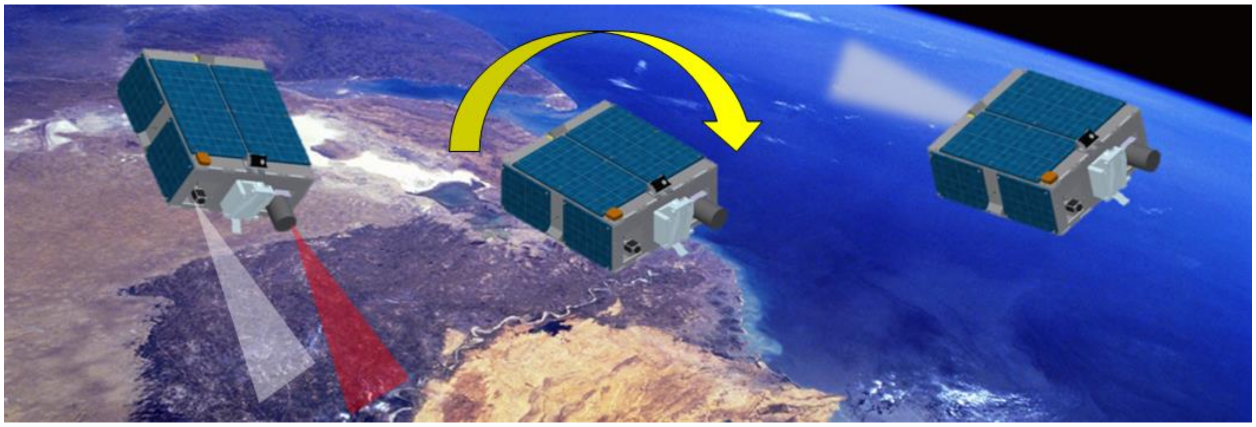
# Prox-1 Guidance, Navigation, & Control Overview: Development, Algorithms, and Integrated Simulation



**Peter Z. Schulte**
**Advisor: Prof. David A. Spencer**
**AE 8900: Special Problems**

Fall 2014
Georgia Institute of Technology
School of Aerospace Engineering

## Abstract

This report describes the development and validation process of a highly automated Guidance, Navigation, & Control (GN&C) subsystem for a small satellite on-orbit inspection application. The resulting GN&C subsystem performs proximity operations (ProxOps) without human-in-the-loop interaction. The report focuses on the description of the GN&C algorithms, the integration and testing of GN&C software, and the development of decision logic to address the question of how such a system can be effectively implemented for full automation. This process is unique because a multitude of operational scenarios must be considered and a set of complex interactions between various GN&C components must be defined to achieve the automation goal. The GN&C subsystem for the Prox-1 satellite is currently under development within the Space Systems Design Laboratory at the Georgia Institute of Technology. The Prox-1 mission involves deploying the LightSail 3U CubeSat, entering into a leading or trailing orbit of LightSail using ground-in-the-loop commands, and then performing automated ProxOps through formation flight and natural motion circumnavigation maneuvers. Operations such as these may be utilized for many scenarios including on-orbit inspection, refueling, repair, construction, reconnaissance, docking, and debris mitigation activities. Prox-1 uses onboard sensors and imaging instruments to perform its GN&C operations during on-orbit inspection of LightSail. Navigation filters perform relative orbit determination based on images of the target spacecraft, and guidance algorithms conduct automated maneuver planning. A slew and tracking controller sends attitude actuation commands to a set of control moment gyroscopes, and other controllers manage desaturation, detumble, and target acquisition/recovery. All Prox-1 GN&C components are developed in a MATLAB/Simulink six degree-of-freedom simulation environment and are integrated using decision logic to autonomously determine when certain actions should be performed. The complexity of this decision logic is the main challenge of this process, and the Stateflow tool in Simulink is used to establish logical relationships and manage data flow between each of the individual GN&C hardware and software components. Once the integrated GN&C simulation is fully developed in MATLAB/Simulink, the algorithms are autocoded to C/C++ and integrated into flight software. The subsystem is tested using hardware-in-the-loop on the flight computers and other hardware.

## Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1148903. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

The author wishes to acknowledge the United States Air Force Office of Scientific Research and Air Force Research Laboratory (AFRL) University Nanosatellite Program for their funding support and technical guidance throughout the Prox-1 flight project lifecycle. Also the Prox-1 team thanks the United States Department of Defense Space Test Program (STP) for providing a ride-along slot for Prox-1 on the STP-2 launch. Josué Muñoz of AFRL is acknowledged for the original development of the Six-Degree-of-Freedom MATLAB/Simulink simulation environment.

The author would also like to thank the following Prox-1 team members who also contributed to the work described herein: Sean Chait, Richard Zappulla, Manan Gandhi, Meet Patel, Aayush Sharma, Midhun Mathew, Tony Nguyen, Jacob Sussman, Mike Van Akin, and Nicholas Zambetti as well as all of those who have worked on the Prox-1 and R3 projects at Georgia Tech. Note that some portions of this report are based on material presented in [1], [2], [3], and [12].

## Nomenclature

| | |
|---|---|
| 3U | Three-Unit (CubeSat) |
| 6DOF | Six-Degree-of-Freedom |
| AD | Attitude Determination |
| ADCS | Attitude Determination and Control Subsystem |
| AFRL | Air Force Research Laboratory |
| APF | Artificial Potential Function |
| BFF | Body-Fixed Frame |
| COB | Center of Brightness |
| CMG | Control Moment Gyroscope |
| DTC | Detumble Controller |
| ECEF | Earth-Centered Earth-Fixed reference frame |
| ECI | Earth-Centered Inertial reference frame |
| FDIR | Fault Detection, Isolation, and Recovery |
| FPF | (imager) Focal Plane Frame |
| FSW | Flight Software |
| FOV | (imager) Field of View |
| Georgia Tech | Georgia Institute of Technology |
| GN&C | Guidance, Navigation, and Control |
| GPS | Global Positioning System |
| IPA | Image Processing Algorithm |
| KOZ | Keep-Out Zone |
| LVI | Launch Vehicle Interface |
| MATLAB | **Mat**rix **Lab**oratory |
| NASA | National Aeronautics and Space Administration |
| NMC | Natural Motion Circumnavigation |
| P-POD | Poly Picosat Orbital Deployer |
| ProxOps | Proximity Operations |
| RelOD | Relative Orbit Determination |
| ROE | Relative Orbital Element |
| RSO | Resident Space Object |
| RSW | Satellite orbit frame (for relative motion) |
| STC | Slew & Tracking Controller |
| STM | State Transition Matrix |
| TAC | Target Acquisition Controller |
| TC | Thruster Controller |
| TR | Torque Rod |

# Table of Contents

## Table of Figures

## Table of Tables

## 1. Prox-1 Mission Description

Prox-1 is a small satellite mission designed, built, and operated by students at the Georgia Institute of Technology (Georgia Tech) under the University Nanosatellite Program at the Air Force Research Laboratory (AFRL). The primary mission of Prox-1 is to demonstrate automated relative trajectory control in Low-Earth Orbit with an uncooperative target for an on-orbit inspection application [3]. The target spacecraft for this mission is The Planetary Society's LightSail, a 3U CubeSat that demonstrates solar sail technology [4]. LightSail is stowed inside of Prox-1 during launch and then deployed using a Poly Picosat Orbital Deployer (P-POD) device.

The mission begins with deployment of Prox-1 as a secondary payload from the SpaceX Falcon Heavy launch vehicle. Prox-1 then uses magnetic torque rods to detumble the spacecraft. Once angular rates have been nulled, a spacecraft checkout phase ensues where on-orbit functionality is established. Following Prox-1 checkout, LightSail is deployed from the P-POD. A period of time is allowed for the two spacecraft to drift apart, and orbit determination is performed on the ground to determine the trajectories of both vehicles. Prox-1 spacecraft checkout and deployment of LightSail are shown in Figures 1 and 2 respectively.

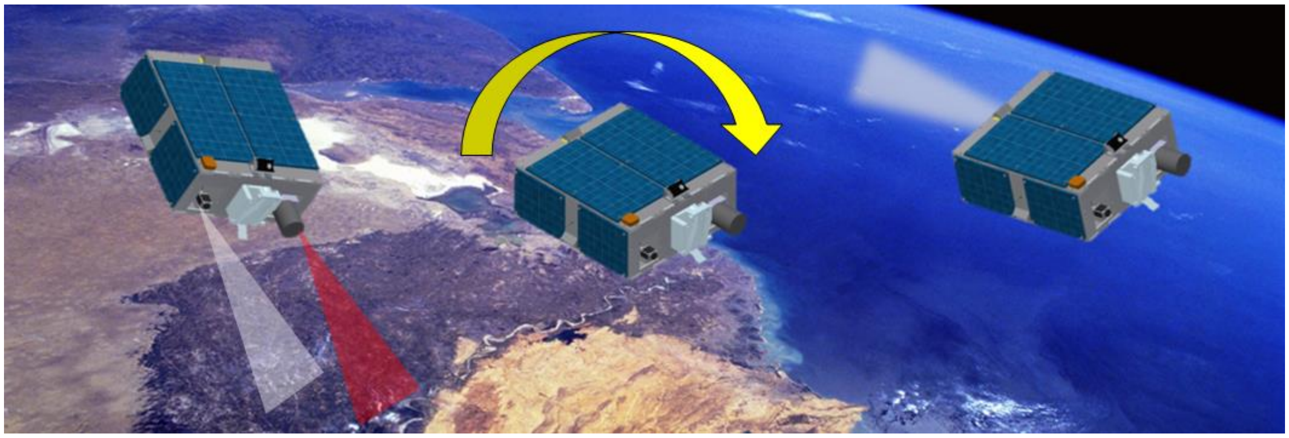


Figure 1: Prox-1 Spacecraft Checkout Phase

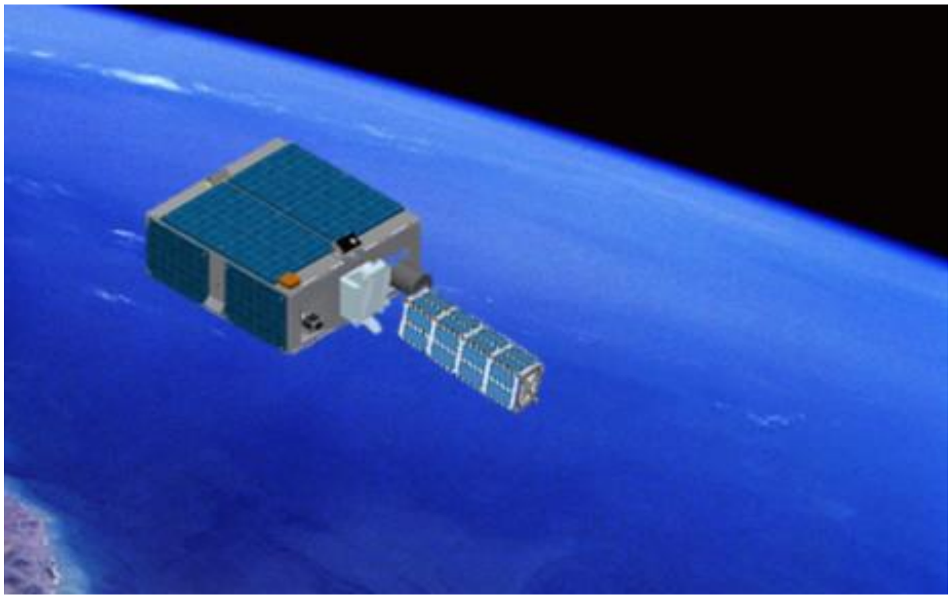


Figure 2: LightSail Deployment from Prox-1

After orbit determination has been completed, ground commands maneuver Prox-1 to within visual sensor range of LightSail using a cold gas thruster developed at The University of Texas at Austin [5]. At this point, automated operations begin. The Prox-1 Guidance, Navigation, and Control subsystem acquires the target spacecraft in its thermal imager field of view (FOV) for relative navigation and maneuvers the spacecraft into formation flight in a leading or trailing orbit with respect to LightSail.

Entry into automated ProxOps is then commanded from the ground. During ProxOps Phase I, Prox-1 performs a rest-to-rest maneuver to move from the initial formation flight location to a point closer to LightSail and station-keeping capability is demonstrated for multiple orbits. Then, during ProxOps Phase II Prox-1 enters into a Natural Motion Circumnavigation (NMC) of LightSail using a relative elliptical orbit. During the ProxOps mission phases, Prox-1 performs all maneuvers without communication from the ground or cooperation from LightSail. ProxOps Phase I and II are shown in Figures 3 and 4 respectively.

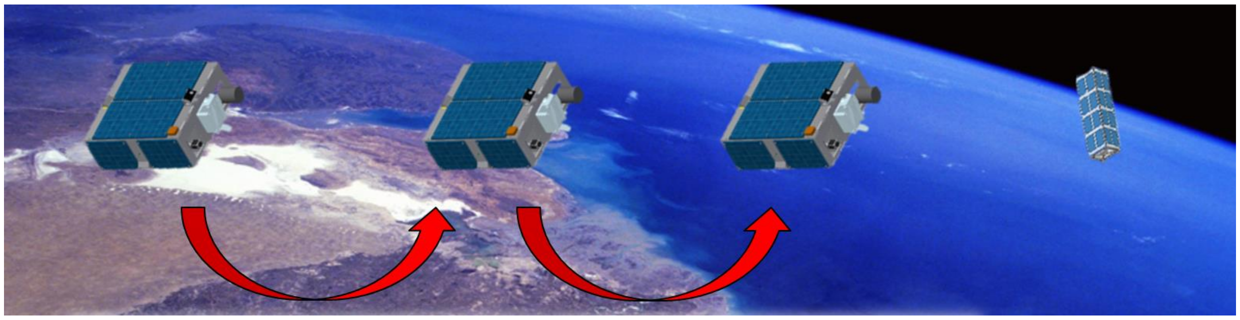


Figure 3: ProxOps Phase I: Rest-to-Rest Approach Strategy

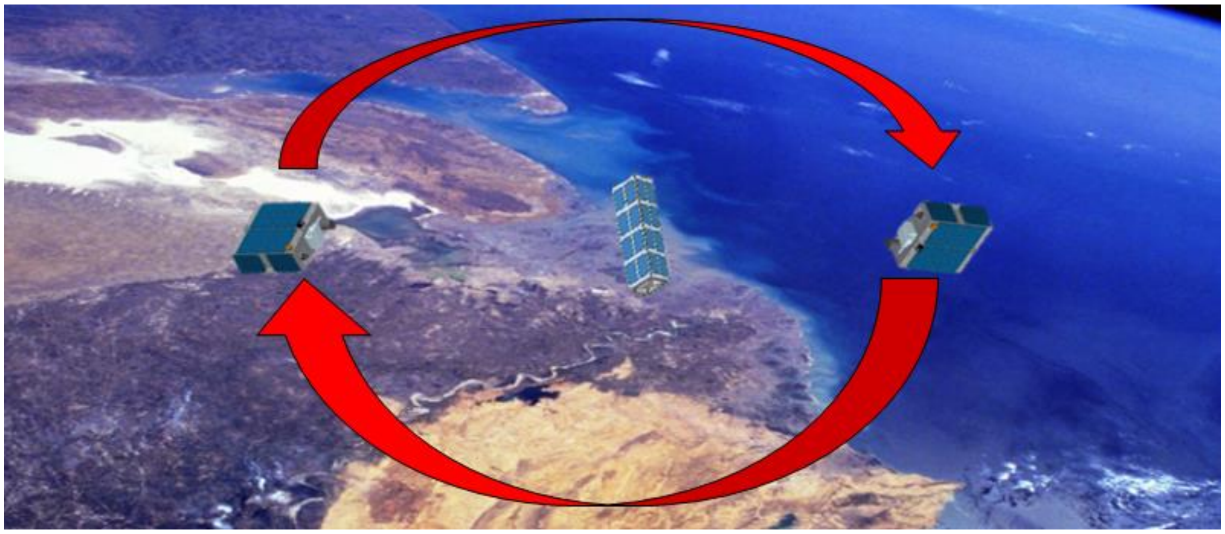


Figure 4: ProxOps Phase II: Natural Motion Circumnavigation

After Prox-1's primary mission is complete, it performs on-orbit inspection to image the deployment of LightSail's 32 $m^2$ solar sail as shown in Figure 5. Finally, once primary and secondary mission requirements are completed, Prox-1 is deorbited using a deployable drag device.

Figure 5: On-orbit inspection of LightSail deployment

## 2. Guidance, Navigation, and Control Subsystem Overview

The Guidance, Navigation, & Control (GN&C) subsystem for Prox-1 is composed of navigation components used to determine the satellite attitude and trajectory state, and guidance and control components to control the attitude and perform propulsive maneuvers. The interactions between various components are shown in Figure 6.

### 2.1. *Navigation Components*

Each of the navigation components is used to determine the state of the spacecraft based on inputs from various sensors. An inertial attitude determination (AD) filter implemented as an extended Kalman filter computes Prox-1's attitude quaternion and angular velocity vector using rate gyroscope, magnetometer, and sun sensor measurements. A microbolometer is used to capture infrared images of LightSail, which are fed into a set of image processing algorithms and a relative orbit determination (RelOD) filter to obtain relative position and velocity [6]. Accelerometer measurements are also taken onboard Prox-1, and a visual camera captures images for use on the ground. Global Positioning System (GPS) measurements are used for onboard inertial orbit determination.

### 2.2. *Guidance Components*

Prox-1's guidance algorithms evaluate state information and determine a set of maneuvers to reach a desired state with respect to LightSail. Translational guidance algorithms utilize relative orbital elements (ROEs) to calculate the desired state and artificial potential functions (APFs) for collision avoidance to stay away from a defined "keep-out-zone" [7].

### 2.3. *Control Components*

Finally, Prox-1 has control components to implement maneuvers commanded by the guidance algorithms and perform other tasks using various actuators. A slew and tracking controller (STC) uses a Control Moment Gyroscope (CMG) unit developed by Honeybee Robotics Spacecraft Mechanisms Corporation for primary attitude control and performs many functions, including slewing to the proper attitude before commanding a thrust maneuver. The STC can also use the CMGs to track LightSail so that it remains in Prox-1's imager FOV. A target acquisition controller (TAC) determines appropriate slew maneuvers for the STC to center the target spacecraft in Prox-1's imaging instrument field of view (FOV). This capability is used for initial target acquisition and for recovery of the target if visual contact is lost. Also, a detumble algorithm is used to damp high angular velocities after launch vehicle separation, and a desaturation algorithm performs angular momentum management to prevent the CMGs from saturating. Finally, a torque rod (TR) controller is used for secondary attitude control and to implement commands from the desaturation and detumble algorithms using three single axis magnetic torque rods designed and manufactured in-house at Georgia Tech.

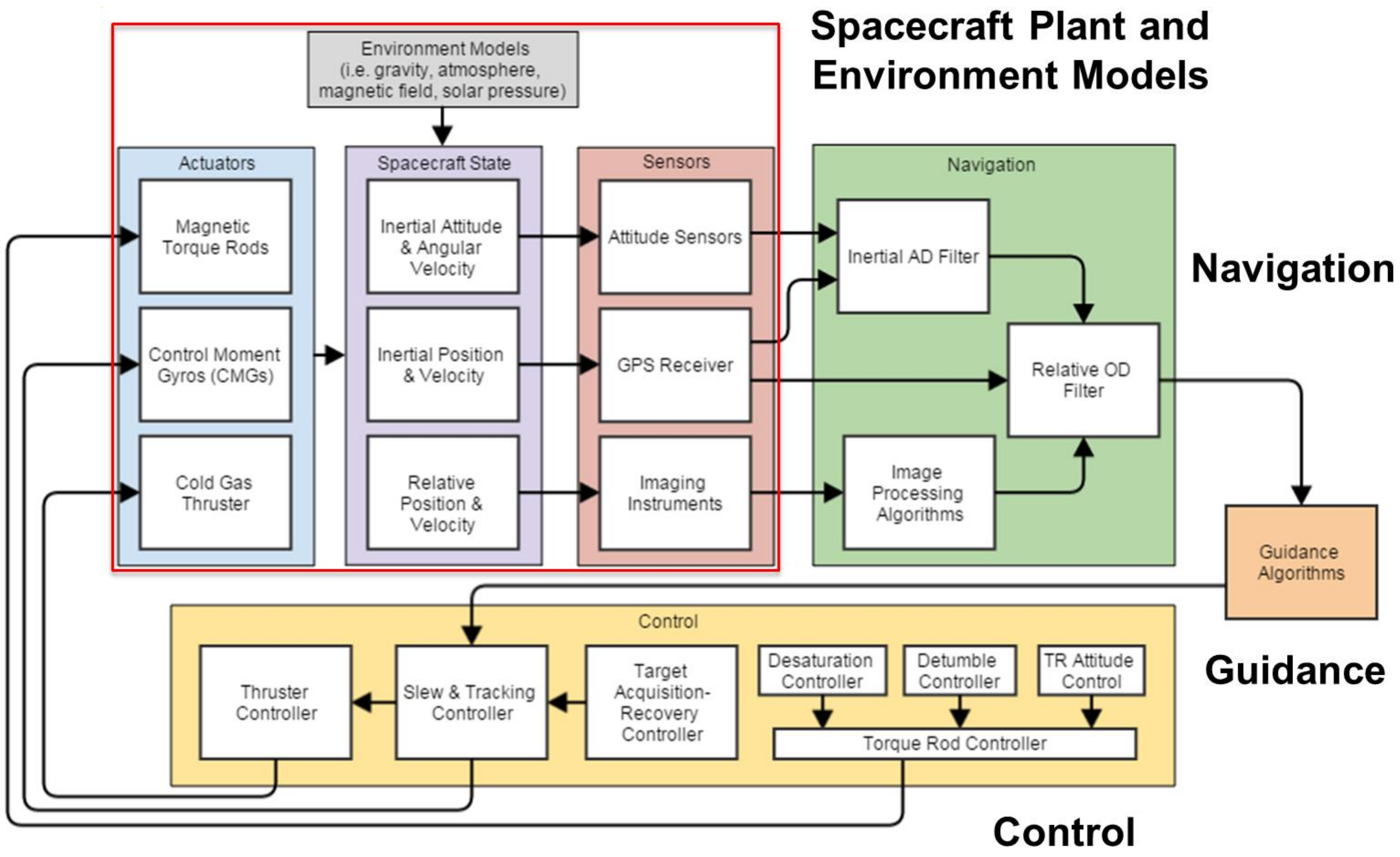

Spacecraft Plant and Environment Models
Environment Models (i.e. gravity, atmosphere, magnetic field, solar pressure)
Actuators
Magnetic Torque Rods
Control Moment Gyros (CMGs)
Cold Gas Thruster
Spacecraft State
Inertial Attitude & Angular Velocity
Inertial Position & Velocity
Relative Position & Velocity
Sensors
Attitude Sensors
GPS Receiver
Imaging Instruments
Navigation
Inertial AD Filter
Relative OD Filter
Image Processing Algorithms
Navigation
Guidance Algorithms
Guidance
Control
Thruster Controller
Slew & Tracking Controller
Target Acquisition-Recovery Controller
Desaturation Controller
Detumble Controller
TR Attitude Control
Torque Rod Controller
Control

Figure 6: Flowchart showing GN&C component interactions.
Components boxed in red are for simulation only and will not be coded into GN&C flight software.

## 2.4. *Reference Frame Definitions*

### 2.4.1. Earth-Centered Inertial (ECI)

The Earth-Centered Inertial (ECI) frame is the main reference frame for determining the absolute position, velocity, and attitude of Prox-1. Prox-1 will use the J2000 ECI frame, which has its origin at the center of Earth. The $\hat{Z}_{J2000}$ axis points toward the North Pole, the $\hat{X}_{J2000}$ direction points toward the mean vernal equinox (♈) at the epoch of 12:00 Terrestrial Time on January 1, 2000, and the $\hat{Y}_{J2000}$ direction is defined by the cross product of the first two basis vectors using the right hand rule. Note that this frame does not rotate with the spin of the Earth. Figure 7 shows a representation of the J2000 ECI Frame.

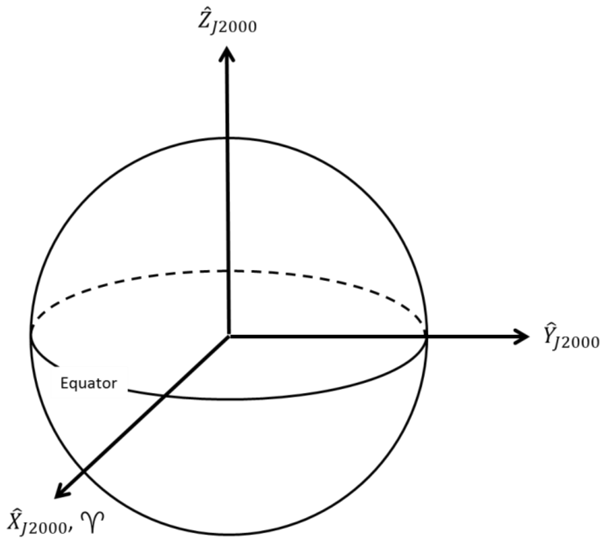


Figure 7: Earth-Centered Inertial reference frame [8]

### 2.4.2. Earth-Centered Earth-Fixed (ECEF)

The Earth-Centered Earth-Fixed (ECEF) frame is similar to the ECI frame in that it is a geocentric frame, however, unlike the ECI frame, its basis vectors are fixed with respect to the surface of the Earth such that the frame rotates with the spin of the Earth. The $\hat{Z}_{ECEF}$ axis points toward the mean rotational axis of the Earth, the $\hat{X}_{ECEF}$ axis points toward the intersection of the equator and the prime meridian (the line of 0° longitude), and the $\hat{Y}_{ECEF}$ axis is defined by the cross product of the first two basis vectors completing the right-handed system. Earth's magnetic field is one of many things conveniently expressed in the ECEF frame, and GPS receivers usually output position and velocity in ECEF coordinates. Figure 8 shows the similarities and differences of the ECEF frame when compared to the ECI frame.

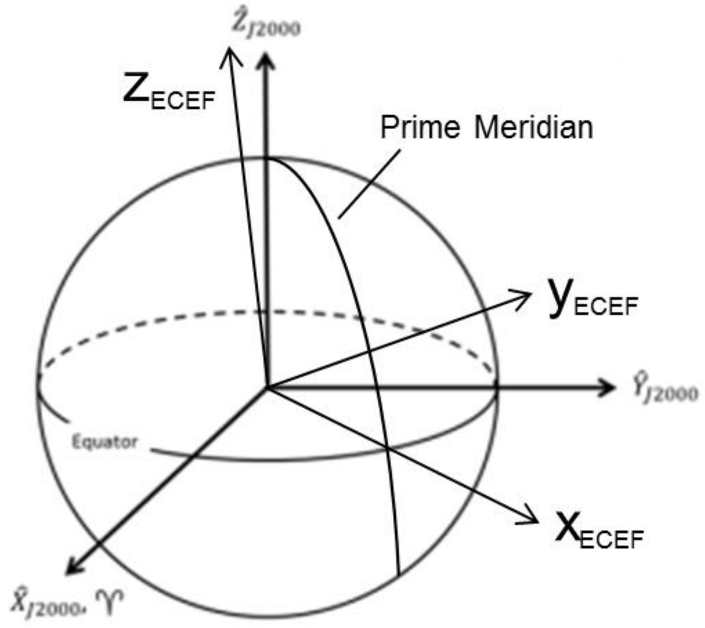


Figure 8: Earth-Centered Earth-Fixed reference frame

### 2.4.3. Satellite Frame (RSW)

The RSW Frame, also known as the Satellite Frame, is an orbit-defined frame whose origin typically lies at the center of mass of the target satellite with the three basic vectors defined by the position vector (radial), velocity vector (in-track) and their cross product (cross-track). Figure 9 shows a visualization of this coordinate frame. Notice that this coordinate frame is constantly moving and rotating in its orbit as the Target moves about the Earth. Note that the RSW frame is also known as the Local-Vertical, Local-Horizontal or Hill frame.

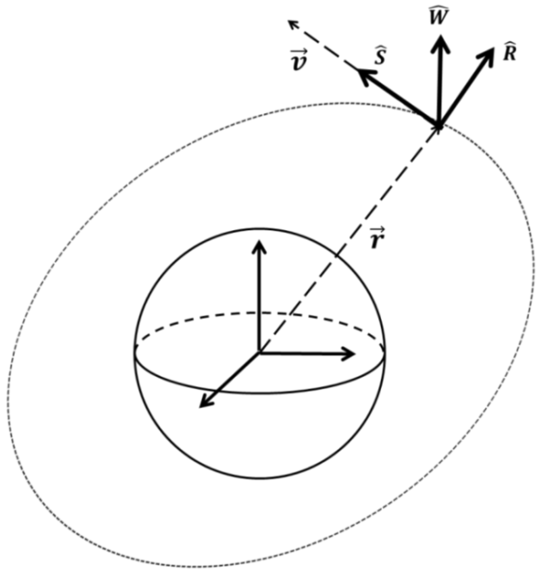


Figure 9: Visual representation of the satellite (RSW) frame

For the Prox-1 mission, it is assumed that no *a priori* knowledge of the Target's inertial position and velocity will be known, thus the **origin** of the RSW frame is located at LightSail's estimated position relative to Prox-1, but the **orientation** of the RSW frame is based on **Prox-1's** inertial position. As a result, the orientation of the basis vectors $\{\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{S}}, \widehat{\boldsymbol{W}}\}$ are defined by Eq. (1), where $\boldsymbol{r}_{chaser}$ and $\boldsymbol{V}_{chaser}$ are the positon and velocity vectors of Prox-1 in the ECI frame and the $^{\mathbf{x}}$ superscript represents the skew function. When the skew function is applied to a vector, a skew symmetric matrix is created which, when multiplied with another vector, produces the same result as a cross product between the two vectors. The following assumptions are applied in calculation of the RSW frame: $\frac{r_{target}}{r_{chaser}}$ is small and $\widehat{\boldsymbol{V}}_{target} \approx \widehat{\boldsymbol{V}}_{chaser}$.

$$\widehat{\boldsymbol{R}} := \frac{\boldsymbol{r}_{chaser}}{||\boldsymbol{r}_{chaser}||} \qquad \widehat{\boldsymbol{W}} := \frac{\boldsymbol{r}^{x}_{chaser}\boldsymbol{V}_{chaser}}{\left|\left|\boldsymbol{r}^{x}_{chaser}\boldsymbol{V}_{chaser}\right|\right|} \qquad \widehat{\boldsymbol{S}} := \frac{\widehat{\boldsymbol{W}}^{x}\widehat{\boldsymbol{R}}}{\left|\left|\widehat{\boldsymbol{W}}^{x}\widehat{\boldsymbol{R}}\right|\right|} \tag{1}$$

### 2.4.4. Body-Fixed Frame (BFF)

The Body-Fixed Frame (BFF), as its name implies, is a coordinate frame that is fixed with respect to the Prox-1 satellite. It is centered at the Lightband Launch Vehicle Interface (LVI) with the three basis vectors $(\hat{x}_b, \hat{y}_b, \hat{z}_b)$ defined such that $\hat{y}_b$ is oriented along the imager boresight, $\hat{z}_b$ is normal to the Lightband LVI and oriented away from the body of the satellite, and $\hat{x}_b$ is defined by the right hand rule. The main purpose of this coordinate frame is to determine the attitude and angular velocity of Prox-1 relative to the RSW Frame. Figure 10 shows the orientation of the body-fixed frame with respect to the Prox-1 spacecraft.

