Lunar Lander Documentation

Porter Burton

I. State Definition

Truth State

The truth state for the navigation system fully describes the dynamics and processes of the lunar lander. The truth state variables are carefully selected to capture the scope of this design which focuses on the use of an optical camera and radar altimeter. There are four important coordinate systems for this simulation: an inertial frame whose origin is at the moons center of mass, a moon frame which rotates with the moon and whose origin is coincident with the inertial frame, a body frame fixed to the lander, and a camera frame which represents the rotation to the on board camera. The inertial and fixed Lunar frames are illustrated in fig(1) where the principle axis of the inertial Lunar frame points towards the 1st point of Ares. The fixed, or moon, frame rotates with the moon

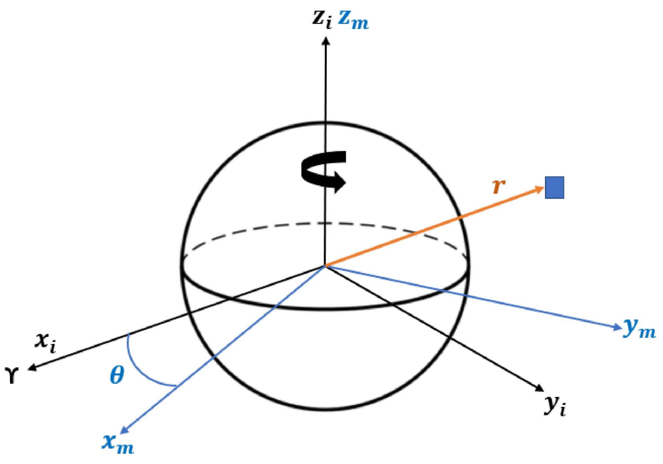


Figure 1. Inertial and Fixed Lunar Coordinate Frames

The body and camera frames are illustrated in fig(2). In the body frame, the z axis is aligned with the thrust direction and in the camera frame the z axis is aligned with the camera's line of sight.

The truth state vector is given by (1). Where $\boldsymbol{r}^i_{b/i}$ is the position of the lander in the inertial frame and $\boldsymbol{v}^i_{b/i}$ is the velocity. q^m_i is a quaternion which represents the attitude rotation of the lander in the moon frame.q^c_b is the rotation to the camera frame from the body frame. b_r is the bias in range, $\boldsymbol{\epsilon}^i_g$ is the bias in gravity, and $\boldsymbol{b}_{accel}$ is the bias in the accelerometer.

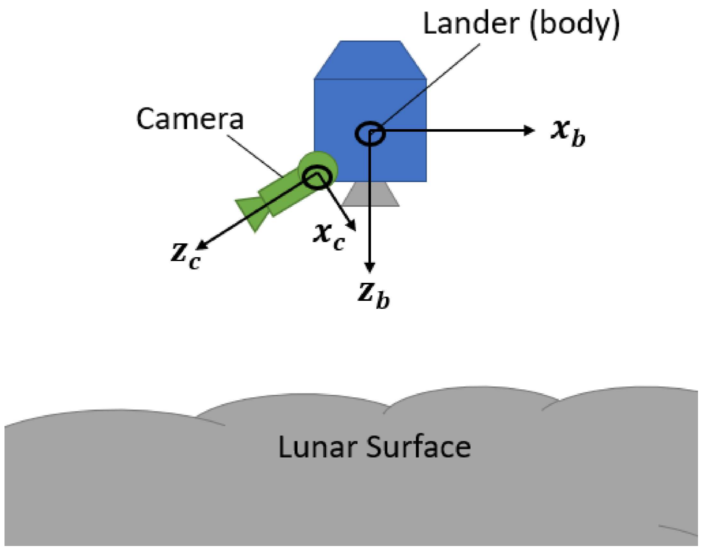


Figure 2. Body and Camera Coordinate Frames

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{r}^i_{b/i} \\ \boldsymbol{v}^i_{b/i} \\ \boldsymbol{q}^m_i \\ \boldsymbol{q}^c_b \\ \boldsymbol{b}_r \\ \boldsymbol{\epsilon}^i_g \\ \boldsymbol{h}_t \\ \boldsymbol{b}_{accel} \end{bmatrix} \tag{1}$$

The first three states of the truth state vector, position, velocity, and orientation, will be the vehicle states. Note, angular rate is not included because we assume there will only by small and slow changes in attitude. Also gyro bias is not included to further simply the problem. The remaining states are parameter states. Including the quaternion and vector components, the truth state contains 25 elements.

