Acta Astronautica 118 (2016) 168–186

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Development of an integrated spacecraft Guidance, Navigation, & Control subsystem for automated proximity operations ☆

Peter Z. Schulte *, David A. Spencer

Space Systems Design Laboratory, Georgia Institute of Technology, 270 Ferst Dr., Atlanta, GA 30332-0150, USA

ARTICLE INFO

Article history:
Received 20 October 2014
Received in revised form
11 August 2015
Accepted 12 October 2015
Available online 23 October 2015

Keywords:
Spacecraft Guidance Navigation & Control
Small satellites
Proximity operations
MATLAB/Simulink
Six degree of freedom simulation
Flight software

ABSTRACT

This paper describes the development and validation process of a highly automated Guidance, Navigation, & Control subsystem for a small satellite on-orbit inspection application, enabling proximity operations without human-in-the-loop interaction. The paper focuses on the integration and testing of Guidance, Navigation, & Control 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 subsystem algorithms must be defined to achieve the automation goal. The Prox-1 mission is currently under development within the Space Systems Design Laboratory at the Georgia Institute of Technology. The mission involves the characterization of new small satellite component technologies, deployment of the LightSail 3U CubeSat, entering into a trailing orbit relative to LightSail using ground-in-the-loop commands, and demonstration of automated proximity operations 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 Guidance, Navigation, & Control 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, thruster firing, and target acquisition/recovery. All Guidance, Navigation, & Control algorithms are developed in a MATLAB/Simulink six degree-of-freedom simulation environment and are integrated using decision logic to autonomously determine when actions should be performed. The complexity of this decision logic is the primary challenge of the automated process, and the Stateflow tool in Simulink is used to establish logical relationships and manage data flow between each of the individual hardware and software components. Once the integrated simulation is fully developed in MATLAB/Simulink, the algorithms are autocoded to C/C++ and integrated into flight software. Hardware-in-the-loop testing provides validation of the Guidance, Navigation, & Control subsystem performance.

☆ This paper was presented during the 65th International Astronautical Congress in Toronto, Canada.

* Corresponding author.
E-mail addresses: pzschulte@gatech.edu (P.Z. Schulte), david.spencer@aerospace.gatech.edu (D.A. Spencer).

http://dx.doi.org/10.1016/j.actaastro.2015.10.010
0094-5765/

1. Introduction

There is an increased need for autonomous Proximity Operations (ProxOps) in such applications as rendezvous and docking for human space exploration and sample return missions, satellite servicing and on-orbit inspection, construction, and debris mitigation. Many autonomous ProxOps systems have been developed and proven in flight for large and complex systems [1–3] and are increasingly being used with small satellites [4,5]. Development of spacecraft Guidance, Navigation, and Control (GN&C) software architectures often involves bringing together individually developed algorithms and evaluating them using both simulation and hardware-in-the-loop testing [6]. The process of spacecraft GN&C algorithm development and integration using Model-Based Design in Simulink and later autocoding into flight software (FSW) has been exercised by the Orion spacecraft team at NASA's Johnson Space Center [7]. Even though Orion uses autocode from Simulink as a cornerstone of its GN&C FSW validation, many of its GN&C algorithms were originally prototyped in C code and later converted to Simulink block diagrams. Previous uses of selective autocoding of GN&C algorithms into flight software include the design of the Space Maneuver Vehicle by Boeing [8], the German Phoenix project by EADS [9], and the Cluster Flight Application by Emergent Space Technologies [10]. In each of these cases, some of the GN&C algorithms and software functions were autocoded from Simulink while others were hand-coded into the final FSW.

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 a non-cooperative target for an on-orbit inspection application [11]. The Prox-1 team is advancing the state-of-the-art by developing all of its GN&C algorithms completely within an integrated MATLAB/Simulink environment rather than integrating externally developed algorithms. The team is also autocoding the entire GN&C flight software rather than hand-picking which algorithms to autocode. The single autocoded GN&C software block is then validated using hardware-in-the-loop testing. By utilizing this integrated approach for GN&C software development and testing, the reliability and robustness of the software is demonstrated, yielding a fully validated GN&C subsystem for flight. This paper will introduce the Prox-1 mission, describe the GN&C algorithms and hardware components, and present the streamlined development and implementation approach that is being utilized by the Prox-1 GN&C subsystem team.

2. Prox-1 mission description

The Prox-1 mission begins with deployment of Prox-1 as a secondary payload from the Space Test Program-2 launch on the SpaceX Falcon Heavy launch vehicle. Once deployed, body-mounted solar panels will begin battery charging, and the flight processor will boot up. Next, magnetic torque rods will detumble the spacecraft. Once angular rates have been nulled, a spacecraft checkout phase ensues where on-orbit functionality is characterized, including the flight qualification of new subsystem technologies including the Honeybee Robotics TORC control moment gyros (CMGs) [12], the Jet Propulsion Laboratory Advanced Micro-Sun Sensor [13], the FLIR/Arizona State University microbolometer thermal imager [14], and the University of Texas at Austin 3D-printed cold gas propulsion unit [15].

The target spacecraft for the Prox-1 mission is The Planetary Society's LightSail, a 3U CubeSat that is designed to demonstrate solar sail technology [16]. LightSail is stowed inside of Prox-1 during launch and is deployed on-orbit using a Poly Picosat Orbital Deployer (P-POD) device following Prox-1 checkout. 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. After orbit determination has been completed, a series of ground-in-the-loop maneuvers will be performed to complete Prox-1 rendezvous with LightSail to within visual sensor range of 100–200 m. At this point, automated ProxOps begin. The Prox-1 GN&C subsystem acquires the target spacecraft in its thermal imager field of view (FOV) for relative navigation and maneuvers the spacecraft into a formation flight orbit with respect to LightSail.

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

3. Guidance, Navigation, & Control (GN&C) subsystem overview

The Guidance, Navigation, & Control (GN&C) subsystem for Prox-1 is made up of navigation algorithms and sensor components used to determine the satellite attitude and trajectory state, guidance algorithms to determine desired trajectories, and control algorithms and actuator components to control the attitude and perform propulsive maneuvers.

3.1. Navigation algorithms and sensor components

Each of the navigation algorithms is used to determine the state of the spacecraft based on inputs from various sensors. The fine attitude determination requirement for the mission is to obtain attitude knowledge within 1.5° per

axis. An inertial attitude determination filter (implemented as an extended Kalman filter) computes Prox-1's attitude quaternion and angular velocity vector using a MicroElectroMechanical Systems (MEMS) rate gyroscope, analog three-axis magnetometer, and coarse 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 filter to obtain relative position and velocity [17]. Accelerometer measurements are also taken onboard Prox-1 using a MEMS inertial measurement unit, and a visual camera captures images for use on the ground. Global Positioning System (GPS) measurements are used for onboard inertial orbit determination. Further details of the Prox-1 sensors are given in Table 1.

3.2. Guidance algorithms

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 to calculate the desired state and artificial potential functions for collision avoidance to stay away from a defined "keep-out-zone" [22]. A target acquisition algorithm determines appropriate slew maneuvers to center the target spacecraft in Prox-1's imaging instrument 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.

3.3. Control algorithms and actuator components

Finally, Prox-1 has control algorithms to implement maneuvers commanded by the guidance algorithms using various actuators. For primary fine attitude control, a slew and tracking controller (STC) uses the CMG unit. The STC commands slews to the proper attitude before allowing thrust maneuvers and tracks LightSail so that it remains in Prox-1's imager FOV. To adequately perform these tasks, Prox-1 has a fine attitude pointing requirement of 3.0° per axis and an angular acceleration requirement of 0.3 deg/s^2 per axis. A thruster controller commands the 50 mN cold gas thruster to fire for a specified amount of time based on guidance algorithm outputs. Finally, a torque rod controller is used for coarse attitude control and to implement commands from the desaturation and detumble algorithms using three single-axis magnetic torque rods. These torque rods are designed and manufactured in-house at Georgia Tech and have the capability to satisfy the coarse attitude control requirements of 40° per axis. Further details of the Prox-1 control actuators are given in Table 2.

3.4. Reference frame definitions

The Earth-Centered Inertial (ECI) J2000 reference frame is used for determining the absolute position, velocity, and attitude of Prox-1. 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 Earth.

The Local Vertical Local Horizontal (LVLH) frame, also known as the RSW frame (because of the unit vector designations below), 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). Fig. 1 shows a visualization of this coordinate frame. Notice that this coordinate frame is constantly translating and rotating in its orbit as the LightSail moves about the Earth.

For the Prox-1 mission, it is assumed that no *a priori* knowledge of LightSail's inertial position and velocity will be known, thus the origin of the LVLH 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 $\left\{\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{S}}, \widehat{\boldsymbol{W}}\right\}$ are defined by Eq. (1), where $\boldsymbol{r}_{Prox1}$ and $\boldsymbol{V}_{Prox1}$ are the positon and velocity vectors of Prox-1 in the ECI frame and the 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 LVLH frame unit vectors: the distance between Prox-1 and LightSail is small compared with the LightSail orbit radius, and the orbital velocity of LightSail is approximately equal to the orbital velocity of Prox-1.

$$\widehat{\boldsymbol{R}} := \frac{\boldsymbol{r}_{Prox1}}{||\boldsymbol{r}_{Prox1}||} \quad \widehat{\boldsymbol{W}} := \frac{\boldsymbol{r}^{x}_{Prox1}\boldsymbol{V}_{Prox1}}{\left|\left|\boldsymbol{r}^{x}_{Prox1}\boldsymbol{V}_{Prox1}\right|\right|} \quad \widehat{\boldsymbol{S}} := \frac{\widehat{\boldsymbol{W}}^{x}\widehat{\boldsymbol{R}}}{\left|\left|\widehat{\boldsymbol{W}}^{x}\widehat{\boldsymbol{R}}\right|\right|} \tag{1}$$

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 parallel to 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 LVLH Frame. Fig. 2 shows the orientation of the Body-Fixed Frame with respect to the Prox-1 spacecraft.