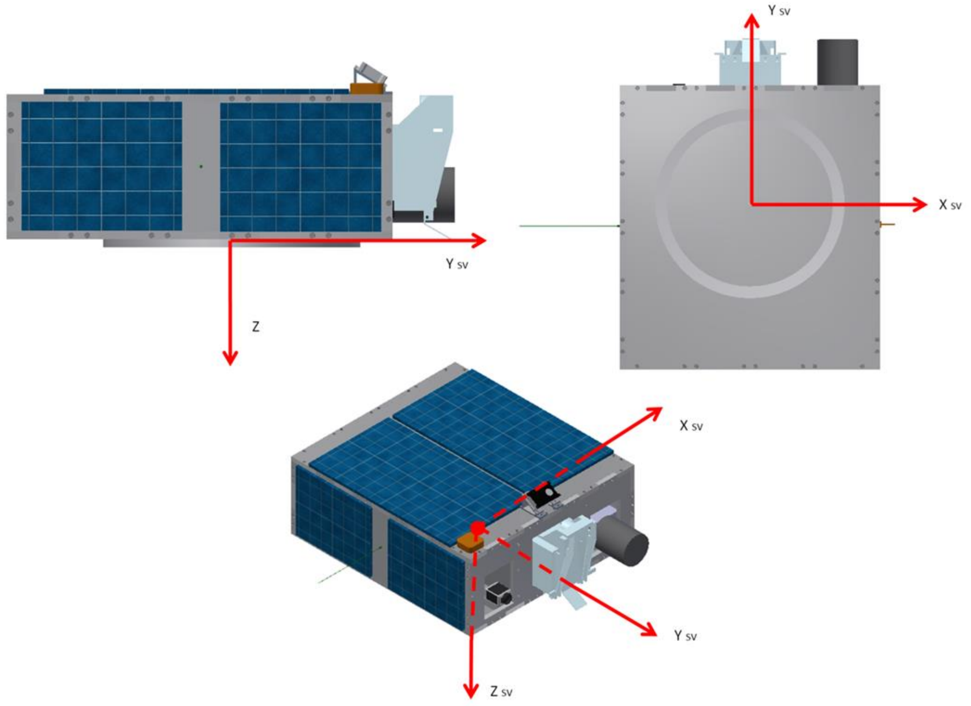


Figure 10: Prox-1 Body-Fixed Frame orientation

### 2.4.5. Imager Focal Plane Frame (FPF)

The imager focal plane frame (FPF) is a frame that is defined and fixed with respect to the microbolometer's focal plane array. It is centered at the upper left corner of the pixel array when viewing the output image with the $\hat{x}_i$ axis pointing down the vertical direction of the pixel array, the $\hat{y}_i$ axis pointing to the right along the horizontal direction of the pixel array, and the $\hat{z}_i$ axis defined as the cross product of the first two basis vectors using the right hand rule as shown in Figure 11.

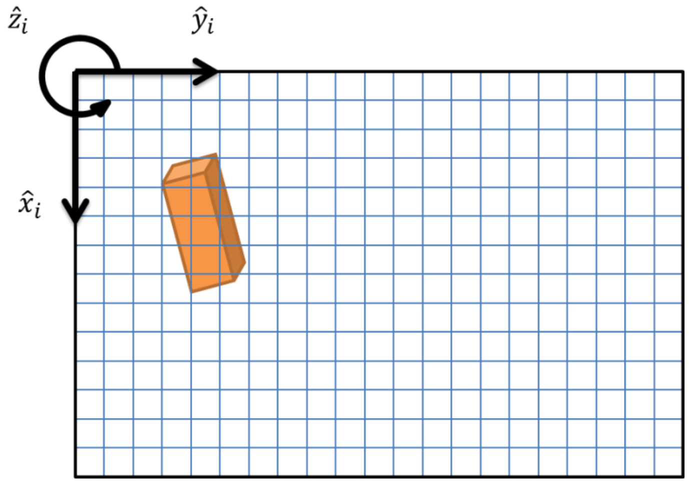


Figure 11: Imager Focal Plane Frame on output pixel array

## 3. Navigation Component Details

### 3.1. *Inertial Attitude Determination Filter*

The Inertial Attitude Determination (AD) filter combines measurements from rate gyroscopes, sun sensors, and a magnetometer to compute the attitude and angular velocity of Prox-1 with respect to the ECI frame. This inertial filter does not include orbit determination because the inertial GPS position and velocity solution is already filtered by the GPS receiver. The following description was originally written by Prox-1 team member Midhun Mathew.

The AD Filter follows the theory of an Extended Kalman Filter to determine Prox-1's current attitude and is based on the book *Optimal Estimation of Dynamic Systems* by Crassidis and Junkins [9]. Inputs include current estimations of attitude, angular velocity, rate gyro bias, and covariance estimates, measurements from various sensors, and the pointing vector of each sensor with respect to the BFF. Initial guesses must also be defined for the attitude, angular velocity, rate gyro bias, and covariance estimates. The filter operates in discrete time and propagates forward the new attitude estimate quaternion, the covariance matrix, the bias of the rate gyros, and the estimated angular velocity. The filter operates in four distinct steps: propagate, gain, update and delay.

The propagation step focuses on converting information from the previous time step, k-1$^+$, to the current time step, k$^-$. During this step, all $\Delta t$ values are assigned to the time step and angular velocity is assumed to be non-zero. The quaternion is propagated using the process shown in Eq. (2), where $\hat{q}_{k+1}^-$ is the propagated quaternion, $\hat{q}_k^+$ is the post-update quaternion estimate, $\widehat{\omega}_k^+$ is the post-update angular velocity estimate, $\Delta t$ is the gyro sampling interval, $\overline{\Omega}(\widehat{\omega}_k^+)$ is given by Eq. (3) and $\hat{\psi}_k^+$ is given by Eq. (4).

$$\hat{q}_{k+1}^- = \overline{\Omega}(\widehat{\omega}_k^+)\,\hat{q}_k^+ \tag{2}$$

$$\overline{\Omega}(\widehat{\omega}_k^+) = \begin{bmatrix} \cos\left(\frac{1}{2}\|\widehat{\omega}_k^+\|\Delta t\right) I_{3x3} - [\hat{\psi}_k^{+x}] & \hat{\psi}_k^+ \\ -\hat{\psi}_k^{+\,T} & \cos\left(\frac{1}{2}\|\widehat{\omega}_k^+\|\Delta t\right) \end{bmatrix} \tag{3}$$

$$\hat{\psi}_k^+ = \frac{\sin\left(\frac{1}{2}\|\widehat{\omega}_k^+\|\Delta t\right)\widehat{\omega}_k^+}{\|\widehat{\omega}_k^+\|} \tag{4}$$

Eq. (5) shows the propagation of the angular velocity $\widehat{w}$ and the bias of the rate gyros $\hat{\beta}$.

$$\widehat{\omega}_k^+ = \widehat{\omega}_k - \hat{\beta}_k^+ \tag{5a}$$

$$\hat{\beta}_{k+1}^- = \hat{\beta}_k^+ \tag{5b}$$

The covariance matrix $P_k$ may be propagated using Eq. (6), utilizing $\mathrm{G}_k$ given by Eq. (7), a discrete time error-state transition matrix $\Phi_k$ given by Eq. (8), and a discrete process noise covariance $\mathrm{Q}_k$ given by Eq. (9), which utilizes the rate random walk $\sigma_v^{\,2}$ and the angle random walk $\sigma_u^{\,2}$. An overview of the propagation step is shown in Figure 12.

$$P_{k+1}^- = \Phi_k P_k^+ \Phi_k{}^T + \mathrm{G}_k \mathrm{Q}_k \mathrm{G}_k{}^T \tag{6}$$

$$\mathrm{G}_k = \begin{bmatrix} -I_{3x3} & 0_{3x3} \\ 0_{3x3} & I_{3x3} \end{bmatrix} \tag{7}$$

$$\Phi_k = \begin{bmatrix} I_{3x3} - [\widehat{\omega}^x]\frac{\sin(\|\widehat{\omega}\|\Delta t)}{\|\widehat{\omega}\|} + [\widehat{\omega}^x]^2\frac{1-\cos(\|\widehat{\omega}\|\Delta t)}{\|\widehat{\omega}\|^2} & [\widehat{\omega}^x]\frac{1-\cos(\|\widehat{\omega}\|\Delta t)}{\|\widehat{\omega}\|^2} - I_{3x3}\Delta t - [\widehat{\omega}^x]^2\frac{\|\widehat{\omega}\|\Delta t - \sin(\|\widehat{\omega}\|\Delta t)}{\|\widehat{\omega}\|^3} \\ 0_{3x3} & I_{3x3} \end{bmatrix} \tag{8}$$

$$\mathrm{Q}_k = \begin{bmatrix} \left(\sigma_v{}^2\Delta t + \frac{1}{3}\sigma_u{}^2\Delta t^3\right) I_{3x3} & -\left(\frac{1}{2}\sigma_u{}^2\Delta t^2\right) I_{3x3} \\ I_{3x3} & I_{3x3} \end{bmatrix} \tag{9}$$

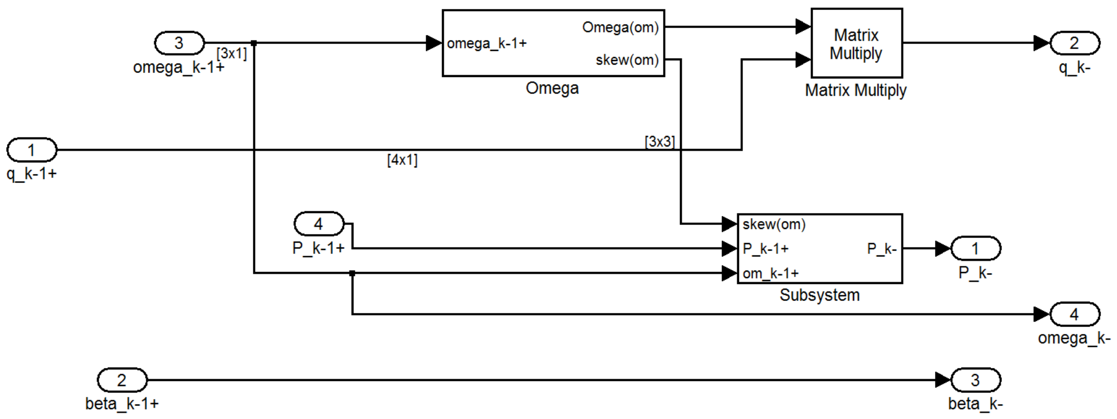
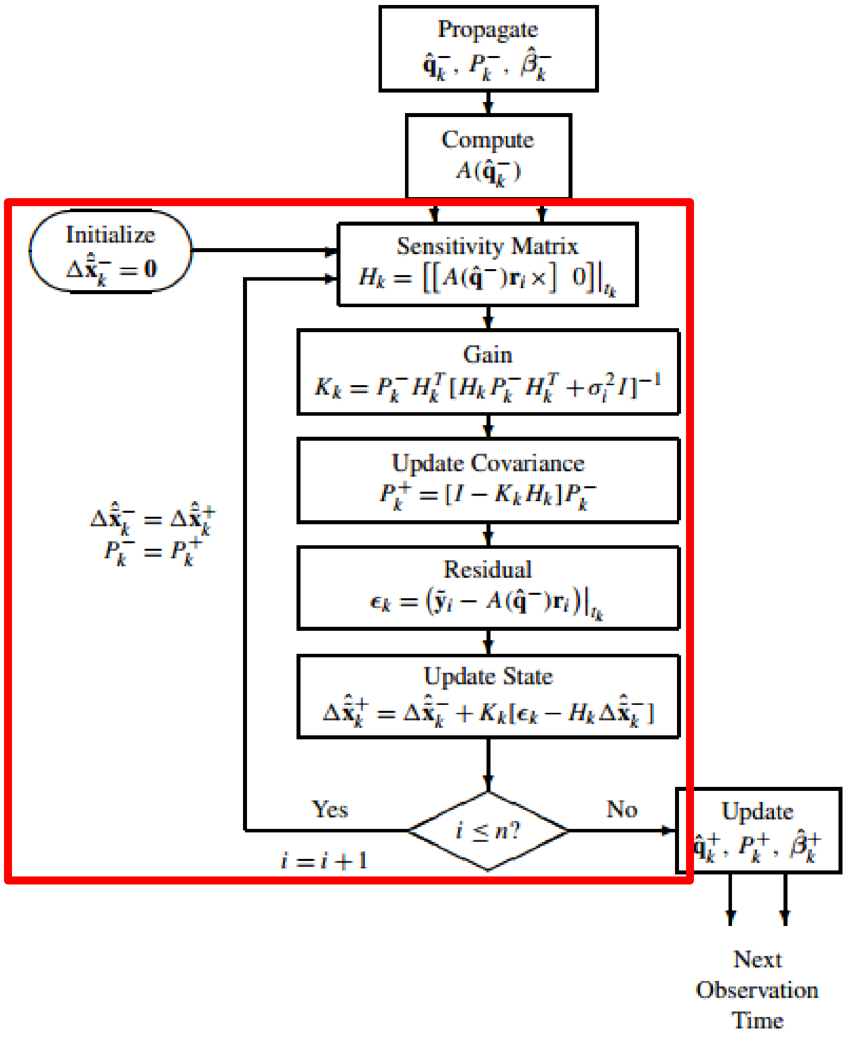


Figure 12: Overall View of AD Filter Propagation Step

The gain step involves calculating the gain caused by each sensor's reading. Given a sensor's pointing vector, the measured pointing vector, the current covariance and the current attitude, the gain step creates a state vector that will be used to update the attitude and rate gyro bias. The sensor update method follows the steps outlined in Figure 13. The gain section works on the principle of superposition. Each sensor input may be interpreted separately and then added together to determine the overall change in state and covariance. For this section of the filter, *A(**q**)* was determined to be the rotation matrix converting from the ECI frame to the BFF frame. The value of $\sigma_i$ was determined to be the noise of each sensor. For reference, the residual $\epsilon_k$ determines the difference in the measured vector pointing of sensors and their actual pointing direction. The measurement obtained from the sensors is represented by $\bar{\boldsymbol{y}}_i$.



Propagate
$\hat{\mathbf{q}}_k^-, P_k^-, \hat{\beta}_k^-$
Compute
$A(\hat{\mathbf{q}}_k^-)$
Initialize
$\Delta\hat{\tilde{\mathbf{x}}}_k^- = \mathbf{0}$
Sensitivity Matrix
$H_k = [[A(\hat{\mathbf{q}}^-)\mathbf{r}_i \times] \ \ 0]|_{t_k}$
Gain
$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + \sigma_i^2 I]^{-1}$
Update Covariance
$P_k^+ = [I - K_k H_k] P_k^-$
Residual
$\epsilon_k = (\tilde{\mathbf{y}}_i - A(\hat{\mathbf{q}}^-)\mathbf{r}_i)|_{t_k}$
Update State
$\Delta\hat{\tilde{\mathbf{x}}}_k^+ = \Delta\hat{\tilde{\mathbf{x}}}_k^- + K_k[\epsilon_k - H_k \Delta\hat{\tilde{\mathbf{x}}}_k^-]$
$\Delta\hat{\tilde{\mathbf{x}}}_k^- = \Delta\hat{\tilde{\mathbf{x}}}_k^+$
$P_k^- = P_k^+$
Yes
$i \le n?$
No
$i = i + 1$
Update
$\hat{\mathbf{q}}_k^+, P_k^+, \hat{\beta}_k^+$
Next Observation Time

Figure 13: Computationally Efficient Attitude Estimation Algorithm [9].

To best utilize the gain step in the simulation, the noise model of each sensor in the filter must be similar to the noise model utilized in the sensor model; a sensor model with signal magnitude dependent noise should be processed with an algorithm that accounts for that signal-to-noise relation. Also, raw sensor measurements must each be processed individually and fully within the AD filter; pre-processed values cannot be fed to the filter from the plant model. For example, to best test the AD Filter, the sun sensor plant model must calculate the actual outputs of each sun sensor plus noise instead of calculating the sun's direction from the orbit and velocities.

The update step uses the updated state from the gain step to create better estimates of attitude, angular velocity and rate gyro bias as seen in Figure 14. The first three elements of the state vector contain the changes to the attitude while the last three elements contain changes to the angular velocity and bias. The update of the attitude is a multiplicative process whereas the bias is updated with additive properties. The updates are performed using Eq. (10)-(12) where $\delta\hat{\alpha}_k^+$ is the vector portion of the error quaternion update state. In Eq. (11) the scalar component of the error quaternion is assumed to be one to within first-order and a small angle approximation has been used to define the vector part of the error quaternion. The $\Xi$ operation is defined in Eq. (12) where $\varepsilon$ is the vector part of the quaternion $q$ and $\eta$ is the scalar part. Eq. (13) is a brute-force normalization of the quaternion that must be performed after the update.

$$\hat{\beta}_k^+ = \hat{\beta}_k^- + \Delta\hat{\beta}_k^+ \tag{10}$$

$$\hat{q}_k^+ = \hat{q}_k^- + \frac{1}{2}\Xi(\hat{q}_k^-)\delta\hat{\alpha}_k^+ \tag{11}$$

$$\Xi(q) = \begin{bmatrix} \eta I + \varepsilon^x \\ -\varepsilon^T \end{bmatrix} \tag{12}$$

$$\hat{q}_k^{+T}\hat{q}_k^+ = 1 \tag{13}$$

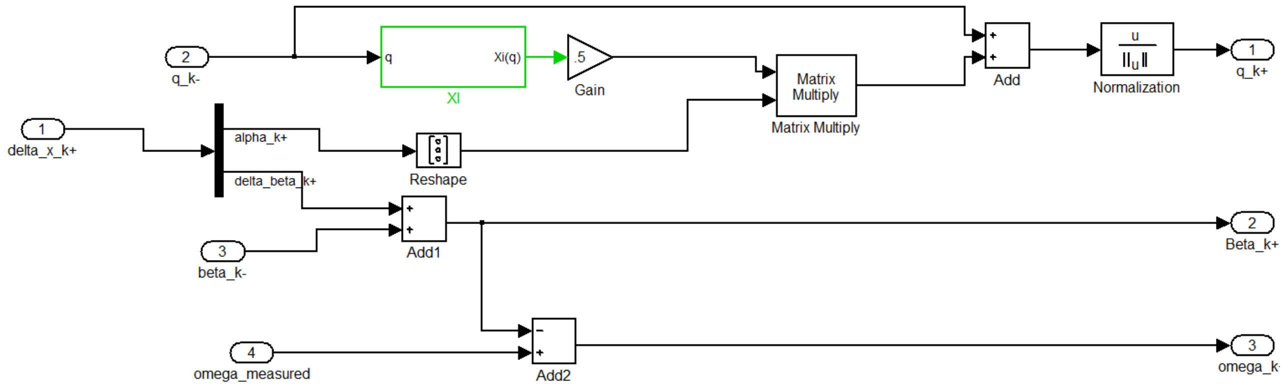


Figure 14: Overall View of AD Filter Update Step

The final major step of the filter is to delay the values that are used in the next time step. Utilizing a delay time block, the current updated attitude estimate, rate gyro bias, angular velocity and covariance are delayed from the k$^+$ time step to the k-1$^+$ time step. These values are now the new starting conditions for the next set of calculations.

For future development, the construction of the filter per the theory developed in the Crassidis text is complete. However, the correct sensor inputs, noise methods and initial conditions must be identified. Currently, the AD filter is being tested with a single sun sensor and magnetometer. The effects of changing these vectors are being observed. Additionally, the effect and magnitude of the rate gyro bias must be more closely observed. Incorrectly processed sensor data can lead to an infinitely increasing rate gyro bias and excessively noisy results; Figure 15 shows the results that come from a steady bias estimate using only rate gyro measurements.

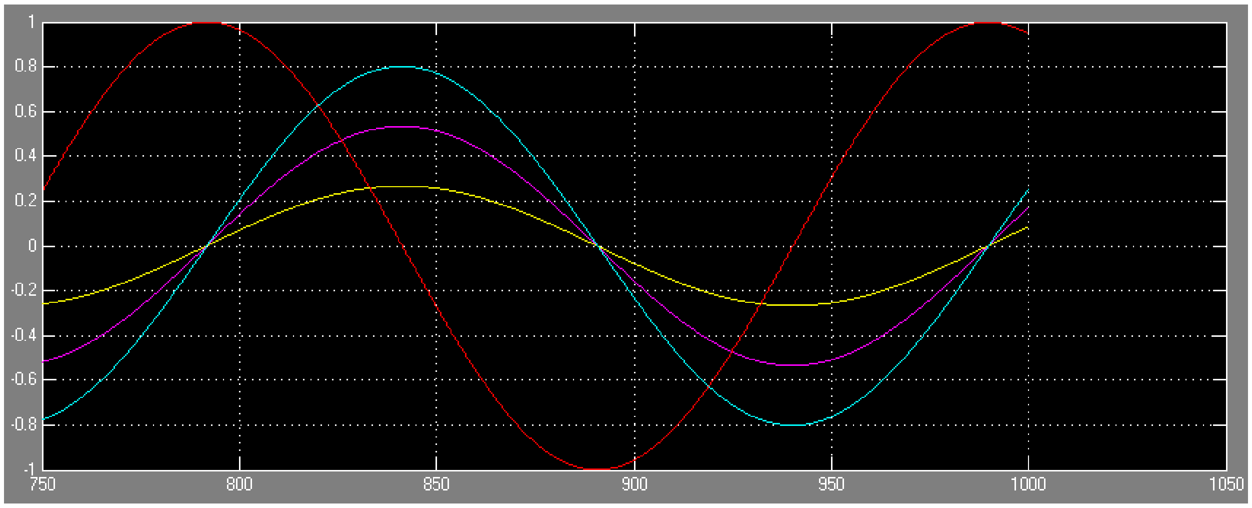


Figure 15: Attitude quaternion estimate with only rate gyro measurements

## 3.2. *Image Processing Algorithms*

The image processing algorithms (IPAs) form the basis of the Prox-1 guidance strategy, as they provide relative positioning information to the rest of the Prox-1 GN&C system. In their current form, the IPA results are shown to vary depending on both the range and orientation of the target or Resident Space Object (RSO). However, through the use of filtering, the precision of the relative position estimate is increased. This description was originally written by Richard Zappulla [2] and is based on material presented by Bellet [10] and Vedie et al. [11].

### 3.2.1. Blobber Algorithm

Immediately following image acquisition, the Blobber Algorithm is the first step in the IPAs and the main process in image processing. The purpose of the Blobber Algorithm is to identify the RSO in the image. The Blobber Algorithm flow diagram is illustrated in Figure 16.

First, the algorithm detects the pixels in an image within a certain range of intensity. For the microbolometer this intensity is analogous to temperature; for the visual camera it is a measure of visible radiance. Next, the algorithm detects groups of pixels that are connected to each other and form a blob (Binary Large Object). Generally, there is not just one blob detected,

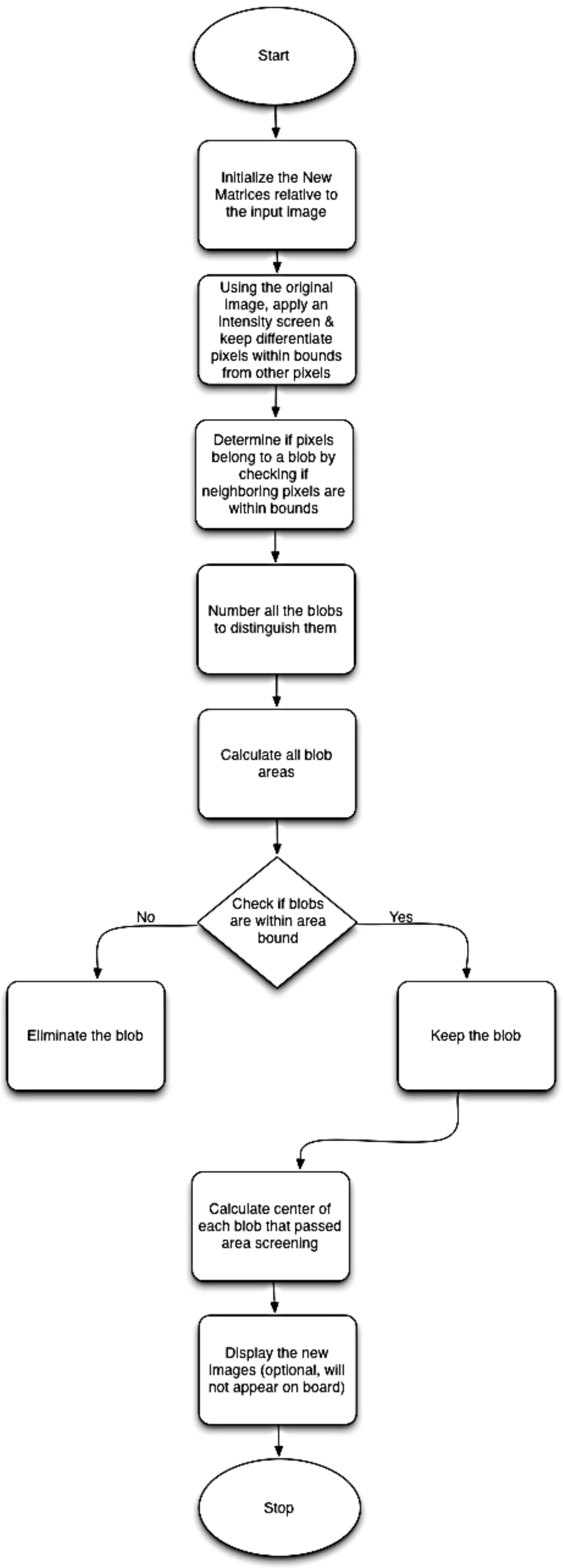

Start
Initialize the New Matrices relative to the input image
Using the original image, apply an intensity screen & keep differentiate pixels within bounds from other pixels
Determine if pixels belong to a blob by checking if neighboring pixels are within bounds
Number all the blobs to distinguish them
Calculate all blob areas
Check if blobs are within area bound
No
Yes
Eliminate the blob
Keep the blob
Calculate center of each blob that passed area screening
Display the new images (optional, will not appear on board)
Stop

Figure 16: Blobber algorithm flow diagram

especially when the image background is the Earth. However the object which has to be localized should be one of the blobs detected. Therefore, to choose the correct blob among the few which are remaining, an area screening must be applied. The correct blob is selected by calculating its area on the image, which is the number of pixels contained in the blob. Indeed, with previous knowledge about the object to detect, it is possible to know the approximate range of the cross-sectional area [10].

The ideal results of the Blobber algorithm are a calculated area for the identified blob and the location of the Center of Brightness (COB) for the blob. The COB is similar to an area centroid or center of mass and is used as the central location of the RSO. COB coordinates are given with respect to the imager's focal plane (FPF) and must be mapped into the body-fixed coordinate frame (BFF) in order to represent the unit vector from Prox-1 to the RSO.

### 3.2.2. Unit Vector Determination

Using the COB coordinates, a unit vector can be found using geometry and the optical properties of the imager. The RSO can at all times be considered to be focused at infinity in relation to the focal length of the camera lenses – that is the distance to the RSO is much greater than the focal length of the lens –the calculation of the unit vector or rotation angles can be determined.

As illustrated by Figure 17, the unit vector $\vec{\boldsymbol{u}}$ can be determined given the focal length of the lens and the position of the COB on the focal plane array. Alternatively, the position vector of RSO can be expressed in spherical coordinates using the radial distance of the COB and the rotation angles $\theta_i$ and $\varphi_i$ shown in Figure 17. The unit vector describing the relative position of the RSO to Prox-1 in BFF is defined in Eq. (14), which uses the rotation matrix from FPF to BFF.

$$\begin{bmatrix} \hat{X}_B \\ \hat{Y}_B \\ \hat{Z}_B \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \sin(\theta_i)\cos(\varphi_i) \\ \sin(\theta_i)\sin(\varphi_i) \\ \cos(\theta_i) \end{bmatrix} \tag{14}$$

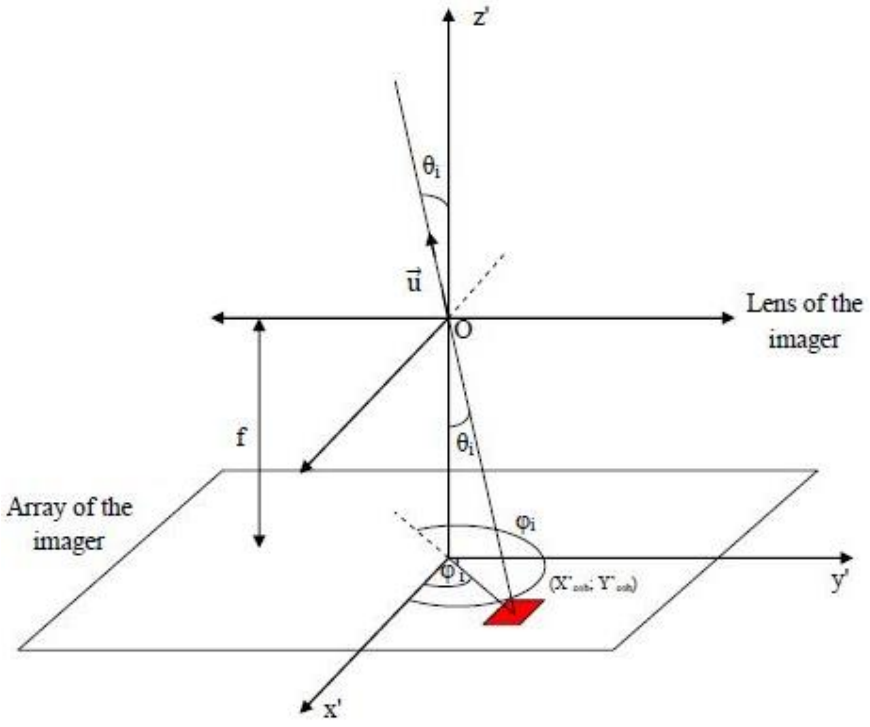


Figure 17: Geometric representation of the relative position vector in FPF

### 3.2.3. Range Estimation

Range estimation occurs only after the RSO has been successfully identified and the unit vector from the Prox-1 spacecraft to the RSO is determined. Range estimation can be broken down into four main steps:

1) Determination of the major and minor axes of the RSO
2) Determination of the ratio of the major axis to the minor axis
3) Estimation of the minimum and maximum apparent areas of the RSO.
4) Determination of the range its associated uncertainty

#### 3.2.3.1. Major and Minor Axis Determination

The major axis of the RSO is the line which maximizes the sum of the squares of the distance between itself and each pixel of the blob belonging to the RSO. Likewise, the minor axis of the RSO is the line which minimizes the sum of the distance between itself and the blob. By definition, these two axes intersect at the COB and are orthogonal to each other.

#### 3.2.3.2. Major-to-Minor Axis Ratio Determination

The ratio of the major-to-minor axis length *r* is determined by comparing the lengths of each axis. The length of each axis is determined by counting the number of pixels along the line defining the respective axis. This can be accomplished by rotating the image of the blob such that the major axis is vertical and the minor axis is horizontal. Since the two axes intersect at the COB, the column and row of the major and minor axis are known respectively. Furthermore, since the blob is a binary matrix populated by ones and zeros, the sum of the column and row containing the COB will yield the length of each axis respectively. However, this method assumes that the intensities for all other pixels not belonging to the RSO are zero.