Design, Navigation, and Error States

The design state vector describes the state the we will ask the Kalman filter to estimate. We will remove the attitude of the vehicle, and the rotation to the camera because these states can be determined outside of the navigation system. Additionally, removing the quaternion quantities greatly simplifies the mathematics when computing the error state. We also remove the position to a feature of interest because we can calculate this value accurately using the position of the lander and the known position vector of that feature. After removing these quantities, the design state vector has 14 individual elements.

A summery of the design states is given by (2) using the same notation as before.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}^i_{b/i} \\ \boldsymbol{v}^i_{b/i} \\ \boldsymbol{b_r} \\ \boldsymbol{\epsilon}^i_g \\ \boldsymbol{h_t} \\ \boldsymbol{b_{accel}} \end{bmatrix} \tag{2}$$

The navigation state vector has the same elements as the design state. Errors will be injected into this state using the calculated errors. Equation (3) summarizes this state.

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{r}^i_{b/i}} \\ \hat{\boldsymbol{v}^i_{b/i}} \\ \hat{\boldsymbol{b_r}} \\ \hat{\boldsymbol{\epsilon}^i_g} \\ \hat{\boldsymbol{h_t}} \\ \hat{\boldsymbol{b_{accel}}} \end{bmatrix} \tag{3}$$

The error state vector will be used to store calculated errors in our system. Again, the error state has the same elements as the design state.

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta r}^i_{b/i} \\ \boldsymbol{\delta v}^i_{b/i} \\ \boldsymbol{\delta b_r} \\ \boldsymbol{\delta \epsilon}^i_g \\ \boldsymbol{\delta h_t} \\ \boldsymbol{\delta b_{accel}} \end{bmatrix} \tag{4}$$

II. State Mapping

Our system will require an efficient method to map between the truth, design, navigation, and error state. These methods are defined and implemented in the simulation. The corrected state estimates $\hat{\boldsymbol{x}}_c$ are given by (5).

$$\hat{\boldsymbol{x}_c} = c(\hat{\boldsymbol{x}}, \boldsymbol{\delta x}) = \boldsymbol{\delta x} + \hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{r}^i_{b/i}} \\ \hat{\boldsymbol{v}^i_{b/i}} \\ \hat{b_r} \\ \hat{\boldsymbol{\epsilon}^i_g} \\ \hat{h_t} \\ \hat{\boldsymbol{b}_{accel}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{r}^i_{b/i} \\ \boldsymbol{v}^i_{b/i} \\ b_r \\ \boldsymbol{\epsilon}^i_g \\ h_t \\ \boldsymbol{b}_{accel} \end{bmatrix} \tag{5}$$

The insertion equation will insert errors into the navigation state and is given by (6).

$$\hat{\boldsymbol{x}} = i(\boldsymbol{x}, \boldsymbol{\delta x}) = \boldsymbol{x} - \boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{r}^i_{b/i} \\ \boldsymbol{v}^i_{b/i} \\ b_r \\ \boldsymbol{\epsilon}^i_g \\ h_t \\ \boldsymbol{b}_{accel} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\delta r}^i_{b/i} \\ \boldsymbol{\delta v}^i_{b/i} \\ \delta b_r \\ \boldsymbol{\delta \epsilon}^i_g \\ \delta h_t \\ \boldsymbol{\delta b}_{accel} \end{bmatrix} \tag{6}$$

Equation (7) computes the estimation errors. This function must be verified to ensure is produces the error state $\boldsymbol{\delta x}$ which was previously defined.