4. Six degree of freedom (6DOF) simulation environment

All GN&C subsystem algorithms 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 algorithm interactions with sensors and actuators and with other GN&C algorithms.

Table 1
Prox-1 sensor specifications.

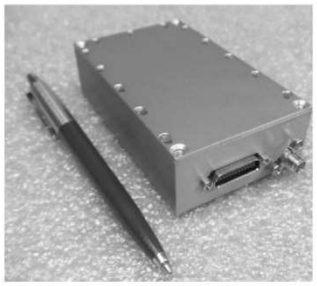
Sensor	Image	Manufacturer	Key specifications
Inertial Measurement Unit (MEMS rate gyro and accelerometer)		Analog Devices, Inc. [18]	Initial gyro bias: 3 deg/s Initial accelerometer bias: 50 mg Sampling frequency: 25 Hz
Analog 3-Axis Magnetometer		SpaceQuest, Ltd. [19]	Sensitivity: 100 μV/nT Sampling frequency: 10 Hz
Advanced Micro Sun Sensor (1)		Jet Propulsion Laboratory [13]	Accuracy: 0.2 deg Sampling frequency: 2 Hz
Coarse Solar Angle Sensors (3)		ELMOS Semiconductor [20]	Angle resolution: 2.7 deg Angle accuracy: 5 deg Sampling frequency: 1 Hz
GPS Receiver		SpaceQuest, Ltd. [21]	Position accuracy: 1.8 m Velocity accuracy: 0.03 m/s Sampling frequency: 20 Hz
Microbolometer Thermal Imager		FLIR/Arizona State University (custom) [14]	Focal length: 100 mm Field of view: 6.2 × 5.0 deg Sampling frequency: 0.1 Hz

Table 2
Prox-1 actuator specifications.

Actuator	Image	Manufacturer	Key specifications
Control Moment Gyroscopes (4)		Honeybee Robotics Spacecraft Mechanisms Corporation [12]	Maximum momentum storage capacity: 86 mN m/s Maximum torque per CMG: 172 mN m Pointing accuracy requirement: 3 deg/axis
Cold Gas Thruster		The University of Texas at Austin (custom) [15]	Max thrust: 50 mN Total ΔV capability: 15 m/s
Torque Rods (3)		Georgia Tech (custom)	Maximum dipole moment: 11.16 A m^2 Pointing accuracy requirement: 40deg /axis

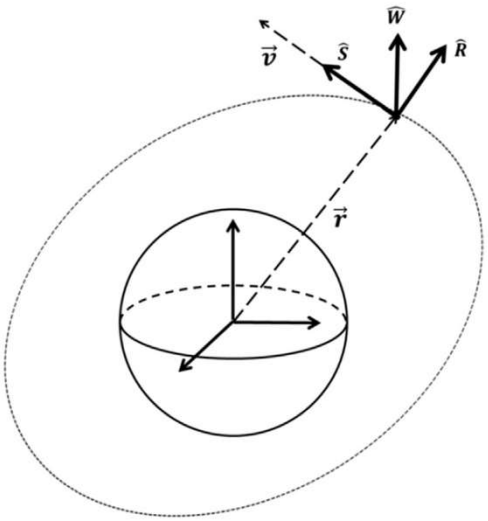


Fig. 1. Visual representation of the LVLH frame.

4.1. 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. A 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 hardware 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 Simulink to embedded processors for hardware-in-the-loop testing.

4.2. 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 LigthSail. 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.

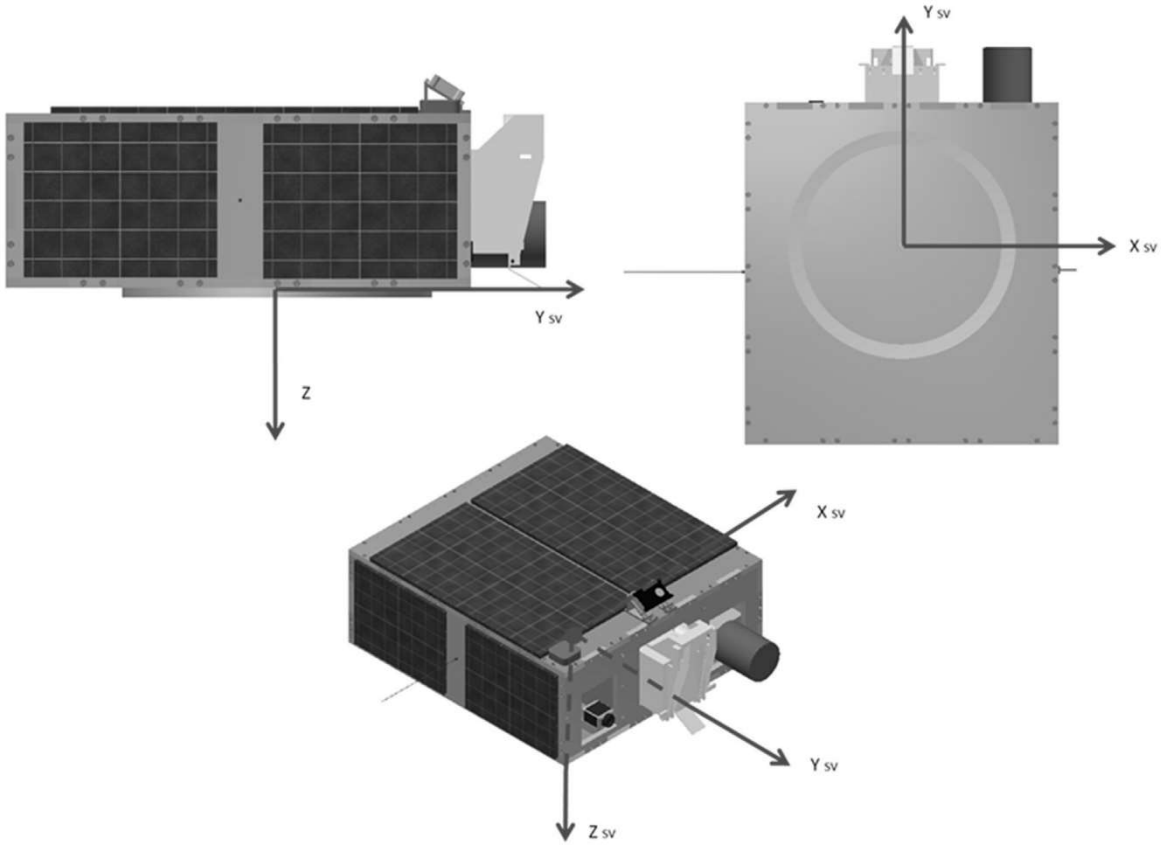


Fig. 2. Prox-1 Body-Fixed Frame orientation.

4.3. 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 the development and testing of all GN&C algorithms by estimating the physical and electrical responses of sensor and actuator hardware based on test data and manufacturer specifications. 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. These simulated images are processed by GN&C algorithms to perform relative navigation testing.

5. GN&C algorithm development

Two main activities are completed within the 6DOF simulation environment: individual GN&C algorithm development and testing of integrated GN&C algorithms. Each GN&C algorithm 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 algorithms. Once basic functionality of the algorithm is established, it is connected to the 6DOF simulation framework for further development and testing. Several examples of algorithm development will be explained in this section.

5.1. Slew and Tracking Controller (STC) development

The Slew and Tracking Controller (STC) is used during ProxOps 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).

5.1.1. STC torque control law development

The core of the STC is an asymptotically stable torque control law, given in Eq. (2), which calculates a desired torque vector $\boldsymbol{\tau}$ in N-m with respect to the Prox-1 BFF based on various inputs.

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

In Eq. (2) above, $\boldsymbol{\omega}_B$ is the current inertial angular velocity of Prox-1 in rad/s expressed in BFF, $\underline{\boldsymbol{J}}$ is the 3×3 inertia tensor of Prox-1 in kg-m^2, $\boldsymbol{\omega}_e$ is the error angular velocity in rad/s defined in Eq. (3), $\underline{\boldsymbol{R}}_e$ is a 3×3 rotation

matrix that maps a vector from the desired reference frame to BFF given in Eq. (4), $\boldsymbol{\omega}_d$ is the desired inertial angular velocity in rad/s expressed in BFF, $\dot{\boldsymbol{\omega}}_d$ is the desired inertial angular acceleration in rad/s^2 expressed in BFF (usually set to zero), η_e is the scalar portion of the error quaternion $\boldsymbol{q}_e$ defined in Eq. (5), $\underline{\boldsymbol{K}}$ is a 3×3 quaternion gain matrix, $\boldsymbol{\varepsilon}_e$ is the vector portion of the error quaternion, and $\underline{\boldsymbol{C}}$ is a 3×3 angular velocity gain matrix.

$$\boldsymbol{\omega}_e = \boldsymbol{\omega}_B - \underline{\boldsymbol{R}}_e\,\boldsymbol{\omega}_d \tag{3}$$

$$\underline{\boldsymbol{R}}_e = \underline{\boldsymbol{\Xi}}^T(\boldsymbol{q}_e)\underline{\boldsymbol{\Psi}}(\boldsymbol{q}_e) \tag{4}$$

$$\boldsymbol{q}_e = [\underline{\boldsymbol{\Xi}}(\boldsymbol{q}_d^{-1})\boldsymbol{q}_d^{-1}]q = \begin{bmatrix}\boldsymbol{\varepsilon_e}\\ \eta_e\end{bmatrix} \tag{5}$$

In Eq. (5) above, $\boldsymbol{q}$ with no subscript is the current attitude quaternion, a unitless four element vector with values ranging between -1 and 1 that represents the rotation of Prox-1's body-fixed reference frame of the Prox-1 BFF with respect to the ECI frame. In Eqs. (4) and (5) above, the $\underline{\boldsymbol{\Xi}}$ operator is defined in Eq. (6) and the $\underline{\boldsymbol{\Psi}}$ operator is defined in Eq. (7). Note that in Eqs. (6) and (7) $\underline{\mathbf{I}}$ is the 3×3 identity matrix.

