MSFC-438

SPARTAN: A High-Fidelity Simulation for Automated Rendezvous and Docking Applications

Michael A. Turbe[1], James H. McDuffie[2], Brandon K. DeKock[3], and Kevin M. Betts[4]
bd Systems, Inc. (a Subsidiary of SAIC), Huntsville, AL, 35802

and

Connie K. Carrington[5]
NASA Marshall Space Flight Center, Huntsville, AL, 35803

bd Systems (a subsidiary of SAIC) has developed the Simulation Package for Autonomous Rendezvous Test and ANalysis (SPARTAN), a high-fidelity on-orbit simulation featuring multiple six-degree-of-freedom vehicles. SPARTAN has been developed in a modular fashion in Matlab/Simulink to test next-generation automated rendezvous and docking guidance, navigation, and control algorithms for NASA's new Vision for Space Exploration. SPARTAN includes autonomous state-based mission manager algorithms responsible for sequencing the vehicle through various flight phases based on on-board sensor inputs and closed-loop guidance algorithms, including Lambert transfers, Clohessy-Wiltshire maneuvers, and glideslope approaches The guidance commands are implemented using an integrated translation and attitude control system to provide 6DOF control of each vehicle in the simulation. SPARTAN also includes high-fidelity representations of a variety of absolute and relative navigation sensors that may be used for NASA missions, including radio frequency, lidar, and video-based rendezvous sensors. Proprietary navigation sensor fusion algorithms have been developed that allow the integration of these sensor measurements through an extended Kalman filter framework to create a single optimal estimate of the relative state of the vehicles. SPARTAN provides capability for Monte Carlo dispersion analysis, allowing for rigorous evaluation of the performance of the complete proposed AR&D system, including software, sensors, and mechanisms. SPARTAN also supports hardware-in-the-loop testing through conversion of the algorithms to C code using Real-Time Workshop in order to be hosted in a mission computer engineering development unit running an embedded real-time operating system. SPARTAN also contains both runtime TCP/IP socket interface and post-processing compatibility with bdStudio, a visualization tool developed by bd Systems, allowing for intuitive evaluation of simulation results. A description of the SPARTAN architecture and capabilities is provided, along with details on the models and algorithms utilized and results from representative missions.

Nomenclature

A	=	amplitude of oscillation
a	=	cylinder diameter
C_p	=	pressure coefficient
C_x	=	force coefficient in the x direction
C_y	=	force coefficient in the y direction
c	=	chord
dt	=	time step
F_x	=	X component of the resultant pressure force acting on the vehicle

[1] Avionics Engineer, Advanced Technology Division, 600 Blvd South, Ste. 304, AIAA Member.
[2] Chief Engineer, Advanced Technology Division, 600 Blvd South, Ste. 304, AIAA Member.
[3] Avionics Engineer, Advanced Technology Division, 600 Blvd South, Ste. 304, AIAA Member.
[4] Engineering Director, Advanced Technology Division, 600 Blvd South, Ste. 304, AIAA Member.
[5] Engineer, Avionics Systems Test Branch,. EV21, Marshall Space Flight Center, Huntsville, AL.

F_y = Y component of the resultant pressure force acting on the vehicle
f, g = generic functions
h = height
i = time index during navigation
j = waypoint index
K = trailing-edge (TE) nondimensional angular deflection rate

I. Introduction

Ball Systems (a subsidiary of SAIC) has developed the Simulation Package for Autonomous Rendezvous Test and Analysis (SPARTAN), a high-fidelity on-orbit simulation featuring multiple six-degree-of-freedom vehicles in order to test next-generation automated rendezvous and docking (AR&D) guidance, navigation, and control (GN&C) algorithms for NASA's new Vision for Space Exploration. SPARTAN has been developed in a modular fashion in Matlab/Simulink in order to support all missions required for future exploration: Low Earth Orbit (LEO) visits to the International Space Station (ISS), LEO assembly of the lunar transfer vehicle, and Low Lunar Orbit (LLO) rendezvous and docking missions. In contrast to current NASA missions, where many spacecraft GN&C functions are performed by the Mission Control Center and onboard astronauts, the AR&D system to fulfill the new Vision for Space Exploration must be capable of operation with minimal inputs from ground-based control centers in order to conduct operations both inside and well outside of LEO.