#### 3.2.3.3. RSO Orientation Estimation

Given *a priori* knowledge of the RSO, a numerical approximation for the projected area as a function of the ratio of the major-to-minor axis length can be derived. For the case of a 3U CubeSat, Figure 18 illustrates the dataset from which the numerical approximations for the maximum and minimum area ratio, given by Eqs. (15) and (16) respectively, are derived. It is important to note that the area ratio is given with respect to $A_0$, the area of the smallest face of the RSO. For a 3U Cubesat, this corresponds to 100 cm$^2$.

$$\frac{A}{A_0}(r)_{max} = \begin{cases} 3.091 & if\ 0.25 \le r < 0.305 \\ 71716r^5 - 144219r^4 - 114743r^3 - 45172r^2 + 8812.5r - 679.42 & if\ 0.305 \le r < 0.495 \\ 2.4853r^4 - 23.737r^3 + 50.09r^2 - 42.372r + 14.977 & if\ r \ge 0.495 \end{cases} \tag{15}$$

$$\frac{A}{A_0}(r)_{min} = \begin{cases} 2.963 & if\ 0.25 \le r < 0.305 \\ 5.51r^4 - 20.70r^3 + 30.73r^2 - 22.02r + 7.36 & if\ r \ge 0.305 \end{cases} \tag{16}$$

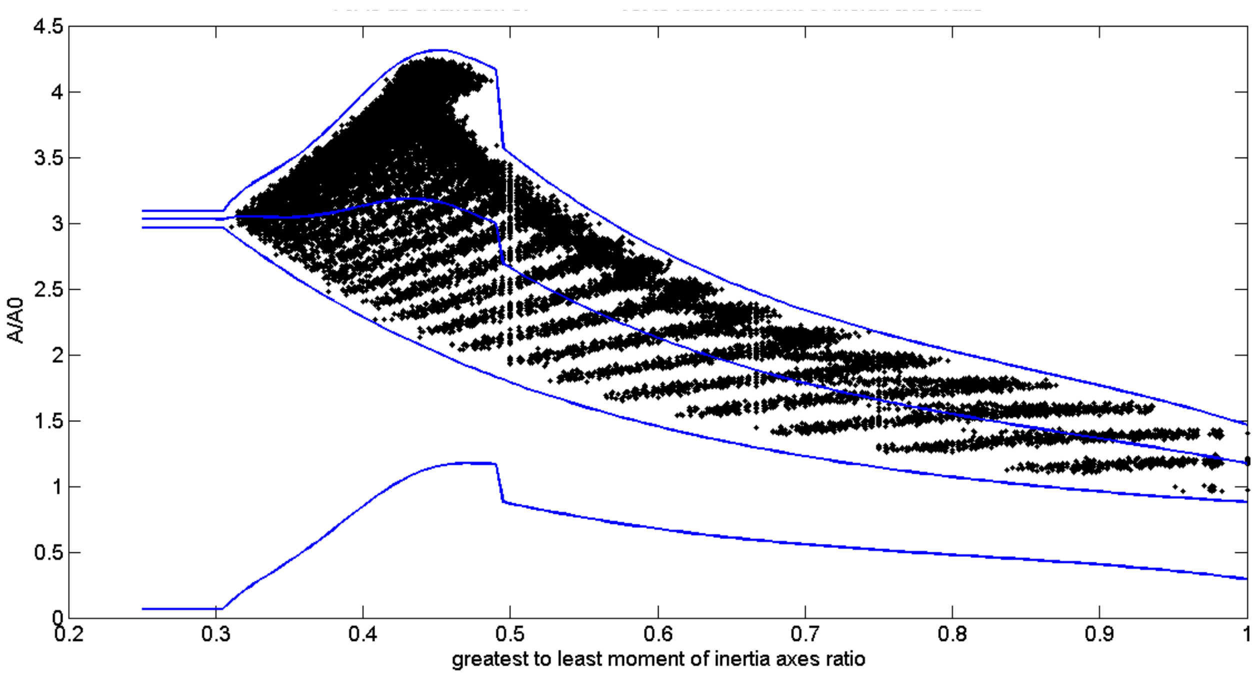


Figure 18: Projected Area as a Function of Axis Length Ratio

#### 3.2.3.4. Range Determination and Uncertainty

Given the focal length of the lens $f$, the area of each pixel $p^2$, the number of pixels in the blob $N$, the numerical approximations for the average range, $\rho$, is given by Eq. (17).

$$\rho = \frac{1}{2}\sqrt{\frac{A_0\left(\left(\frac{A}{A_0}\right)_{min}+\left(\frac{A}{A_0}\right)_{max}\right)}{Np^2}} \tag{17}$$

The uncertainty of the IPAs was shown to be primarily dependent upon the uncertainty in the area ratio function computed from the body axis ratio. The resulting estimated mean relative uncertainty is approximately 16% [10].

### 3.2.4. IPA Testing

#### 3.2.4.1. Image Generation for IPA Testing

To prepare for the implementation of closed loop simulation of the GN&C subsystem, a Simulink-based image generator was developed to mimic the input from the microbolometer into the IPAs. The image generator implementation was performed in two steps. The first step consisted of the implementation of a MATLAB function that can reliably generate a black and white image which accounts for location of the RSO within the FOV, inertial orientation of the RSO in space, and distance between Prox-1 and the RSO. The second step consisted of the integration of the image generator into Simulink so that the images being generated represent what the microbolometer would actually see in space, without consideration of Earth in the background or other possible satellites that the microbolometer might observe.

The MATLAB code that generates image matrices does this by representing the dimensions of the RSO and plotting them in a three-dimensional figure. Each of the individual faces is rotated with Euler angles provided by a 3-2-1 rotation. The corresponding coordinates are then translated by a vector distance given as an input, which represents the location of the RSO within the FOV. To account for the range, all physical dimensions are multiplied by a scaling factor. The scaling factor is calculated as $\mathrm{s} = \frac{20}{\mathrm{d}_1}$, where $\mathrm{d}_1 = 2\rho\tan(\frac{\alpha}{2})$ corresponds to the total amount of units being plotted for the aspect ratio of the image to be precise, and $\alpha$ is the angular aperture of the microbolometer's FOV. It is important to notice that the angle $\alpha$ corresponds to the aperture of the Y-axis in the FPF.

Next, the sides of the RSO are filled in. After rotating the view to an orientation that the imager will see, the aspect ratio is adjusted and the image is inverted (black to white and white to black) to create a representative image. It is possible to save a version of the image for reference, as well as to add the Earth as a background by simple inclusion of one more input.

The second part of the open loop implementation of the IPAs is the integration into Simulink. The various inputs to the Simulink image generator are: relative location of the RSO to Prox-1, orientation with respect to Prox-1, and the range from Prox-1. The resulting Simulink block diagram is illustrated in Figure 19.

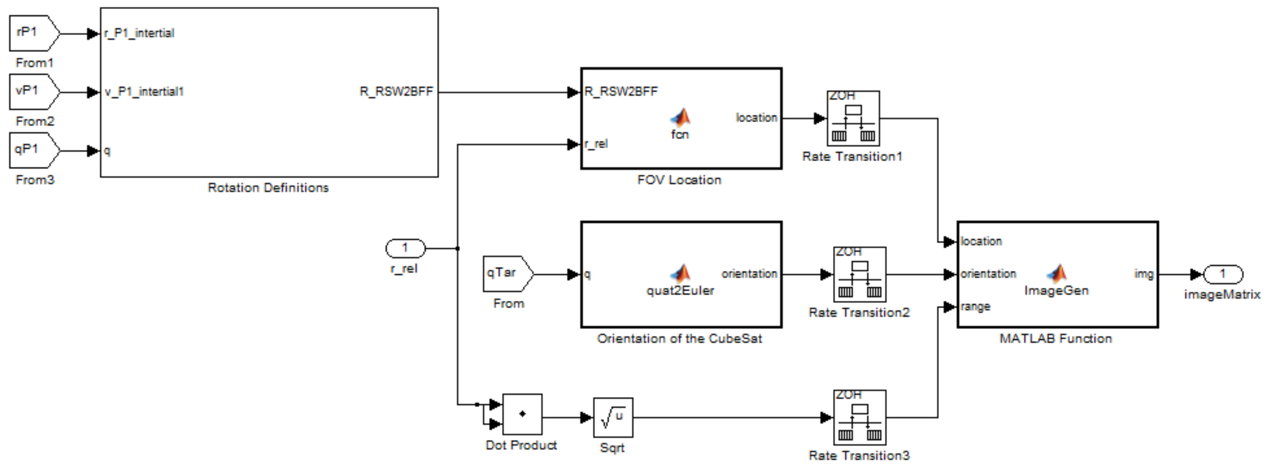


Figure 19: Simulink block diagram for image generator implementation

### 3.2.4.2. Location and Orientation Calculation

The location of the RSO within the FOV of the microbolometer is calculated by expressing the relative position vector from Prox-1 to the RSO in BFF and projecting it onto the y-axis of the BFF. Subtracting this projection from the relative position vector allows for the calculation of a vector projected onto the $\hat{X}_{BFF}\hat{Z}_{BFF}$ plane.

It is important to accurately describe the orientation of the RSO with respect to Prox-1 as the uncertainty of the IPAs is dependent on the area ratio calculation. The three Euler angles of interest are computed from the inertial quaternion of the RSO in the simulation. The Euler angles used to perform the 3-2-1 rotation are defined by Eq. (18) and the resulting rotation matrix to produce a 3-2-1 rotation is given by Eq. (19).

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix} atan2(\,2\cdot\ (q_0q_1+q_2q_3)\,, 1-2\cdot(q_1^2+q_2^2)\,) \\ \mathrm{asin}(\,2\cdot(q_0q_2-q_3q_1)\,) \\ atan2(\,2\cdot\ (q_0q_3+q_2q_1)\,, 1-2\cdot(q_3^2+q_2^2)\,)\end{bmatrix} \tag{18}$$

$$R_{3-2-1} = R(\phi)R(\theta)R(\psi) \tag{19}$$

### 3.2.4.3. Covariance Matrix Determination

The covariance of the IPAs was analyzed and aided in determination of the measurement noise of the RelOD filter. The inputs to the image generator were used as the truth data set. The covariance analysis contained a permutation of the three Euler angles for the orientation of the

RSO, where $\alpha_i \in \{0°, 45°, 90°\}$, as well as the location of the RSO in the FOV of the simulated imager. From this analysis, the standard deviation of the range error over all the orientations was obtained and is listed in Table 1 and illustrated graphically in Figure 20. Furthermore, Figure 21 illustrates the ratio of error-to-range as a function of range.

Table 1: Standard deviation of the error in range measurement as a function of range

| *Range (m)* | *1-σ Range Error (m)* |
|---|---|
| 40 | 4.123 |
| 60 | 7.526 |
| 80 | 10.462 |
| 100 | 12.783 |
| 120 | 13.553 |
| 140 | 16.027 |

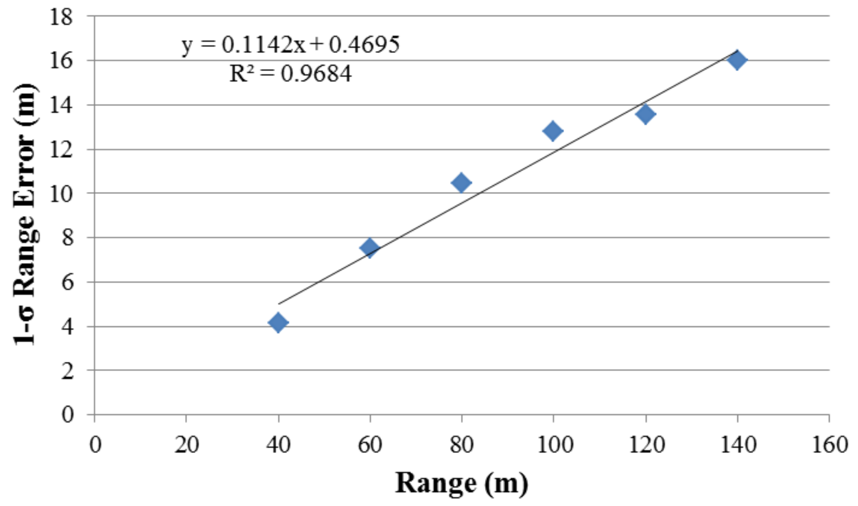


Figure 20: Standard deviation of the error in range measurements as a function of range

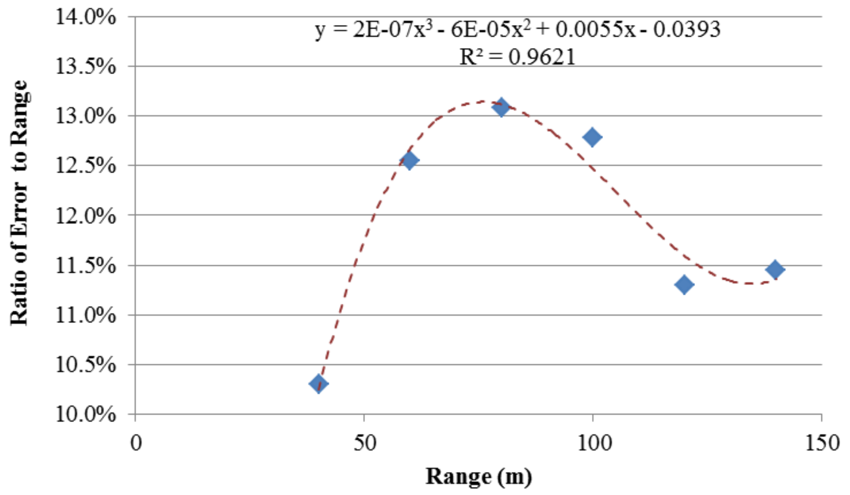


Figure 21: Ratio of (Range Error / Range) as a function of range

#### 3.2.4.4. IPA Boundary Analysis

In order to study the boundaries of the IPAs and their capability of detecting the RSO location under different orientations, three specific orientations were selected as illustrated by Figure 22.

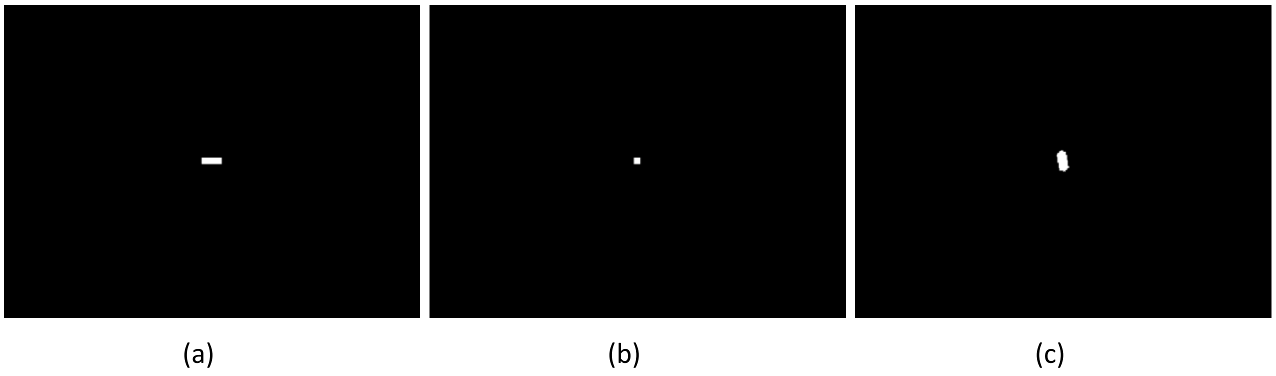


Figure 22: RSO orientations studied for image processing algorithm reliability. (a) Euler angles [$\phi,\theta,\psi$]=[0°,0°,0°]. (b) Euler angles [$\phi,\theta,\psi$]=[0°,0°,90°]. (c) Euler angles [$\phi,\theta,\psi$]=[45°,45°,45°].

These orientations expose the IPAs to both the minimum and maximum visible areas of the RSO. A simulation was created where every orientation was reproduced 24,400 different times for ranges between 30 and 150 m, in steps of 2 m, for 20 locations on each of the axes. The intent was to span the entire FOV of the simulated imager. From the output results, three metrics were used in determining whether a measurement was successful: (1) successfully found the RSO (minimum number of pixels found), (2) correctly determined the aspect ratio to within a small error of the truth, and (3) error in range is less than the fitted error given as $0.1142x + 0.4695$ where *x* is the range measurement.

Figure 23 illustrates the number of sample cases for which the IPAs failed to provide an accurate estimate of the location of the RSO as a function of true range in meters. As expected, the orientation $\{0°, 0°, 0°\}$ provided the highest reliability with 85% success. This is the case for which the IPAs are most finely tuned, since the CubeSat is represented by its exact aspect ratio. On the other hand, the cases for $\{45°, 45°, 45°\}$ and $\{0°, 0°, 90°\}$ provided a low success rate. The success rate for these was 32.75% and 25.26% respectively. The failure distribution of the $\{45°, 45°, 45°\}$ test case is a bimodal distribution with a minimum occurring between the ranges of 100m and 110m.

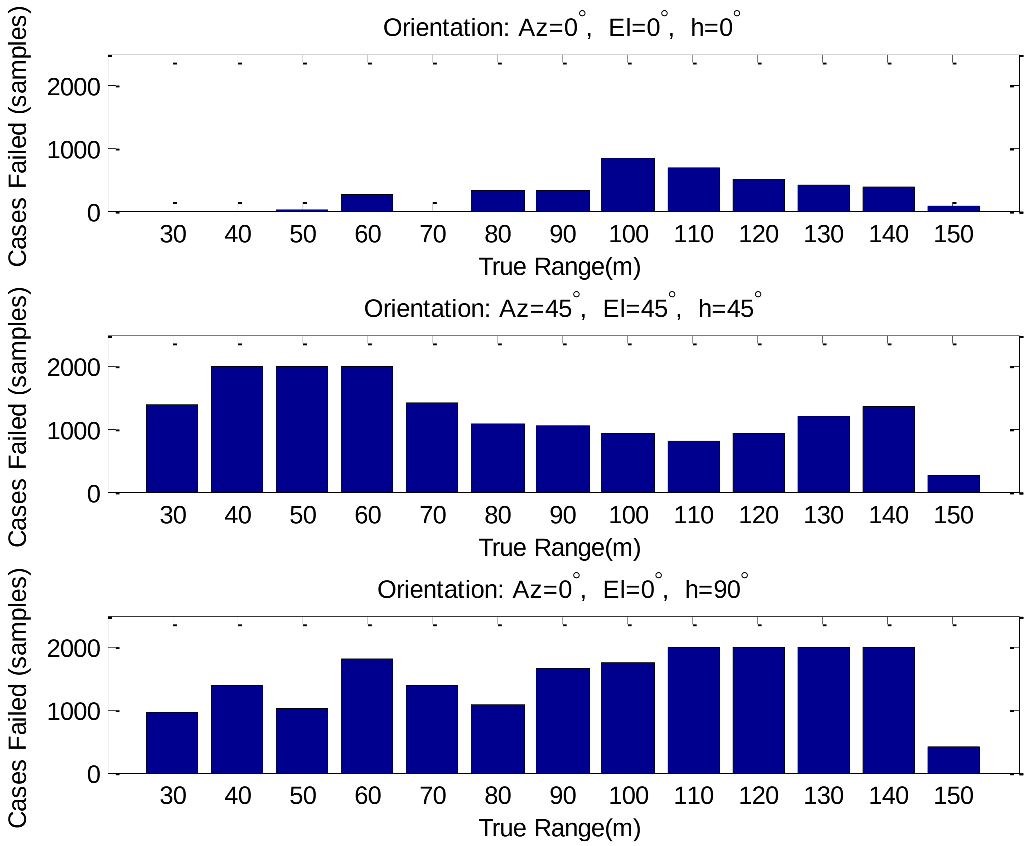


Figure 23: Number of cases for which the IPAs failed to approximate range to RSO as a function of range for different orientations

## 3.3. *Relative Orbit Determination Filter*

The relative orbit determination (RelOD) Kalman filter takes BFF unit vectors and range estimates from the IPAs and generates RSW relative position and velocity estimates. The following description was originally presented by Sean Chait [3] and Richard Zappulla [2].

### 3.3.1. State Prediction

The general model for the state transition matrix (STM) associated with the RelOD filter is derived from the Closhessy Wiltshire equations. The model has a homogenous term (Φ), which holds regardless of the forces acting on the spacecraft, and a particular solution (Ψ), which is activated only when the spacecraft executes a thrust force $\underline{f}$ and is constant only when the thrust is of a constant magnitude. Assuming a constant thrust level for a given maneuver, the spacecraft's motion can be modeled using a state transition matrix without requiring integration, as shown in Eq. (20) where $\underline{X}$ is the RSW state vector $[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ ]^T$, $t$ is the current time, and $t_0$ is the initial time.

$$\underline{X}(t-t_0) = \underline{X_h}(t-t_0) + \underline{X_p}(t-t_0) = \mathbf{\Phi}(t-t_0)\underline{X}(t_0) + \mathbf{\Psi}(t-t_0)\underline{f}(t) \tag{20}$$

However, imaging will not occur during thruster firing. Therefore, for the purpose of the Kalman filter the particular solution can be ignored, although it will be reexamined for use in the state propagator. This reduces our state and covariance estimation step to Eqs. (21) and (22) where *k* is the current timestep, k+1 is the next timestep, ***P*** is the covariance estimate, the STM **Φ** is defined by Eq. (23) and *n* is the mean motion of the orbit.

$$\underline{\mathbf{X}}_{\mathbf{k+1}} = \mathbf{\Phi}(t_k - t_0)\boldsymbol{X}_k \tag{21}$$

$$\underline{\boldsymbol{P}}_{k+1} = \mathbf{\Phi}(t_k - t_0)\boldsymbol{P_k}\mathbf{\Phi}(t_k - t_0)' + \mathbf{Q}(\mathrm{k}) \tag{22}$$

$$\mathbf{\Phi}(t) = \begin{bmatrix} 4-3\cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} & \frac{2-2\cos(nt)}{n} & 0 \\ 6(\sin(nt)-nt) & 1 & 0 & \frac{2\cos(nt)-2}{n} & \frac{4\sin(nt)-3nt}{n} & 0 \\ 0 & 0 & \cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} \\ 3n\sin(nt) & 0 & 0 & \cos(nt) & 2\sin(nt) & 0 \\ 6n(\cos(nt)-1) & 0 & 0 & -2\sin(nt) & -3+4\cos(nt) & 0 \\ 0 & 0 & -n\sin(nt) & 0 & 0 & \cos(nt) \end{bmatrix} \tag{23}$$

In Eq. (22) ***Q*** is the process noise, which represents all possible perturbing forces acting on the spacecraft such as solar radiation pressure and J2 Earth oblateness effects, that are not accounted for in the STM. Process noise is defined by Eq. (24), where $\underline{\boldsymbol{G}}$ is given by Eq. (25). Given a small timestep that is much less than the orbital period, the definition reduces to Eq. (26).

$$Q_{\mathrm{k}} = \int_{t_0}^{t_k} \underline{\mathbf{\Phi}}(t_k, t_0)\underline{\boldsymbol{G}}Q\underline{\boldsymbol{G}}^T\underline{\mathbf{\Phi}}(t_k, t_0)^T d\tau \tag{24}$$

$$\underline{\boldsymbol{G}} = \begin{bmatrix} \underline{\mathbf{0}}_{3x3} & \underline{\mathbf{0}}_{3x3} \\ \underline{\mathbf{0}}_{3x3} & \underline{\boldsymbol{I}}_{3x3} \end{bmatrix} \tag{25}$$

$$Q(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{xrsw}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{yrsw}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{zrsw}^2 \end{bmatrix} \Delta t \tag{26}$$

σ ~ N(0,1e-6) is a zero-mean normal distribution with a variance of 1e-6 which is a function of the expected magnitude of perturbations not accounted for by the STM. Process noise is thus constant for the purpose of this algorithm and is only dependent on the sample rate at which the imager is currently operating, therefore allowing for its prior calculation and storage.

### 3.3.2. Prediction Update

The predicted state and covariance estimates are updated through use of the measurement residual $\tilde{\boldsymbol{y}}_{\boldsymbol{k+1}}$ and the updated Kalman gain $\boldsymbol{K}_{k+1}$. The measurement residual is calculated using

Eq. (27) where the actual measurement is $\boldsymbol{z_k}$ and the predicted measurement is determined by the measurement model $\mathbf{G}(\mathbf{X}(\mathbf{t_{k+1}}))$. Kalman gain is then updated in Eq. (28) via the mapping matrix $\boldsymbol{H}_{\boldsymbol{k+1}}$ and the measurement noise $\boldsymbol{R}_k$. The updated state and covariance estimates are then calculated using Eqs. (29) and (30).

$$\tilde{\boldsymbol{y}}_{\boldsymbol{k+1}} = \boldsymbol{z_k} - \mathbf{G}(\mathbf{X}(\mathbf{t_{k+1}})) \tag{27}$$

$$\boldsymbol{K}_{k+1} = \underline{\boldsymbol{P}}_{k+1}\boldsymbol{H}_{k+1}^{\boldsymbol{T}}\left(\boldsymbol{H}_{k+1}\underline{\boldsymbol{P}}_{k+1}\boldsymbol{H}_{k+1}^{\boldsymbol{T}} + \boldsymbol{R}_k\right)^{-\mathbf{1}} \tag{28}$$

$$\boldsymbol{X}_{\boldsymbol{k+1}} = \underline{\mathbf{X}}_{\mathbf{k+1}} + \boldsymbol{K}_{k+1}\tilde{\boldsymbol{y}}_{\boldsymbol{k+1}} \tag{29}$$

$$\boldsymbol{P}_{\boldsymbol{k+1}} = (\boldsymbol{I} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{k+1})\underline{\boldsymbol{P}}_{k+1}(\boldsymbol{I} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{k+1})^T + \boldsymbol{K}_{k+1}\boldsymbol{R}_k\boldsymbol{K}_{k+1}{}^{\boldsymbol{T}} \tag{30}$$

By definition, a regular Kalman filter acts on a linear system and has a simplified measurement model and subsequent mapping matrix. Since the IPAs ultimately provide a relative position vector measurement in the RSW frame (transformed from BFF based on the current inertial position and attitude of Prox-1), the predicted measurement simply becomes the relative position vector as determined by the current state estimate without any transformation. The measurement model and mapping matrix therefore are given by Eqs. (31) and (32) respectively.

$$\mathbf{G}\big(\mathbf{X}(\mathbf{t_{k+1}})\big) = \underline{\mathbf{X}}_{\mathbf{k+1}}(1{:}\,3) \tag{31}$$

$$\boldsymbol{H}_{\boldsymbol{k+1}} = [I_{3x3} \quad 0_{3x3}] \tag{32}$$

The measurement noise matrix, $\boldsymbol{R}$, is derived from the expected noise of the measurement produced by the IPAs. Analysis of the IPAs has yielded that the expected error in the algorithms is not a constant value but rather a function of range. Thus, the expected measurement noise is updated dynamically using Eq. (33), based upon the range of the current predicted measurement where *c* is a tunable scaling parameter and $\sigma_i$ is given by Eq. (34) where $e_i$ is the normal distribution N(0,1).

$$\boldsymbol{R}^{\boldsymbol{Sensor}} = \begin{bmatrix} \sigma_{x,BFF}^2 & c\sigma_{x,BFF}\sigma_{y,BFF} & c\sigma_{x,BFF}\sigma_{z,BFF} \\ c\sigma_{x,BFF}\sigma_{y,BFF} & \sigma_{y,BFF}^2 & c\sigma_{y,BFF}\sigma_{z,BFF} \\ c\sigma_{x,BFF}\sigma_{z,BFF} & c\sigma_{y,BFF}\sigma_{z,BFF} & \sigma_{z,BFF}^2 \end{bmatrix} \tag{33}$$

$$\sigma_i = \left|\underline{\mathrm{X}}_{\mathrm{k+1}}(1{:}\,3)\right|_{BFF} e_i \tag{34}$$

### 3.3.3. State Estimate Propagator

During operations, constant relative position measurements will not be provided to the relative orbit determination filter for two reasons. First, image processing and data storage is computationally expensive; therefore there will be a finite sample rate (on the order of seconds) between each image to account for the limitations imposed by the Prox-1 flight computer. Second, thrusting maneuvers will require slews that move LightSail out of the microbolometer's

FOV. Although no relative position measurements are acquired during this period, Prox-1's GN&C systems still require a relative state estimate for maneuver planning and execution. It is for this reason that a state estimate propagator must be introduced into Prox-1's navigation system to continuously provide relative state estimates between periods of measurement.

The State Estimate Propagator will consist of two components: the steady state propagator and the force propagator. The steady state propagator will be used in all states where no measurement is being taken, which include but are not limited to: time period between measurements, slew periods, and periods where LightSail is temporarily lost or out of range. The force propagator will be used to update the state estimate when a known a thrust maneuver is applied. The updated state will aid in faster convergence of the Kalman filter once the measurement period begins again. The propagated estimate will also aid in reacquisition of LightSail after slew maneuvers during which the relative position of the target will have changed. Prox-1 tracking controllers will therefore be configured to slew to the expected location of LightSail and not the previous location, minimizing unnecessary search periods.

### 3.3.4. RelOD Test Scenario

In this scenario, a natural motion circumnavigation (NMC) maneuver is simulated with an initial position of -75m in the along-track direction and an initial radial velocity. The orbit is designed such that there is no motion in the cross-track direction. The Kalman filter is given an incorrect initial position with an incorrect along-track component as well as small cross-track and radial components. The initial state vector given to the Kalman filter also has incorrect radial and cross-track velocity components. The simulation was then propagated for one orbit (~90 minutes).The unfiltered range estimates are shown in Figure 24 and the filtered relative position and velocity estimates are shown in Figure 25.

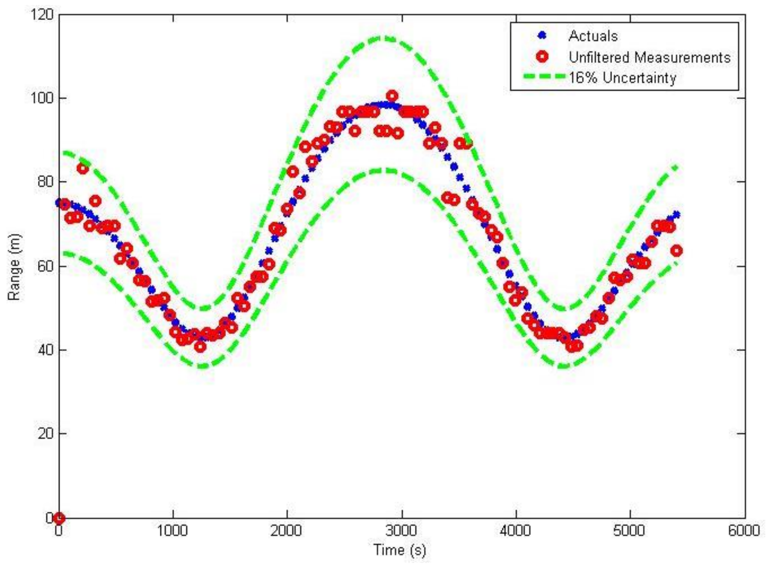


Figure 24: Unfiltered range measurements from IPAs for sample scenario

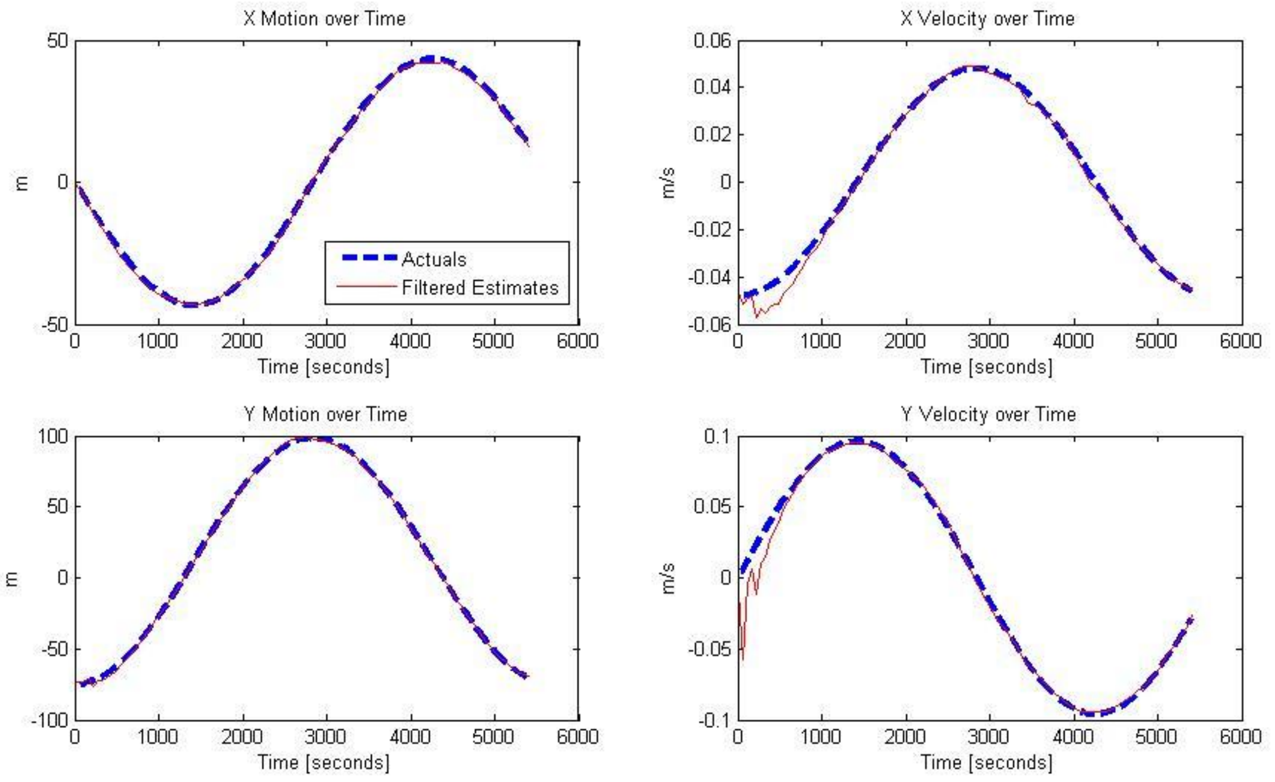


Figure 25: Relative navigation filter results

Figure 25 shows that the filter rapidly converges with minimal steady state error. The magnitude of the relative position vector error is within ±5 meters and the relative velocity error is within 3-sigma bounds of ± 0.02 m/s. The filter is able to do this even though the relative orientation and range of LightSail is constantly changing, which results in variability of the accuracy of the IPAs. Further refinement of the RelOD system will allow for higher fidelity state estimates.