$$\underline{\boldsymbol{\Xi}}(\boldsymbol{q}) = \begin{bmatrix}\eta\underline{\mathbf{I}} + \boldsymbol{\varepsilon}^{\boldsymbol{x}}\\ -\boldsymbol{\varepsilon}^{\boldsymbol{T}}\end{bmatrix} \tag{6}$$

$$\underline{\boldsymbol{\Psi}}(\boldsymbol{q}) = \begin{bmatrix}-\boldsymbol{\varepsilon}^{x} + \eta\underline{\mathbf{I}}\\ -\boldsymbol{\varepsilon}^{T}\end{bmatrix} \tag{7}$$

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. (2) through(5). The desired quaternion $\boldsymbol{q}_d$ is then computed using the method presented by Shuster [23]. Development of the STC torque law involved using a basic standalone Simulink block diagram, shown in Fig. 3, with a constant input for $\boldsymbol{q}_d$, zero vector inputs for $\boldsymbol{\omega}_d$, and $\dot{\boldsymbol{\omega}}_d$, and a zero matrix input for $\underline{\boldsymbol{R}}_e$. This initial simulation included basic unperturbed angular kinematic equations and assumed perfect execution of the desired torque commands. 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 resulted in the selection of the gain constants 0.05*$\underline{\mathbf{I}}$ for $\underline{\boldsymbol{K}}$ and 0.5*$\underline{\mathbf{I}}$ for $\underline{\boldsymbol{C}}$.

To demonstrate performance of the torque controller, a test scenario was run in the standalone STC simulation using the selected controller gains. The initial angular velocity was set to zero, and the initial error quaternion was set to a value of [0.5 −0.5 0.5 0.5]. $\boldsymbol{q}_d$ was set to the unity quaternion [0 0 0 1], $\boldsymbol{\omega}_d$ and $\dot{\boldsymbol{\omega}}_d$ were set to zero vectors, and $\underline{\boldsymbol{R}}_e$ was set to the 3×3 zero matrix. The moments of inertia for Prox-1 were set to the following in kg-m^2: $I_{xx}=0.9349$, $I_{yy}=0.9349$, $I_{zz}=1.4605$, and the products of inertia were set to zero. This scenario corresponds with a simple case where a pointing error exists that the torque controller must correct.

Fig. 4 shows the simulation results using four output plots. From top to bottom, the first plot in shows the commanded torque in the Prox-1 BFF in N-m. The second plot shows the error quaternion of Prox-1, as defined in Eq. (5) above. The third plot shows the inertial angular velocity of Prox-1 in rad/s measured in the Prox-1 BFF. The final plot shows the inertial angular acceleration of Prox-1 in rad/s^2 measured in the Prox-1 BFF. The performance of the controller represents critical damping, and the desired conditions are reached within 60 s. The maximum angular velocity of 0.06 rad/s in this scenario corresponds to a maximum slew rate of about 3.44° per second.

5.1.2. STC Simulink logic development

Once the basic torque control law was developed and tested, additional logic was written to transform other expected inputs, such as a desired direction to fire the thruster (Tcmd), into quaternion representation. In subsequent stages of development, additional logic was added to allow tracking of the relative position vector to maintain LightSail within Prox-1's imager field of view (the "tracking" element of the slew and tracking controller). 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 $\boldsymbol{\omega}_d$ is

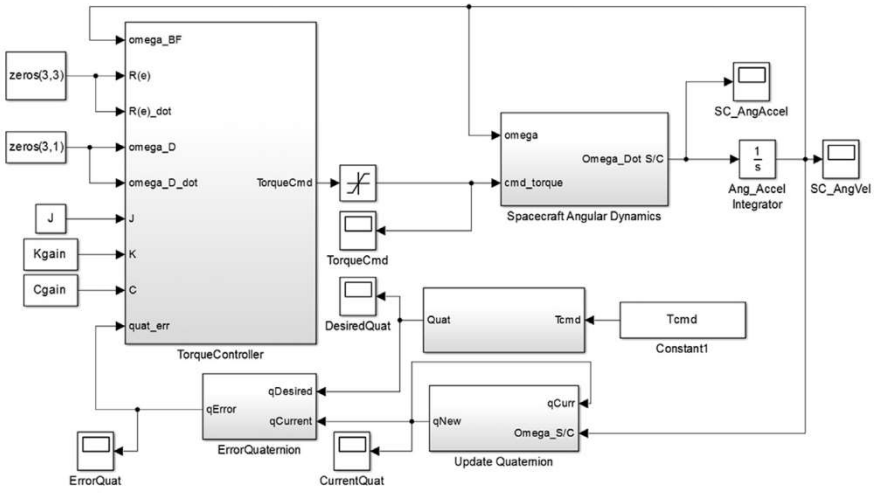


Fig. 3. Standalone Simulink block diagram for initial slew and tracking controller component development.

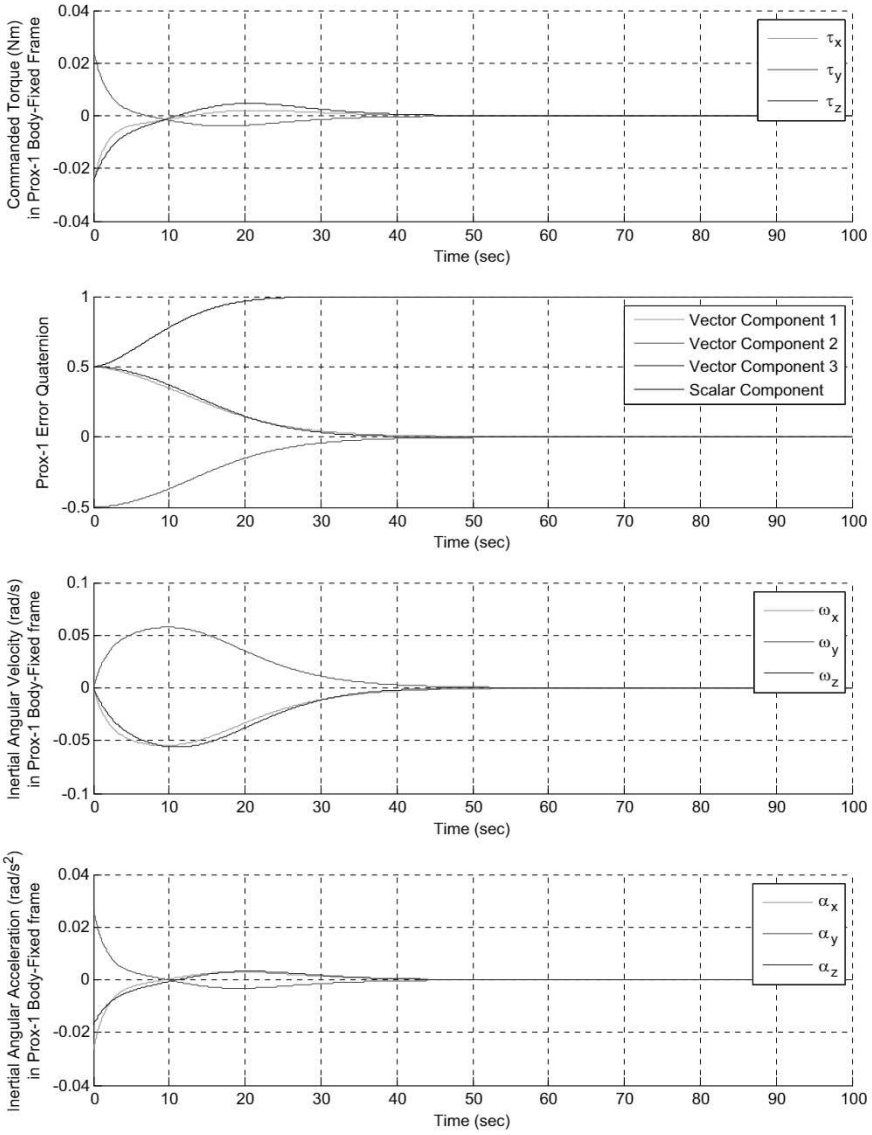


Fig. 4. Torque controller performance test.

determined using Eq. (8) where $\boldsymbol{r}_{rel}$ is the LVLH relative position vector, $\boldsymbol{v}_{rel}$ is the relative velocity vector, and $\underline{\boldsymbol{R}}_{BFF}^{LVLH}$ is the transformation matrix from LVLH to BFF.

$$\boldsymbol{\omega}_d = \underline{\boldsymbol{R}}_{BFF}^{RSW} \frac{\boldsymbol{r}_{rel}^{\times} \boldsymbol{v}_{rel}}{\|\boldsymbol{r}_{rel}\|^2} \tag{8}$$

The desired slew vector is set to $\boldsymbol{r}_{rel}$ and the camera boresight axis is pointed along $-\boldsymbol{r}_{rel}$. Also, upon integration of the environmental and hardware plant models of the 6DOF simulation, a cost function was added 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. This tracking logic has been validated in the simulation using the image generation tool, confirming that the STC can maintain LightSail within its FOV when the relative states are known. A target acquisition and recovery controller is being developed to find LightSail initially and recover LightSail if it drifts out of Prox-1's FOV. The target acquisition controller will also track LightSail until a relative state estimate is obtained.

As additional complexities were added to the STC, it became apparent that simply developing additional logic using more Simulink blocks would lead to an unwieldy design solution. Stateflow provided an intuitive and elegant way to overcome this issue. A Stateflow chart was developed, as shown in Fig. 5, which allowed the creation of various modes within the STC: Standby, Slew_to_Thrust, Wait_for_Thruster, and Track. Each mode is represented by a tan box in the chart, and blue arrows between the boxes represent transitions between the modes.