The main objectives of the simulation are to:

- Test sensor requirements in a digital "closed-loop fashion"
  - Simulation runs from long-range rendezvous through docking
- Test Kalman filter algorithms in realistic fashion
  - Interactions with closed-loop guidance and control is only to assess whether sensor package accuracy is sufficient
- Simulate all Constellation missions that will require AR&D functionality
  - LEO ISS, LEO LTV Assembly, Low Lunar Orbit R&D Missions
  - Ability to achieve with minimal changes to the simulation initialization file
- Perform Monte Carlo analyses to determine subsystem performance and robustness
  - Vary simulation parameters from Matlab wrapper file to determine statistical measures of system performance
- Develop framework capable of real-time mission computer code generation
  - Autocoding Simulink diagrams into C can be conducted using Mathworks toolboxes (Real-Time Workshop + Embedded Coder)
  - Code can be used to support real-time HWIL testing

Autonomous state-based mission manager algorithms in SPARTAN have been developed for sequencing the vehicle through its various flight phases based on on-board sensor inputs. The software is also charged with detecting deviations from the nominal flight profile. Depending on the magnitude of the deviation, the vehicle trajectory can be re-planned to achieve mission objectives or aborted through the use of Collision Avoidance Maneuvers (CAMs) to avoid contact with the target vehicle. Advanced GN&C algorithms have been developed in SPARTAN that are responsible for achieving the flight phase goals commanded by the mission manager. The guidance algorithms can be used to autonomously maneuver the vehicle from initial orbit insertion through completion of the desired rendezvous and docking sequence, including a variety of relative motion translation profiles and station-keeping at various hold points. Closed-loop guidance algorithms are used to reject orbital disturbances to ensure the vehicle follows the commanded trajectory within the desired relative velocity profile. A variety of guidance algorithms have been implemented in order to perform performance characterizations: Lambert transfers, Clohessy-Wiltshire maneuvers, co-elliptic rendezvous strategies for proximity operations, and straight line and glideslope algorithms for terminal approach. The guidance commands are implemented using translation and attitude control systems to provide 6DOF control of each vehicle in the simulation.

SPARTAN includes representations of a variety of absolute and relative navigation sensors that may be used for NASA missions. Error models for radio frequency (RF), laser rangefinder, and video-based rendezvous sensors are currently incorporated into the simulation. Proprietary navigation sensor fusion algorithms have been developed that allow the integration of these sensor measurements to create a single optimal estimate of the relative state of the vehicles. The extended Kalman filter framework for integrating measurements based on the vehicles' orbital

mechanics and the high-fidelity sensor error models provide a solution with increased accuracy relative to any single sensor. The navigation algorithms also contain built-in measurement consistency checks that assist sensor fault detection, isolation, and recovery (FDIR) by applying statistical tests to compare the consistency of the measurements across all available vehicle sensors, flagging measurements that are significantly abnormal compared to the a priori estimates of the sensor error model parameters.

Future work on the project focuses on extending the versatility of the system. Performance of the complete proposed AR&D system --- including software, sensors, and mechanisms --- will be rigorously evaluated using Monte Carlo dispersion analysis. The results of the analysis will be used to more accurately determine sensor and mechanism requirements for future space missions. The ARTEMIS code will support Hardware-in-the-Loop (HWIL) testing before the end of Fiscal Year 2007. Select algorithms have been converted to C code using Real-Time Workshop in order to be hosted in a mission computer engineering development unit running an embedded real-time operating system (VxWorks). Portions of the GN&C algorithms will be subjected to HWIL testing in MSFC's Flight Robotics Laboratory (FRL) in late summer of 2007. The FRL is one of the world's premier test facilities for AR&D HWIL testing of sensors, mechanisms, and flight software; the facility includes a 3DOF flat floor and an 8DOF overhead crane for relative trajectory simulations.