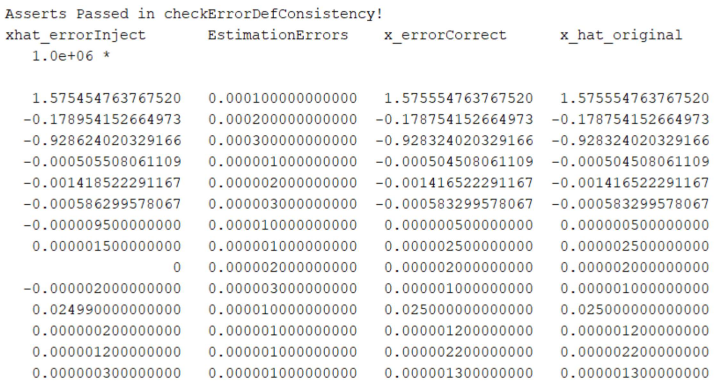
$$\boldsymbol{\delta x} = e(\hat{\boldsymbol{x}}, \boldsymbol{x}) = \boldsymbol{x} - \hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{r}^i_{b/i} \\ \boldsymbol{v}^i_{b/i} \\ b_r \\ \boldsymbol{\epsilon}^i_g \\ h_t \\ \boldsymbol{b}_{accel} \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{r}^i_{b/i}} \\ \hat{\boldsymbol{v}^i_{b/i}} \\ \hat{b_r} \\ \hat{\boldsymbol{\epsilon}^i_g} \\ \hat{h_t} \\ \hat{\boldsymbol{b}_{accel}} \end{bmatrix} \tag{7}$$

Finally, we need a mapping to extract the design states from the truth state vector. The design states are a subset of the truth states, so to map the design states a simple matrix multiplication is used. The mapping is given by (8) where **0** represents a matrix of zeros.

$$\boldsymbol{x} = m(\boldsymbol{x_t}) = \begin{bmatrix} I_{6x6} & 0 & 0 \\ 0 & 0_{8x8} & 0 \\ 0 & 0 & I_{8x8} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}^i_{b/i} \\ \boldsymbol{v}^i_{b/i} \\ q^m_i \\ q^c_b \\ b_r \\ \boldsymbol{\epsilon}^i_g \\ h_t \\ \boldsymbol{b}_{accel} \end{bmatrix} \tag{8}$$

Validation of Mappings

The mappings are verified using a Matlab simulation. To test the mappings, a truth state and navigation state were initialized using practical initial conditions. Then, errors are injected, calculated and corrected. The simulation showed that the original navigation state and the corrected state after error injection were the same within a tolerance of 10^{-12}. This proves the mapping functions are functional. Fig (3) shows transient values of these states after they we able to pass the tolerance asserts in checkErrorDefConsistency.m in the provided script. Is is seen, the corrected state matches the input state before errors were injected.



```
Asserts Passed in checkErrorDefConsistency!
xhat_errorInject      EstimationErrors    x_errorCorrect      x_hat_original
   1.0e+06 *

   1.575454763767520   0.000100000000000   1.575554763767520   1.575554763767520
  -0.178954152664973   0.000200000000000  -0.178754152664973  -0.178754152664973
  -0.928624020329166   0.000300000000000  -0.928324020329166  -0.928324020329166
  -0.000505508061109   0.000001000000000  -0.000504508061109  -0.000504508061109
  -0.001418522291167   0.000002000000000  -0.001416522291167  -0.001416522291167
  -0.000586299578067   0.000003000000000  -0.000583299578067  -0.000583299578067
  -0.000009500000000   0.000010000000000   0.000000500000000   0.000000500000000
   0.000001500000000   0.000001000000000   0.000002500000000   0.000002500000000
                   0   0.000002000000000   0.000002000000000   0.000002000000000
  -0.000002000000000   0.000003000000000   0.000001000000000   0.000001000000000
   0.024990000000000   0.000010000000000   0.025000000000000   0.025000000000000
   0.000000200000000   0.000001000000000   0.000001200000000   0.000001200000000
   0.000001200000000   0.000001000000000   0.000002200000000   0.000002200000000
   0.000000300000000   0.000001000000000   0.000001300000000   0.000001300000000
```