The Stateflow chart is a block within the Simulink diagram for the STC, as shown in Fig. 6. Inputs are fed into the Stateflow block based on the angular state and desired conditions, and outputs from the block are fed into the

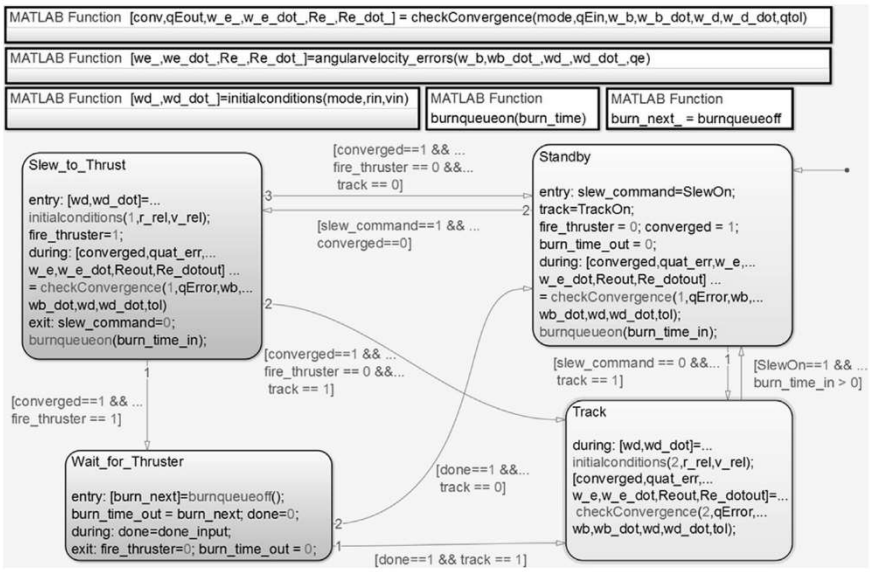


Fig. 5. Stateflow diagram for slew and tracking controller mode logic, including transition conditions. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

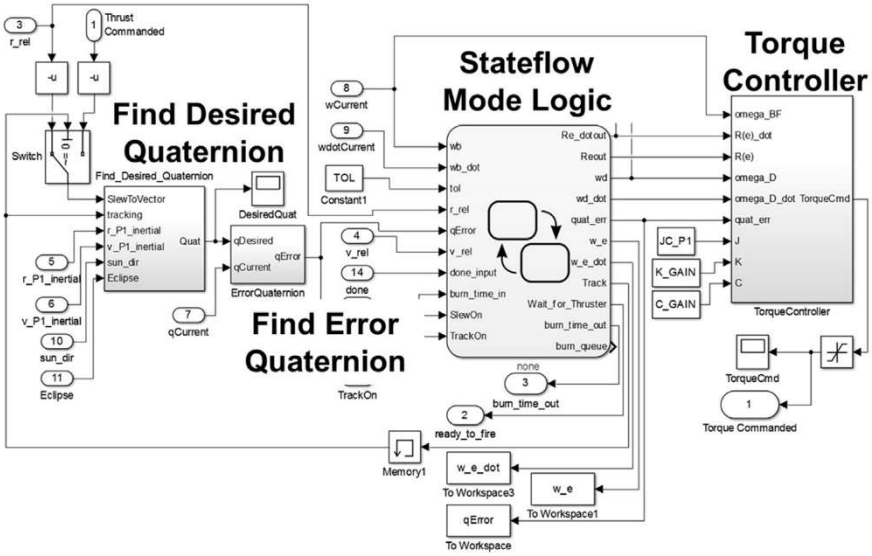


Fig. 6. Simulink diagram for the slew and tracking controller model. The Stateflow chart is a tan block in the middle, inputs are connected to blocks by arrows from the left, and outputs are connected from blocks to the right.

torque control law. Boolean variables are set within and outside the Stateflow chart to determine which mode is active. Each transition arrow contains a logical condition. When the condition connected to an outgoing arrow from the active mode is met, the chart will change its active mode.

Code within each mode determines the action taken by STC, calls embedded MATLAB functions within the chart (shown in the top left corner 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. This feature allows to developer to test and debug the mode logic by running the simulation and watching the live animation along with Scope data.

5.2. Thruster controller (TC) development

The thruster controller (TC) is a simple algorithm that controls the Prox-1 thruster state. The TC outputs an on/off command to the thruster based on a burn duration and execution time received from the STC. The inputs from STC are total burn time and a “ready-to-fire” Boolean variable.

Early versions of the simulation before the STC was added omitted the thruster controller and assumed that the thruster fired in the desired direction as soon as the

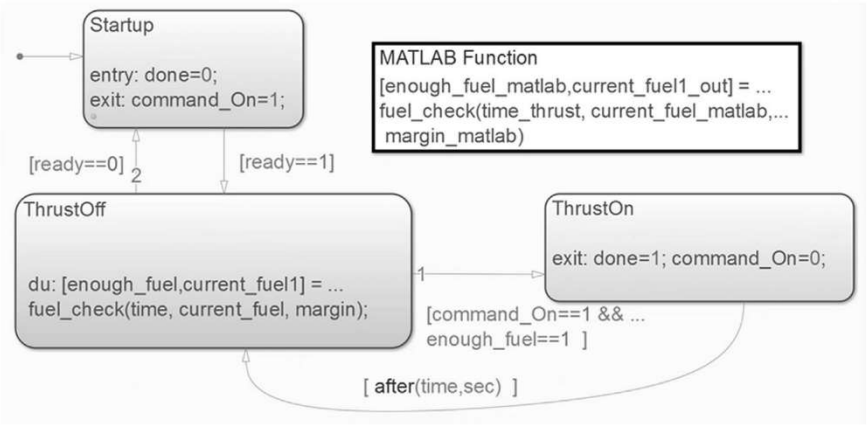


Fig. 7. Standalone Stateflow chart for TC logic.

guidance algorithms commanded a burn. With the STC added in the loop, it is necessary to delay the thruster firing to allow the STC to slew Prox-1 to the proper orientation before firing the thruster. The thruster controller performs this function by waiting to execute a burn command until the "ready-to-fire" flag is set to true.

Initially, the TC logic was developed in a standalone Stateflow diagram, shown in Fig. 7. The diagram has three states: Standby, ThrustOff, and ThrustOn. The chart will remain in standby until the "ready" flag becomes true. At that time, a check for available fuel is run, and if sufficient fuel is available, the thruster is fired for the specified amount of time. After the firing is complete the "done" flag is set to true, and the state changes to ThrustOff until "ready" becomes false again. Initially, the standalone TC chart was fed by constants for the ready flag and burn time that were defined in the initialization file. It was soon modified to receive inputs from and send outputs to external models, as shown in Fig. 8. This allows integration with the STC and thruster plant models.

5.3. Detumble controller (DTC) development

The detumble controller (DTC) is triggered after separation from the launch vehicle and calculates a magnetic moment required to slow the initial 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 [24], shown in Eq. (9) where $\hat{\boldsymbol{b}}$ is the ECI unit vector direction of the Earth magnetic field at Prox-1's location, $\boldsymbol{\omega}$ is the current angular velocity of Prox-1 in ECI, and k_ω is a constant given in Eq. (10) where Ω is the mean motion of Prox-1's orbit, ξ_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.

$$\boldsymbol{m} = -\frac{k_\omega}{\|\boldsymbol{b}\|}\hat{\boldsymbol{b}} \times \left[\left(\boldsymbol{I}_3 - \hat{\boldsymbol{b}}\hat{\boldsymbol{b}}^T\right)\boldsymbol{\omega}\right] \tag{9}$$

$$k_\omega = 2\Omega(1 + \sin\ \xi_m)J_{min} \tag{10}$$

Originally, the DTC was developed in a simplified version of the 6DOF sim with no other controllers present. A high initial angular velocity condition was input to the sim and the controller was developed to damp that initial condition. DTC was then integrated into the 6DOF sim. The DTC Stateflow logic diagram is shown in Fig. 9. 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 into Standby_Detumble until the angular velocity limits are violated again and Detumbling is allowed by higher level mode logic.

To demonstrate the performance of the detumble controller, a test scenario was run 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 consistent with the torque rod capability. In this case, the convergence criteria were angular velocities of less than 0.01 rad/s (or 0.573 deg/s) in each axis. This scenario corresponds to a realistic case of the expected tumbling Prox-1 will experience after release from the launch vehicle. Fig. 10 shows the simulation results using four output plots. The first plot shows the magnetic moment commanded by the detumble controller. The second plot shows the actual moment executed by the torque rods. Note that the commanded moment is not directly realized but is fed into the torque rod controller, which turns the torque rods on and off to approximate it using a "bang-bang" control strategy since the torque rods can only execute fully positive, fully negative, or zero magnetic moment. The third plot shows the inertial angular velocity of Prox-1 in rad/s measured in the Prox-1 BFF.