SPARTAN also contains a direct interface to bdStudio, a visualization tool developed by bd Systems. The visualization tool allows simulation results to be intuitively understood by all project stakeholders and also aids as a simulation debugging tool. SPARTAN uses a TCP/IP socket connection to send simulation data to bdStudio for run-time animation of simulation results; bdStudio also supports post-run animation generation with selectable frame rate from stored data files. SPARTAN and bdStudio both support visualization using relative or inertial data. bdStudio has the capability to visualize all applicable AR&D missions, including earth and lunar orbit scenarios.

The remaining sections of this document will provide an overview of the SPARTAN simulation architecture and capabilities. An overview of the models and algorithms used will be provided, along with representative results for rendezvous missions.

II. Simulation Overview

A. Background

SPARTAN is designed to examine all phases of proximity operations, rendezvous, and docking, with special emphasis on the final phase of rendezvous when the two spacecraft are in close proximity to each other. This is defined as the relative range being less than 10 kilometers. At these distances, close attention must be paid to the relative motion of the vehicles. It is customary that one vehicle is referred to as the target vehicle, and the other is known as the chaser vehicle. It is assumed in SPARTAN that the target vehicle is passive and will not actively maneuver itself to ensure a successful rendezvous. Instead, responsibility for the success of the maneuver lies entirely with the chaser.

In this phase of rendezvous, it is common to describe the motion of the chaser vehicle relative to the target vehicle. However, experiments with inertial relative motion descriptions have shown that a relative motion description in inertial coordinates is not accurate enough in the presence of disturbances such as geopotential anomaly and atmospheric drag. Therefore, the relative motion of the chaser is usually expressed in the target Local-Vertical, Local-Horizontal (LVLH) frame. This frame is centered on the target and is defined by the following unit vectors:

$$\begin{aligned} \hat{\mathbf{k}} &= -\hat{\mathbf{R}} && \text{R - bar} \\ \hat{\mathbf{j}} &= \hat{\mathbf{k}} \times \hat{\mathbf{V}} && \text{H - bar} \\ \hat{\mathbf{i}} &= \hat{\mathbf{j}} \times \hat{\mathbf{k}} && \text{V - bar} \end{aligned} \tag{1}$$

Here $\hat{\mathbf{R}}$ is a unit vector aligned with the target inertial geocentric radius vector, and $\hat{\mathbf{V}}$ is a unit vector aligned with the target inertial velocity. Note that the H-bar unit vector is in the opposite direction of the orbit angular momentum vector. These unit vectors can be used to construct a direction cosine matrix which transforms vectors expressed in the Earth-Centered Inertial (ECI) frame to vectors expressed in the LVLH frame:

$$\mathbf{D}_{L/I} = \begin{bmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \end{bmatrix}^T \tag{2}$$

Equations of motion for relative motion can be derived for the LVLH frame. Let the vector positions of the chaser and target spacecraft relative to the center of the Earth be $\mathbf{r}$ and $\mathbf{r}_T$, and let the vector describing the relative position of the chaser with respect to the target be $\boldsymbol{\rho}$. Therefore, the chaser position can be expressed as:

$$\mathbf{r} = \mathbf{r}_T + \boldsymbol{\rho} \tag{3}$$

Differentiating this equation twice with respect to an inertial coordinate frame yields:

$$\ddot{\mathbf{r}} = \ddot{\mathbf{r}}_T + \ddot{\boldsymbol{\rho}} + 2(\boldsymbol{\omega} \times \dot{\boldsymbol{\rho}}) + \dot{\boldsymbol{\omega}} \times \boldsymbol{\rho} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{\rho}) \tag{4}$$

where $\boldsymbol{\omega}$ represents the orbital angular rate vector, $\ddot{\mathbf{r}}$ represents the inertial acceleration of the chaser, $\ddot{\mathbf{r}}_T$ represents the inertial acceleration of the target, $\ddot{\boldsymbol{\rho}}$ represents the acceleration of the chaser with respect to the target, $2(\boldsymbol{\omega} \times \dot{\boldsymbol{\rho}})$ represents the Coriolis acceleration, $\dot{\boldsymbol{\omega}} \times \boldsymbol{\rho}$ represents the Euler acceleration, and $\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{\rho})$ represents the centripetal acceleration.