Figure 3. Error Injection, Calculation, and Correction Comparison

III. State Propagation

Truth State

We now must define the differential equations for the truth state so that the states can be propagated through time in the simulation. The first two states are position and velocity, whose equations are show by (9,10). The derivative of position is simply velocity, and the derivative of velocity is expressed using a sum of accelerations on the body. The thrust acceleration is given by a_{thr}, and accelerations from gravity are given by a_{grav} which is given by (11). A process noise w_a is also included in the sum of acceleration components. Thrust acceleration must be generated by a guidance algorithm and will be modeled in the truth state. A simple guidance algorithm is given by (12) which was used for the Apollo missions. Note, to determine a realistic landing trajectory the value of t_{go} must be chosen carefully, since this guidance law always produces a convex solution. The desired states, $\boldsymbol{r}_{t_f}$, $\boldsymbol{v}_{t_f}$, and $\boldsymbol{a}_{t_f}$ must be specified at the final time and control how and where the lander will touchdown.

$$\dot{\boldsymbol{r}}^i_{b/i} = \boldsymbol{v}^i_{b/i} \tag{9}$$

$$\dot{\boldsymbol{v}}^i_{b/i} = \boldsymbol{a}^i_{grav} + \boldsymbol{a}^i_{thr} + \boldsymbol{\epsilon}^i_g + \boldsymbol{w}^i_a \tag{10}$$

$$\boldsymbol{a}^i_{grav} = \frac{-\mu}{\left\|\boldsymbol{r}^i_{b/i}\right\|^3}\boldsymbol{r}^i_{b/i} \tag{11}$$

$$\boldsymbol{a}^i_{thr} = \boldsymbol{a}^i_{tf} + \frac{6}{t_{go}(\boldsymbol{v}^i_{tf} - \dot{\boldsymbol{v}}^i_{b/i})} + \frac{12}{t^2_{go}(\boldsymbol{r}^i_{tf} - \boldsymbol{r}^i_{b/i} - \boldsymbol{v}^i_{b/i}t_{go})} \tag{12}$$

Next we define the equations related to rotations, which are quaternion equations. We must include the angular velocity of the moon's rotation to compute the rate of change of the moon quaternion. We also assume the camera is fixed to the body so the rate of change of the camera quaternion is zero. Equations (13-15) summarize the propagation of these state using quaternion mathematics.

$$\dot{q}^m_i = \frac{`1}{2}\boldsymbol{\omega}^m_{m/i} \otimes q^m_i \tag{13}$$

$$\boldsymbol{\omega}^m_{m/i} = \begin{bmatrix} 0 \\ 0 \\ \Omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2.7x10^{-6} \end{bmatrix} \frac{rad}{s} \tag{14}$$

$$\dot{q}^c_b = 0 \tag{15}$$

The bias states are modeled as exponentially correlated random variables (ECRVs.) To propagate these states a time constant, τ, and process noise, w, is included. Using the ECRV model, these states are given by (16,17).

$$\dot{b}_r = -\frac{1}{\tau_r}b_r + w_r \tag{16}$$

$$\dot{b}_{accl} = -\frac{1}{\tau_{accl}}b_{accl} + w_{accl} \tag{17}$$

The last two states are the gravity error model and the height of the terrain. A perpendicular velocity term is included in both these equations in order to realize the correct units. This is the lateral velocity of the lander, or the velocity in the direction parallel to the Lunar surface. Each term also includes process noise and a distance correlation factor d. Equations (18-21) summarize the how these states will be modeled to propagate.