## 4. Guidance Component Details

Several guidance strategies are employed for the Prox-1 mission using Artificial Potential Functions (APFs) and Relative Orbital Elements (ROEs). Currently two formulations of the guidance algorithms have been developed: one using solely APFs and another using solely ROEs. Another strategy is in development using ROEs for maneuver planning with APFs included for collision avoidance [7].

### 4.1. *APF Guidance Algorithms*

The following description of the APF-only guidance strategy is based on the concept described by Martinson and Munoz [12] and was originally presented by Sean Chait [3]. The decision to use APFs as the primary station keeping and collision avoidance mechanism stems from two factors. First, APFs consist solely of arithmetic operators and are therefore very computationally inexpensive, a favorable condition for a nanosatellite mission with limited computing capabilities. Secondly, the use of APFs as attractors and repellers, similar to the concept of sources and sinks often used for fluid dynamics analysis, aids in guaranteeing that a global minimum is created.

The current formulation creates a global minimum via an attractive potential at the designated goal location and a series of repulsive potentials to generate keep-out zones (KOZ) and obstacles analogous to the boundary conditions of the guidance scenario. The attractive potential $\underline{\boldsymbol{\phi}}_A$ is defined in Eq. (35) where $(\boldsymbol{r} - \boldsymbol{r}^*)$ is the relative position of the Prox-1 to the goal position in the RSW frame, $\underline{\boldsymbol{Q}}_{\boldsymbol{A}} \epsilon\, \mathbb{R}^{3x3}$ is a positive definite shaping matrix, and $\kappa_A \epsilon\, \mathbb{R}^1$ is a small positive gain. It is important to note that this goal position is *not* synonymous with the target spacecraft location; rather it is the location *relative to the target* that we wish Prox-1 to achieve.

$$\underline{\boldsymbol{\phi}}_A = \frac{\kappa_A}{2}(\boldsymbol{r} - \boldsymbol{r}^*)^{\boldsymbol{T}} \underline{\boldsymbol{Q}}_{\boldsymbol{A}} (\boldsymbol{r} - \boldsymbol{r}^*) \tag{35}$$

The repulsive potential $\underline{\boldsymbol{\phi}}_R$ is defined by Eq. (36) where $(\boldsymbol{r} - \boldsymbol{r}^{\boldsymbol{\Delta}})$ is the relative position of the chaser to the target spacecraft, $\kappa_R \epsilon\, \mathbb{R}^1$ is a small positive gain, and $\underline{\boldsymbol{P}}_{\boldsymbol{A}} \epsilon\, \mathbb{R}^{3x3}$ is a positive definite shaping matrix for the repulsive potential. This shaping matrix is used to encapsulate the KOZ of the desired trajectory. This is mathematically implemented via a sudden increase in the repulsive potential at the border of the designated KOZ, making it impossible for the spacecraft to ever reach this boundary condition.

$$\underline{\boldsymbol{\phi}}_R = \frac{\kappa_R}{2} \frac{(\boldsymbol{r} - \boldsymbol{r}^*)^{\boldsymbol{T}} \underline{\boldsymbol{Q}}_{\boldsymbol{A}} (\boldsymbol{r} - \boldsymbol{r}^*)}{(\boldsymbol{r} - \boldsymbol{r}^{\boldsymbol{\Delta}})^{T} \underline{\boldsymbol{P}}_{\boldsymbol{A}} (\boldsymbol{r} - \boldsymbol{r}^{\boldsymbol{\Delta}}) - 1} \tag{36}$$

In this mission, a KOZ with a radius of 25 meters will always be defined around LightSail so as to mitigate the chance of re-contact with the target spacecraft. Throughout the entirety of the mission, the relative position of LightSail will be monitored and this repulsive potential will be enacted should it be determined that we are currently moving dangerously close to the target. The total potential function $\underline{\boldsymbol{\phi}}$ is defined in Eq. (37) as the superposition of the attractors and repellers, which in the case of Prox-1 consists of only two potentials: an attractive potential guiding the spacecraft to the target location and a repulsive potential used for collision avoidance.

$$\underline{\boldsymbol{\phi}} = \underline{\boldsymbol{\phi}}_A + \underline{\boldsymbol{\phi}}_R \tag{37}$$

To ensure that a solution is achieved, the potential is treated as a Lyapunov candidate function [12], signaling the following criteria in Eq. (38) must be met. By enforcing these requirements, ultimately Eq. (39) must be satisfied, where $\boldsymbol{V}_D \epsilon\, \mathbb{R}^3$ is the current desired velocity along the total potential in the RSW frame and the basis for the guidance algorithm.

$$\begin{aligned} &1.\ \underline{\boldsymbol{\phi}}(\boldsymbol{r},t) > 0 \qquad \forall \qquad \boldsymbol{r} \neq 0 \\ &2.\ \underline{\boldsymbol{\phi}}(\boldsymbol{r},t) = 0 \qquad for \qquad \boldsymbol{r} = 0 \\ &3.\ \dot{\underline{\boldsymbol{\phi}}}(\boldsymbol{r},\dot{\boldsymbol{r}},t) < 0 \qquad \forall \qquad \boldsymbol{r},\dot{\boldsymbol{r}} \rightarrow 0 \end{aligned} \tag{38}$$

$$-\nabla\underline{\boldsymbol{\phi}} = \boldsymbol{V}_D \tag{39}$$

When examining a rest-to-rest maneuver, two goals are desired: first that $(\boldsymbol{r} - \boldsymbol{r}^*)$ converges to zero and second that the final relative velocity is zero. However this simple formulation poses potential issues for the system defined above. As shown in Eqs. (35) and (36), these potential formulations are a function of relative position only and thus present the problem that unless the desired goal position is an inherently stable position as shown by the Clohessy Wiltshire equations, a state of constant thrust will be required to maintain station-keeping. Although this may be possible in a larger spacecraft, this formulation is not useful in the Prox-1 mission architecture and would be difficult to implement using a single thruster. Fortunately, these issues can be mitigated through the development of an appropriate guidance strategy.

It can be shown that a stationary relative orbit, without the need for constant station-keeping maneuvers, can be achieved if a purely along-track goal position is targeted. As such, this algorithm will only be used to target stationary, along-track goal positions. To further mitigate collision risk, goal positions in the half-plane opposite the current chaser position, will not be allowed in this strategy. When driving to a goal location, successful execution will be achieved when the desired location is achieved and relative velocity is driven to zero within predetermined acceptable error bounds. The relative state will therefore be continuously monitored for deviation from these desired bounds and will trigger the potentials when undesirable drift is detected.

A second possible issue can be seen by examining Eqs. (35) and (36). As the potentials are solely functions of relative position, a solution will be calculated in each iteration and thus dictates a state of continuous thrust. Although this would result in rapid convergence and may be desirable for a highly maneuverable vehicle, continuous thrust in a small vehicle will quickly diminish the craft's fuel reserves. Also, it is desirable to update the current relative state estimate and would not be possible if the spacecraft's imagers are oriented away from the target for a long, continuous period of time. From these considerations, further rules governing the guidance algorithms were developed. First, a finite time to allow for successful completion of the burn maneuver and re-acquisition of the target is allotted. After imaging resumes, the RelOD filter will obtain an updated solution. Upon successful convergence, the guidance algorithms will be triggered again and another burn implemented. To prevent excessive burning, a minimum wait period (which includes the burn and state calculation period) has been implemented in cases where only small maneuvers are required and the filters rapidly converge. The final guidance law for rest-to-rest maneuvers can be defined using the following steps, where $\boldsymbol{r}^*$ is the goal position in the RSW frame and $\boldsymbol{r}$ is the relative position between Prox-1 and LightSail in the RSW frame:

1. Check target position criteria
$$\boldsymbol{r}^*(1) = 0$$
$$\boldsymbol{r}^*(2) \neq 0$$
$$\boldsymbol{r}^*(3) = 0$$
$$sign(\boldsymbol{r}^*(2)) = sign(\boldsymbol{r}(2))$$
2. Compute Desired Velocity: $-\nabla\underline{\boldsymbol{\phi}} = \boldsymbol{V}_D$
3. Compute Error Velocity and Burn: $\boldsymbol{V}_{cmd} = \boldsymbol{V}_{chaser} - \boldsymbol{V}_D$
4. Slew to thrust orientation (using STC) and perform burn
5. Re-acquire LightSail via propagated relative state solution (using TAC)
6. Determine relative state solution convergence (using RelOD filter)
7. If $|\boldsymbol{r} - \boldsymbol{r}^*| > error_{allowable}$, recalculate burn
8. If $t_{wait} > t_{min}$ , execute maneuver

This strategy is demonstrated by the following test scenario. In this scenario, a rest-to-rest maneuver is examined. Initially, the chaser is in a 120 m stable leading orbit with no relative velocity. A goal position with a trailing orbit of 50 m is desired with a KOZ of 25 m around the target vehicle. The goal position and velocity is achieved within tolerances in only 1,200 seconds at an expense of 8 grams of fuel (using the hydrazine thruster design that has since been changed to a cold gas thruster). To examine the station-keeping ability of the algorithm, the scenario is propagated for approximately 90 minutes. For the remaining 4,200 seconds of the orbit, only 2 grams of fuel (again hydrazine) are required, which is well within acceptable mission bounds. The resulting relative position and velocity plots are shown in Figure 26.

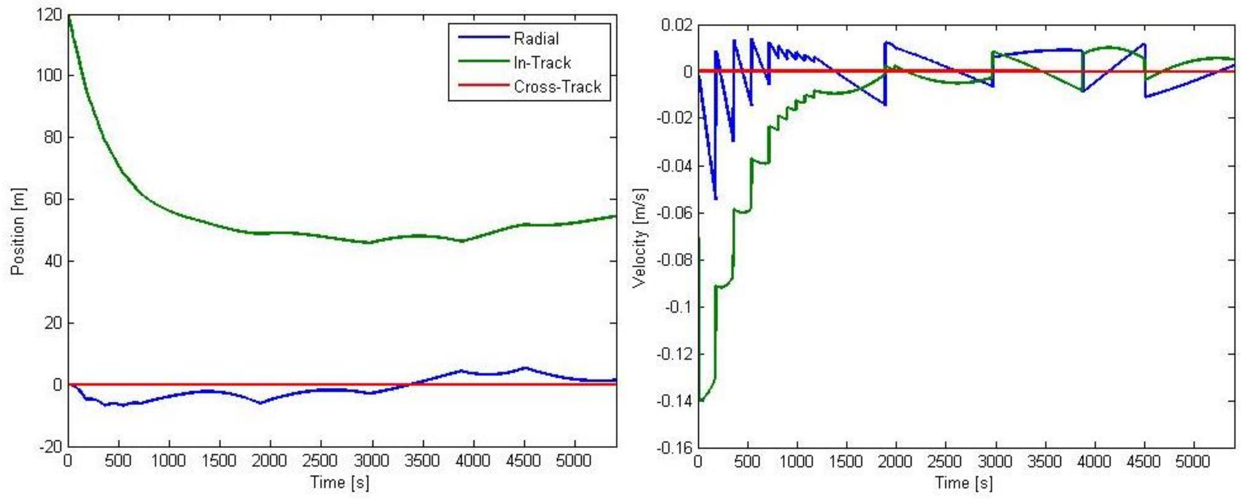


Figure 26: APF Guidance Algorithm Rest-to-Rest and Station Keeping Test Scenario

## 4.2. *ROE Guidance Algorithms*

The Relative Orbital Element (ROE) guidance formulations have been developed by the Prox-1 principal investigator Professor David Spencer, with the assistance of Thomas Lovell from AFRL. The discussion in this section is based on material presented by Lovell and Spencer [13].

### 4.2.1. Relative Orbital Elements

ROEs provide a convenient method to describe a relative orbit about a target spacecraft with a circular orbit. Similar to classical Keplerian orbit elements, ROEs provide six parameters to completely describe the orbit of a chaser spacecraft (i.e. Prox-1) with respect to a target spacecraft (i.e. LightSail). The six ROEs are the instantaneous center of the relative ellipse ($x_d$, $y_d$), the semi-major axis of the relative ellipse $a_r$, the relative eccentric anomaly $E_r$, the amplitude of the z-motion $A_z$, and the phase angle along the z-direction $\psi$, defined in Eqs. (40)-(45) where $x_0$, $y_0$, $z_0$, $\dot{x}_0$, $\dot{y}_0$, and $\dot{z}_0$ are the initial relative states of the chaser in the RSW frame, $t_0$ is the initial time, $t$ is the current time, $n$ is the mean motion of the target's orbit.

$$x_d = 4x_0 + \frac{2\dot{y}_0}{n} \tag{40}$$

$$y_d = -\left(6nx_0 + 3\dot{y}_0\right)\left(t - t_0\right) - \frac{2\dot{x}_0}{n} + y_0 \tag{41}$$

$$a_r = \sqrt{\left(6x_0 + \frac{4\dot{y}_0}{n}\right)^2 + \left(\frac{2\dot{x}_0}{n}\right)^2} \tag{42}$$

$$E_r = \operatorname{atan2}\left(\dot{x}_0, 3nx_0 + 2\dot{y}_0\right) + n\left(t - t_0\right) \tag{43}$$

$$A_z = \sqrt{z_0^2 + \left(\frac{\dot{z}_0}{n}\right)^2} \tag{44}$$

$$\psi = \operatorname{atan2}\left(nz_0, \dot{z}_0\right) + n\left(t - t_0\right) \tag{45}$$

The geometric meanings of the ROEs $x_d$, $y_d$, $a_r$, and $E_r$ are illustrated in Figure 27, where D(x,y) is the position of the deputy (chaser) spacecraft along the relative ellipse, P is the periapsis point of the relative ellipse, and Q($x_q$,$y_q$) is the position of the chaser spacecraft on a circumscribed circle about the relative ellipse.

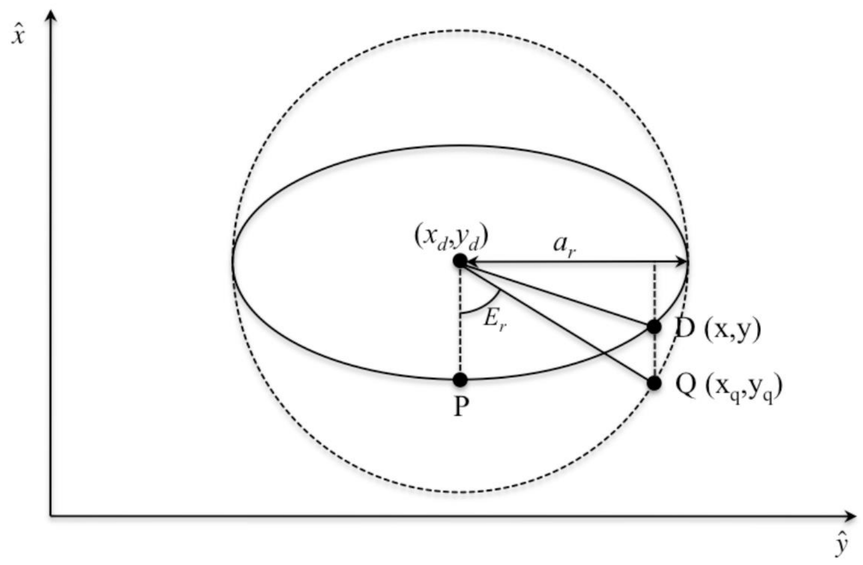


Figure 27: Relative orbit geometry in the RSW x-y frame [13]

The geometric meanings of the ROEs $A_z$ and $\psi$ are illustrated in Figure 28, where the deputy (chaser) spacecraft is located at point $D_0$ at time $t_0$ and point D at time $t$. $F_0$ and F are the initial and current positions along a circle of radius $A_z$ and $G_0$ is located at the point where a line parallel to the z-axis passing through $F_0$ intersects the x-axis.

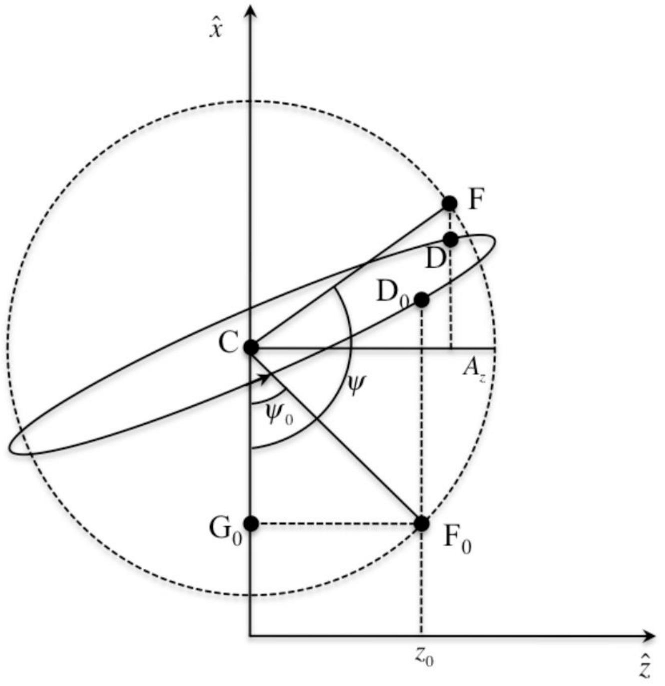


Figure 28: Z-motion phase angle geometry, projected onto the x-z plane [13]

### 4.2.2. Station-Keeping in a Leading or Trailing Orbit

To perform station-keeping and rest-to-rest maneuvers using ROEs, a series of impulsive thrust maneuvers is executed by Prox-1 to achieve a desired $y_d$ location $y_{tgt}$ either leading or trailing LightSail with $x_d = 0$, $a_r = 0$, and $A_z = 0$. Table 2 shows these four maneuvers, where the superscript "-" indicates ROEs before the specified maneuver and "+" indicates desired ROEs after the maneuver, $\Delta V$ is the impulse change in velocity to be applied in m/s. It is interesting to note that maneuver 2 results in a secular drift in the $y$-axis that reaches the $y_{tgt}$ location exactly $s$ orbits after the maneuver is executed.

Table 2: Station-keeping maneuver sequence summary [13]

| Maneuver # | Maneuver Location | $\Delta\bar{V}$ Components | Purpose |
|---|---|---|---|
| 1 | Any | $\Delta V_y = -\frac{n}{2} x_{d_0}$ | Stop secular drift in $\hat{y}$. |
| 2 | $x = 0$ | $\Delta V_y = \frac{n\left(y_d^- + a_r^- - y_{tgt}\right)}{6\pi s}$ | Initiate secular drift in $\hat{y}$ such that $s$ orbits after maneuver, orbit crosses $x = 0, y = y_{tgt}$. |
| 3 | $x = 0, y = y_{tgt}$ | $\Delta V_x = \frac{n}{2}\left(y_{d_0} - y_d^+\right) - \frac{3}{2} n x_{d_0}\left(t_b - t_0\right)$ <br> $\Delta V_y = -\frac{n}{2} x_{d_0}$ | Fix motion in $\hat{x} - \hat{y}$ plane at $x = 0, y = y_{tgt}$. |
| 4 | $z = 0$ | $\Delta V_z = -nA_z \cos\psi^-$ | Zero-out $\hat{z}$-motion. |

To illustrate the ROE station-keeping strategy, an example scenario is presented using the initial and target conditions shown in Table 3, where the targeted station-keeping position is a leading orbit 100 m ahead of LightSail in the y-direction. The initial conditions, if left uncontrolled, would result in the spacecraft drifting in the –y direction. The maneuvers performed are summarized in Table 4 and the trajectories are shown in Figure 29. The maneuver successfully places Prox-1 into a fixed position with respect to LightSail.

Table 3: Station-keeping example [13]

| | |
|---|---|
| Initial Conditions | $t_0 = 0$<br>$x_{d_0} = 3$ m<br>$y_{d_0} = 100$ m<br>$a_{r_0} = 2$ m<br>$E_{r_0} = 0$<br>$A_{z_0} = 1$ m<br>$\psi_0 = \frac{\pi}{2}$ rad |
| Target Conditions | $x_d = 0$<br>$y_d = y_{d_{sp}} = 100$ m<br>$a_r = 0$<br>$A_z = 0$ |

Table 4: Station-keeping example maneuver summary [13]

| Maneuver | Time from Initial State (s) | $\Delta V_x$ (m/s) | $\Delta V_y$ (m/s) | $\Delta V_z$ (m/s) |
|---|---|---|---|---|
| 1 | 5,676.981 | 0 | -1.6602E-03 | 0 |
| 2 | 15,611.699 | 0 | -3.5632E-04 | 0 |
| 3 | 38,319.624 | -2.2136E-03 | 3.5633E-04 | 0 |
| 4 | 41,158.115 | 0 | 0 | 1.1068E-03 |

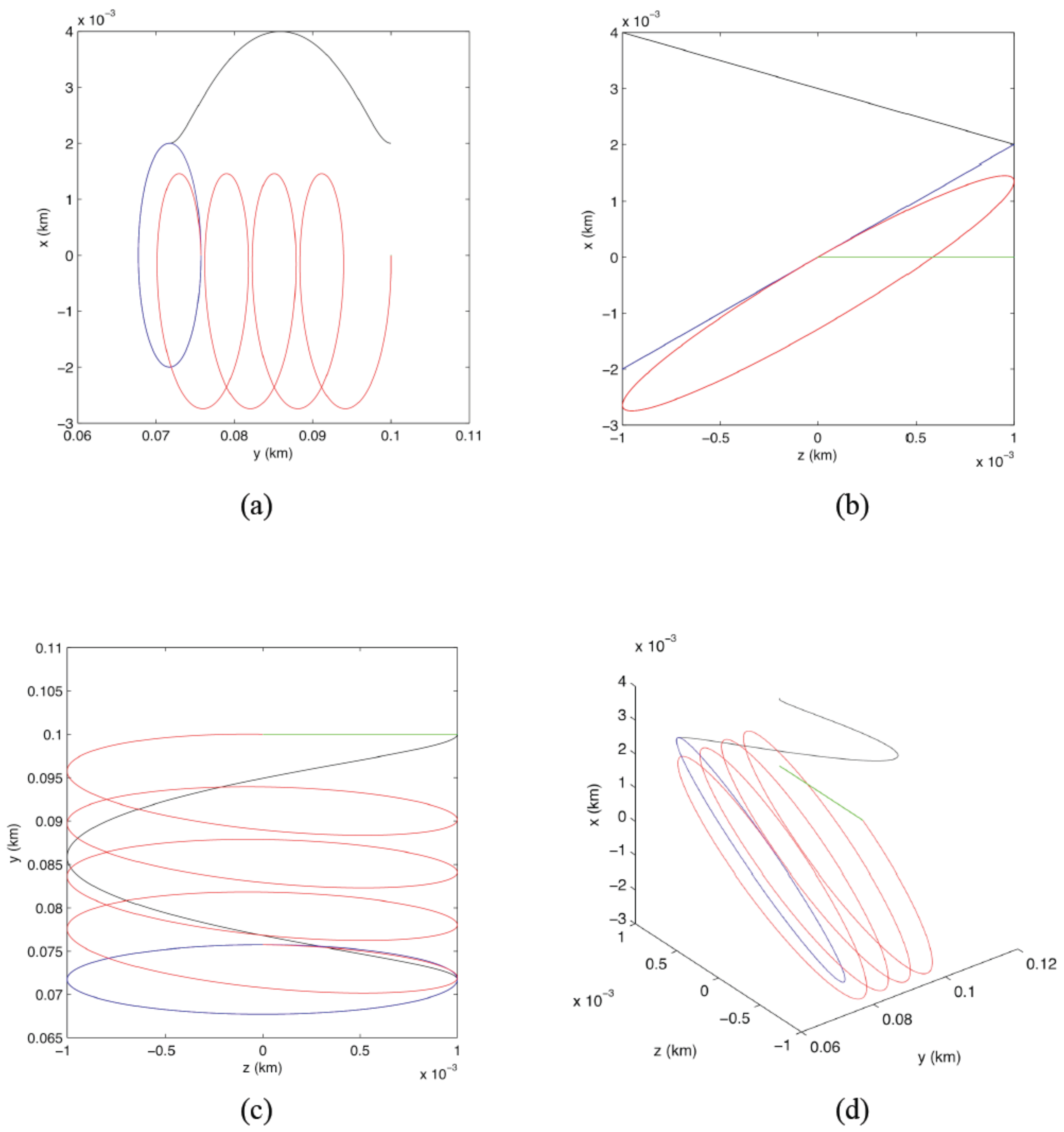


Figure 29: Station-keeping example: (a) y-x projection, (b) z-x projection, (c) z-y projection, (d) Three-dimensional trajectory plot [13]

### 4.2.3. Natural Motion Circumnavigation

To enter into Natural Motion Circumnavigation (NMC) about LightSail, Prox-1 will execute a single impulse maneuver to target the following ROEs: $x_d = 0$, $y_d = 0$, $a_r = y_{d0}$, and $A_z = A_{ztgt}$. The impulsive maneuver contains two components defined by Eqs. (46) and (47).

$$\Delta V_x = \frac{n}{2} y_{d_0} \tag{46}$$

$$\Delta V_z = \pm n A_{z_{tgt}} \tag{47}$$

The following example shows a transition from a 100 m leading orbit to a 100 m circular NMC, as summarized in Table 5. The trajectory plots are shown in Figure 30.

Table 5: Example of transition from leading orbit to NMC motion [13]

| | |
|---|---|
| Initial Conditions | $t_0$, $x_{d_0} = 0$, $y_{d_0} = 0.100$ km, $a_{r_0} = 0$, $A_{z_0} = 0$ |
| Target Conditions | $x_d^+ = 0$, $y_d^+ = 0$, $a_r^+ = 0.100$ km, $A_z^+ = \sqrt{\frac{3}{4}} a_r^+ = 0.0866$ km |
| Solutions | $\Delta V_x = 5.5339E-02$ m/s<br>$\Delta V_y = 0$<br>$\Delta V_z = 9.5847E-05$ m/s<br>$E_r^+ = \frac{\pi}{2}$ rad<br>$\psi^+ = 0$ |

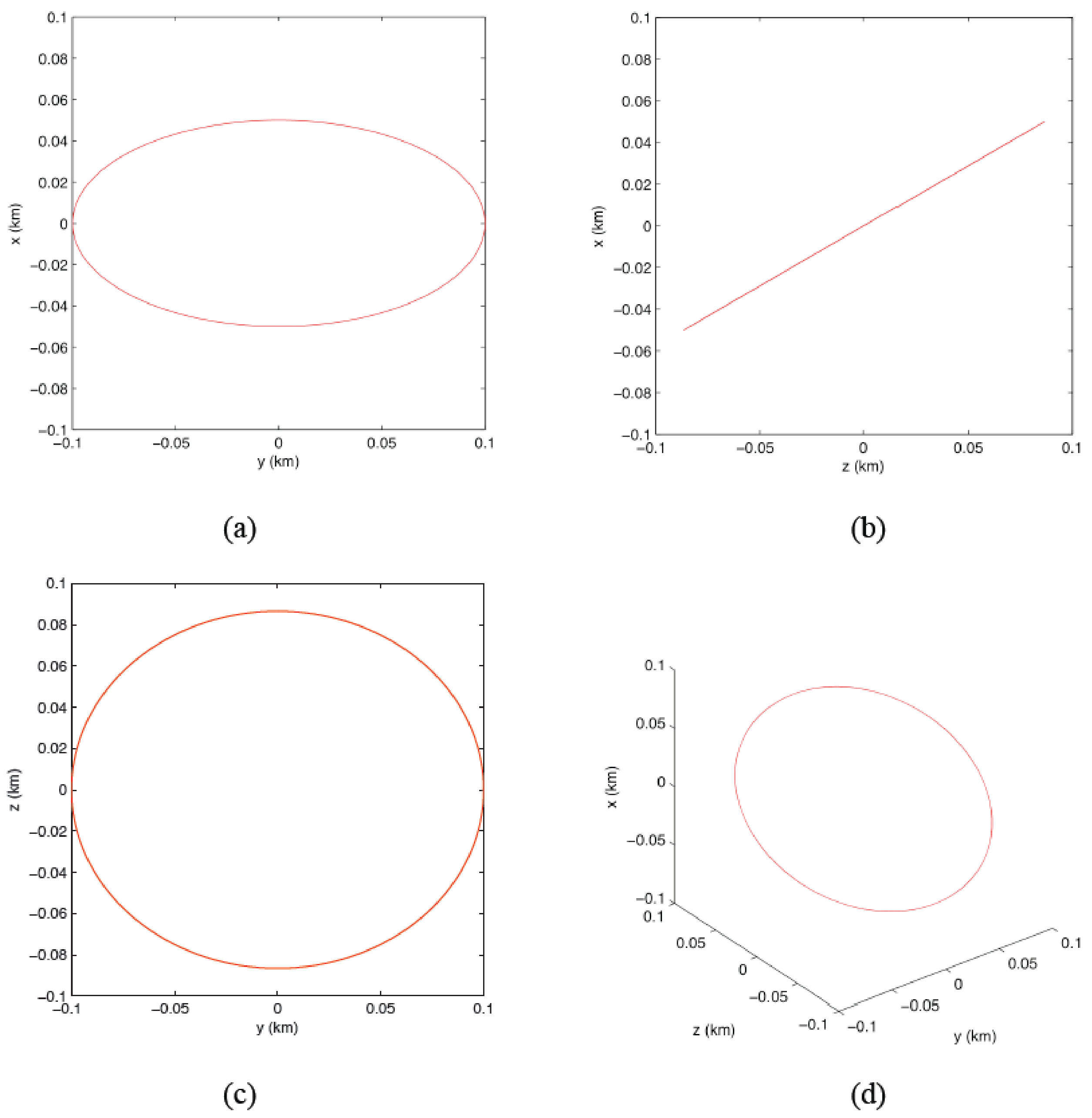


Figure 30: Natural motion circumnavigation circular orbit initiated from a leading orbit: (a) y-x projection, (b) z-x projection, (c) z-y projection, (d) Three-dimensional trajectory plot [13]

## 5. Control Component Details

### 5.1. *Slew & Tracking Controller*

The Slew and Tracking Controller (STC) is used during Proximity Operations for precise attitude control. It produces a torque command for the CMGs based on the current angular state (attitude, angular velocity, and angular acceleration) given a desired angular state. The STC also contains additional logic to allow tracking of the relative position vector to maintain LightSail within Prox-1's imager field of view (the "tracking" element of STC). Also, a cost function is used to determine the optimal direction for solar panel pointing. This cost function allows Prox-1 to track LightSail as a primary pointing objective while simultaneously maintaining maximum power production as a secondary objective. Finally, the STC has the capability to track any arbitrary input for an RSW vector direction and ECI angular velocity; currently this capability is used to input desired slew maneuvers from the Target Acquisition Controller.