Now define the inertial acceleration of the chaser to be:

$$\ddot{\mathbf{r}} = \mathbf{g} + \mathbf{A} \tag{5}$$

where $\mathbf{g}$ is the gravitational acceleration and $\mathbf{A}$ is the acceleration applied by external forces, such as thrust and atmospheric drag. Resolving the above equations into the x, y, and z LVLH components, solving for the relative accelerations with the assumption that $|\boldsymbol{\rho}| << |\mathbf{r}_T|$, and finally assuming a circular orbit results in the Clohessy-Wiltshire (CW) equations:

$$\begin{aligned} \ddot{x} - 2\omega z &= A_x \\ \ddot{y} + \omega^2 y &= A_y \\ \ddot{z} + 2\omega \dot{x} - 3\omega^2 z &= A_z \end{aligned} \tag{6}$$

An analytical homogeneous solution to this system can be easily found using the method of Laplace transforms if the external acceleration is assumed to be zero. The continuous time solution is given by:

$$x(t) = \left(\frac{4\dot{x}_0}{\omega} - 6z_0 \right) \sin(\omega\tau) - \frac{2\dot{z}_0}{\omega} \cos(\omega\tau) + (6\omega z_0 - 3\dot{x}_0)\tau + \left(x_0 + \frac{2\dot{z}_0}{\omega} \right) \tag{7}$$

$$y(t) = y_0 \cos(\omega\tau) + \frac{\dot{y}_0}{\omega} \sin(\omega\tau) \tag{8}$$

$$z(t) = \left(\frac{2\dot{x}_0}{\omega} - 3z_0 \right) \cos(\omega\tau) + \frac{\dot{z}_0}{\omega} \sin(\omega\tau) + \left(4z_0 - \frac{2\dot{x}_0}{\omega} \right) \tag{9}$$

where $\tau = t - t_0$ and t_0 is the initial time. The initial relative positions and velocities are represented by $x_0, y_0, z_0, \dot{x}_0, \dot{y}_0,$ and $\dot{z}_0$. The homogeneous Clohessy-Wiltshire equations can be written in state-variable form:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \ddot{x}(t) \\ \ddot{z}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2\omega \\ 0 & 3\omega^2 & -2\omega & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \\ \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \dot{y}(t) \\ \ddot{y}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} \tag{11}$$

From these equations, it is evident that another form of the general solution to the Clohessy-Wiltshire equations is:

$$\mathbf{x}(t) = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^{\mathrm{T}} = \mathbf{\Phi}(t, t_0)\mathbf{x}(t_0) \tag{12}$$

Here the state transition matrix relating the initial condition to the final state at time t is given by:

$$\mathbf{\Phi}(t, t_0) = \begin{bmatrix} \mathbf{\Phi}_{\mathbf{RR}}(\tau) & \mathbf{\Phi}_{\mathbf{R\dot{R}}}(\tau) \\ \mathbf{\Phi}_{\mathbf{\dot{R}R}}(\tau) & \mathbf{\Phi}_{\mathbf{\dot{R}\dot{R}}}(\tau) \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} \mathbf{\Phi}_{11}(\tau) & \mathbf{\Phi}_{12}(\tau) \\ \mathbf{\Phi}_{21}(\tau) & \mathbf{\Phi}_{22}(\tau) \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{\Phi}_{11}(-\tau) & \mathbf{\Phi}_{12}(-\tau) \\ \mathbf{\Phi}_{21}(-\tau) & \mathbf{\Phi}_{22}(-\tau) \end{bmatrix} \tag{14}$$