$$\dot{\boldsymbol{\epsilon}}^i_g = -\frac{\|\boldsymbol{v}_\perp\|}{d_g}\boldsymbol{\epsilon}^i_g + \boldsymbol{w}_g \tag{18}$$

$$\dot{h} = -\frac{\|\boldsymbol{v}_\perp\|}{d_h}h + w_h \tag{19}$$

$$\boldsymbol{v}_\perp = \boldsymbol{v}^i_{b/i} - \boldsymbol{\omega}^m_{m/i} \times \boldsymbol{r}^i_{b/i} - (\boldsymbol{v}^i_{b/i} \cdot \hat{i}_r)\hat{i}_r \tag{20}$$

$$\hat{i}_r = \frac{\boldsymbol{r}^i_{b/i}}{\left\|\boldsymbol{r}^i_{b/i}\right\|} \tag{21}$$

Measurement Models

The truth states are now fully defined, however we also need a truth sensor model. The sensors modeled are an accelerometer and an optical camera. It is now convenient to define an acceleration term for non-gravitational terms which will be called a.

$$\boldsymbol{a}^i = \boldsymbol{a}^i_{thr} + \boldsymbol{w}^i_a \tag{22}$$

The accelerometer is them modeled with a bias and noise term in (23). A transformation matrix, T^b_i, is needed to transform accelerations to the body frame where the sensor will be located.

$$\tilde{\boldsymbol{a}}^b = T^b_i\boldsymbol{a}^i + \boldsymbol{b}_a + \boldsymbol{n}_a \tag{23}$$

The camera measure model is more complex, Fig(4) shows vectors which help visualize how the model is constructed.

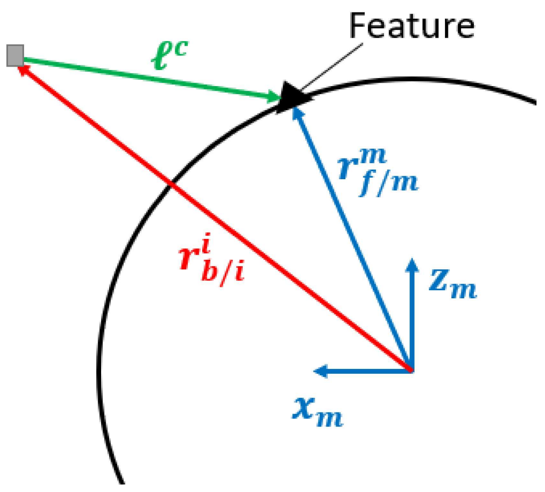


Figure 4. Line of Sight (LOS) vector ℓ_c

Now, the line of sight vector, ℓ, is given by vector algebra and coordinate system transformations in (24), where $\boldsymbol{r}^m_{f/m}$ is the position to a feature of interest as expressed in the moon frame.

$$\boldsymbol{\ell}^c = T_b^c T_i^b (T_m^i \boldsymbol{r}_{f/m}^m - \boldsymbol{r}_{b/i}^i) = \begin{bmatrix} \ell_x \\ \ell_y \\ \ell_z \end{bmatrix} \tag{24}$$

With the line of sight vector defined, the camera measurement model is given by (25) which includes a noise term v_c.

$$\tilde{\boldsymbol{z}} = \begin{bmatrix} \frac{\ell_x}{\ell_z} \\ \frac{\ell_y}{\ell_z} \end{bmatrix} + v_c \tag{25}$$

Design and Navigation State Model

Now that the truth model is defined we will define the design and navigation state equations. The design state model can easily be determined from the truth state model and is show by (26). the design model has the same states as the navigation states.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{r}}_{b/i}^i \\ \dot{\boldsymbol{v}}_{b/i}^i \\ \dot{b}_r \\ \dot{\boldsymbol{\epsilon}}_g^i \\ \dot{h} \\ \dot{b}_{accl} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{b/i}^i \\ \boldsymbol{a}^i + \boldsymbol{\epsilon}_g^i + \boldsymbol{a}_{grav} \\ -\frac{1}{\tau_r} b_r + w_r \\ -\frac{\|\boldsymbol{v}_\perp\|}{d_g} \boldsymbol{\epsilon}_g^i + \boldsymbol{w}_g \\ -\frac{\|\boldsymbol{v}_\perp\|}{d_h} h + w_h \\ -\frac{1}{\tau_{accl}} b_{accl} + w_{accl} \end{bmatrix} \tag{26}$$