The core of the STC is the asymptotically stable torque control law, which calculates a desired torque vector $\boldsymbol{\tau}$ in BFF based on various inputs [2]. The control law is given in Eq. (48), where $\boldsymbol{\omega}_B$ is the current angular velocity of Prox-1 in BFF, $\underline{\boldsymbol{J}}$ is the 3x3 inertia tensor of Prox-1, $\boldsymbol{\omega}_e$ is the error angular velocity defined in Eq. (49), $\underline{\boldsymbol{R}}_e$ is a 3x3 rotation matrix that maps a vector from the desired reference frame to BFF given in Eq. (51), $\boldsymbol{\omega}_d$ is the desired angular velocity in BFF, $\dot{\boldsymbol{\omega}}_d$ is the desired angular acceleration in BFF (usually set to zero), $\eta_e$ is the scalar portion of the error quaternion defined in Eq. (50), $\boldsymbol{\varepsilon}_e$ is the vector portion of the error quaternion, $\boldsymbol{q}$ is the current attitude quaternion of the Prox-1 BFF with respect to ECI, $\underline{\boldsymbol{K}}$ is the quaternion gain matrix, and $\underline{\boldsymbol{C}}$ is the angular velocity gain matrix. The $\boldsymbol{\Xi}$ operator is defined in Eq. (12) and the $\boldsymbol{\Psi}$ operator is defined in Eq. (52).

$$\boldsymbol{\tau} = \boldsymbol{\omega}_B^x \underline{\boldsymbol{J}} \boldsymbol{\omega}_B - \underline{\boldsymbol{J}} \boldsymbol{\omega}_e^x \underline{\boldsymbol{R}}_e \boldsymbol{\omega}_d + \underline{\boldsymbol{J}} \underline{\boldsymbol{R}}_e \dot{\boldsymbol{\omega}}_d - 2\eta_e \underline{\boldsymbol{K}} \boldsymbol{\varepsilon}_e - \underline{\boldsymbol{C}} \boldsymbol{\omega}_e \tag{48}$$

$$\boldsymbol{\omega}_e = \boldsymbol{\omega}_B - \underline{\boldsymbol{R}}_e \, \boldsymbol{\omega}_d \tag{49}$$

$$\boldsymbol{q}_e = \left[ \underline{\boldsymbol{\Xi}}(\boldsymbol{q}_d^{-1}) \; \boldsymbol{q}_d^{-1} \right] \boldsymbol{q} = \begin{bmatrix} \boldsymbol{\varepsilon}_e \\ \eta_e \end{bmatrix} \tag{50}$$

$$\underline{\boldsymbol{R}}_e = \underline{\boldsymbol{\Xi}}^T(\boldsymbol{q}_e) \underline{\boldsymbol{\Psi}}(\boldsymbol{q}_e) \tag{51}$$

$$\underline{\boldsymbol{\Psi}}(\boldsymbol{q}) = \begin{bmatrix} -\boldsymbol{\varepsilon}^x + \eta \underline{\mathbf{I}} \\ -\boldsymbol{\varepsilon}^T \end{bmatrix} \tag{52}$$

Every function of the STC is performed by manipulating the variables of the desired angular state: $\boldsymbol{q}_d$ , $\boldsymbol{\omega}_d$, and $\dot{\boldsymbol{\omega}}_d$ in Eqs. (48)-(50). In order to determine the desired states, complicated Simulink logic is employed, along with embedded MATLAB code. The desired ECI to BFF rotation matrix is computed from a desired pointing vector in the RSW frame using a MATLAB function called `SlewVector2Rotation`. The desired quaternion $\boldsymbol{q}_d$ is then computed using the procedure in Table 6, which utilizes a method presented by Shuster [14].

Table 6: Procedure to compute desired quaternion

| | |
|---|---|
| 1. Compute components of rotation matrix $\underline{\boldsymbol{R}}$ using `SlewVector2Rotation` | $\underline{\boldsymbol{R}}$: 3x3 Matrix |
| 2. Evaluate only the arguments of Equations (53)a, (54)a, (55)a, and (56)a: | $\eta$: Scalar |
| Equation (53)a: $\eta_4 = \pm \frac{1}{2}\sqrt{Tr(\underline{\boldsymbol{R}}) + 1}$ | |
| Equation (54)a: $\eta_1 = \pm \frac{1}{2}\sqrt{1 + \underline{\boldsymbol{R}}_{1,1} - \underline{\boldsymbol{R}}_{2,2} - \underline{\boldsymbol{R}}_{3,3}}$ | |
| Equation (55)a: $\eta_2 = \pm \frac{1}{2}\sqrt{1 - \underline{\boldsymbol{R}}_{1,1} + \underline{\boldsymbol{R}}_{2,2} - \underline{\boldsymbol{R}}_{3,3}}$ | |
| Equation (56)a: $\eta_3 = \pm \frac{1}{2}\sqrt{1 - \underline{\boldsymbol{R}}_{1,1} - \underline{\boldsymbol{R}}_{2,2} + \underline{\boldsymbol{R}}_{3,3}}$ | |
| 3. Determine the largest value between $\eta_1, \eta_2, \eta_3, \eta_4$ | |
| if $(\eta_1 > \{\eta_2, \eta_3, \eta_4\})$ Use Equations (53): | |
| $\eta_1 = \pm \frac{1}{2}\sqrt{1 + \underline{\boldsymbol{R}}_{1,1} - \underline{\boldsymbol{R}}_{2,2} - \underline{\boldsymbol{R}}_{3,3}} \quad (a)$ | |
| $\eta_2 = \frac{1}{4\eta_1}\left(\underline{\boldsymbol{R}}_{1,2} + \underline{\boldsymbol{R}}_{2,1}\right) \quad (b)$ | |
| $\eta_3 = \frac{1}{4\eta_1}\left(\underline{\boldsymbol{R}}_{1,3} + \underline{\boldsymbol{R}}_{3,1}\right) \quad (c)$ | |
| $\eta_4 = \frac{1}{4\eta_1}\left(\underline{\boldsymbol{R}}_{2,3} - \underline{\boldsymbol{R}}_{3,2}\right) \quad (d)$ | |
| else if $(\eta_2 > \{\eta_1, \eta_3, \eta_4\})$ Use Equations (54): | |
| $\eta_2 = \pm \frac{1}{2}\sqrt{1 - \underline{\boldsymbol{R}}_{1,1} + \underline{\boldsymbol{R}}_{2,2} - \underline{\boldsymbol{R}}_{3,3}} \quad (a)$ | |
| $\eta_1 = \frac{1}{4\eta_2}\left(\underline{\boldsymbol{R}}_{1,2} + \underline{\boldsymbol{R}}_{2,1}\right) \quad (b)$ | |
| $\eta_3 = \frac{1}{4\eta_2}\left(\underline{\boldsymbol{R}}_{2,3} + \underline{\boldsymbol{R}}_{3,2}\right) \quad (c)$ | |
| $\eta_4 = \frac{1}{4\eta_2}\left(\underline{\boldsymbol{R}}_{3,1} - \underline{\boldsymbol{R}}_{1,3}\right) \quad (d)$ | |
| else if $(\eta_3 > \{\eta_2, \eta_1, \eta_4\})$ Use Equations (55): | |
| $\eta_3 = \pm \frac{1}{2}\sqrt{1 - \underline{\boldsymbol{R}}_{1,1} - \underline{\boldsymbol{R}}_{2,2} + \underline{\boldsymbol{R}}_{3,3}} \quad (a)$ | |
| $\eta_1 = \frac{1}{4\eta_3}\left(\underline{\boldsymbol{R}}_{1,3} + \underline{\boldsymbol{R}}_{3,1}\right) \quad (b)$ | |
| $\eta_2 = \frac{1}{4\eta_3}\left(\underline{\boldsymbol{R}}_{2,3} + \underline{\boldsymbol{R}}_{3,2}\right) \quad (c)$ | |
| $\eta_4 = \frac{1}{4\eta_3}\left(\underline{\boldsymbol{R}}_{1,2} - \underline{\boldsymbol{R}}_{2,1}\right) \quad (d)$ | |
| else: Use Equations (56): | |
| $\eta_4 = \pm \frac{1}{2}\sqrt{Tr(\underline{\boldsymbol{R}}) + 1} \quad (a)$ | |
| $\eta_1 = \frac{1}{4\eta_4}\left(\underline{\boldsymbol{R}}_{2,3} - \underline{\boldsymbol{R}}_{3,2}\right) \quad (b)$ | |
| $\eta_2 = \frac{1}{4\eta_4}\left(\underline{\boldsymbol{R}}_{3,1} - \underline{\boldsymbol{R}}_{1,3}\right) \quad (c)$ | |
| $\eta_3 = \frac{1}{4\eta_4}\left(\underline{\boldsymbol{R}}_{1,2} - \underline{\boldsymbol{R}}_{2,1}\right) \quad (d)$ | |
| 4. Compute $\boldsymbol{q}_d = [\eta_1;\ \eta_2;\ \eta_3;\ \eta_4]$ using appropriate set of equations from Step 3. | $\boldsymbol{q}_d$: 4x1 Vector |

The desired angular acceleration is currently assumed to be zero, however, the desired pointing vector and angular velocity are based on different inputs in different situations. Various modes for STC are represented by a state machine in Simulink called a Stateflow chart, which is shown in Figure 31. The Stateflow chart is a block within the Simulink diagram for the STC. Inputs are fed into the Stateflow block based on the angular state and desired conditions, and outputs from the block are fed into the torque control law. The following modes currently exist within the STC: Standby, Slew_to_Thrust, Wait_for_Thruster, Track, and ManualSlew. Each mode is represented by a tan box in the chart, and blue arrows between the boxes represent transitions between the modes. Boolean variables are set within and outside the Stateflow chart to determine which mode to activate. Each transition arrow contains a logical condition to switch between modes based on these Boolean variables. When the condition connected to an outgoing arrow from the active mode is met, the chart will change its active mode automatically.

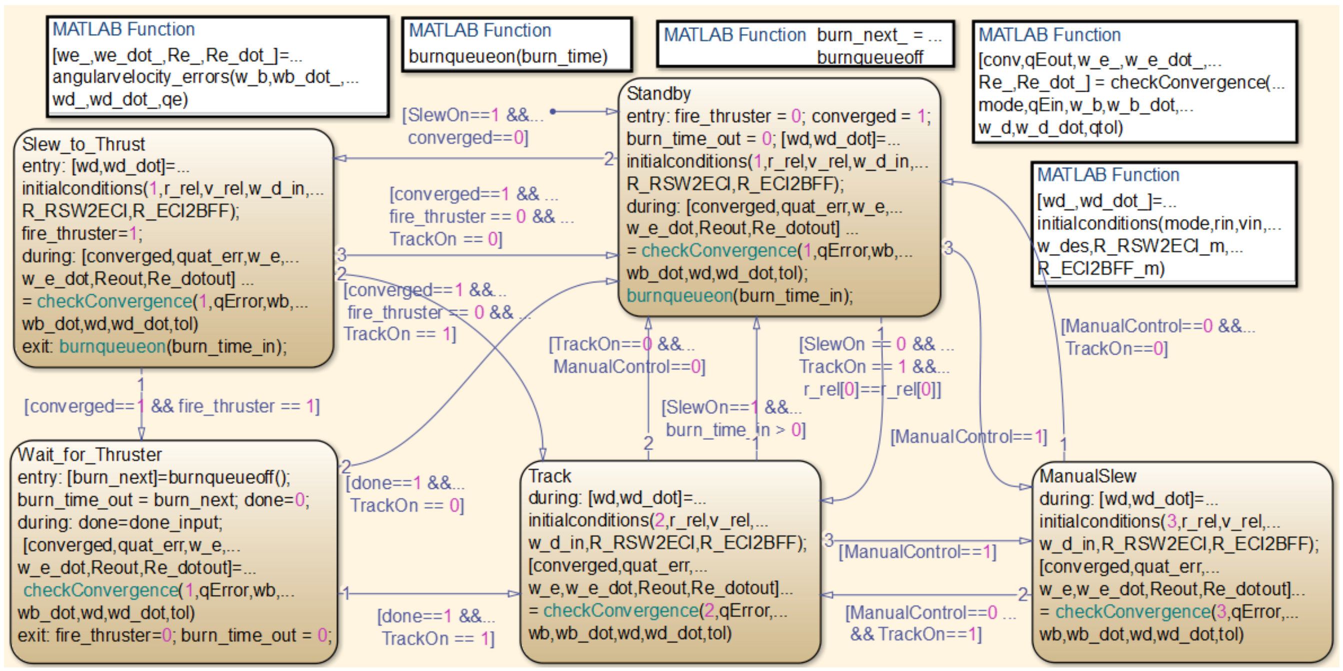

MATLAB Function
[we_,we_dot_,Re_,Re_dot_]=...
angularvelocity_errors(w_b,wb_dot_,...
wd_,wd_dot_,qe)
MATLAB Function
burnqueueon(burn_time)
MATLAB Function burn_next_ = ...
burnqueueoff
MATLAB Function
[conv,qEout,w_e_,w_e_dot_,...
Re_,Re_dot_] = checkConvergence(...
mode,qEin,w_b,w_b_dot,...
w_d,w_d_dot,qtol)
MATLAB Function
[wd_,wd_dot_]=...
initialconditions(mode,rin,vin,...
w_des,R_RSW2ECI_m,...
R_ECI2BFF_m)
Standby
entry: fire_thruster = 0; converged = 1;
burn_time_out = 0; [wd,wd_dot]=...
initialconditions(1,r_rel,v_rel,w_d_in,...
R_RSW2ECI,R_ECI2BFF);
during: [converged,quat_err,w_e,...
w_e_dot,Reout,Re_dotout] ...
= checkConvergence(1,qError,wb,...
wb_dot,wd,wd_dot,tol);
burnqueueon(burn_time_in);
Slew_to_Thrust
entry: [wd,wd_dot]=...
initialconditions(1,r_rel,v_rel,w_d_in,...
R_RSW2ECI,R_ECI2BFF);
fire_thruster=1;
during: [converged,quat_err,w_e,...
w_e_dot,Reout,Re_dotout] ...
= checkConvergence(1,qError,wb,...
wb_dot,wd,wd_dot,tol)
exit: burnqueueon(burn_time_in);
Wait_for_Thruster
entry: [burn_next]=burnqueueoff();
burn_time_out = burn_next; done=0;
during: done=done_input;
[converged,quat_err,w_e,...
w_e_dot,Reout,Re_dotout]=...
checkConvergence(1,qError,wb,...
wb_dot,wd,wd_dot,tol)
exit: fire_thruster=0; burn_time_out = 0;
Track
during: [wd,wd_dot]=...
initialconditions(2,r_rel,v_rel,...
w_d_in,R_RSW2ECI,R_ECI2BFF);
[converged,quat_err,...
w_e,w_e_dot,Reout,Re_dotout]...
= checkConvergence(2,qError,...
wb,wb_dot,wd,wd_dot,tol)
ManualSlew
during: [wd,wd_dot]=...
initialconditions(3,r_rel,v_rel,...
w_d_in,R_RSW2ECI,R_ECI2BFF);
[converged,quat_err,...
w_e,w_e_dot,Reout,Re_dotout]...
= checkConvergence(3,qError,...
wb,wb_dot,wd,wd_dot,tol)
[SlewOn==1 &&...
converged==0]
[converged==1 && ...
fire_thruster == 0 && ...
TrackOn == 0]
[converged==1 &&...
fire_thruster == 0 &&...
TrackOn == 1]
[converged==1 && fire_thruster == 1]
[done==1 &&...
TrackOn == 0]
[done==1 &&...
TrackOn == 1]
[TrackOn==0 &&...
ManualControl==0]
[SlewOn == 0 &&...
TrackOn == 1 &&...
r_rel[0]==r_rel[0]]
[SlewOn==1 &&...
burn_time_in > 0]
[ManualControl==1]
[ManualControl==0 &&...
TrackOn==0]
[ManualControl==1]
[ManualControl==0 ...
&& TrackOn==1]
1
2
3

Figure 31: Stateflow diagram for STC mode logic, including transition conditions

Code within each mode determines what the STC should be doing, calls embedded MATLAB functions within the chart (shown in the top of Figure 31 as a group of silver blocks), and sets the output data that is sent to the torque control algorithm. Stateflow even animates the chart while the simulation is running, adding a bold blue outline around the active mode, which moves along transition arrows when a new mode is activated. In this way the developer can create, test, and debug the mode logic by running the simulation and watching the live animation along with Scope data showing plots of the outputs. Each of the modes will now be explained in detail.

### 5.1.1. Standby Mode

Standby Mode is the default condition for the STC, as indicated by the default transition, a blue arrow with no conditions that is only connected to the Standby block. During Standby Mode, the desired angular velocity is set to zero, and the desired pointing vector is set to the command thrust vector output by the guidance algorithms. Generally during Standby mode this vector is zero, and the last commanded vector is tracked. Standby mode can only be exited on one of three conditions, all controlled by higher level mode logic at the GN&C subsystem level which will be explained in a later section:

1.) Switch to Track mode if tracking is allowed by higher level logic and a relative positon vector estimate has been generated
2.) Switch to Slew_to_Thrust mode if slew is allowed by higher level logic and a new desired thrust vector/time is received from guidance.
3.) Switch to ManualSlew mode if commanded by higher level logic

### 5.1.2. Slew_to_Thrust Mode

If condition 2 from the previous section is met, the STC enters Slew_to_Thrust mode. During this mode desired angular velocity is set to zero, the desired slew vector is the commanded thrust direction, and the $\boldsymbol{y_{BFF}}$ axis is aligned with this vector. The $\boldsymbol{x_{BFF}}$ axis is determined by the cross product between $\boldsymbol{y_{BFF}}$ and the last command for $\boldsymbol{z_{BFF}}$, and the $\boldsymbol{z_{BFF}}$ axis is determined by the cross product between $\boldsymbol{x_{BFF}}$ and $\boldsymbol{y_{BFF}}$. Finally, the rotation matrix $\underline{\boldsymbol{R}}$ from ECI to BFF is computed using Eq. (57).

$$\underline{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{x_{ECI}} \cdot \boldsymbol{x_{BFF}} & \boldsymbol{y_{ECI}} \cdot \boldsymbol{x_{BFF}} & \boldsymbol{z_{ECI}} \cdot \boldsymbol{x_{BFF}} \\ \boldsymbol{x_{ECI}} \cdot \boldsymbol{y_{BFF}} & \boldsymbol{y_{ECI}} \cdot \boldsymbol{y_{BFF}} & \boldsymbol{z_{ECI}} \cdot \boldsymbol{y_{BFF}} \\ \boldsymbol{x_{ECI}} \cdot \boldsymbol{z_{BFF}} & \boldsymbol{y_{ECI}} \cdot \boldsymbol{z_{BFF}} & \boldsymbol{z_{ECI}} \cdot \boldsymbol{z_{BFF}} \end{bmatrix} \tag{57}$$

Slew_to_Thrust mode is exited when the error quaternion converges to within a pre-specified tolerance and one of the following conditions is met:

1.) Switch to Wait_for_Thruster if thruster firing is allowed.
2.) Switch to Track mode if thruster firing is not allowed and tracking is allowed.
3.) Switch to Standby mode if thruster firing is not allowed and tracking is not allowed.

### 5.1.3. Wait_for_Thruster Mode

Wait_for_Thruster mode is activated when the thruster is pointed in the desired direction. Once the mode is activated, a desired burn time is sent to the thruster. The STC continues to point the thruster toward the desired thrust vector direction during this time. Once this burn has been executed, a flag is returned indicating completion and one of the following two modes is entered:

1.) Switch to Track mode if thruster firing is complete and tracking is allowed.
2.) Switch to Standby mode if thruster firing is complete and tracking is not allowed.

### 5.1.4. Track Mode

During tracking mode, the STC uses the relative position and velocity estimates and attempts to keep LightSail within Prox-1's imager FOV. The desired angular velocity in BFF $\boldsymbol{\omega}_d$ is determined using Eq. (58) where $\boldsymbol{r}_{rel}$ is the RSW relative position vector, $\boldsymbol{v}_{rel}$ is the relative velocity vector, and $\underline{\boldsymbol{R}}_{BFF}^{RSW}$ is the transformation matrix from RSW to BFF.

$$\boldsymbol{\omega}_d = \underline{\boldsymbol{R}}_{BFF}^{RSW} \frac{\boldsymbol{r}_{rel}^{x}\boldsymbol{v}_{rel}}{\left\|\boldsymbol{r}_{rel}\right\|^2} \tag{58}$$

The desired slew vector is set to $\boldsymbol{r}_{rel}$ and the camera boresight axis ($\boldsymbol{y}_{BFF}$) is pointed along -$\boldsymbol{r}_{rel}$. If Prox-1 is in eclipse, the $\boldsymbol{x}_{BFF}$ and $\boldsymbol{z}_{BFF}$ axes are computed using the same method as in Slew_to_Thrust mode. However, because of high power usage of the CMGs during ProxOps, if the sun is visible then preferential solar panel pointing is enabled, where the STC continues to track the target while also pointing the top solar panel of Prox-1 towards the Sun. To achieve preferential solar panel pointing, $\boldsymbol{x}_{BFF}$ and $\boldsymbol{z}_{BFF}$ are computed based on the procedure in Table 7.

Table 7: STC Preferential Solar Panel Pointing Calculation

| | |
|---|---|
| 1. Set $\boldsymbol{y}_{BFF}$ to coincide with -$\boldsymbol{r}_{rel}$. | |
| 2. Make an initial guess for $\boldsymbol{z}_{BFF}$ using Equation (59): $$\boldsymbol{z}_{b0} = \boldsymbol{y}_{BFF}{}^{x}\hat{\boldsymbol{s}} \tag{59}$$ | $\hat{\boldsymbol{s}}$**:** unit vector direction to the Sun |
| 3. Rotate $\boldsymbol{z}_{b0}$ about the $\boldsymbol{y}_{BFF}$ axis in 0.5 degree increments to find all unit vectors normal to $\boldsymbol{y}_{BFF}$ using Equation (60), which utilizes the Euler angle rotation matrix for rotation about an arbitrary axis [15] where $\varphi$ is the rotation angle: $$\boldsymbol{z}_{bi} = \begin{bmatrix} (1-\cos\varphi)a_1^2+\cos\varphi & (1-\cos\varphi)a_1a_2+a_3\sin\varphi & (1-\cos\varphi)a_1a_3-a_2\sin\varphi \\ (1-\cos\varphi)a_2a_1-a_3\sin\varphi & (1-\cos\varphi)a_2^2+\cos\varphi & (1-\cos\varphi)a_2a_3+a_1\sin\varphi \\ (1-\cos\varphi)a_3a_1+a_2\sin\varphi & (1-\cos\varphi)a_3a_2-a_1\sin\varphi & (1-\cos\varphi)a_3^2+\cos\varphi \end{bmatrix} \boldsymbol{z}_{b0} \tag{60}$$ | $a_1 = y_{BFF_x}$ $a_2 = y_{BFF_y}$ $a_3 = y_{BFF_z}$ |
| 4. Calculate the cost for each possible $\boldsymbol{z}_{bi}$ using Equation (61) $$J(\boldsymbol{z}_{bi}) = \left\|(-\boldsymbol{z}_{bi}\cdot\hat{\boldsymbol{s}}) - 1\right\| \tag{61}$$ | $J$ : scalar cost function |
| 5. Select the $\boldsymbol{z}_{BFF}$vector that minimizes the cost function and normalize to obtain a unit vector | |

There are three possible conditions to exit Tracking mode:

1.) Switch to Standby (and then to Slew_to_Thrust) mode if slew is allowed and a thruster firing command is received.
2.) Switch to Standby mode if tracking and manual slew are turned off by higher level logic.
3.) Switch to ManualSlew mode if commanded by higher level logic.

### 5.1.5. ManualSlew Mode

ManualSlew mode simply inputs a desired angular velocity and pointing vector sent directly from outside of STC. Currently this method is used to follow a desired slew trajectory for Target Acquisition and Recovery. There are two possible conditions to exit ManualSlew mode:

1.) Switch to Standby mode if tracking is not allowed and manual slew is no longer commanded by higher level logic.
2.) Switch to Track mode if tracking is allowed and manual slew is no longer commanded by higher level logic.

### 5.1.6. STC Performance Test

The STC controller gains were selected such that the system is critically damped, with the damping coefficient set to one. An analysis of control input limitations, slew time, and final error [2] resulted in the selection of the gain constants 0.05*$I_3$ for $\underline{\boldsymbol{K}}$ and 0.5*$I_3$ for $\underline{\boldsymbol{C}}$. Given an initial error quaternion of [0.5 -0.5 0.5 0.5], the resulting control torque input is shown in Figure 32.

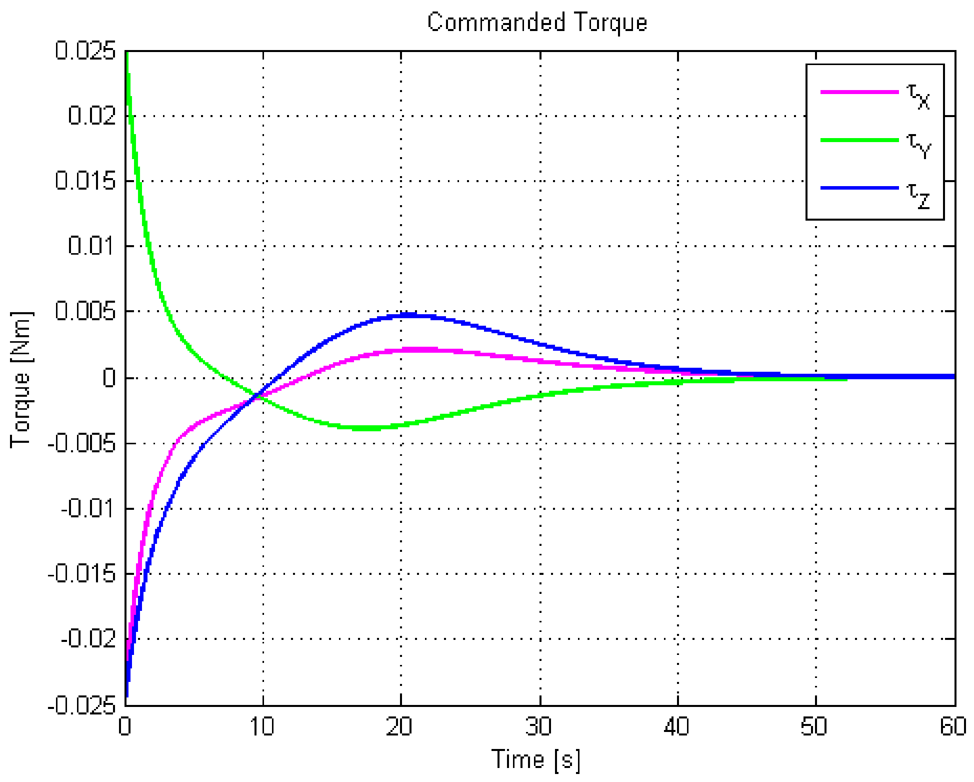
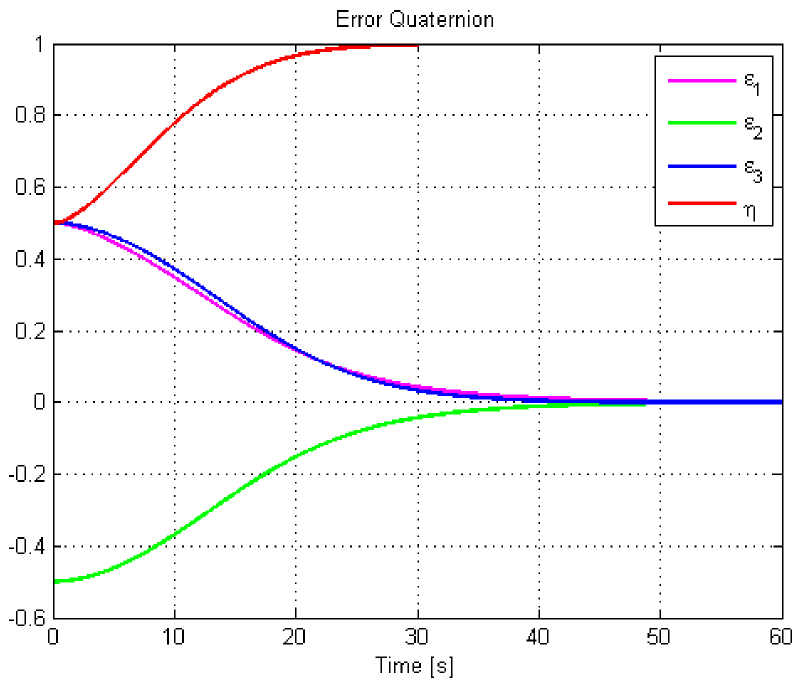
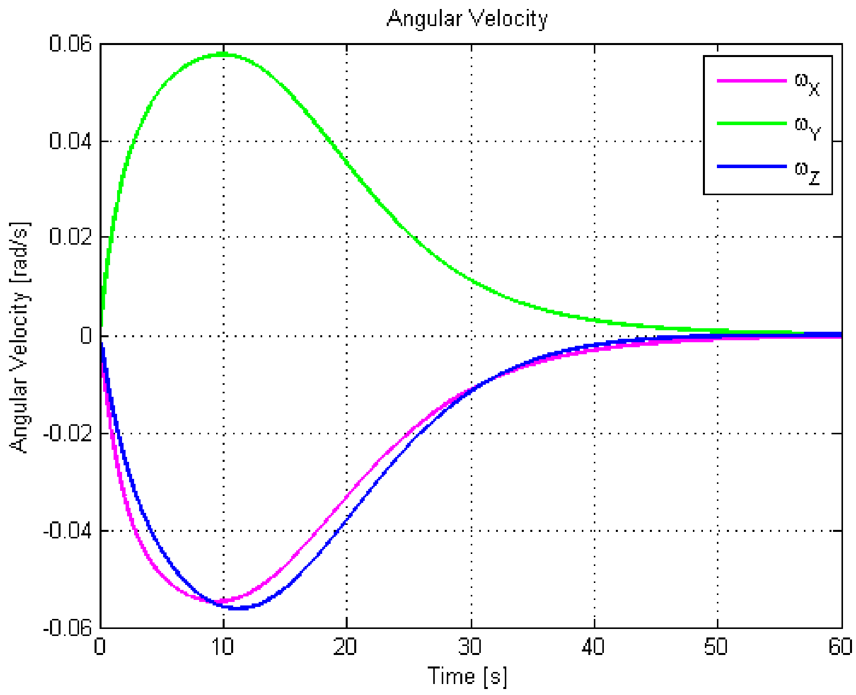


Figure 32: STC control torque input

The resulting outputs of the system, the error quaternion and spacecraft angular velocity, are shown in Figure 33. The error quaternion output in Figure 33(a) demonstrates that the selected gains do in fact produce a critically damped system. The tracking controller and mode logic were developed and tested later during integration with other GN&C components.



**(a)**



**(b)**

Figure 33: STC performance parameters: (a) error quaternion, (b) spacecraft angular velocity

## 5.2. *Target Acquisition & Recovery*

The Target Acquisition Controller (TAC) is used to locate or recover LightSail whenever it is not within Prox-1's imager FOV. The TAC produces slew commands which are executed using the ManualSlew mode of STC. The following description is based on a report written by Prox-1 team member Manan Gandhi.

The current version of the TAC has functioning Stateflow logic and integrated into the STC and overall GN&C mode logic. The primary logic paths include one where the Prox-1 has a previous relative position vector of the RSO from the RelOD filter and one where Prox-1 does not have this relative position vector.

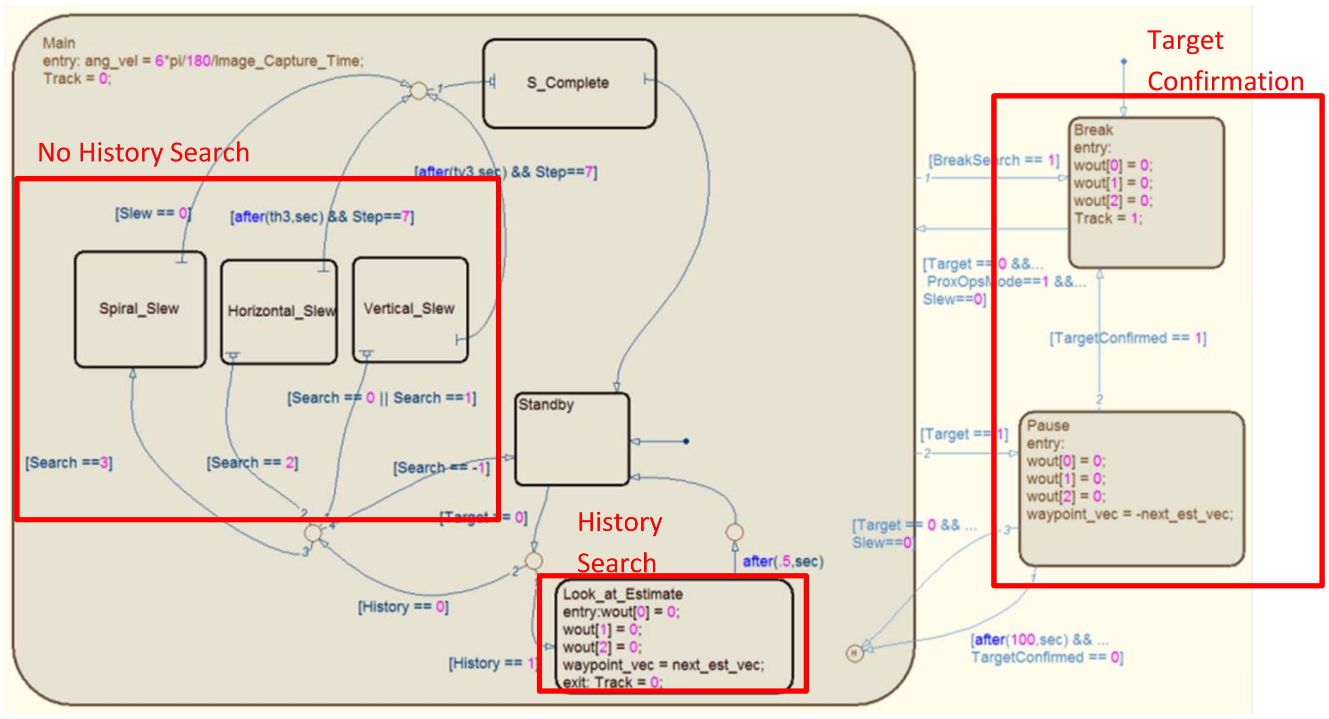


Figure 34: Target Acquisition & Recovery Stateflow chart

Figure 34 shows a Stateflow chart that is the core of the target acquisition controller. The two highlighted sections within the main chart represent the two main states of the TAC: search with relative position history and search with no reliable relative position history.