$$\mathbf{\Phi}_{\mathbf{RR}} = \begin{bmatrix} 1 & 0 & 6\omega(1-\cos(\omega\tau)) \\ 0 & \cos(\omega\tau) & 0 \\ 0 & 0 & 3\omega\sin(\omega\tau) \end{bmatrix} \tag{15}$$

$$\mathbf{\Phi}_{\mathbf{R\dot{R}}} = \begin{bmatrix} \frac{1}{\omega}(4\sin(\omega\tau) - 3\omega\tau) & 0 & \frac{2}{\omega}(1-\cos(\omega\tau)) \\ 0 & \frac{1}{\omega}\sin(\omega\tau) & 0 \\ -\frac{2}{\omega}(1-\cos(\omega\tau)) & 0 & \frac{1}{\omega}\sin(\omega\tau) \end{bmatrix} \tag{16}$$

$$\mathbf{\Phi}_{\mathbf{\dot{R}R}} = \begin{bmatrix} 0 & 0 & 6\omega(1-\cos(\omega\tau)) \\ 0 & -\omega\sin(\omega\tau) & 0 \\ 0 & 0 & 3\omega\sin(\omega\tau) \end{bmatrix} \tag{17}$$

$$\mathbf{\Phi}_{\mathbf{\dot{R}\dot{R}}} = \begin{bmatrix} 4\cos(\omega\tau) - 3 & 0 & 2\sin(\omega\tau) \\ 0 & \cos(\omega\tau) & 0 \\ -2\sin(\omega\tau) & 0 & \cos(\omega\tau) \end{bmatrix} \tag{18}$$

While the Clohessy-Wiltshire equations can be used to create a relative motion simulation of one vehicle, SPARTAN tracks both vehicles independently in inertial space. The translational states of each vehicle are then

differenced and transformed to the LVLH frame to provide information on the relative motion. This approach allows for more accurate results since it does not rely on the assumptions used to develop the equations listed above, such as small relative distances and circular orbits. For example, as the relative distance between the target and chaser increases, the Clohessy-Wiltshire equations become less accurate. This results from frame misalignment between the target LVLH frame and the chaser LVLH frame. By tracking the vehicles independently using inertial states, accurate results can be obtained, even at large separation distances. These results can then be compared to the results predicted by the Clohessy-Wiltshire equations to determine the impact of errors like frame misalignment. Understanding the accuracy of Clohessy-Wiltshire solutions for various conditions is important because relative guidance algorithms often rely on those solutions to predict future relative motion.

B. Architecture

SPARTAN consists of a Simulink model and a wrapper of Matlab m-files. These m-files are responsible for setting up and launching the Simulink model. After the simulation is complete, these files also process and display the results. The main file for SPARTAN is the execution m-file; this file is used to execute the entire simulation session. Additionally, the user chooses the case to be run and sets various simulation execution settings in this file. This file will then call each of the other wrapper functions and the Simulink model in the proper order.

When the execution file is run, the parameter m-file for the desired case is run. This loads the simulation case-specific parameters into the Matlab workspace. It also designates which parameter files are to be used for the target vehicle, the chaser vehicle, and the celestial body that the vehicles are orbiting. Next, the additional parameter files are run by the load m-file. At this point, the initialization m-file is called. This file performs limited error checking to ensure that the user has loaded a feasible simulation case. This m-file also formats any existing variables and calculates any additional variables needed by the Simulink model. When the initialization file has finished execution, all the variables needed by the Simulink model have been loaded to the Matlab workspace.

At this point, the execution file calls the Simulink model, and the simulation runs until a stop condition is reached. The SPARTAN Simulink model is separated into two loops, one for the target vehicle and one for the chaser vehicle. Both loops are further separated into three subsystems: the sensors, which take in the current states and generate the sensor outputs; the flight computers, which take in the sensor outputs and generate the actuator commands; and the plant, which takes in the actuator commands and generates the current state derivatives. Those state derivatives are then integrated to provide the next time step's states. There are two additional high-level subsystems. One is responsible for controlling the Simulink model's execution, and the other is responsible for interfacing with bdStudio. This layout is illustrated in Fig.1, which shows the highest level of the Simulink model. Each of the subsystems shown in Fig. 1 is further divided into subsystems to enhance readability and modularity.