For the navigation state, the accelerometer sensor data will be used to estimate non-gravitational accelerations. and noise terms are dropped from the truth state. A summary of the navigation state propagation is given by (27).

$$\dot{\hat{\boldsymbol{x}}} = \begin{bmatrix} \dot{\hat{\boldsymbol{r}}}_{b/i}^i \\ \dot{\hat{\boldsymbol{v}}}_{b/i}^i \\ \dot{\hat{b}}_r \\ \dot{\hat{\boldsymbol{\epsilon}}}_g^i \\ \dot{\hat{h}} \\ \dot{\hat{b}}_{accl} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{v}}_{b/i}^i \\ \tilde{\boldsymbol{a}}^i + \hat{\boldsymbol{\epsilon}}_g^i + \hat{\boldsymbol{a}}_{grav} \\ -\frac{1}{\tau_r} \hat{b}_r \\ -\frac{\|\boldsymbol{v}_\perp\|}{d_g} \hat{\boldsymbol{\epsilon}}_g^i \\ -\frac{\|\boldsymbol{v}_\perp\|}{d_h} \hat{h} \\ -\frac{1}{\tau_{accl}} \hat{b}_{accl} \end{bmatrix} \tag{27}$$

Note, the accelerometer measurements are found in the body frame so we must define (28) where T_b^i is a transformation matrix from the body to the inertial frame. In this simulation, we will assume the thruster will always be pointed in the direction of the inertial velocity and is aligned with one of the principle axis of the body. This assumption makes the transformation matrix T_i^b easy to calculate.

$$\tilde{\boldsymbol{a}}^i = T_b^i (\tilde{\boldsymbol{a}}^b - \boldsymbol{b}_{accl} - \boldsymbol{n}_{accl}) \tag{28}$$

Propagation in Simulation

Thus far, we have not included any linearization, covariance, or Kalman updates to our simulation. We simply, want to demonstrate the state propagation with synthesized measurements in the navigation state. With no process noise, we can model the truth state to be ideal, based on dynamic equations, and compare how the navigation state of the lander compares to the truth. We can also use the navigation state to predict nonlinear measurements and compare them to the synthesized measurements in the form of residuals. As mentioned before, a good guess for simulation time must be found for the guidance law to work effectively. This time changes based on how far the lander must travel to its landing site. For this lunar lander simulation, t_{go} is set to 550 s in order to generate a good trajectory with the sample initial conditions.

For simplicity, the trajectory for this lander simulation is held in the (x,y) inertial plane. The lander starts approximately 10 km above and 400 km behind the desired landing site at the final time. The lander has an initial velocity in the x direction and begins in a stable parking orbit. A feature is also observed 100 km behind the landing site to generate LOS measurements. The trajectory of the lander is shown in Fig(5).