The overall logic of the controller is as follows: TAC constantly sends commands to the STC in order to perform the desired slew maneuvers to conduct a thorough search. The desired pointing vector from TAC is output to STC in RSW and the desired angular velocity is output to STC in ECI. During every point of the search, an input from the IPAs is constantly updating whether or not the RSO has been acquired and confirmed. If an object that appears to be the RSO is located on the camera, the TAC algorithm immediately pauses and attempts to keep the RSO within the imager FOV. Then the controller waits for a set amount of time in order to give the IPAs time to confirm whether the object in view is the desired RSO and to allow the RelOD filter to converge on a solution for relative position and velocity.

During a search with a reliable current relative position vector, the TAC simply outputs a track command to the STC. This Track command instructs the STC to slew towards the last reliable relative position vector. This functionality will point the spacecraft towards the last known location of the RSO in hopes of locating it.

During the search with an outdated or unconverged relative position vector (i.e. initial acquisition or recovery after losing the RSO for a long period of time), the TAC begins a series of search patterns. These search patterns are currently designed to search through a given angular sweep in three ways. The first is a local vertical search, the second is a local horizontal search, and finally there is the potential to add an additional spiral search. These search patterns are commanded to the STC by outputting a constant desired angular velocity vector, as well as a series of waypoint vectors illustrated in Figure 35. These waypoint vectors provide the STC with a path to follow as it rotates at the given constant angular velocity and allow the STC to resolve an error quaternion to determine whether it has arrived at the desired orientation.

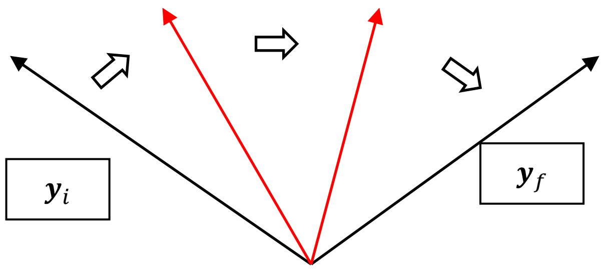


Figure 35: Waypoint Vector example showing a slew from $\mathbf{y}_i$ to $\mathbf{y}_f$

In the current iteration of the TAC control logic, the following inputs are required from the various GN&C components:

- Slew status flag (logical) from STC [outputs the current state of the ManualSlew mode]
- Relative position to LightSail (unit vector) from the RelOD filter
- Target Acquired flag (logical) from the IPAs
- Target Confirmed flag (logical) from the IPAs
- Image sample rate (scalar) from the IPAs
- Stop Search flag (logical) from higher mode logic.
- Rotation Matrices RSW2ECI and BFF2RSW from the RelOD filter

In the current iteration of the control logic, the following outputs are sent to various GN&C components:

- ECI angular velocity command (vector) to STC
- Tracking Command flag (logical) to STC
- RSW waypoint vector to STC
- TargetConfirmed flag (logical) to higher level mode logic

Internally within the TAC there are three main search patterns that are utilized to locate the RSO: local vertical, local horizontal, and spiral slew. Local vertical is the performed first because the RSO is expected to be either ahead of or behind Prox-1 in its orbit. The TAC first aligns Prox-1's BFF with the RSW frame with the imager pointing along the velocity vector (S). Then, the local vertical search rotates Prox-1 about the radial (R) axis, first pointing the imager ($Y_{BFF}$) toward the cross-track direction (W). This is the V_1 maneuver shown in Figure 36. Then, Prox-1 rotates about the –R axis back through the velocity vector direction and points the imager toward the –W axis: the V_2 maneuver. Finally, Prox-1 returns to point its imager along the velocity vector: the V_3 maneuver that completes a 180 degree sweep of the SW-plane ahead of Prox-1 in its orbit. This set of three maneuvers can be modified to span the opposite 180 degrees or the entire 360 degree plane depending on the final ConOps of the mission. A mission with Prox-1 rendezvous in a trailing orbit of LightSail was selected for the initial demonstration of the TAC.

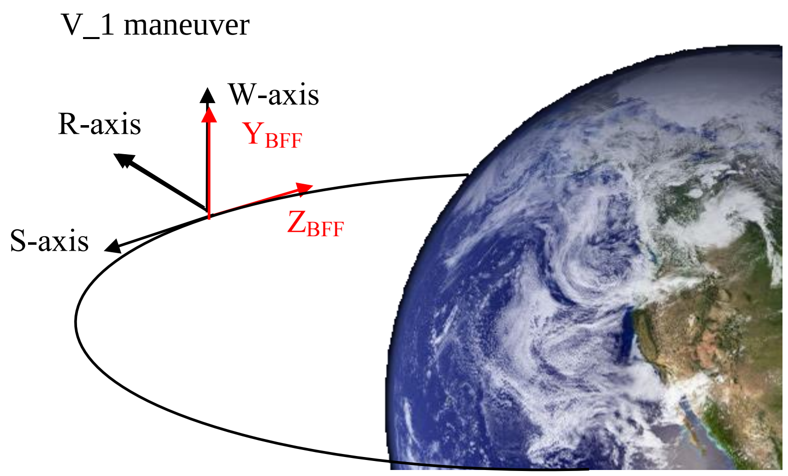
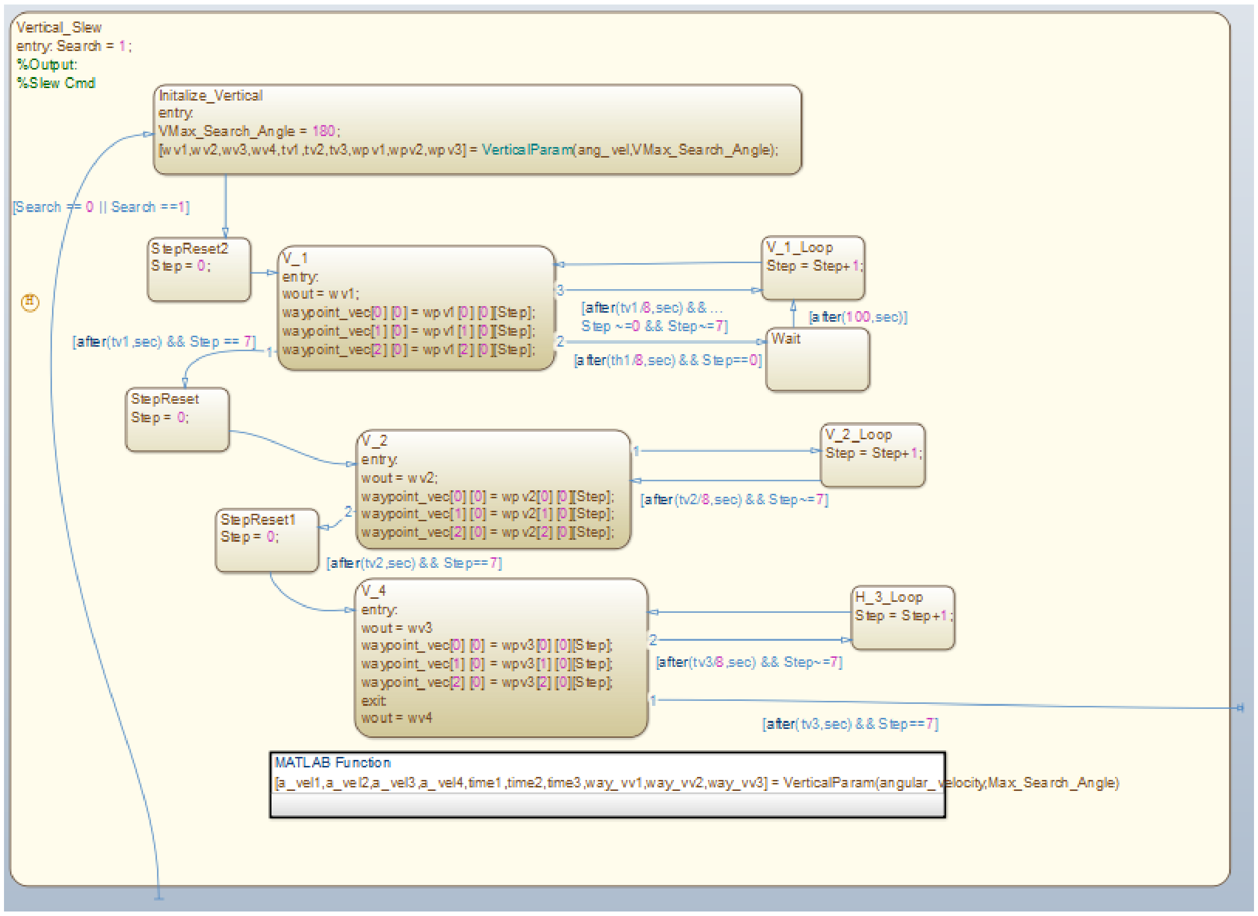


Figure 36: Local vertical TAC search along the SW-plane

In order to command these maneuvers, a sub-chart of the TAC Stateflow chart from Figure 34 is activated, as shown in Figure 37. Currently, on entry into the Vertical_Slew state, a step iterator is initialized to 0 and the Search command is set up to continue to the next search pattern after the completion of the vertical search.The Initialize_Vertical state sets the maximum search angle that will be passed through for the Horizontal Search. It then runs an embedded MATLAB script that calculates the desired slew rate (based on the imager sample rate), the required time delays between each maneuver, and the required waypoint vectors for the maneuver. The states V_1 and

V_1_Loop demonstrate the use of each waypoint vector. For each step, V_1 sends a waypoint vector and the desired angular velocity to the STC. After a certain amount of time, the next waypoint is immediately sent to the STC. After the V_1 maneuver is completed, the next two maneuvers V_2 and V_3 sequentially begin.



```
Vertical_Slew
entry: Search = 1;
%Output:
%Slew Cmd

Initalize_Vertical
entry:
VMax_Search_Angle = 180;
[wv1,wv2,wv3,wv4,tv1,tv2,tv3,wpv1,wpv2,wpv3] = VerticalParam(ang_vel,VMax_Search_Angle);

StepReset2
Step = 0;

V_1
entry:
wout = wv1;
waypoint_vec[0] [0] = wpv1 [0] [0][Step];
waypoint_vec[1] [0] = wpv1 [1] [0][Step];
waypoint_vec[2] [0] = wpv1 [2] [0][Step];

V_1_Loop
Step = Step+1;

Wait

StepReset
Step = 0;

V_2
entry:
wout = wv2;
waypoint_vec[0] [0] = wpv2[0] [0][Step];
waypoint_vec[1] [0] = wpv2[1] [0][Step];
waypoint_vec[2] [0] = wpv2[2] [0][Step];

V_2_Loop
Step = Step+1;

StepReset1
Step = 0;

V_4
entry:
wout = wv3
waypoint_vec[0] [0] = wpv3[0] [0][Step];
waypoint_vec[1] [0] = wpv3[1] [0][Step];
waypoint_vec[2] [0] = wpv3[2] [0][Step];
exit:
wout = wv4

H_3_Loop
Step = Step+1;

MATLAB Function
[a_vel1,a_vel2,a_vel3,a_vel4,time1,time2,time3,way_vv1,way_vv2,way_vv3] = VerticalParam(angular_velocity,Max_Search_Angle)
```

[Search == 0 || Search ==1]
3
[after(tv1/8,sec) && ...
Step ~=0 && Step~=7]
2
[after(th1/8,sec) && Step==0]
[after(100,sec)]
1
[after(tv1,sec) && Step == 7]
1
[after(tv2/8,sec) && Step~=7]
2
[after(tv2,sec) && Step==7]
2
[after(tv3/8,sec) && Step~=7]
1
[after(tv3,sec) && Step==7]

Figure 37: Vertical slew Stateflow sub-chart for TAC

The next search pattern is the local horizontal search, a rotation about the W-axis. The TAC first aligns Prox-1's BFF with the RSW frame with the imager pointing along the velocity vector (S). Then, the local vertical search rotates Prox-1 about the cross-track (W) axis, first pointing the imager ($Y_{BFF}$) toward nadir direction; this is the H_1 maneuver shown in Figure 38. Then, Prox-1 rotates about the W-axis back through the velocity vector direction and points the imager to zenith (radial) direction: the H_2 maneuver. Finally, Prox-1 returns to point its imager along the velocity vector again: the H_3 maneuver that completes a 180 degree sweep of the RS-plane ahead of Prox-1 in its orbit. In order to command these slews, an internal Stateflow chart very similar to the local vertical slew chart sends commands to the STC.

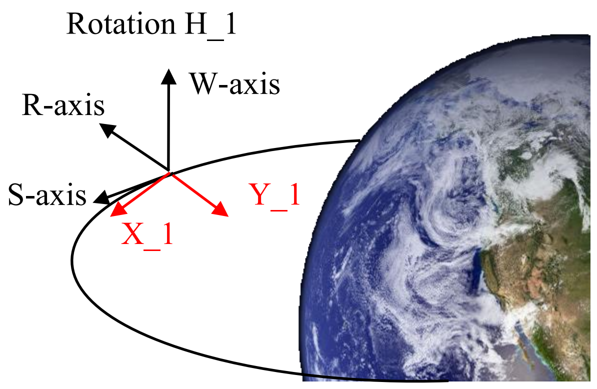


Figure 38: Local horizontal TAC search along the RS-plane

The final search pattern is the spiral search pattern, which has not yet been completely implemented and tested in Stateflow. This pattern is visualized in Figure 39. The $Y_{BFF}$-axis is rotated around the S-axis or the last known position vector in order to search around where the RSO could potentially be located.

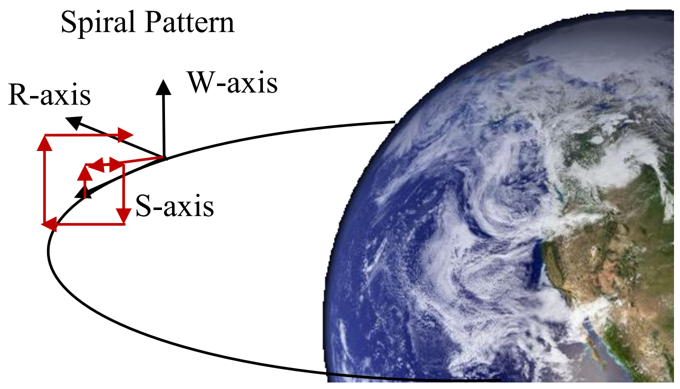
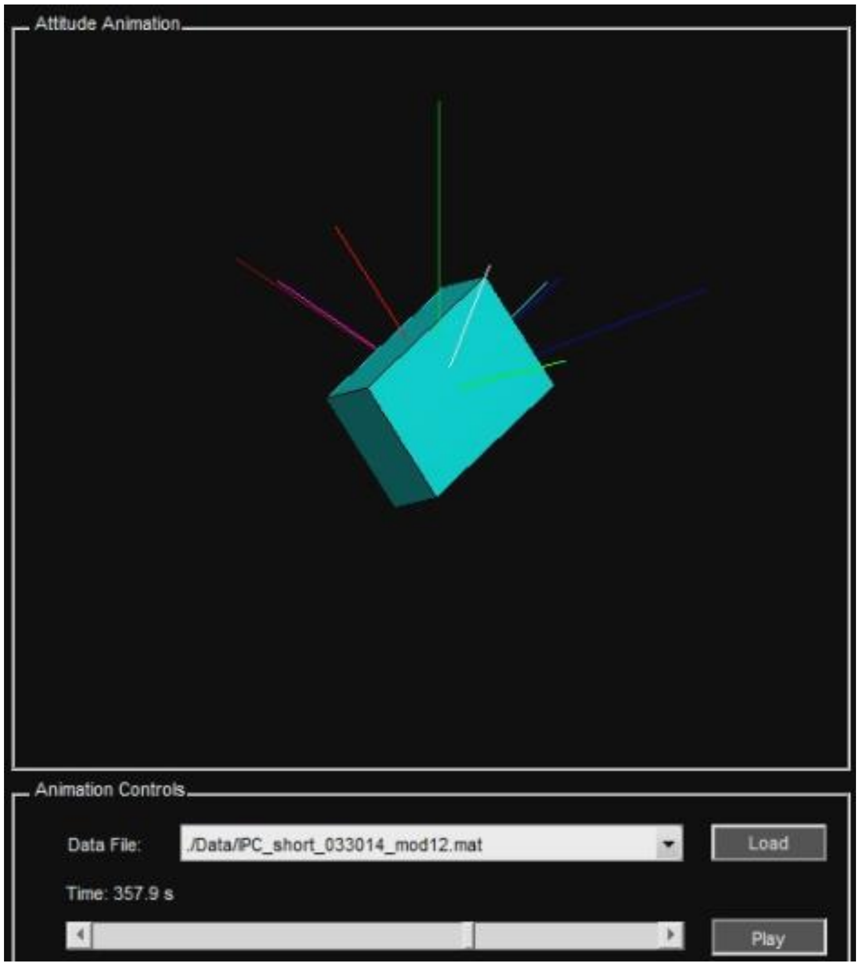


Figure 39: Spiral TAC search pattern

During integration and testing of the TAC, a graphical user interface shown in Figure 40 was used to ensure that the desired rotations were being executed properly. This allowed for an animation of the attitude of the Prox-1 BFF with respect to the constant ECI frame and the time-varying RSW frame. The animation is based on data output by the Prox-1 GN&C simulation after it finishes running and is very helpful for debugging the Stateflow logic and MATLAB code defining the TAC maneuvers. This tool may be integrated with future mission planning or operations displays and could be expanded to illustrate the relative direction to the Sun, the relative position to LightSail, antenna pointing, or other attitude considerations.



./Data/IPC_short_033014_mod12.mat

Time: 357.9 s

Figure 40: Attitude animation of Prox-1 rotation for TAC development and testing

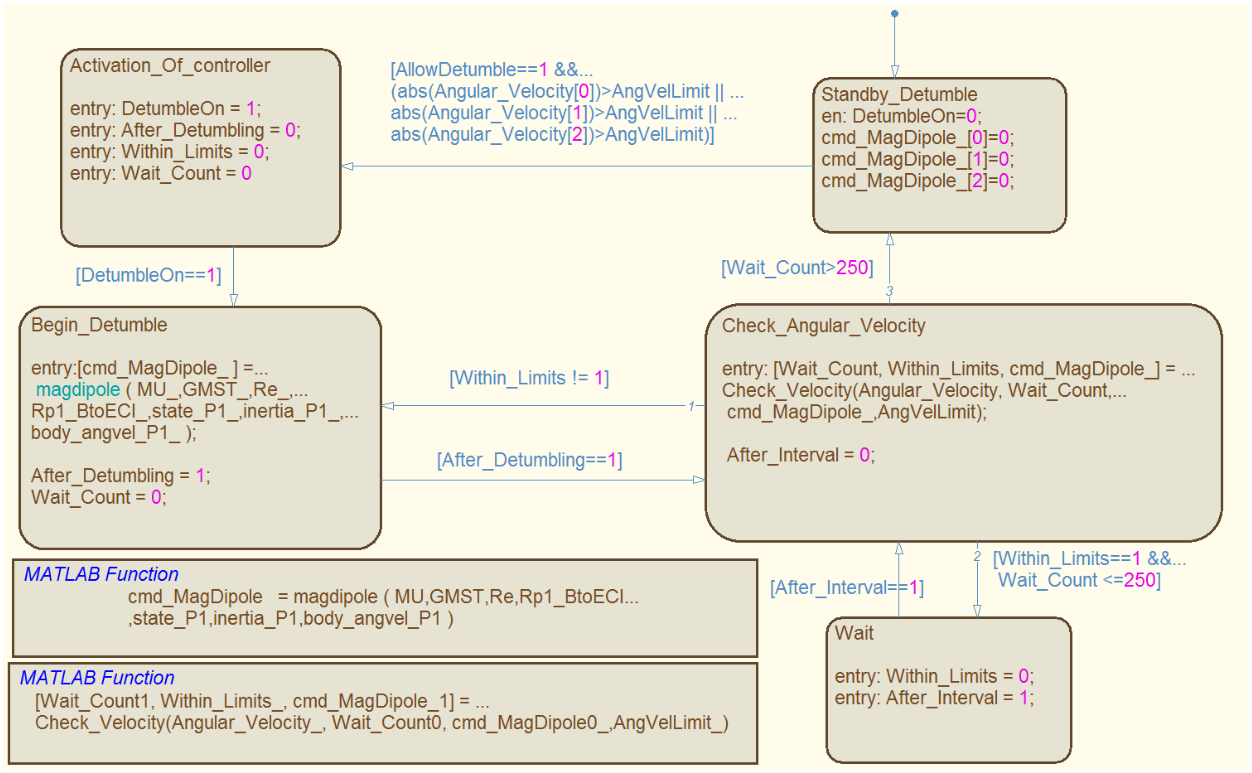
## 5.3. *Detumble Controller*

The detumble controller (DTC) determines when Prox-1 has an angular velocity that is too high and calculates a magnetic moment to damp the rotation rate of the spacecraft using its torque rods. The magnetic control torque $\boldsymbol{M}$ is generated by taking the cross product of the magnetic dipole moment $\boldsymbol{m}$ with the current magnetic field $\boldsymbol{b}$. The magnetic dipole moment $\boldsymbol{m}$ is calculated using the method presented by Avanzini and Giuletti [16], shown in Eq. (62) where $\hat{\boldsymbol{b}}$ is the unit vector direction of the Earth's magnetic field at Prox-1's location, $\boldsymbol{\omega}$ is the current angular velocity of Prox-1, and $k_\omega$ is a constant given in Eq. (63) where $\Omega$ is the mean motion of Prox-1's orbit, $\xi_m$ is the inclination of Prox-1's orbit with respect to Earth's magnetic equator plane, and $J_{min}$ is Prox-1's minimum principal moment of inertia in BFF.

$$\boldsymbol{m} = -\frac{k_\omega}{\|\boldsymbol{b}\|}\hat{\boldsymbol{b}} \times \left[\left(\boldsymbol{I_3} - \hat{\boldsymbol{b}}\hat{\boldsymbol{b}}^{\boldsymbol{T}}\right)\boldsymbol{\omega}\right] \tag{62}$$

$$k_\omega = 2\Omega(1 + \sin\xi_m)J_{min} \tag{63}$$

The DTC Stateflow logic diagram is shown in Figure 41. When the DTC is initialized, it enters the Standby_Detumble state. If Detumble is allowed by higher level logic and angular rate limits are exceeded, Activation_Of_controller mode begins and several logical flags are initialized. Then the DTC transitions to Begin_Detumble, where a MATLAB function calculates the required magnetic dipole moment to damp the rotation, and a command to produce the moment is sent to the torque rods via the torque rod controller. DTC then transitions to Check_Angular_Velocity and a second MATLAB function determines if the angular velocity limits have been satisfied. If angular velocity is not within limits, then DTC returns to Begin_Detumble and calculates a new magnetic moment. If the angular velocity is within limits, a wait timer is started. As the timer advances, the angular velocity check continues. Once the timer expires, DTC moves back into Standby_Detumble until the angular velocity limits are violated again and detumbling is allowed by higher level mode logic.


```
Activation_Of_controller

entry: DetumbleOn = 1;
entry: After_Detumbling = 0;
entry: Within_Limits = 0;
entry: Wait_Count = 0

[AllowDetumble==1 &&...
(abs(Angular_Velocity[0])>AngVelLimit || ...
abs(Angular_Velocity[1])>AngVelLimit || ...
abs(Angular_Velocity[2])>AngVelLimit)]

Standby_Detumble
en: DetumbleOn=0;
cmd_MagDipole_[0]=0;
cmd_MagDipole_[1]=0;
cmd_MagDipole_[2]=0;

[DetumbleOn==1]

[Wait_Count>250]

Begin_Detumble

entry:[cmd_MagDipole_] =...
 magdipole ( MU_,GMST_,Re_,...
Rp1_BtoECI_,state_P1_,inertia_P1_,...
body_angvel_P1_ );

After_Detumbling = 1;
Wait_Count = 0;

[Within_Limits != 1]

[After_Detumbling==1]

Check_Angular_Velocity

entry: [Wait_Count, Within_Limits, cmd_MagDipole_] = ...
Check_Velocity(Angular_Velocity, Wait_Count,...
 cmd_MagDipole_,AngVelLimit);

 After_Interval = 0;

MATLAB Function
            cmd_MagDipole   = magdipole ( MU,GMST,Re,Rp1_BtoECI...
            ,state_P1,inertia_P1,body_angvel_P1 )

MATLAB Function
  [Wait_Count1, Within_Limits_, cmd_MagDipole_1] = ...
  Check_Velocity(Angular_Velocity_, Wait_Count0, cmd_MagDipole0_,AngVelLimit_)

[After_Interval==1]

[Within_Limits==1 &&...
 Wait_Count <=250]

Wait

entry: Within_Limits = 0;
entry: After_Interval = 1;
```

Figure 41: Stateflow diagram for Detumble Controller to generate magnetic dipole command

A test scenario is shown in Figure 42, with an initial tumbling rate of 3 deg/s in each axis. A saturation limit of 11.16 Am$^2$ is used for the magnetic moment output because of the limitations of the torque rods. In this case, the convergence criteria were angular velocities of less than 0.01 rad/s (or 0.573 deg/s) in each axis. The magnetic moment plot shows that the initial magnetic moment command was outside the capability of the torque rods, however after about 5 seconds the necessary moment is attainable. Note that the moment output is not directly realized but is fed into the torque rod controller, which turns the torque rods on and off to approximate it. In this case, the angular velocity goes to zero and the attitude becomes constant after 50 seconds.

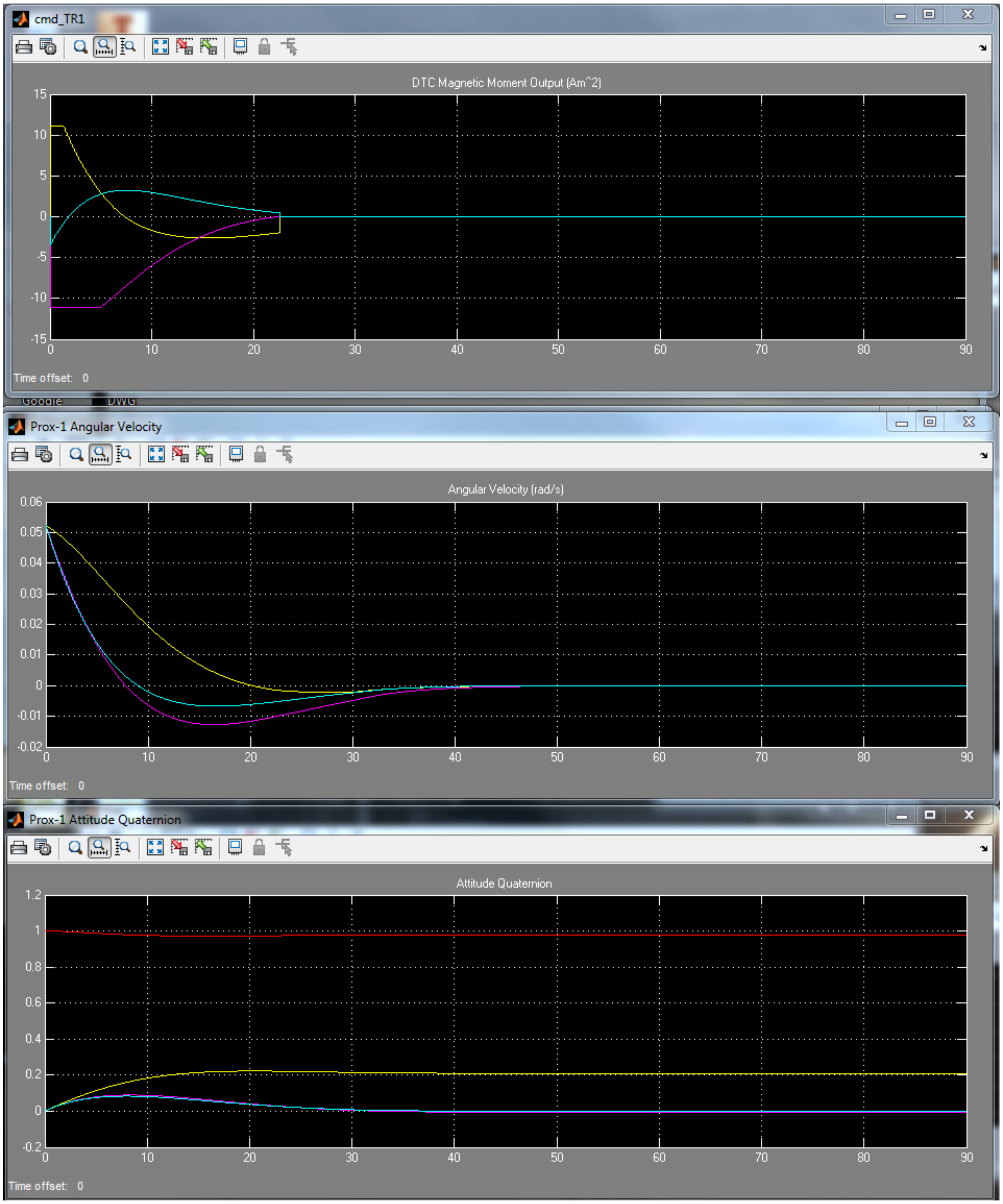


Figure 42: Results from DTC test scenario

### 5.4. *Desaturation Controller*

The CMGs on Prox-1 will have a maximum momentum capacity, and once this capacity is reached they will be saturated and no longer effective for attitude control. The desaturation controller monitors for momentum saturation of the CMGs and performs a desaturation maneuver. Allowable pre-maneuver saturation levels will determined through integrated testing with the CMG model. The process for desaturation is illustrated in the preliminary Stateflow diagram in Figure 43. The current saturation levels are evaluated and compared to a chosen saturation limit. If this limit is violated, desaturation begins immediately. Also, saturation is checked before the beginning of each maneuver, and if the CMGs are close to reaching the saturation limit, the maneuver will not be performed and Prox-1 will desaturate the CMGs before continuing. Desaturation is performed by commanding each of the four CMG gimbal wheel rates to zero. This process induces a rotation of Prox-1, so after desaturation is complete, DTC will be activated to detumble the spacecraft and dump the excess momentum. Finally, proximity operations will resume after detumble is completed.