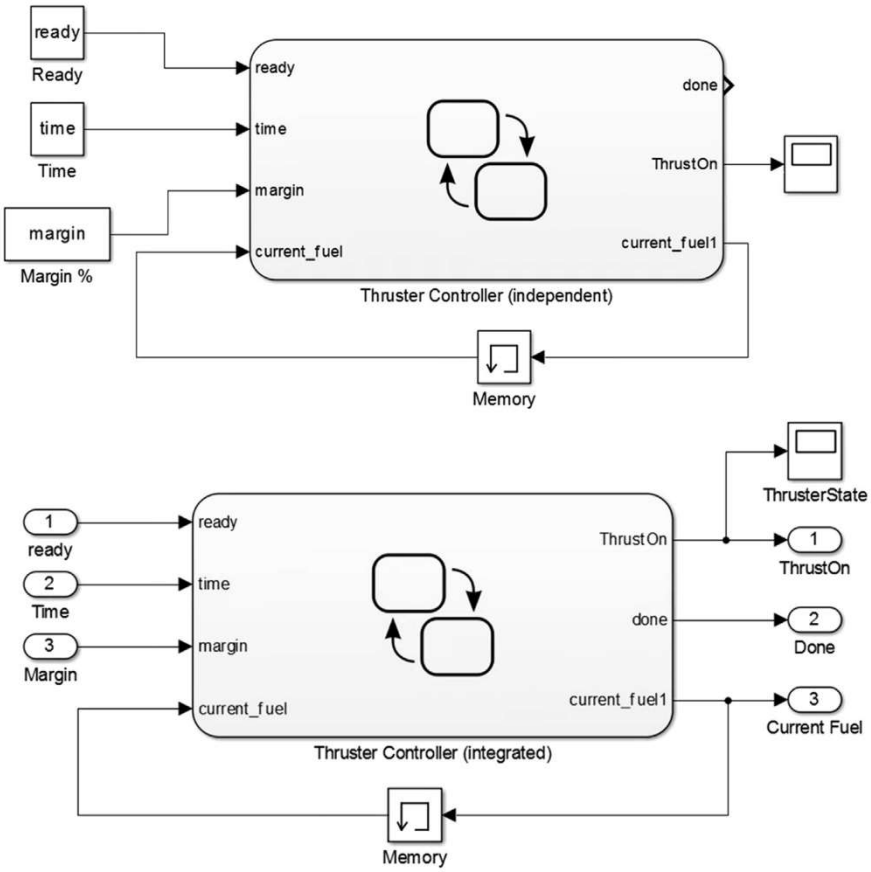


Fig. 8. Simulink diagrams for the thruster controller. Top is the standalone version, bottom is the integrated version.

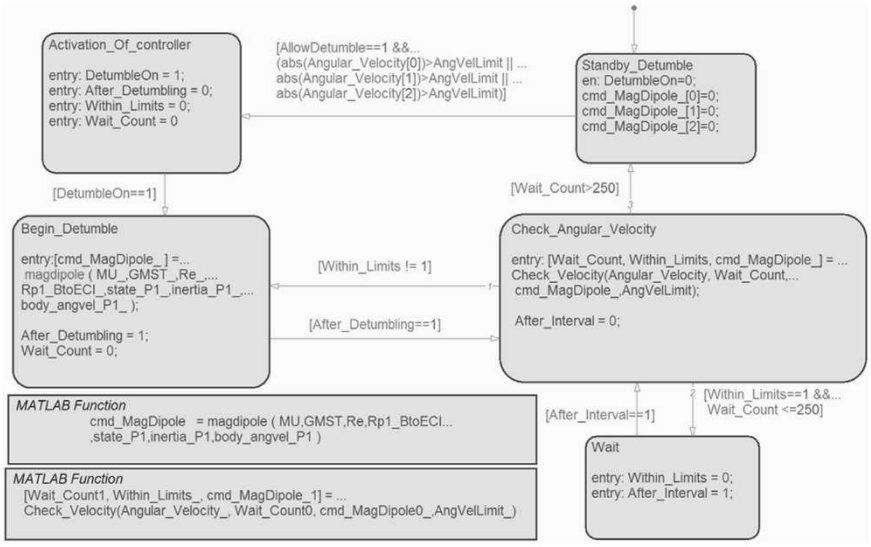


Fig. 9. Stateflow logic diagram for the detumble controller to generate magnetic dipole command.

The final plot shows the attitude quaternion of the Prox-1 BFF with respect to the ECI frame. The commanded magnetic moment plot shows that the initial magnetic moment required was outside the capability of the torque rods, however after a few seconds the necessary moment is attainable. The commanded moment goes to zero around 22 s when all of the angular velocities drop below 0.01 rad/s. In this case, all the angular velocities converge to zero and the attitude quaternion becomes constant after 50 s, showing that the spacecraft has successfully nulled its rotation rates.

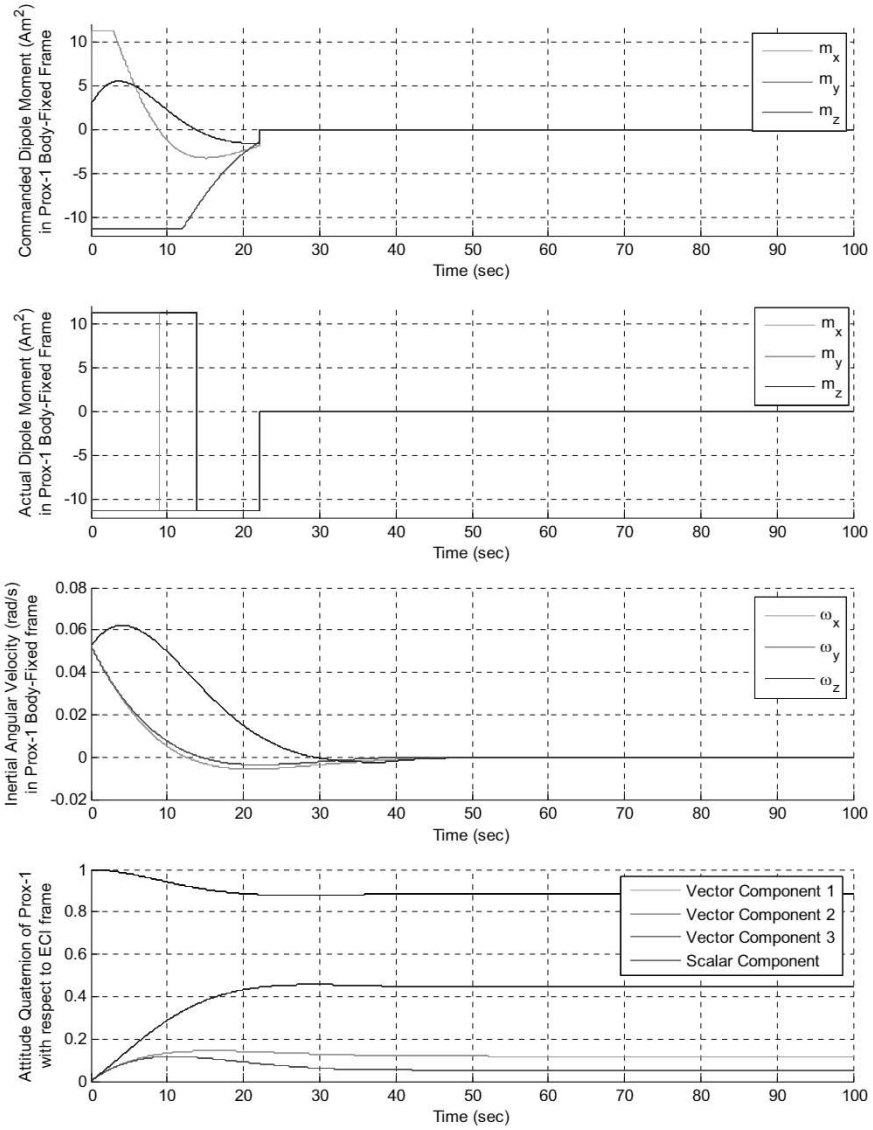


Fig. 10. Detumble controller performance test.

6. GN&C simulation integration

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

6.1. Simulation integration process

To achieve the current version of the "master simulation" in the 6DOF environment, the GN&C team begins with the standalone simulation used to test the guidance algorithms. This initial sim, shown in Fig. 11, contains the foundation for the 6DOF sim: the Prox-1 and LightSail plant and environment models. The only GN&C algorithm in this version of the sim is the guidance block, which receives relative position and velocity inputs directly from the spacecraft plant (rather than from navigation algorithms) 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 functional guidance block. To account for effects of other algorithms as they are added, the guidance block is continually tuned throughout the integration process.

The next step in the integration process involves bringing in the STC and TC, described in Sections 5.1 and 5.2 respectively. The resulting integrated sim is shown in Fig. 12. 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

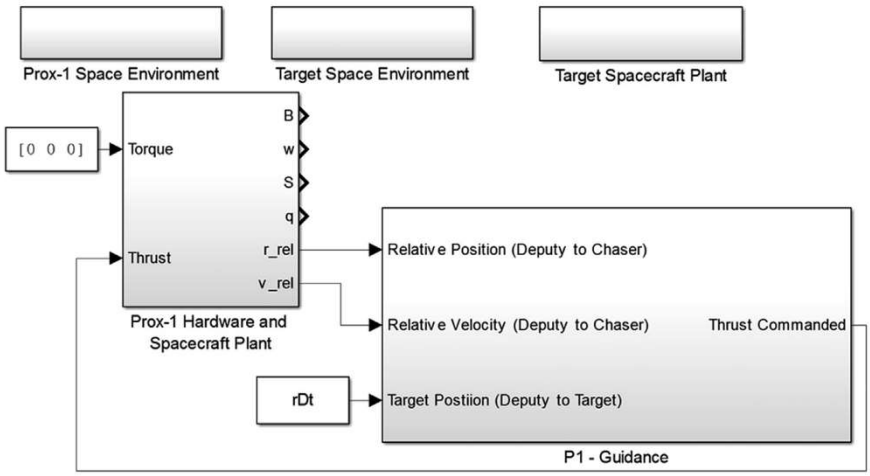


Fig. 11. Standalone guidance testing simulation.