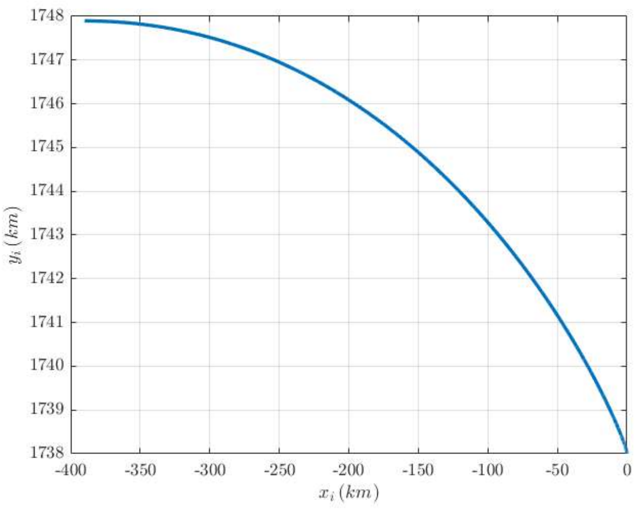


Figure 5. Lunar Lander Trajectory

The thrust accelerations, generated by the guidance law are shown in fig(6), note the sudden cutoff in thrust at the end of the simulation. This means to achieve the desired touchdown velocity and acceleration, the lander had to touch down while still thrusting. Also, fig(7) shows the velocity of the lander converging to the desired values.

We see, with no state correction the navigation state is able to predict the true states, but the values begin to drift as the simulation runs longer. Fig(8) shows the error in the position states for the landing burn. Over 550 seconds the position estimate drifted by 5 m in the y direction and 1.5 m in the x direction. Since most of the motion for this trajectory was across the x direction, it is interesting that the y position has more error.

Now, fig(9) shows the error in the velocity states. The velocity error is incredibly small over the whole simulation, on the order of cm/s. However, even just a small error lead to the potion errors above.

For now, process noise is not included in the bias, gravity, and height states. Also, these states are not coupled with any other state, except for the gravity bias $\boldsymbol{\epsilon}_g^i$ in velocity. So theses state propagate exactly the same as the truth states and there are no errors.

A feature is also added to the simulation to test the camera LOS measurements. The position to a feature ,$\boldsymbol{r}_f^i$ in the inertial