After the simulation is complete, the execution file then calls the post-processing m-file. This file formats the data recorded by Simulink during execution and calculates any values of interest that were not saved during simulation. Finally, the execution file calls a plotting m-file to display the simulation results. The execution file can also save the simulation data for later analysis and/or visualization. Certain results, like residual position errors and fuel expenditure, can also be displayed on the screen.

A similar execution flow is followed if a Monte Carlo dispersion analysis is run. In this scenario, a dispersion m-file is called that loops through a subset of the functions above for each dispersion case. In each pass through this loop, the nominal case parameters are loaded to the workspace. A function then changes selected nominal values to the desired dispersed values based on a dispersion input file. The simulation is then initialized and executed. A minimal set of results is saved to save hard drive while still allowing for proper evaluation of the simulated mission. At this point, the dispersion function performs the next dispersion case until all have been performed. After completion of the entire set of dispersion runs, the minimum and maximum values of user-selected variables are provided, along with out-of-bounds checks on any variables of interest. The Simulink model and dispersion function are designed to allow for identical results to be obtained from identical initial conditions, even when run on a different computer, through careful selection of random number generator seeds. This allows a single case to be rerun so that all variables of interest can be recorded and the results analyzed in greater detail. The SPARTAN Monte Carlo architecture is also compatible with an in-house distributed computing utility. This utility allows multiple computers on a local-area network to run various cases of a Monte Carlo batch at the same time, greatly reducing the amount of time to complete the Monte Carlo analysis.

C. Capabilities

The SPARTAN architecture provides several benefits by combining a Simulink model with Matlab wrapper functions. The use of m-files for initialization avoids hard-coded values in the Simulink model, making it as generic as possible. The m-files also provide robust capabilities for plotting data compared to the limited functionality of

Simulink's scopes. The case parameter m-files are designed to allow for rapid reconfiguration of the simulation. If the parameter file has already been created, only one line must be changed in the execution file to run an entirely different simulation case. Additionally, the case parameter file designates which parameter files to use for the vehicles and orbital body. This ensures that those values are located in only one place, allowing them to be updated easily and allowing new vehicles and orbital bodies to be easily added to the simulation. The use of separate files for the vehicle and body parameters also allows the user to easily change which vehicles are being simulated and which body the vehicles are orbiting. Such changes only require altering two lines in the case parameter file. Finally, the SPARTAN architecture enables Monte Carlo dispersion runs to be easily performed.

SPARTAN also provides the capability to interface with a visualization tool, as mentioned earlier. This interface is specific to bdStudio, a tool which has been enhanced on this project to specifically support AR&D visualization needs. bdStudio, shown in Fig. 2, is a C++ program that utilizes Python scripting and the OpenGL library to create an environment where the user can introduce models of vehicles and orbital bodies that can be animated by data. It also supports on-screen plotting of data, movie recording, and the additional of effects like thrusters, sensor beams, stars, and accurate sun lighting/shadows. SPARTAN can record a data file to be used for visualization after completion of the simulation or can pass data to bdStudio during simulation through a TCP/IP socket connection for runtime visualization. This tool is invaluable in understanding simulation results and presenting them in an easy-to-understand format.

SPARTAN also contains timing control, enabling it to be run in real-time or a faster multiple of real-time. This functionality has been coupled with an interface for joystick commands and with run-time visualization by bdStudio to allow for man-in-the-loop execution of the SPARTAN simulation. Currently, manual control of the attitude channel has been tested, with extension to manual translational control possible in the future. The manual attitude control has been developed with the option to command either desired moments or desired angular rates, with a toggle switch enabling full control or fine control.