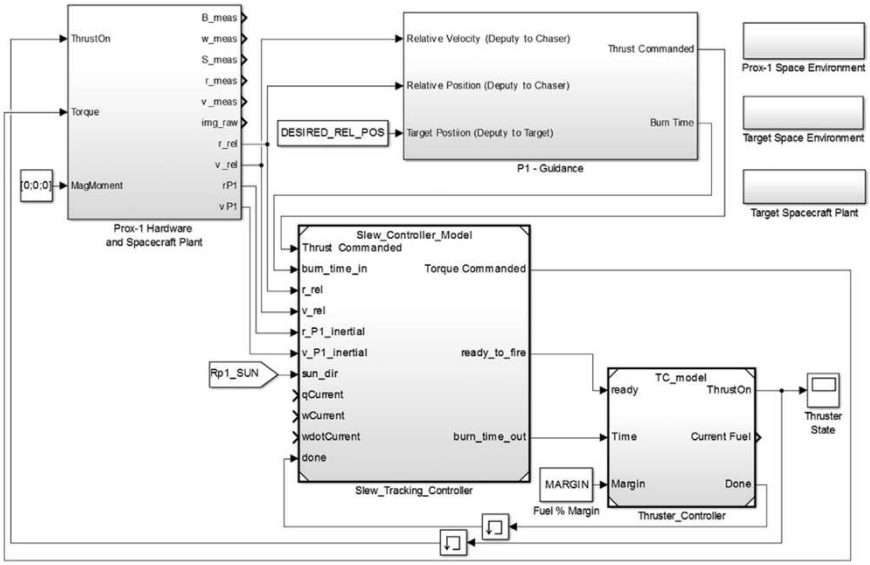


Fig. 12. Integrated sim including guidance, slew & tracking, and thruster controller algorithms.

these are sent to the STC. The STC also takes inputs of inertial and relative position and velocity directly from the spacecraft plant, as well as the sun vector directly from the space environment models, again bypassing the navigation algorithms. Torque commands from the STC are output directly to the spacecraft plant model (bypassing the complex and computationally intensive CMG hardware model). The "ready-to-fire" flag and burn time are sent from STC into TC, which sends thruster firing commands to the plant models and feeds back a "done" flag to STC that indicates when the firing is complete.

Once all of these connections are made, an integrated functional test of the simulation is run, as described in Section 6.2 below. Outputs of the simulation are examined to determine if the algorithms are functioning properly. At each stage of integration, all of the algorithms are verified to perform in the expected manner. If they do not perform as expected, design changes are made and the integration cycle is iterated.

Finally, the DTC is added to the simulation and mode logic is developed in a Stateflow diagram, resulting in the master simulation shown in Fig. 13. 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 for now). The plant and environment models, boxed in red in Fig. 13, are only used for simulation and are not implemented in flight software.

The tan block in the middle of the master simulation is the Stateflow GN&C mode logic, shown in Fig. 14. This block includes two states: 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

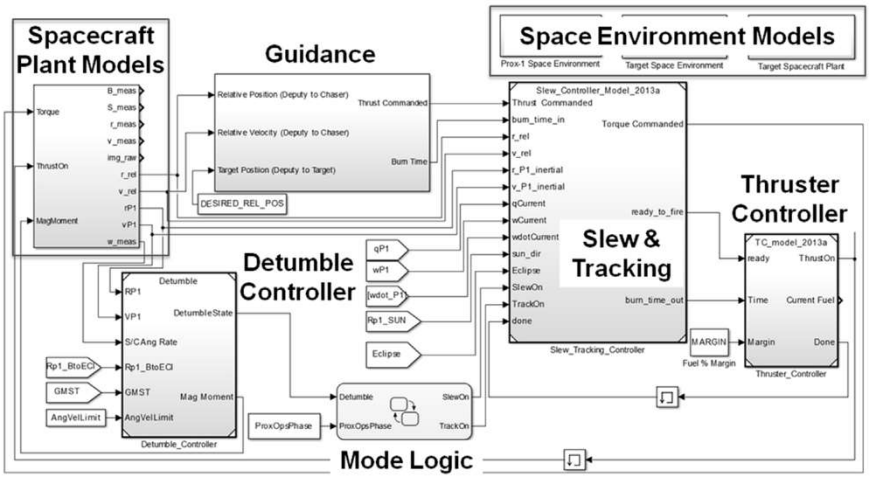


Fig. 13. Master simulation including guidance, slew & tracking, thrust control, detumble control, and mode logic. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

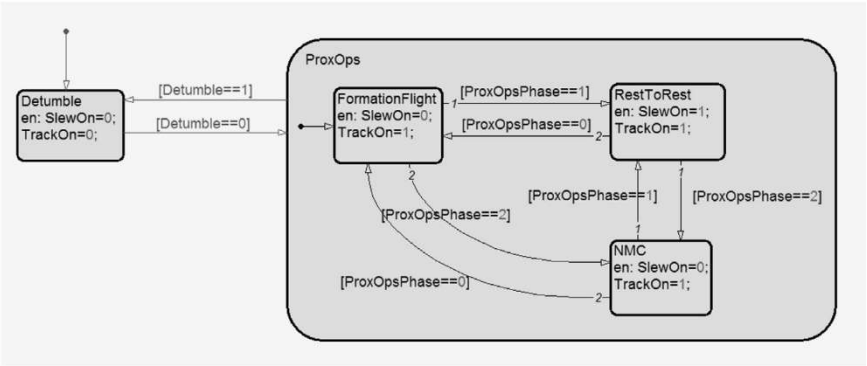


Fig. 14. Stateflow mode logic for integrated master simulation.

(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.

6.2. Integrated simulation performance test

The simulation conditions for the integrated GN&C subsystem test are as follows: LightSail begins in 24° inclination circular orbit with an altitude of 720 km. Prox-1 begins in a trailing orbit at the same altitude and inclination but 100 m behind LightSail in the along-track direction. There is no initial relative velocity between the two spacecraft. Fig. 15 shows the simulation results using four output plots. From top to bottom, the first plot shows the LVLH relative position of Prox-1 with respect to LightSail in meters. The second plot shows the LVLH relative velocity of Prox-1 with respect to LightSail in m/s. The third plot shows the state of Prox-1's thruster (1=on, 0=off). The final plot shows the attitude quaternion for Prox-1 with respect to the ECI frame. The timescale for all of these outputs is in seconds. Table 3 provides relevant simulation parameters for this scenario. Note that the orbit of LightSail is defined within the simulation initialization file, and the dynamics of Prox-1 are defined relative to LightSail using initial conditions for relative position and velocity.

In this scenario, the relative position begins with a −100 m offset in the along-track direction, which closes to −50 m by the end of the simulation as Prox-1's guidance algorithms command thruster burns to maneuver Prox-1 toward LightSail. This is termed a “Rest-to-Rest” maneuver because it begins and ends with near-zero relative velocity. The dips in the attitude quaternion plot represent slew maneuvers that point Prox-1's thruster in the desired direction to execute thruster burns commanded by the guidance algorithms. These slew maneuvers are necessary because Prox-1 has a single fixed thruster that must be rotated to the proper direction before each burn. These burns are shown as step functions on the thruster state plot. By comparing the third and fourth plots it can be seen that the thruster burns occur just after the peak of the dips in the attitude plot, when Prox-1 is pointed in the proper direction for thrusting. After each burn, Prox-1 slews back

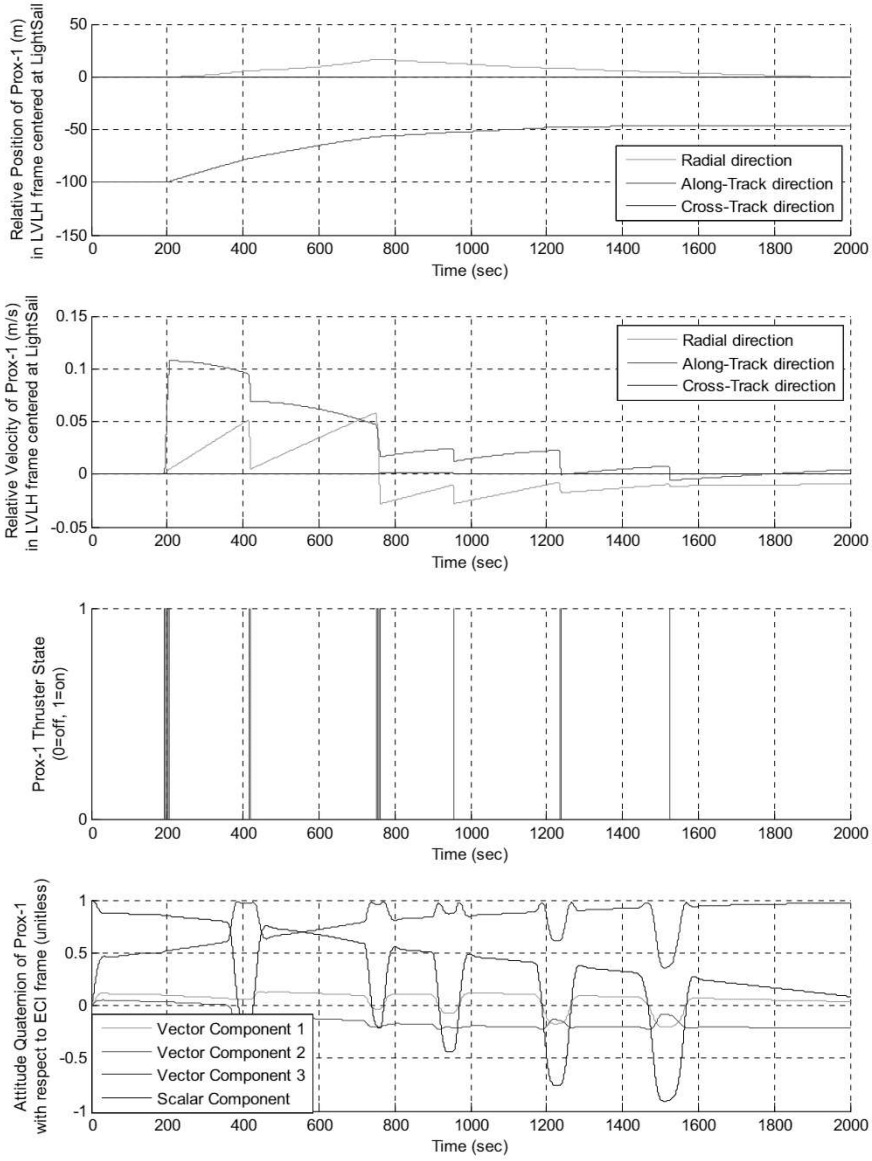


Fig. 15. Integrated simulation results for Rest-to-Rest maneuver.

to point its cameras at LightSail, continuing to track it and capture images. It can be seen by comparing the second and third plots that thruster firings cause discontinuities in relative velocity, which the guidance algorithms then use to guide Prox-1 to the desired final relative position.