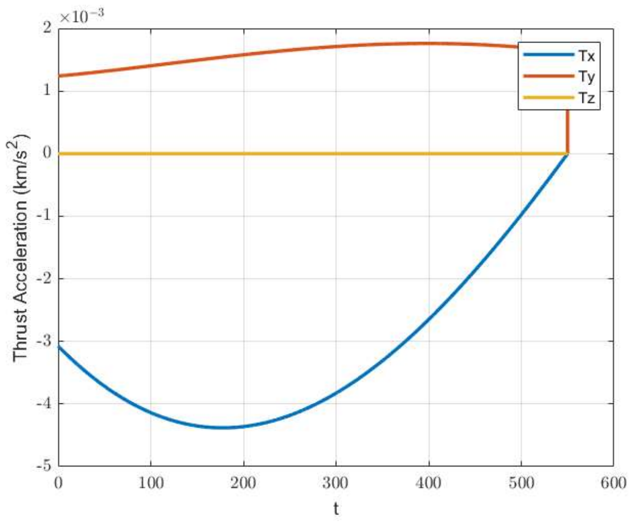


Figure 6. Thrust Commands Given by the Guidance Law

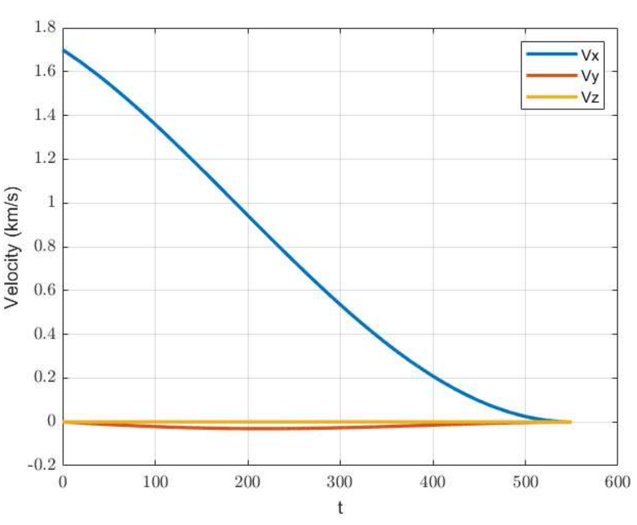


Figure 7. Truth State Velocity

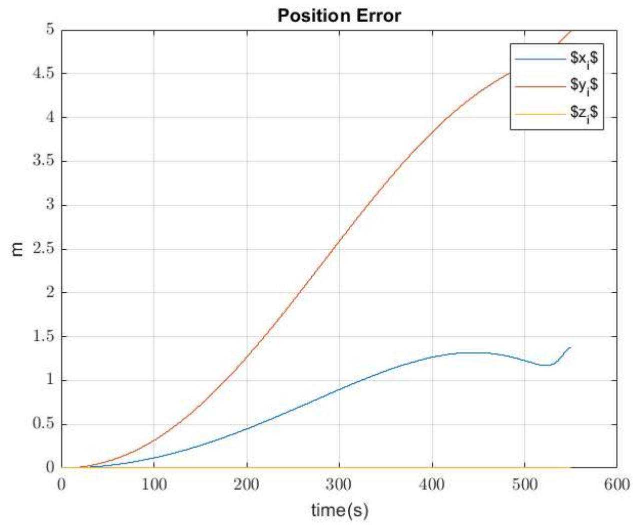


Figure 8. Position Error

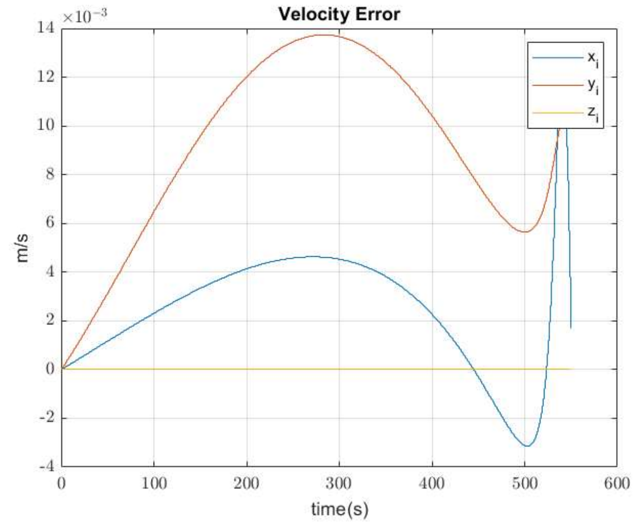


Figure 9. Velocity Error

plane is given and is assumed to be in the camera's line of sight. With this feature location, the LOS vector can easily be computed and the measurement is synthesized. Comparing the synthesized measurement to the predicted measurement, we see the residuals are very small, on the order of 10^{-6}, until near the end of the simulation where some singularity seems to arise. The residuals of the camera measurements are shown in fig(10).

Overall, this simulation shows the truth and navigation states are propagating as expected and the dynamic modeling seems to be correct. We will continue to add elements to make this model more complex, but for now this is a good check to see things are running properly.

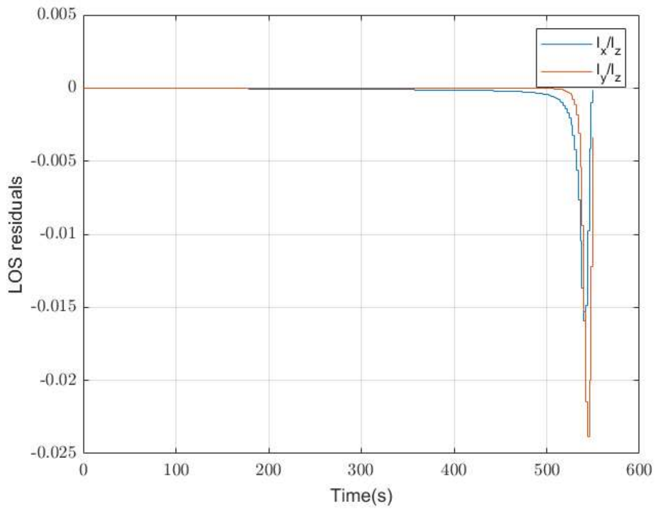


Figure 10. LOS Residuals