III. Simulation Details

The following subsections address the various components of the SPARTAN simulation in greater detail. Equations and references are provided for both the environmental models used and the implemented GN&C algorithms. The nomenclature used for vectors in the following sections consists of a subscript that denotes the frame in which the vector is expressed. The superscript identifies the frame or quantity in question, followed by a slash and an identification of the reference point or frame. Therefore, $\boldsymbol{\omega}_B^{B/I}$ would represent the angular rate of the body frame with respect to the inertial frame, expressed in the body frame. Some of these identifiers may be omitted if unnecessary.

A. Equations of Motion

SPARTAN currently provides 6DOF simulation of two vehicles. For each vehicle, the following state vector is used:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_I^{B/I} & \mathbf{v}_I^{B/I} & \mathbf{q}^{B/I} & \boldsymbol{\omega}_B^{B/I} \end{bmatrix}^T \tag{19}$$

It consists of the position, velocity, attitude quaternion, and angular rate of the vehicle, respectively. The attitude is tracked using a quaternion to avoid the singularities present in the Euler angle formulation. The following standard equations of motion are used (note that the quaternion derivative expression is provided for a quaternion with the scalar component in the fourth element):

$$\dot{\mathbf{p}}_{\mathbf{I}}^{\mathbf{B/I}} = \mathbf{v}_{\mathbf{I}}^{\mathbf{B/I}} \tag{20}$$

$$\dot{\mathbf{v}}_{\mathbf{I}}^{\mathbf{B/I}} = \mathbf{F}_{\mathbf{I}}/m \tag{21}$$

$$\dot{\mathbf{q}}^{\mathbf{B/I}} = \frac{1}{2}\begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \boldsymbol{\omega}_{\mathbf{B}}^{\mathbf{B/I}} \tag{22}$$

$$\dot{\boldsymbol{\omega}}_{\mathbf{B}}^{\mathbf{B/I}} = \mathbf{I}_{\mathbf{B}}^{-1}\left(\mathbf{M}_{\mathbf{B}} - \boldsymbol{\omega}_{\mathbf{B}}^{\mathbf{B/I}} \times \mathbf{I}_{\mathbf{B}} \boldsymbol{\omega}_{\mathbf{B}}^{\mathbf{B/I}}\right) \tag{23}$$

Here the vehicle mass is represented by m, and vehicle moment of inertia about the center of mass and expressed in the body frame is represented by $\mathbf{I}_{\mathbf{B}}$. Simulink's fixed-step 4th–order Runge-Kutta algorithm is currently used for the integration of these states.

B. Gravity

SPARTAN is designed to simulate rendezvous missions in orbit around either the Earth or the Moon. The parameters used in SPARTAN for these celestial bodies are provided in Table 1. The SPARTAN gravity model currently includes non-spherical effects (up to J4 effects). This gravity model, available from Reference, is given as:

$$\mathbf{F}_{\mathbf{I}}^{\mathbf{grav}} = m\mathbf{g}_{\mathbf{I}} = -m\frac{\mu}{r^3}\left[\mathbf{r}_{\mathbf{I}} + a^2\left(3J_2 p_2 + 4aJ_3 p_3 + 5a^2 J_4 p_4\right)\mathbf{r}_{\mathbf{I}} - c_\phi\left(J_2 q_2 + aJ_3 q_3 + a^2 J_4 q_4\right)\mathbf{s}_\varphi\right] \tag{24}$$

Here, r represents the norm of $\mathbf{r}_{\mathbf{I}}$, the inertial position vector of the vehicle. The values of the gravitational parameter, μ, and the J gravitational coefficients in Eq. (29) are provided in Table 2. Expressions for calculating the various simplifying terms are not included here, but are available in Reference. Through use of a simulation flag, the non-spherical effects can be turned off by replacing the constant J gravitational terms with zeros. This reduces Eq. (24) to the Keplerian gravity model. Accuracy of the Earth gravity model can be increased easily by the inclusion of additional terms and coefficients to capture higher-order effects. For improved accuracy in lunar orbits, the lunar gravity model has been considered but not yet implemented. Gravity gradient effects can also be included, and the gravity gradient moment can be described by the following equation:

$$\mathbf{M}_{\mathbf{B}}^{\mathbf{gg}} = \frac{3\mu}{r^4}\mathbf{r}_{\mathbf{B}} \times \frac{1}{r}\mathbf{I}_{\mathbf{B}}\mathbf{r}_{\mathbf{B}} \tag{25}$$

