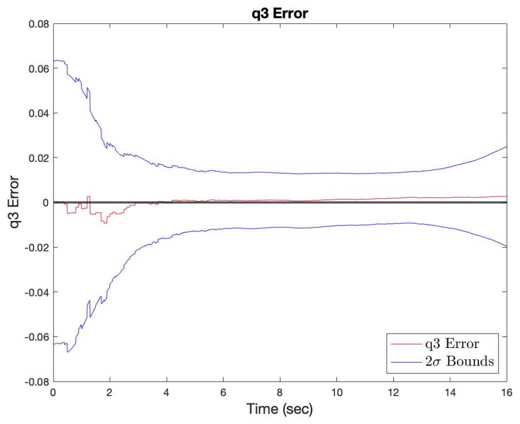

X-Velocity Error
X-Velocity Error
Time (sec)
X-Velocity Error
2σ Bounds

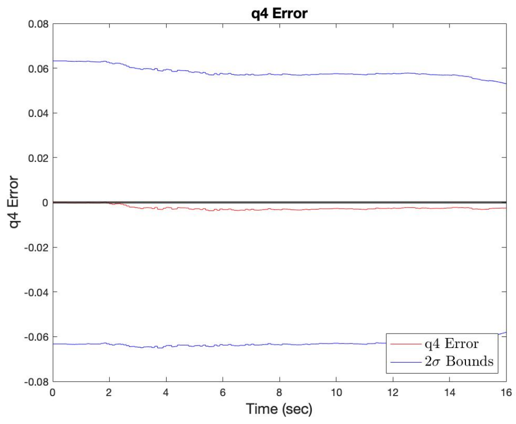

Y-Velocity Error
Y-Velocity Error
Time (sec)
Y-Velocity Error
2σ Bounds

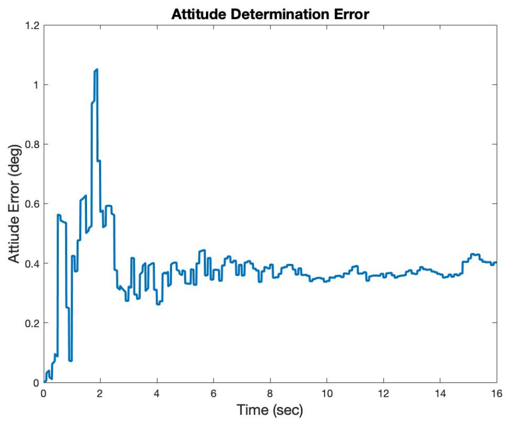

Z-Velocity Error
Z-Velocity Error
Time (sec)
Z-Velocity Error
2σ Bounds

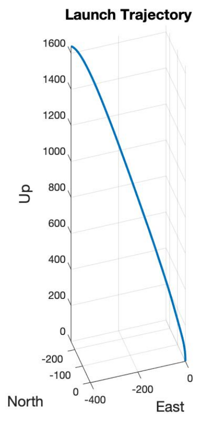

q1 Error
q1 Error
2σ Bounds
Time (sec)

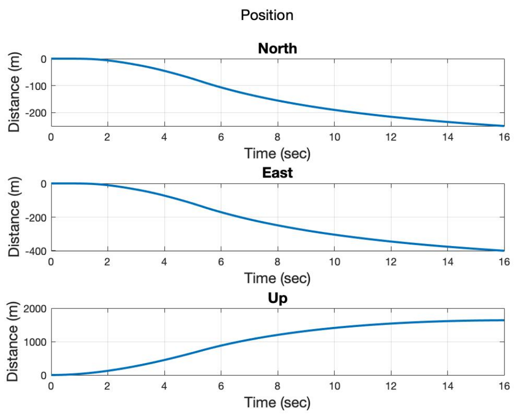

q2 Error
q2 Error
2σ Bounds
Time (sec)

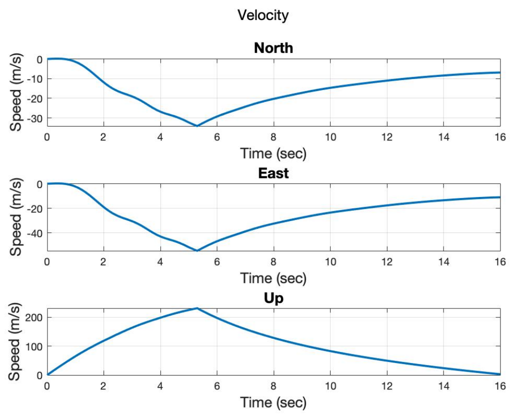

q3 Error
q3 Error
2σ Bounds
Time (sec)

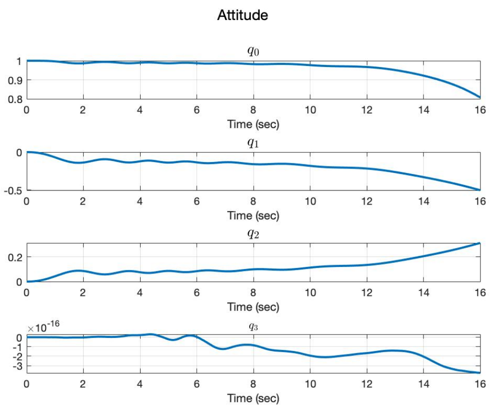

q4 Error
q4 Error
2σ Bounds
Time (sec)

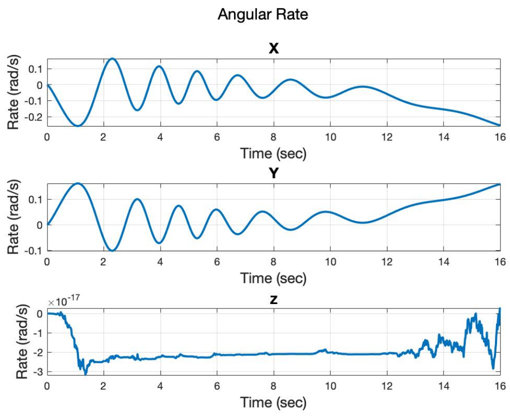

Attitude Determination Error
Attiude Error (deg)
Time (sec)


Launch Trajectory
Up
North
East


Position
North
East
Up
Distance (m)
Time (sec)


Velocity
North
East
Up
Speed (m/s)
Time (sec)


Attitude
q_0
q_1
q_2
q_3
$\times 10^{-16}$
Time (sec)


Angular Rate
X
Y
Z
$\times 10^{-17}$
Rate (rad/s)
Time (sec)