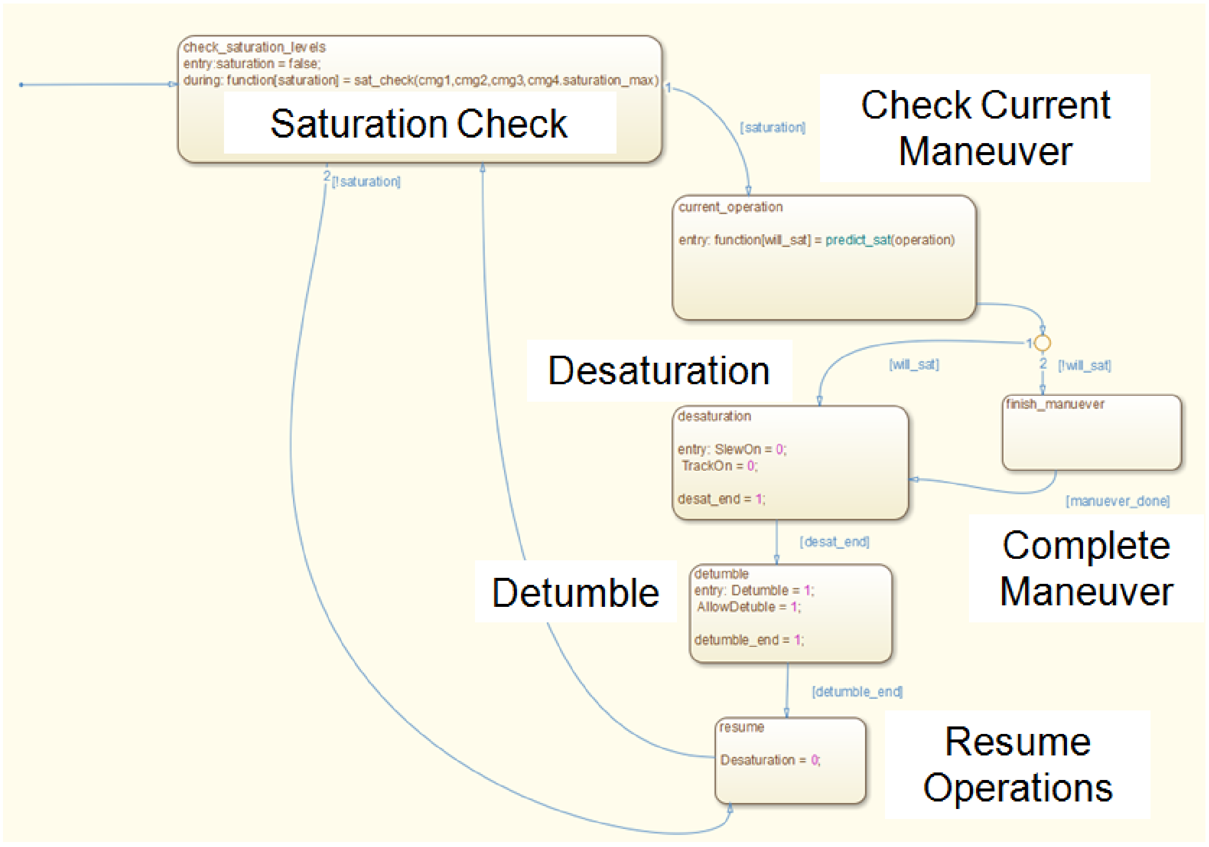


Figure 43: Preliminary desaturation controller Stateflow chart

### 5.5. *Torque Rod Controller*

The torque rod controller will be used for detumbling and for coarse attitude control when ProxOps mode is not active and the CMGs are turned off. The torque rods for Prox-1 have been designed and manufactured in-house at Georgia Tech. There are three separate rods, which are mounted so that they apply moments aligning with each of the three axes of the Prox-1 BFF as shown in Figure 44. Each rod can apply a total dipole moment of 11.16 $Am^2$ in either the positive or negative direction. This is achieved by running a constant electric current through the rod in one direction or the other. An on/off or “bang-bang” control strategy is used for each of the three torque rods, along with a choice of torque direction along each axis. The rods can be operated one at a time or simultaneously.

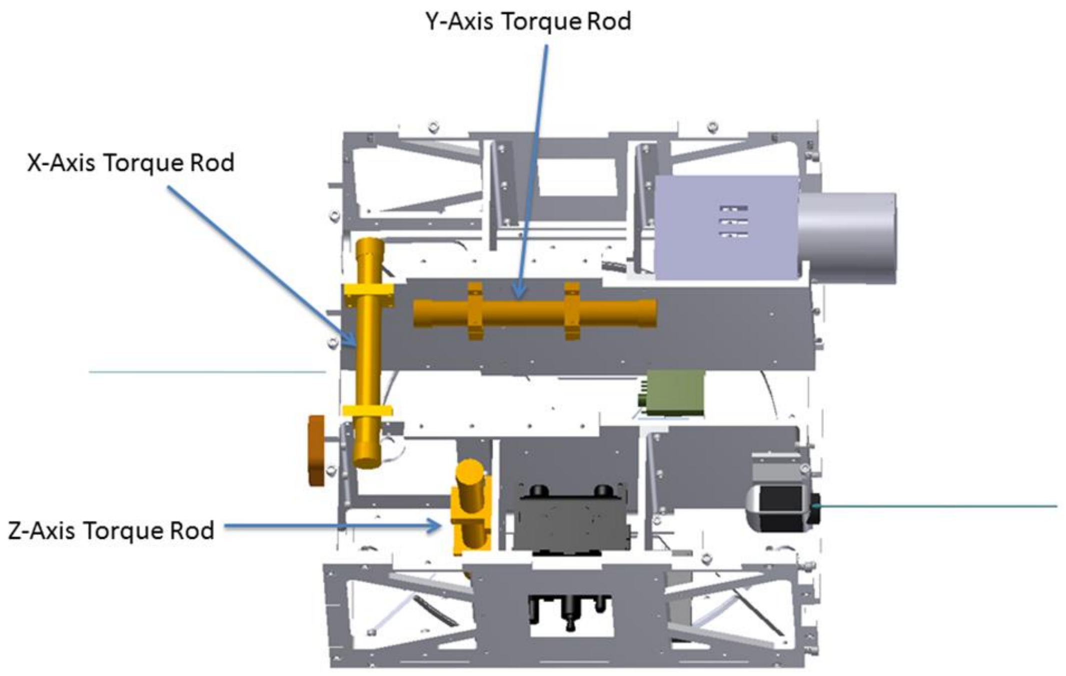


Figure 44: Torque rod mounting on Prox-1 structure

The torque rod controller has been implemented and tested with the DTC. The commands sent to the torque rods during the DTC test scenario from Section 5.3 are shown in Figure 45. The commands consist of zero, positive one, or negative one for each of the three torque rods, indicating if the current flow should be turned off, turned on in the positive direction, or turned on the negative direction. These commands will be executed by the Attitude Determination and Control Subsystem (ADCS) microcontroller, which is implemented on an Arduino board.

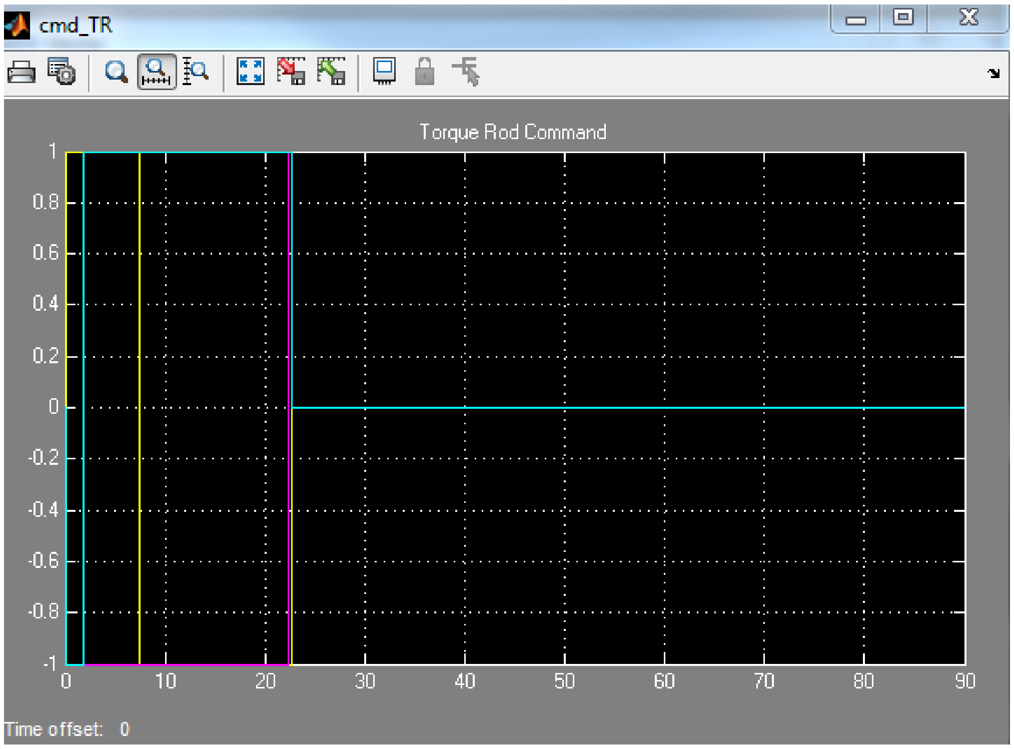


Figure 45: Torque rod commands for DTC test scenario

## 6. Six-Degree-of-Freedom Simulation Environment

Most components of the Prox-1 GN&C subsystem are developed and tested within a six degree-of-freedom (6DOF) simulation environment using MATLAB and Simulink version 2012a. This "block diagram" environment allows real-time or accelerated simulation of GN&C component interactions with sensors and actuators and with other GN&C components.

### 6.1. *Simulink Features*

Simulink provides many different block libraries to develop complicated simulation and control functions. One key Simulink block allows the developer to include embedded MATLAB code within the Simulink environment by connecting data ports that correspond to input and output variables of a MATLAB function. Simulink also provides Scope blocks, which allow the developer to view data plots changing in real time as a simulation runs. Another key Simulink tool called Stateflow allows graphical development and testing of decision logic using state machines. The Simulink environment also has the ability to connect and run simulations on external computers using xPC Target. This capability allows for accelerated simulation speeds for complicated models such as the electrical dynamics model of the CMGs. xPC Target also provides the capability to connect to embedded processors for hardware-in-the-loop testing.

## 6.2. *6DOF Environmental Framework*

The framework for the 6DOF simulation environment includes space environment models for tracking time, Earth rotation, Sun, Earth, and Moon location, and Earth magnetic field dipole. The simulation uses perturbed two-body orbital mechanics and rigid body attitude dynamics for both Prox-1 and LightSail. Environmental disturbances modeled include J2-J6 Earth oblateness effects, gravity gradient torque, aerodynamic drag, magnetic torque, solar radiation pressure, and 3rd body gravitational effects from the Sun and Moon. The states tracked by the simulation include attitude quaternions, inertial positions and velocities of both spacecraft, and relative position and velocity, all of which can be transformed between various reference frames. In addition, a MATLAB initialization script is run prior to starting the simulation to allow the user to set various constants and initial conditions for different simulation cases.

## 6.3. *Spacecraft Hardware Plant Models*

The 6DOF simulation environment also includes plant models for various sensors and actuators on Prox-1. These models provide a realistic environment for development and testing of GN&C components by estimating the physical and electrical responses of sensor and actuator hardware based on test data and manufacturer specifications. For example, the physical first order response of the thruster is estimated using transfer functions and random noise and biases are added to modeled sensor measurements to introduce errors to the system. In addition, a power production model has been developed to track the estimated amount of power produced by the solar panels during various ProxOps phases based on the position and attitude of Prox-1 relative to the Sun. Finally, an image generation tool produces simulated images of LightSail as seen by Prox-1 based on their relative positions and attitudes.

## 6.4. *GN&C Component Development Example*

Two main activities are completed within the 6DOF simulation environment: individual GN&C component development and testing of integrated GN&C components. Each GN&C component for Prox-1 is developed individually in a basic version of the simulation environment. Initially, simple equations are used in place of complex plant models, and often constant values are used to represent inputs from other GN&C components. Once basic functionality of the component is established, it is connected to the 6DOF simulation framework for further development and testing. An example of component development will be explained in this section and an example of simulation integration will be explained in the following section.

Early development of the Slew and Tracking Controller (STC) involved using a basic standalone Simulink block diagram, shown in Figure 46, with a constant input for desired attitude quaternion and zero vector inputs for desired angular velocity and acceleration. This initial simulation included basic unperturbed angular kinematic equations and assumed perfect execution of the desired torque commands.

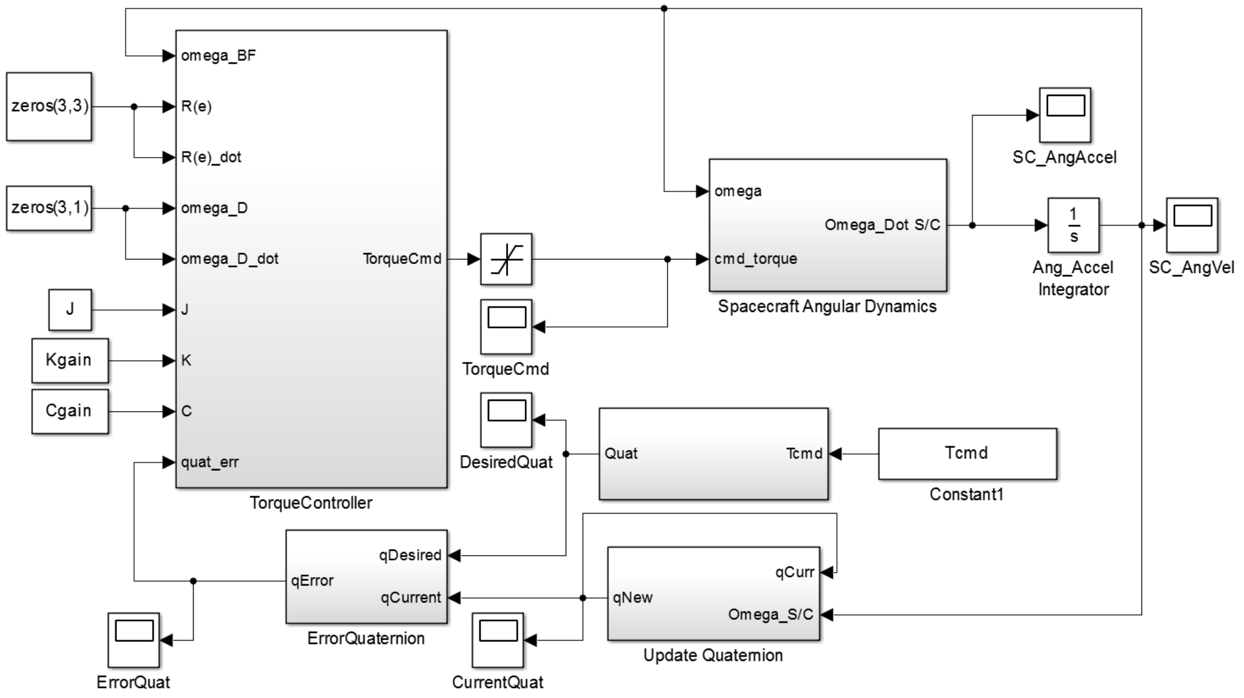


Figure 46: Standalone Simulink block diagram for slew and tracking controller development

Once a basic torque control law was developed and tested, additional logic was written to transform other expected inputs, such as a desired vector direction to fire the thruster (Tcmd), into quaternion representation. In subsequent stages of development, additional logic to allow tracking of LightSail and preferential solar panel pointing was added.

As additional complexities were added to the STC, it became apparent that simply developing additional logic using more Simulink blocks or complicated embedded MATLAB functions would lead to an unwieldy design solution. Stateflow provided an intuitive and elegant way to overcome this issue. A Stateflow chart was developed, as described in Section 5.1, and was integrated as a block within the Simulink diagram for the STC, as shown in Figure 47.

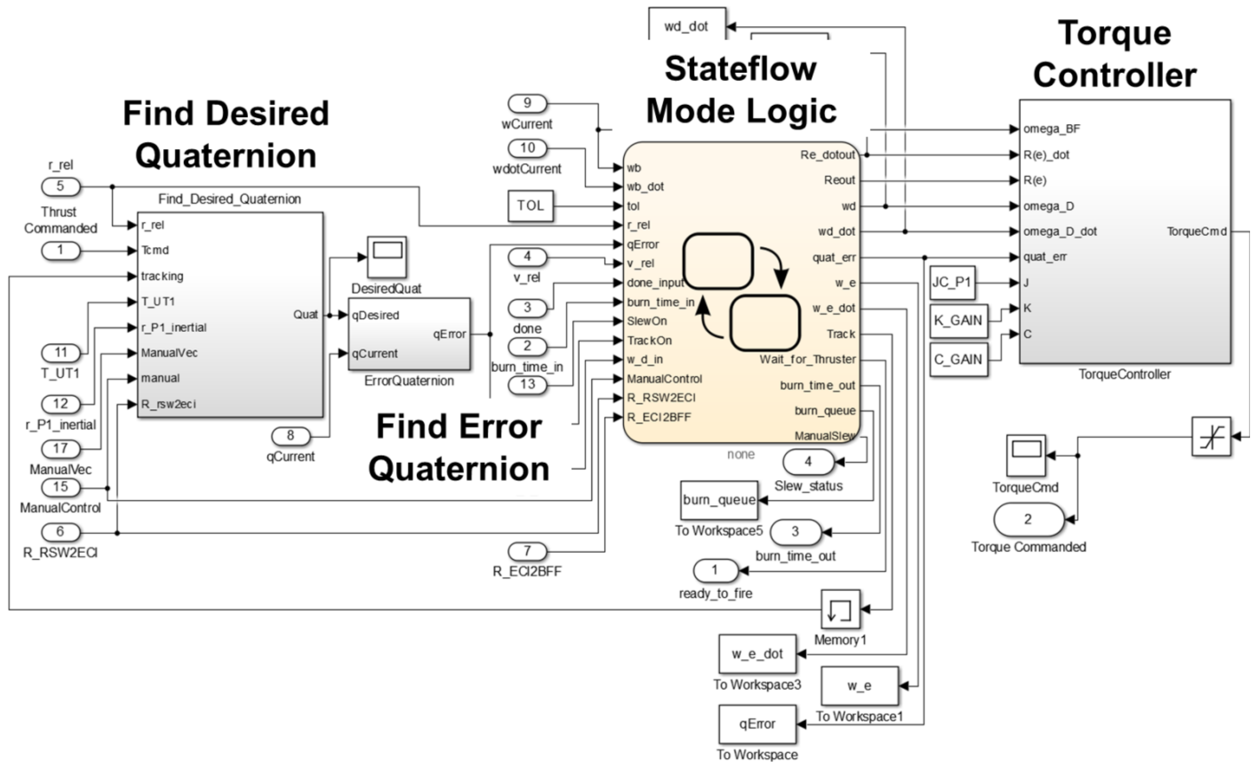


Figure 47: Simulink diagram for the STC model. The Stateflow chart is a tan block in the middle

## 6.5. *GN&C Simulation Integration Example*

Once an individual GN&C component is developed, it is ready to be integrated into the 6DOF Simulation environment along with other components. During this integration process, further developments and modifications are made to each of the components to ensure that they work well together. Also, as components are linked together, test simulation cases are run to ensure that each component is performing as expected. An example follows to illustrate the process of component integration.

In this example, to achieve an integrated "master simulation" in the 6DOF environment, the user begins with the standalone simulation used to test the APF guidance algorithms. This initial sim, shown in Figure 48, contains the foundation for the 6DOF sim: the Prox-1 and LightSail plant and environment models. The only GN&C component in this version of the sim is the APF Guidance block, which receives relative position and velocity inputs directly from the spacecraft plant (rather than from navigation) and outputs thrust commands directly to the thruster plant (assuming instantaneous slewing to the proper attitude). Although these simplifications do not completely represent the reality of operating Prox-1, they are sufficient to develop a first-cut functional guidance block. To account for effects of other components as they are added, the guidance block is continually changed and tuned throughout the integration process.

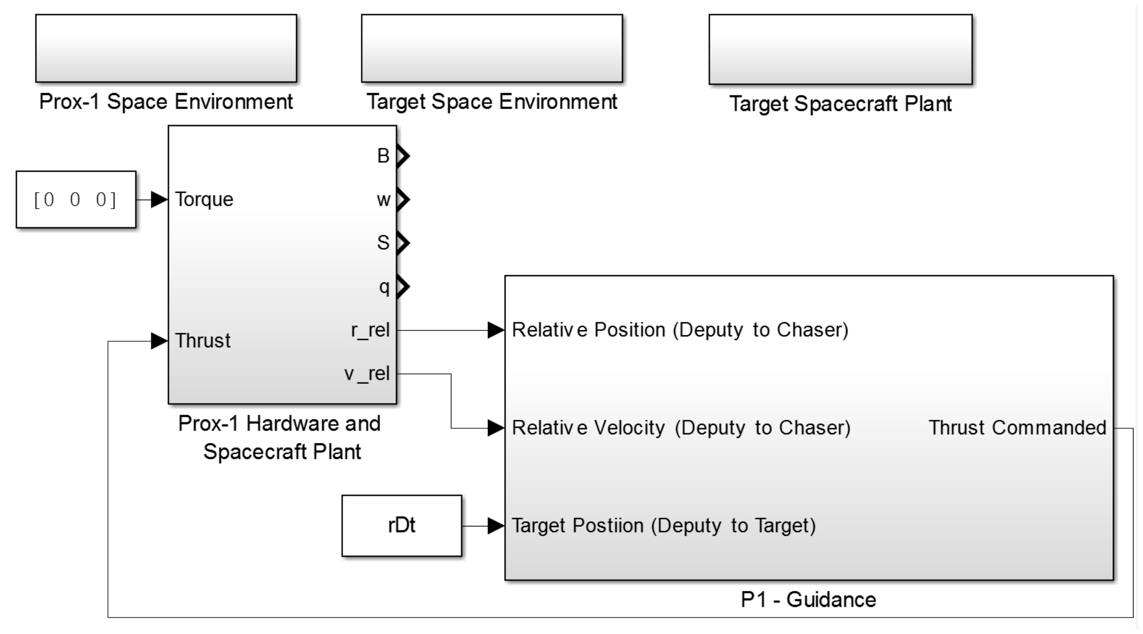

Prox-1 Space Environment
Target Space Environment
Target Spacecraft Plant
[0 0 0]
Torque
Thrust
B
w
S
q
r_rel
v_rel
Prox-1 Hardware and Spacecraft Plant
Relative Position (Deputy to Chaser)
Relative Velocity (Deputy to Chaser)
Thrust Commanded
rDt
Target Postiion (Deputy to Target)
P1 - Guidance

Figure 48: Standalone guidance testing simulation

The next step in the integration process involves bringing in the STC and the Thruster Controller (TC). The resulting integrated sim is shown in Figure 49. Note that originally the TC was designed as a GN&C software component for the hydrazine thruster design, but it has since been replaced by a hardware plant model for a propulsion subsystem microcontroller after changing to a cold gas thruster design. The outputs of the Guidance block are altered to include both a thrust command (unit vector direction) and a burn time associated with each burn command; both of these are sent to the STC. The STC also takes inputs of inertial and relative position/velocity directly from the spacecraft plant, as well as the sun vector directly from the space environment models, again bypassing the navigation filters.

Torque commands from the STC are output directly to the spacecraft plant model (bypassing the complex and computationally intensive CMG model). The “ready-to-fire” flag and burn time are sent from STC into TC, which sends thruster firing commands to the thruster plant model and feeds back a “done” flag to STC that indicates when the firing is complete.

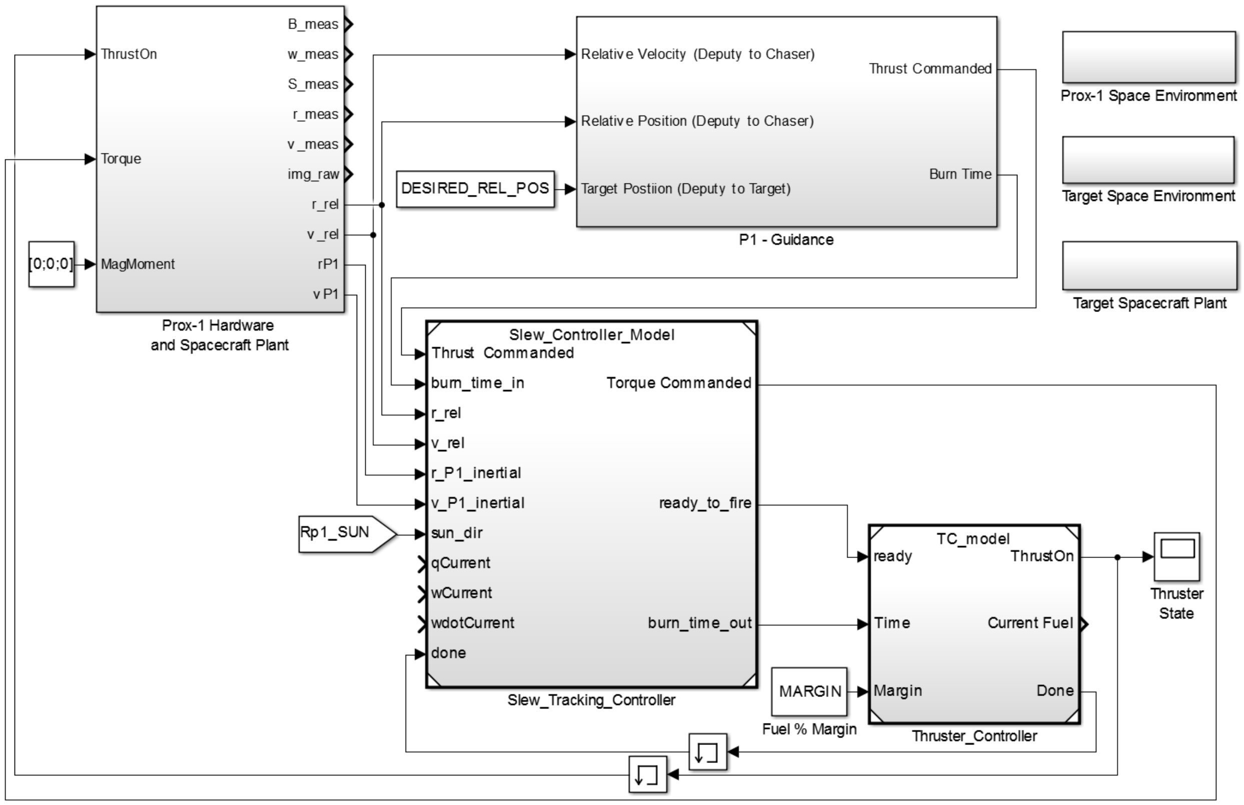


Figure 49: Integrated sim including guidance, STC, and thruster controller components

Once all of these connections are made, an integrated functional test of the simulation is run and scope outputs such as those in Figure 50 are used to determine if the components are functioning properly. From top to bottom, the plots in Figure 50 show the relative position between Prox-1 and LightSail in meters (magenta = along-track, cyan = cross-track, yellow = radial), the thruster state (1 = on, 0 = off), and Prox-1's attitude quaternion (a unitless 4 element vector with values ranging between -1 and 1). The timescale for all of the scope outputs is in seconds.

In this simulation example, the relative position starts with a 120 meter offset in the along-track direction, which closes to 50 meters as the guidance algorithms command thruster burns. The dips in the attitude quaternion plot represent slew maneuvers that point the thruster in the desired direction to execute burns commanded by the guidance algorithms. These burns are then shown as square pulses on the thruster state plot. After many tweaks, the desired result is achieved, and the next component can be integrated. At each stage of integration, it should be verified that all of the components perform in the expected manner. If they do not, design changes are made and the integration cycle is iterated. Although the results shown here assume a hydrazine thruster, the system has been shown to work with the new cold gas thruster design.

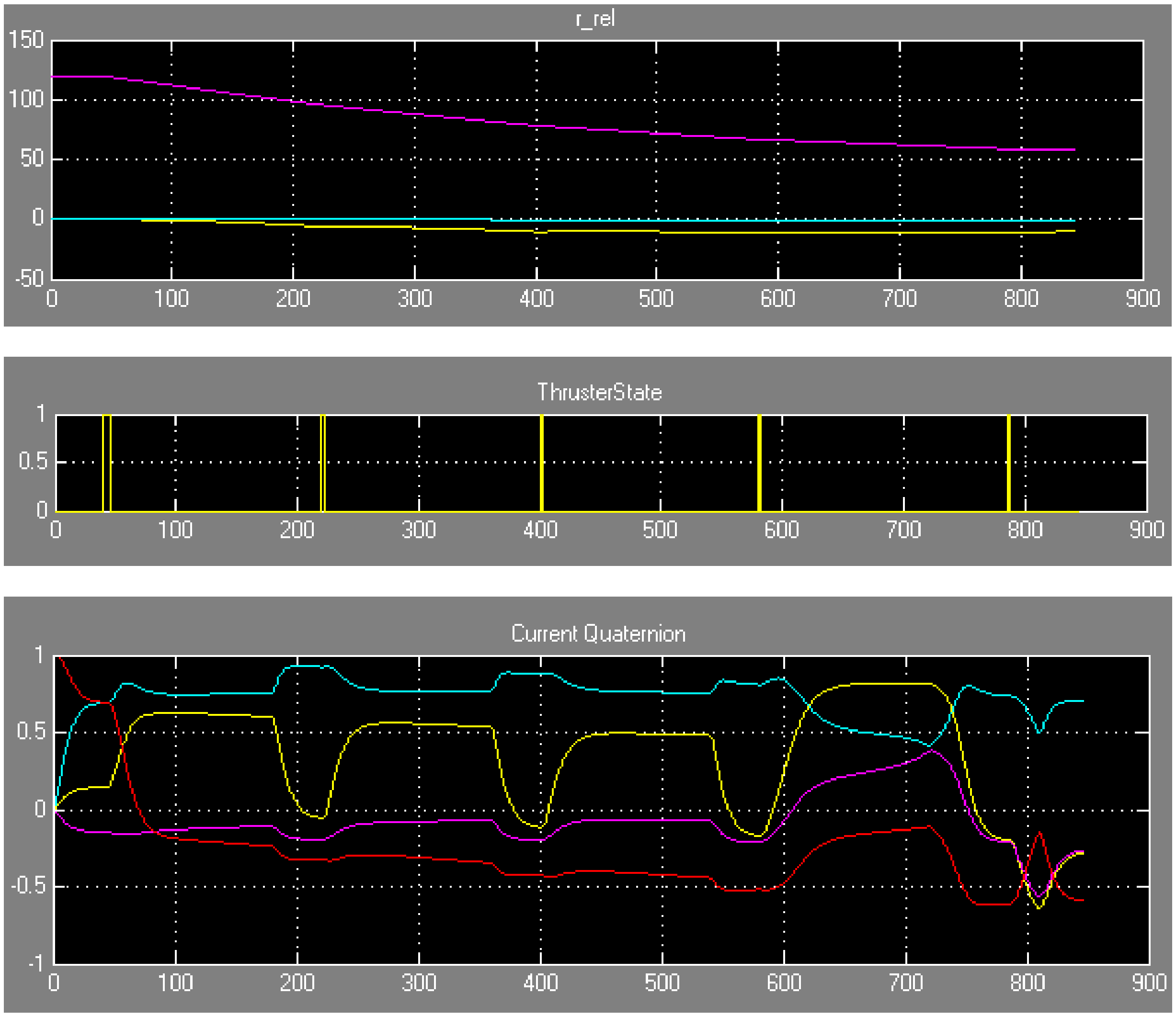


Figure 50: Integrated sim results for Guidance/STC/TC

Finally, the DTC is added to the simulation and mode logic is developed in a Stateflow diagram, resulting in the master simulation shown in Figure 51. The DTC block takes inputs from the spacecraft plant and environment models and outputs a magnetic moment directly to the torque rod plant model (bypassing the torque rod controller in this example simulation). The plant and environment models, boxed in red in Figure 51 are only used for simulation and will not be coded into flight software.