6.3. Current and future work

As of this writing, the Prox-1 GN&C team is completing the development and integration of various algorithms in Simulink, including the target acquisition and recovery controller, inertial attitude determination filter, desaturation controller, and torque rod controller. Updates are also being made to other algorithms. When all algorithms are in their final form, the entire GN&C subsystem will be tuned and tested rigorously within Simulink. The flowchart in Fig. 16 shows how all of the algorithms and hardware components will eventually fit together in the final GN&C subsystem design.

7. Autocoding from MATLAB/Simulink to C/C++

The Simulink design of the GN&C algorithms is autocoded into C/C++ using Simulink Coder for integration with flight software (FSW). Some modifications are be made to ensure the models are codable, such as avoiding the use of incompatible MATLAB functions.

7.1. Autocoding process

The process the Prox-1 team has developed for autocoding from a Simulink master simulation to C code is

Table 3
Simulation conditions for integrated performance test.

Variable name	Description	Value	Units
EPOCH	Initial time of simulation start	[2012 08 16 17 00 00]	[yr mon day h min s]
dt	Simulation timestep	0.01	s
RP	Periapsis above the surface	720e3	m
e_LS	Eccentricity	0.0	n/a
Omega_LS	Right Ascension of Ascending Node	0	deg
inc_LS	Inclination	24	deg
omega_LS	Argument of periapsis	0	deg
f0_LS	True anomaly at epoch	50	deg
MASS_P1	Mass of Prox-1	54.686	kg
JC_P1	Inertia tensor of Prox-1	2.367 0.043 0.194 0.043 2.804 −0.133 0.194 −0.133 2.457	kg m^2
w0_P1	Initial angular velocity of the Prox-1 BFF relative to the ECI frame	[0 0 0]	rad/s
q0_P1	Initial unit quaternion relating Prox-1 BFF to ECI frame	[0 0 0 1]	n/a
r0_rel	Initial relative position of Prox-1 to LightSail (LVLH)	[0 −100 0]	m
v0_rel	Initial relative velocity of Prox-1 to LightSail (LVLH)	[0 0 0]	m/s
DESIRED_REL_POS	Desired relative position from Prox-1 to LightSail	[0 −50 0]	m
CMG_MAX_TORQUE	Max CMG wheel torque	0.112	N m
K_GAIN	Slew Controller K Gain	0.05*eye(3,3)	n/a
C_GAIN	Slew Controller C Gain	0.5*eye(3,3)	n/a
TOL	Slew Controller Tolerance	[0.01 0.001]	n/a
ZONAL_HARMONICS	Zonal harmonic constants (J1–J6)	1 1.08262668355e−3 −2.53265648533e−6 -1.61962159137e−6 -2.27296082869e−7 5.40681239107e−7	n/a
Cd	Drag coefficient for each face of both spacecraft	2.2	n/a
Cr	Reflectivity coefficient for each face of both spacecraft	1.0	n/a
A_LS	Average area of each face of LightSail	2.333e−4	m^2
A_P1	Average area of each face of Prox-1	0.1276	m^2

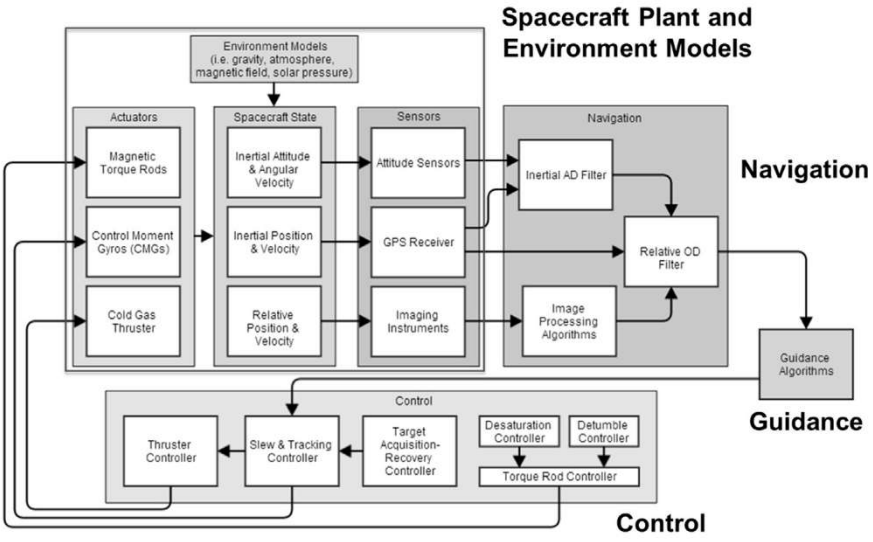


Fig. 16. Flowchart showing final GN&C component interactions.

described in this section. This process is illustrated at a high level by the flowchart in Fig. 17.

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 such as the Prox-1 GN&C master simulation (MasterSim) shown in Fig. 13 cannot be autocoded if it does not run properly in simulation. In the version of the MasterSim shown above, the Simulink diagram includes many GN&C algorithms. Blocks outlined in red are only used for simulation purposes, while other blocks will be autocoded into GN&C flight software (FSW). For configuration control, most of the GN&C algorithm blocks are integrated into the master simulation using model reference blocks. These allow each algorithm to be saved as a separate Simulink model file that can be integrated into multiple master simulations.

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 file 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 algorithm block as shown in Fig. 18. The spacecraft plant and environment models remain in separate blocks and will not be autocoded. Before autocoding of the GN&C algorithms 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. Many errors can occur during this stage, and each error must be understood and corrected. Any errors encountered are documented for future team members to reference. 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 validate that the outputs match those of the Simulink model.

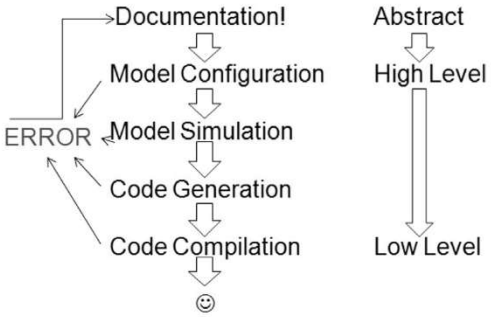


Fig. 17. Autocoding process flowchart.

7.2. Integration with flight software

After the C code has been generated for use on the BeagleBoard, it must be integrated with FSW. The Prox-1 GN&C code is integrated as a monolithic sub-routine that is called during each timestep. This process is illustrated in Fig. 19, 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

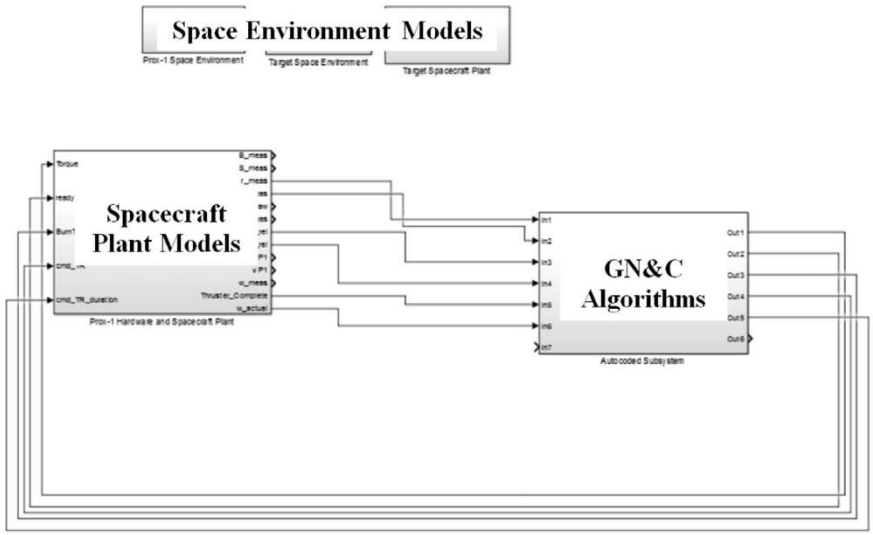


Fig. 18. Master sim model prepared for autocoding by creating a single GN&C block.

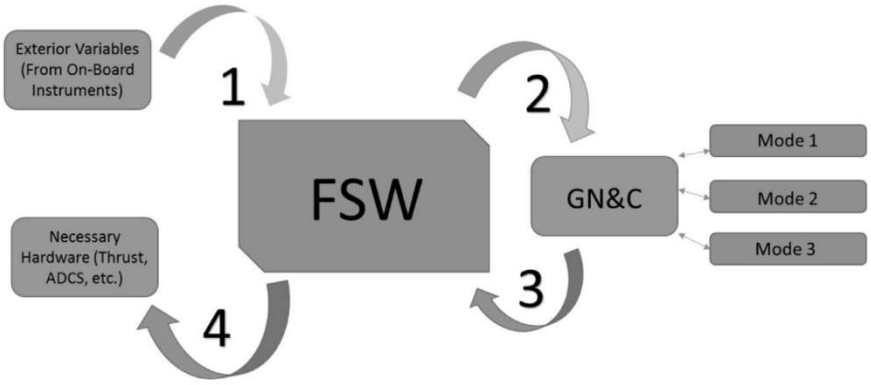


Fig. 19. GN&C/FSW interface illustration.

Propulsion and Attitude Determination and Control Subsystem (ADCS) microcontrollers (4). 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 a Guidance command has been successfully executed.

The actual mechanism of integration between GN&C and FSW involves the Core Flight Executive (CFE), which is code produced by NASA that greatly facilitates the production of flight software [25]. Once the C code is generated from Simulink, all files are transferred to the Prox-1 CFE github repository. Under the CFE apps directory, a new *gnc_app* directory is created, and all the autocoded files are relocated here. A small C wrapper called *gnc_app* is then created within the Prox-1 FSW implementation using CFE.