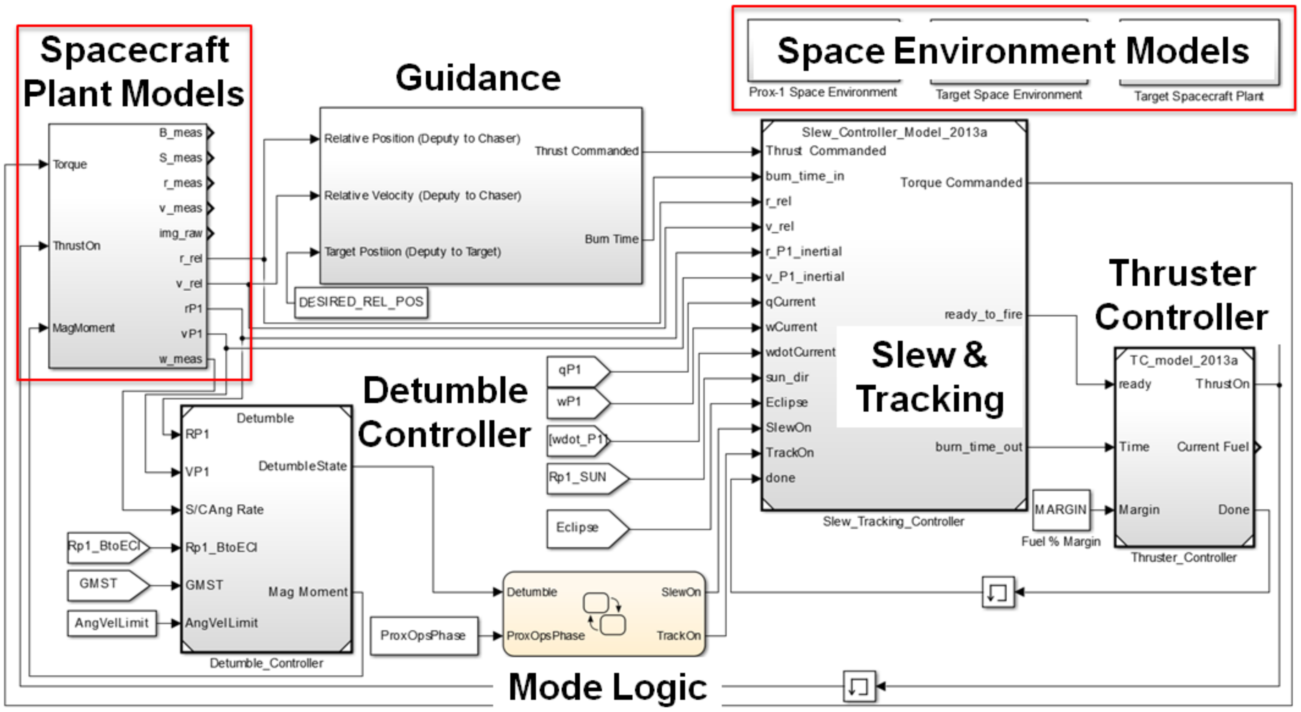


Figure 51: Master simulation including guidance, STC, TC, DTC, and mode logic

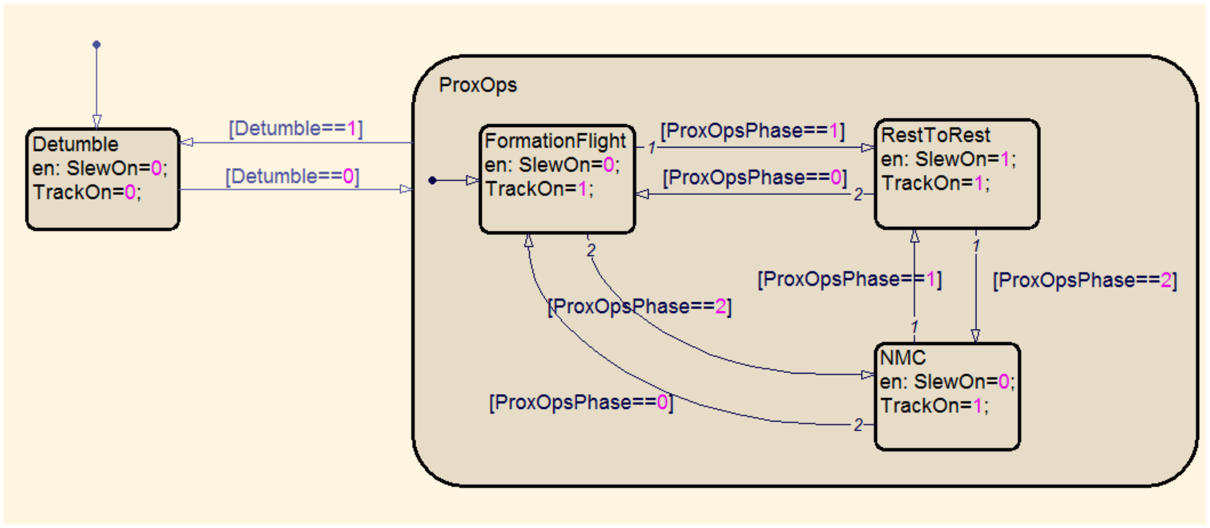


Figure 52: Stateflow mode logic for integrated master simulation

The tan block in the middle of the master simulation is the Stateflow GN&C mode logic, shown in Figure 52. In this example, there are two states in the GN&C mode logic: Detumble and ProxOps. Additional GN&C modes are added as development of the integrated autonomous system continues. Detumble is the initially active state (to damp any high initial angular rates), and once angular rates have been damped to within acceptable limits, the Detumble flag is set to zero and ProxOps mode is activated. ProxOps mode enables or disables both slew and tracking based on which ProxOps phase is active (phase 0 = FormationFlight, phase 1 = RestToRest, phase 2 = NMC). Both the Detumble and the ProxOpsPhase flags are inputs to the mode logic block, and SlewOn/TrackOn are outputs. In order for the STC to enter its SlewToThrust mode, the SlewOn flag must be set to 1. To enter the Track mode, the TrackOn flag must be set to 1. If both flags are set to 1, STC will perform commanded slew maneuvers then return to tracking LightSail. If neither flag is set to 1, STC will remain in Standby mode.

## 6.6. *Current Version of the Master Simulation*

In its current version as of December 2014, the Prox-1 6DOF master simulation is slightly more complicated than the simulation presented in the previous section. The Simulink diagram, shown in Figure 53, now includes additional components: the RelOD filter, the torque rod (TR) controller, target acquisition controller, and flight software emulator. Gray blocks are only used for simulation purposes, while white blocks will be autocoded into GN&C flight software (FSW). For configuration control, most of the GN&C component blocks are integrated into the master simulation using model reference blocks. These allow each component to be saved as a separate Simulink file that can be integrated into multiple master simulations. Another change in the current simulation is the addition of some feedback loops between GN&C components, such as Boolean variables that are fed back from the STC and mode logic blocks into the TAC block.

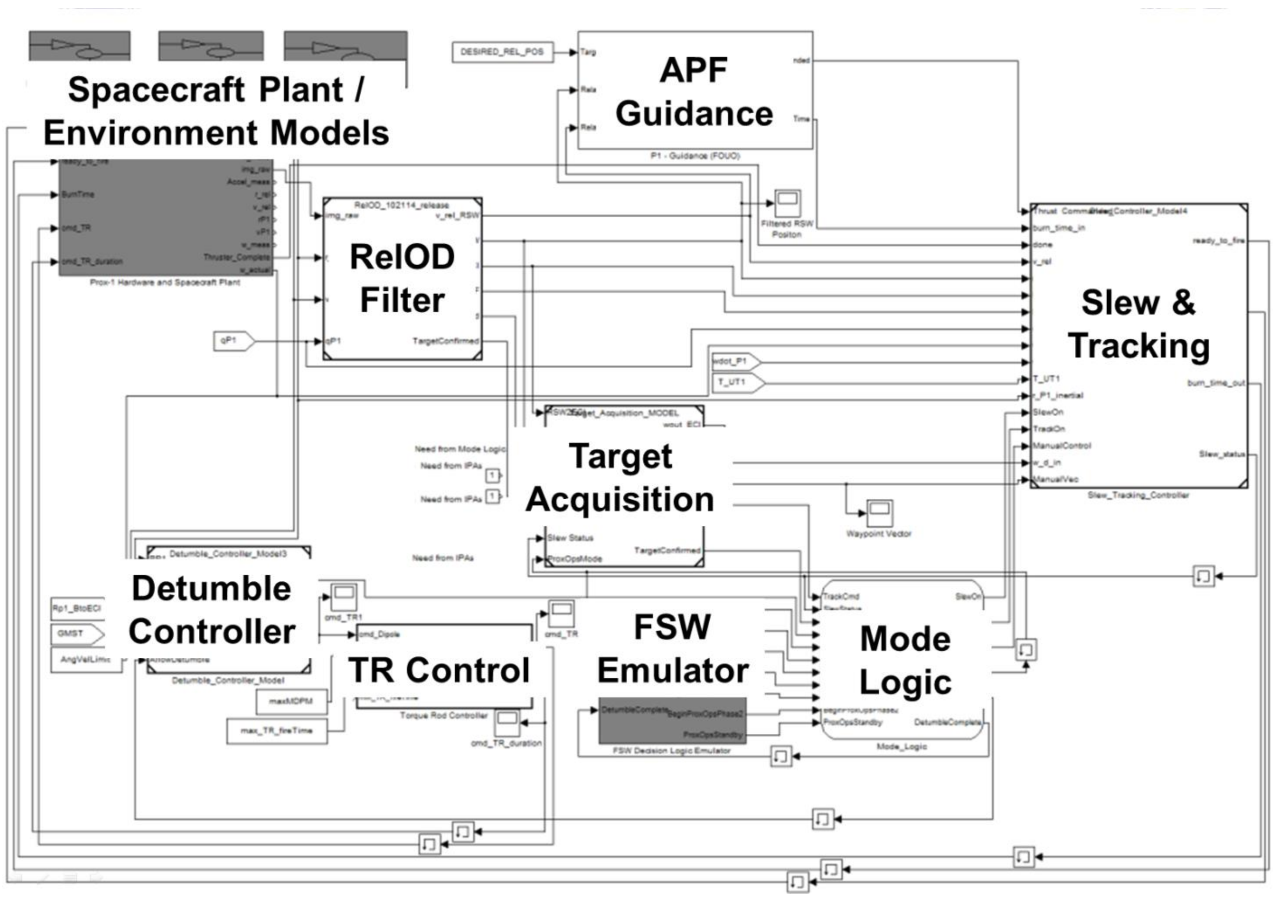


Figure 53: Current version of the Prox-1 6DOF master simulation

The mode logic block has also been updated as shown in Figure 54 to more accurately represent the way that Prox-1 will handle mode transitions in flight. There are four primary modes: Startup, NormalMode, ProxOpsMode, and SafeMode. Entry into and transition between these modes is commanded by FSW external to the GN&C algorithms. Currently, NormalMode includes only the Detumble state and a Standby state, though it will eventually also encompass coarse TR attitude control. ProxOps mode includes two states: ProxOpsActive and TargetAcquisition. TargetAcquisition becomes active whenever the RelOD filter determines that the RSO is not in the imager FOV and the TAC is activated until the target is found and the RelOD filter converges. When a converged RelOD solution is available, the ProxOpsActive state becomes active. There are 4 possible phases within this state: Standby (Phase -1), Formation Flight (Phase 0), RestToRest (Phase 1), and NMC (Phase 2). Transition between any of these four phases can be commanded by FSW depending on feedback received from GN&C and from ground commands. Finally, SafeMode simply disables the STC and puts everything in standby. Eventually, a survival TR attitude control algorithm will be implemented in SafeMode to maintain solar panel and antenna pointing for power production and communication respectively.

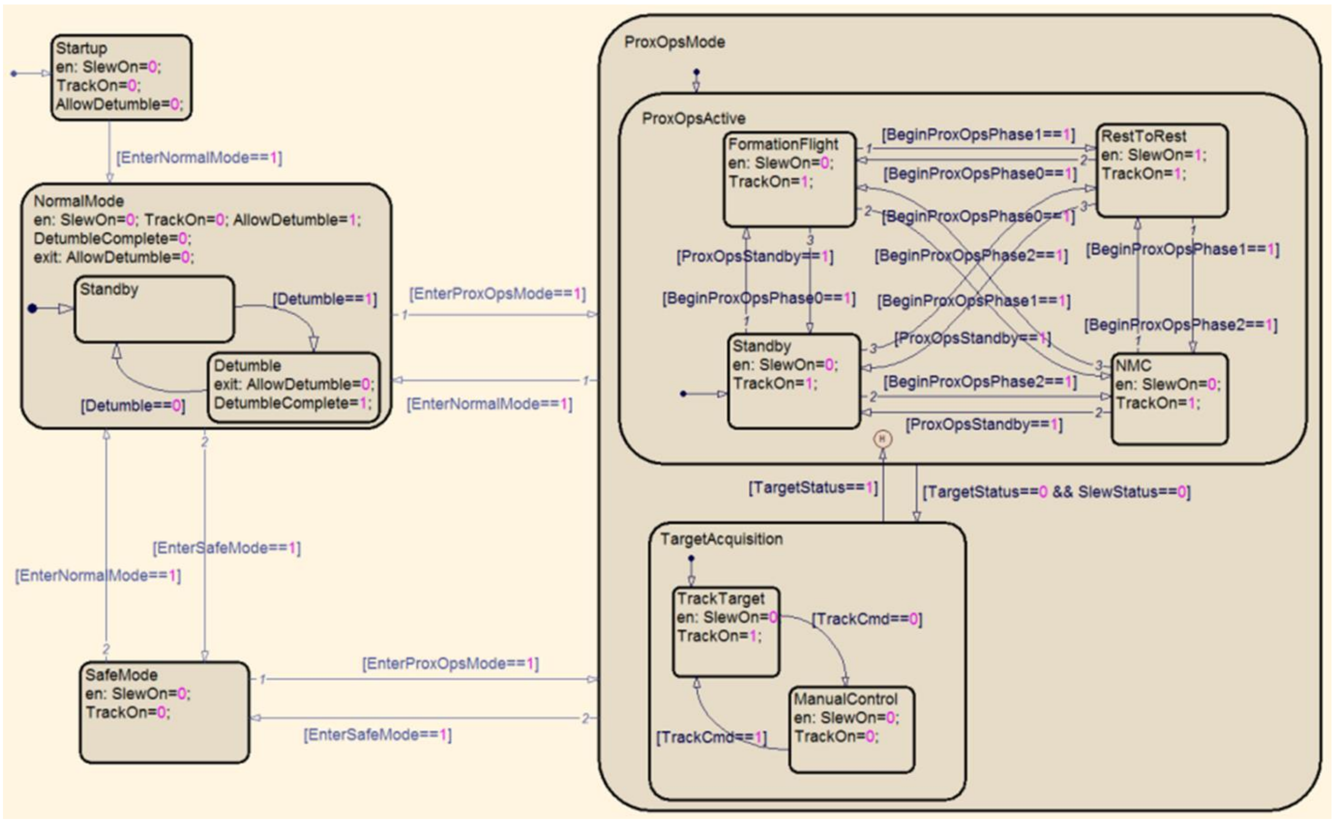


Figure 54: Prox-1 mode logic Stateflow chart for current master simulation

```
%Safe Mode
if CurrentMode==0

%Normal Mode
elseif CurrentMode==1
   EnterNormalMode=1;
   %Transition to ProxOpsMode once Detumble is
complete
   if DetumbleComplete==1;
      EnterNormalMode=0;
      CurrentMode=2;
   end
%ProxOps Mode
elseif CurrentMode==2
   EnterProxOpsMode=1;
   if ProxOpsMode==true
      %Prox Ops Phase 0 (Stationkeeping)
      ProxOpsPhase=0;
      BeginProxOpsPhase0=1;
      if ProxOpsPhase==0
         %need conditions to move to other phases
      end
   end
%Error State
else
   error('Bad mode state: possible values are 0=safe;
1=normal; 2=prox ops)');
end
```

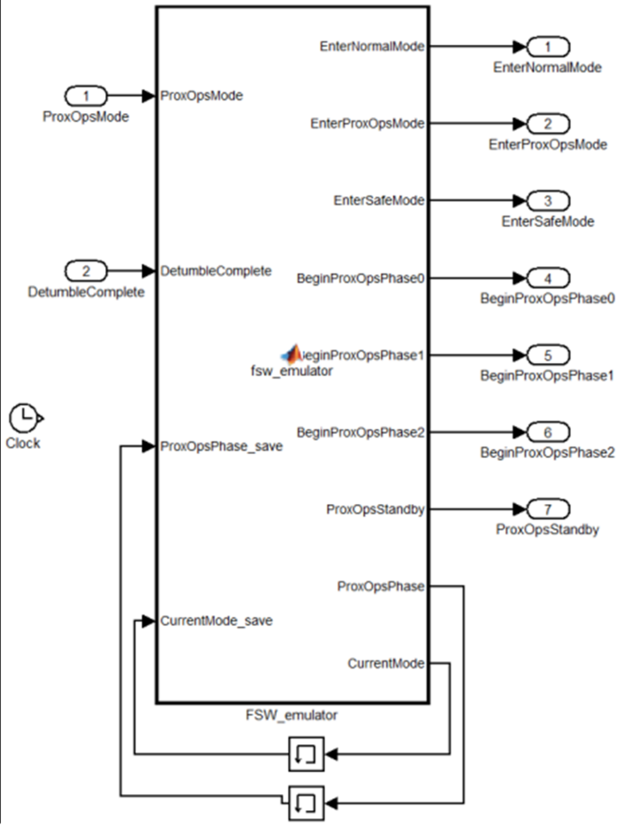


Figure 55: Flight software emulator for master simulation mode logic commands

The FSW emulator shown in Figure 55 was developed as a simple embedded MATLAB block to send simulated commands to the GN&C mode logic. It contains simple if statements that transition between modes and ProxOps phases. Currently, the FSW emulator commands GN&C to change from NormalMode to ProxOpsMode once Detumble is completed and to begin Stationkeeping (ProxOps Phase 0) upon entering ProxOpsMode. This can be used as a template for the FSW team as they develop more complex mode logic that takes into account information from all of the subsystems onboard Prox-1 to make mode and phase decisions during flight.

## 7. Autocoding to Flight Software

The Simulink design of the GN&C algorithms for Prox-1 is autocoded into C/C++ using Simulink Coder for integration with FSW. Some modifications are made to ensure the models are codable, such as avoiding the use of incompatible MATLAB functions. This process of GN&C algorithm development and integration in Simulink and later autocoding into flight software has been pioneered by the Orion spacecraft team at NASA's Johnson Space Center in Houston, Texas [17].

## 7.1. *Autocoding Process*

The process for autocoding from a Simulink master simulation to C code is described in this section based on work completed by Prox-1 team members Jacob Sussman and Meet Patel. This process is illustrated at a high level by the flowchart in Figure 56.

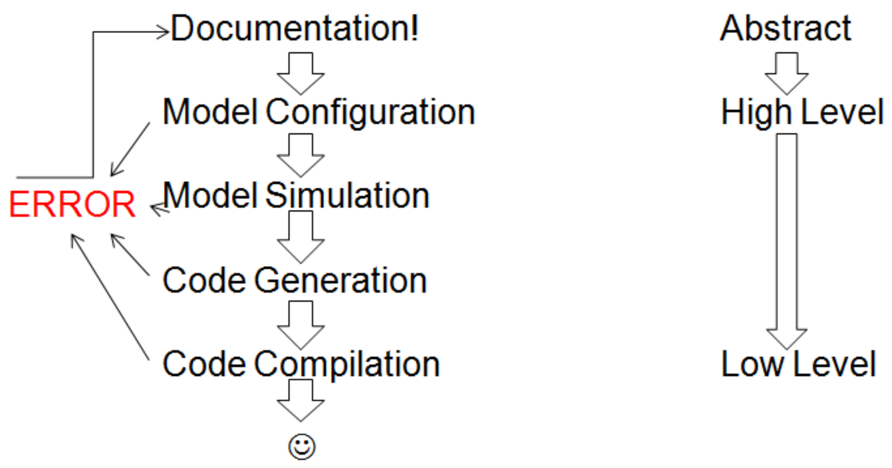


Figure 56: Autocoding process flowchart

The first step in this process is documentation, which is critical for capturing detailed instructions and lessons learned such as best practices and how to deal with common errors. Next is model configuration, which involves setting a multitude of parameters within the Simulink model to be autocoded. Proper model configuration allows for smoother model simulation and code generation. Also, these configuration parameters can optimize the resulting generated code to run on a specific embedded processor. Once the proper configuration settings are determined, they can be saved as a separate file and maintained using configuration control. Developers can then apply these standard configuration settings to any model by importing that file in Simulink.

The next step is model simulation. A Simulink model cannot be autocoded if it does not run properly in simulation. After it is verified that the simulation model can run and provides the desired outputs, any blocks included from separate files as model references are copied and pasted into a single Simulink model to simplify the autocoding process. The model is then reconfigured so that all GN&C blocks to be autocoded are combined into a single monolithic GN&C block as shown in Figure 57. Before autocoding begins, the reconfigured simulation is run again to ensure that no changes in performance or outputs have been introduced by the reconfiguration process.

Once all of these changes have been made, the model is ready for code generation. This is the stage in the process where most errors occur, and each error must be understood and corrected. The complete detailed set of instructions for autocoding is included in a separate technical memorandum which explains why many errors occur and how to avoid them. After the code is generated, it must be compiled to run on the BeagleBoard XM flight computer. Finally, the compiled code should be run to verify that the outputs match those of the Simulink model.

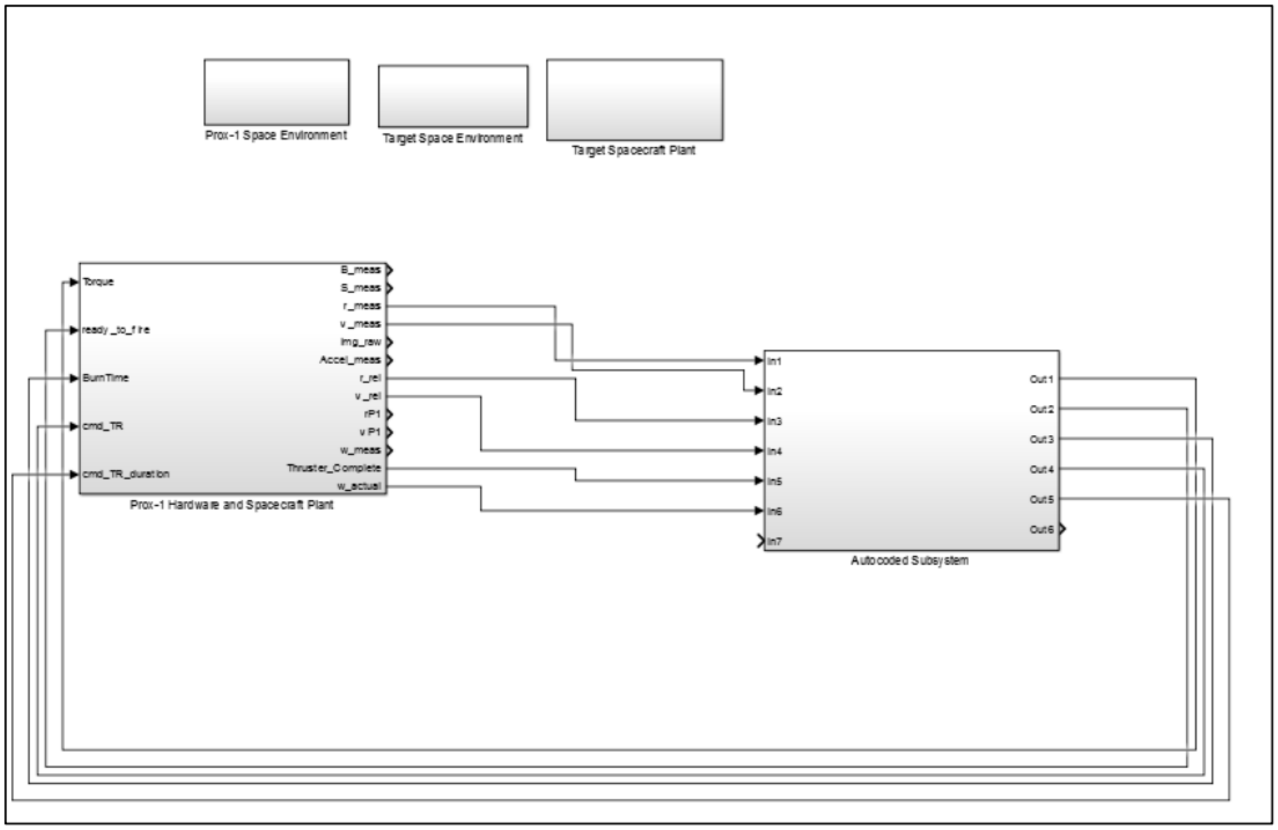


Figure 57: Master sim model prepared for autocoding by creating a single GN&C block

## 7.2. *Integration with Flight Software*

After the C code has been generated and compiled for use on the BeagleBoard, it must be integrated with FSW for use on-board Prox-1. In the current understanding of the Prox-1 GN&C and FSW teams, the GN&C code will be integrated as a monolithic sub-routine that is called during each timestep. This process is illustrated in Figure 58, which shows that external variables are collected by FSW from various sensors (1) and sent into the GN&C code (2), which then operates in various modes. GN&C returns commands to FSW (3), which then distributes those commands to actuator hardware such as the Propulsion and ADCS microcontrollers. In (2) and (3) FSW and GN&C will also exchange mode logic variables, such as a command to enter ProxOpsMode from the ground or an indication from GN&C that an NMC command has been successfully executed.

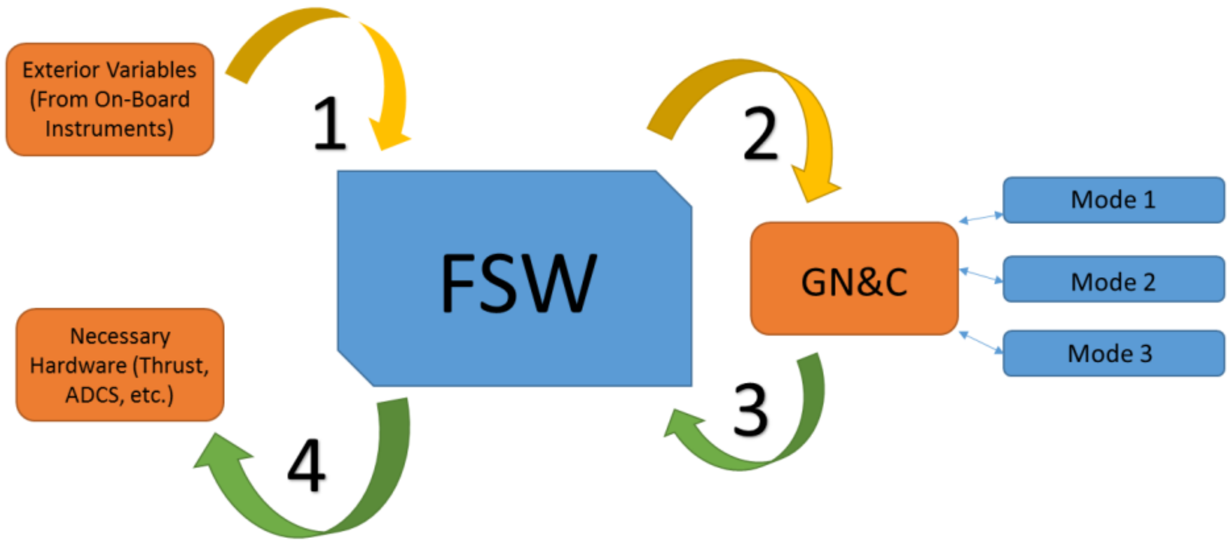

Exterior Variables (From On-Board Instruments)
1
2
FSW
GN&C
Mode 1
Mode 2
Mode 3
3
Necessary Hardware (Thrust, ADCS, etc.)
4

Figure 58: GN&C/FSW interface illustration

## 8. Forward Work

As of this writing, the Prox-1 GN&C team is completing the development of various components in Simulink, including the inertial attitude determination filter, desaturation controller, and torque rod coarse attitude control. Other components are currently in the integration phase, such as the RelOD filter, Target Acquisition Controller, and ROE guidance algorithms. Updates and upgrades are also being made to other components such as STC as development and integration move forward. When all components are in their final form, the entire GN&C subsystem will be tuned and tested rigorously within Simulink. Also, Fault Detection, Isolation, and Recovery (FDIR) methods will be implemented to actively take into account imperfections and failures in the GN&C system and correct for them in real time.

As new versions of the integrated simulation are completed, they will be autocoded into C/C++ and tested to ensure that they produce similar results to the Simulink models. A preliminary version of the Prox-1 GN&C algorithms has already been autocoded and the compiled version of the code has been run on the BeagleBoard. Next, these algorithms will be integrated with FSW code and tested in connection with actual flight hardware, including sensors, actuators, and subsystem microcontrollers. Then, an integrated "Day in the Life" test will bring together all of Prox-1's hardware and software systems. As FDIR methods are implemented in the GN&C system, additional tests using a hardware-in-the-loop platform will verify that the methods are able to effectively deal with system errors. Finally, an optional test will place mock-ups of Prox-1 and LightSail on an air-bearing floor at NASA's Marshall Space Flight Center in Hunstville, Alabama to evaluate spacecraft system performance in a full operational scenario.

## 9. Conclusion

A complete autonomous Guidance, Navigation, and Control (GN&C) subsystem is under development for the Prox-1 satellite project. This subsystem enables the spacecraft to perform inertial and relative navigation using various sensors, plan ProxOps maneuvers to approach and circumnavigate an uncooperative target, and execute those maneuvers using various actuators. Each GN&C component is developed within the MATLAB/Simulink 6DOF simulation environment and integrated together to create a complete design solution. By autocoding this design into C/C++ with Simulink Coder, integration and testing of GN&C algorithms are greatly simplified. Finally, hardware-in-the-loop testing will validate that the algorithms can be executed as intended on flight-like hardware and FDIR methods will enable the GN&C system to actively detect and account for imperfections in the system.

## References

[1] Schulte, P. and Spencer, D., "Development of an Integrated Spacecraft Guidance, Navigation, & Control Subsystem for Automated Proximity Operations," *65th International Astronautical Congress*, Toronto, ON, Canada, Sep-Oct 2014.

[2] Zappulla, R., "Prox-1 Guidance, Navigation, & Control Algorithms," Georgia Institute of Technology, School of Aerospace Engineering, Atlanta, GA, May 2013.

[3] Chait, S. and Spencer, D., "Prox-1: Automated Trajectory Control for On-Orbit Inspection," AAS 14-066, *37$^{th}$ Annual AAS Guidance & Control Conference*, Breckenridge, Colorado, Jan-Feb 2014.

[4] Biddy, C., and Svitek, T., "LightSail-1 Solar sail Design and Qualification," *41st Aerospace Mechanisms Symposium*, Pasadena, California, May 2012.

[5] Arestie, S, Lightsey, E.G, and Hudson, B, "Development of a Modular, Cold Gas Propulsion System for Small Satellite Applications," *Journal of Small Satellites*, Vol. 1, No. 2, October 2012, pp. 63-74.

[6] Walker, L. "Automated Proximity Operations Using Image-Based Relative Navigation," SSC12-VII-3, *26th Annual USU/AIAA Conference on Small Satellites,* Logan, Utah, August 2012.

[7] Spencer, D. "Automated Relative Proximity Operations Trajectory Control Using Relative Orbital Elements," Ph.D. Dissertation, Guggenheim School of Aerospace Engineering, Georgia Institute of Technology (pending).

[8] Bate, R.R., Mueller, D.D., and White, J.E., *Fundamentals of Astrodynamics*, New York: Dover Publications, 1971.

[9] Crassidis, J.L. and Junkins, J.L., *Optimal Estimation of Dynamic Systems*, Boca Raton, FL: CRC Press, LLC, 2004.

[10] Bellet, E., and Spencer, D., "Detection and Localization of a Target on a Thermal Image Using a 'Blobber' Algorithm," Georgia Institute of Technology, School of Aerospace Engineering, Atlanta, GA 2011.

[11] Vedie, N., Spencer, D., and Walker, L., "Autonomous Ocean Current Geolocation from Orbit," *Journal of Small Satellites,* Vol. 2, No. 1, July 2013.

[12] Martinson, N., Munoz, J.D., and Wiens, G.J., "A new method of guidance control for autonomous rendezvous in a cluttered space environment," *AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, 2007.

[13] Lovell, T.A. and Spencer, D.A., "Maneuver Design Using Relative Orbital Elements," *Journal of the Astronautical Sciences*, accepted, Sept 2014.

[14] Shuster, M.D., "Survey of Attitude Representations," *Journal of the Astronautical Sciences*, Vol. 41, pp. 439-517, 1993.

[15] Hughes, P.C., *Spacecraft Attitude Dynamics*, Mineola, NY: Dover Publications, 2004.

[16] Avanzini, G. and Giulietti, F., "Magnetic Detumbling of a Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 4, July-August 2012.

[17] Jackson, M.C. and Henry, J.R., "Orion GN&C Model Based Development: Experience and Lessons Learned," AIAA-2012-5036, *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota, August 2012.