The main function of *gnc_app.c* listens for messages from other applications, mainly *Prox_app*, the master application for Prox-1 FSW. The messages will contain a data type, along with a command ID that tells GN&C which input parameter the data corresponds to. The *gnc_app* main function also has logic that determines if all thirteen inputs necessary for GN&C to run have been received. If not, the main function continues in a loop. Once all the inputs are received, *gnc_app* calls *MasterSubsystem0.c*, which is the main auto-coded C file from Simulink that contains all of the integrated GN&C algorithms. The Master Subsystem takes the inputs and calculates an output structure. The output structure is forwarded back to the main application of *gnc_app*, and then sent via a software bus to other applications.

To determine how to compile the application, *MasterSubsystem0.c* and its dependencies are all compiled independently outside of CFE, alongside a test function generated by the autocoder, *ert_main.c*. The makefile within the *for_build* CFE subdirectory is then modified to include all the necessary dependencies and the *–lm* C flag necessary for compilation. Everything else is taken care of by the CFE, so that when the whole of flight software is compiled, *gnc_app* compiles as well. The *gnc_app* and *MasterSubsystem0* functions have been compiled and run in a Linux environment identical to the flight computer.

7.3. Autocoding validation

The Prox-1 team developed a process to validate that the outputs created by the autocoded C version of the GN&C algorithms matched the corresponding outputs given by the Simulink program in MATLAB. This allows the team to validate that the autocoded program (from here on referred to as autocode) performs properly and there are no errors in the autocodig process. Testing of the FSW/GN&C integration has been performed through an external daemon server that poses as a serial device interacting with CFE. Since the flight computer does not receive actual relevant data from hardware until it is in orbit, the daemon simulator is necessary to test the performance of the GN&C autocode in an open-loop manner. The daemon server obtains simulated input data from text files. The simulated inputs are collected from the original Simulink code, which has a hardware simulation module. The inputs are fed through the serial port one by one, interacting first with the serial app, then going through the main *Prox_app*, and finally making their way to *gnc_app*. The first test ensures that the autocoded GN&C function does not execute the first iteration until it receives all 13 inputs. The next test runs GN&C through a certain number of iterations while comparing the output structures to the outputs in the original Simulink model outputs. The team used this validation process with several variations of the MasterSim Simulink program and was able to conclude that the outputs from the autocode matched the outputs produced by the Simulink program within a very small degree of error. Although a closed-loop test was not performed, it was validated that the C code provides the same results as the Simulink models at each individual open-loop timestep.

Once all previous validation steps are complete, a closed-loop test will be performed using hardware-in-the-loop simulation with the FSW on the BeagleBoard connected to the Spacecraft Plant and Environment models in Simulink via MATLAB's xPC Target toolbox. This test will validate that the completed *gnc_app* within the FSW code performs in the same manner as the GN&C algorithms in Simulink using the same Simulink simulation plant and environment models. Then, an integrated "Day in the Life" test will bring together all of Prox-1's hardware and software systems for a final evaluation before flight.

8. Conclusion

A complete autonomous Guidance, Navigation, and Control (GN&C) subsystem is being developed 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. All GN&C algorithms are developed within the MATLAB/Simulink 6DOF simulation environment and integrated together to create a complete design solution. By autocoding this entire 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. The Prox-1 mission advances the state-of-the-art by utilizing a streamlined, fully integrated approach to GN&C FSW development that is unique and rigorous for a small satellite mission.

Acknowledgments

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 authors and do not necessarily reflect the views of the National Science Foundation.

The authors wish 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 6DOF MATLAB/Simulink simulation environment.

The authors would also like to thank the following Prox-1 team members who also contributed significantly to the work described herein: Sean Chait, Richard Zappulla, Meet Patel, Aayush Sharma, Jacob Sussman, Nolan Coulter, and Matthew Krumweide, as well as the numerous students who have worked on the Prox-1 and R^3 projects at Georgia Tech.

References

[1] S. Ueda, T. Kasai, H. Uematsu, HTV rendezvous technique and GN&C design evaluation based on 1st flight on-orbit operation result, In: Proceedings of AIAA/AAS Astrodynamics Specialist Conference, Toronto, Canada, August 2010.

[2] E. De Pasquale, ATV Jules Verne: a step by step approach for in-orbit demonstration of new rendezvous technologies, In: Proceedings of the 12th International Conference on Space Operations, Stockholm, Sweden, June 2012.

[3] R.B. Friend, Orbital express program summary and mission overview, In: Proceedings of Sensors and Systems for Space Applications II Conference, Orlando, Florida, March 2008.

[4] M. Delpech, J.C. Berges, S. Djalal, et al., Preliminary results of the vision based rendezvous and formation flying experiments performed during the PRISMA extended mission, Adv. Astronaut. Sci. 145 (2012) 1375–1390. IAA-AAS-DyCoSS1-12-07.

[5] M. Sabatini, G.B. Palmerini, P. Gasbarri, A testbed for visual based navigation and control during space rendezvous operations, In: Proceedings of the 65th International Astronautical Congress, Toronto, Canada, October 2014.

[6] S. Nolet, Development of a guidance, navigation, and control architecture and validation process enabling autonomous docking to a tumbling satellite (Ph.D. dissertation), Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2007.

[7] M.C. Jackson, J.R. Henry, Orion GN&C model based development: experience and lessons learned, In: Proceedings of AIAA Guidance, Navigation, and Control Conference, AIAA-2012-5036, Minneapolis, Minnesota, August 2012.

[8] V.H. Nguyen, A.K. Chaudhary, D. Poladian, V.L. Wong, M.J. Zyss, The GN&C of the SMV (space maneuver vehicle) flight test vehicle: rapid design of an unpowered autolanding system, In: Proceedings of Annual AAS Rocky Mountain Guidance and Control Conference, AAS GN&C 98-024, Breckenridge, Colorado, February 1998, pp. 264–265.

[9] R. Da Costa, M. Markus, G. Ortega, Rapid prototyping tool for development and validation of GN&C onboard software, In: Proceedings of the 54th International Astronautical Congress, Vol. 3, Bremen, Germany, October 2003 1287 1294.

[10] S.M. Stewart, L. Ward, S. Strand, Distributed GN&C flight software simulation for spacecraft cluster flight, In: Proceedings of the 37th Annual AAS Guidance and Control Conference, AAS 14-032, Breckenridge, Colorado, Jan–Feb 2014.

[11] S. Chait, D. Spencer, Prox-1: automated trajectory control for on-orbit inspection, In: Proceedings of the 37th Annual AAS Guidance & Control Conference, AAS 14-066, Breckenridge, Colorado, Jan–Feb 2014.

[12] Microsat CMG Attitude Control Array, Honeybee Robotics Spacecraft Mechanics Corporation, Broolyn, New York. ⟨http://www.honeybeerobotics.com/wp-content/uploads/2014/03/Honeybee-Robotics-MicrosatCMGs.pdf⟩, 2015 (accessed 2.4.15).

[13] S. Mobasser, C. Liebe, J. Naegle, C. Lee, Flight qualified micro sun sensor for Mars applications, in: Proceedings of the 2nd International Conference on Recent Advances in Space Technologies, RAST 2005, Istanbul, Turkey, June 2005, pp. 234–239.

[14] L. Walker, D. Spencer, R3: thermal imaging and rapid feature detection for small satellites, In: Proceedings of the AIAA/8th Responsive Space Conference, AIAA- RS8-2010-3002, El Segundo, CA, April 2010.

[15] S. Arestie, E.G. Lightsey, B. Hudson, Development of a modular, cold gas propulsion system for small satellite applications, J. Small Satell. 1 (2) (2012) 63–74.

[16] C. Biddy, T. Svitek, LightSail-1 solar sail design and qualification, in: Proceedings of the 41st Aerospace Mechanisms Symposium, Pasadena, California, May 2012.

[17] L. Walker, Automated proximity operations using image-based relative navigation, in: Proceedings of the 26th Annual USU/AIAA Conference on Small Satellites, SSC12-VII-3, Logan, Utah, August 2012.

[18] Six Degrees of Freedom Inertial Sensor: ADIS1636/ADIS16365, Norwood, MA: Analog Devices, Inc., Data Sheet. ⟨http://www.analog.com/media/en/technical-documentation/data-sheets/ADIS16360_16365.pdf⟩, 2009–2012 (accessed 23.4.15).

[19] MAG-3 Three-Axis Magnetometer, SpaceQuest, Ltd. Interface Control Document. ⟨http://www.spacequest.com/s/MAG-3.pdf⟩, 2014 (accessed 23.4.15).

[20] Integrated Solar Angle Sensor E910.86, ELMOS Semiconductor AG, Dotrmund, Germany. Data Sheet. ⟨http://www.elmos.de/uploads/tx_commerce/Infosheet_e910_86.pdf⟩, 2010 (accessed 4/23/15).

[21] GPS12-V1 Space-Qualified GPS Reciever, SpaceQueset, Ltd., Fairfax, Virginia. ⟨http://www.spacequest.com/s/GPS12-V1-Receiver-Product-Sheet-Ver-4.pdf⟩, 2014 (accessed 23.4.15).

[22] D. Spencer, Automated Trajectory Guidance for Proximity Operations Using Relative Orbital Elements (Ph.D. dissertation), Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, 2015.

[23] M.D. Shuster, Survey of attitude representations, J. Astronaut. Sci. 41 (1993) 439–517.

[24] G. Avanzini, F. Giulietti, Magnetic detumbling of a rigid spacecraft, J. Guid Control Dyn 35 (4).

[25] Core Flight Executive (cFE), Goddard Space Flight Center, Maryland. ⟨http://opensource.gsfc.nasa.gov/projects/cfe/index.php⟩, February 2015 (accessed 31.3.15).