C. Aerodynamics

SPARTAN also supports the inclusion of residual drag effects on orbiting vehicles. Atmospheric properties are modeled using a lookup table based on the US 1976 Standard Atmosphere model. Implementation of higher fidelity atmosphere models, such as the Jacchia Reference Atmosphere has been considered but not yet implemented. Aerodynamic forces and moments are currently estimated with a fairly simple model. Aerodynamic effects can be disabled through use of a simulation flag and are automatically disabled if the simulation case corresponds to a lunar orbit. The current aerodynamic force model is expressed by the following equations

$$C_D = 0.01 + \frac{2\alpha}{\pi/2} \tag{26}$$

$$C_L = \begin{cases} 5.6C_D, & \text{if } C_D \le 0.25 \\ 1.4 - 0.6571C_D, & \text{if } C_D > 0.25 \end{cases} \tag{27}$$

$$\mathbf{F}_\mathbf{B}^{\text{aero}} = \mathbf{D}_{\mathbf{B/W}} \mathbf{F}_\mathbf{W}^{\text{aero}} = \mathbf{D}_{\mathbf{B/W}} \begin{bmatrix} -C_D q_\infty S & 0 & -C_L q_\infty S \end{bmatrix}^{\mathrm{T}} \tag{28}$$

Here, S represents the aerodynamics reference area, and $\mathbf{D}_{\mathbf{B/W}}$ represents the transformation matrix from the wind frame to the body frame. An expression for the dynamic pressure is provided in Eq.(29) below. The velocity with respect to the atmosphere is found by subtracting the inertial velocity of the vehicle from the velocity of the atmosphere (using the assumption that the atmosphere rotates rigidly with the Earth). The resultant relative velocity vector can be transformed from the inertial frame to the vehicle body frame for use in the calculation of the angle of attack, α, and sideslip, β.

$$q_\infty = \frac{1}{2}\rho_\infty v^2 = \frac{1}{2}\rho_\infty \left| \mathbf{v}_\mathbf{I} - \boldsymbol{\omega}_\mathbf{I}^\mathbf{E} \times \mathbf{r}_\mathbf{I} \right|^2 \tag{29}$$

In this equation, the atmospheric density, ρ, is provided by the atmosphere model and the Earth angular rotation vector is represented as $\boldsymbol{\omega}_\mathbf{I}^\mathbf{E}$. The aerodynamic moment model is expressed as follows, with l representing the aerodynamic reference length:

$$C_R = -0.1, \quad C_P = -\frac{\alpha}{\pi/2}, \quad C_Y = \frac{\beta}{\pi/2} \tag{30}$$

$$\mathbf{M}_\mathbf{B}^{\text{aero}} = \begin{bmatrix} C_R q_\infty Sl & C_P q_\infty Sl & C_Y q_\infty Sl \end{bmatrix}^{\mathrm{T}} \tag{31}$$

D. Vehicle Mass Properties

SPARTAN currently includes parameter files for four vehicles: the International Space Station (ISS), the Crew Exploration Vehicle (CEV), the ascent stage of the Lunar Surface Access Module (LSAM), and the Earth Departure stage (EDS). Basic mass properties for each vehicle are listed in Table 2. The mass of the chaser vehicle is constantly updated during simulation to account for propellant usage. This implementation currently does not update moments and products of inertia or center of gravity positions, but such functionality can be added as needed.

E. Actuators

Due to the passive role of the target vehicle during rendezvous, the target vehicle actuators are used only for attitude control. The active model consists of an idealized effector, which perfectly applies the desired attitude control moment to the vehicle. The actuator model implemented for the chaser vehicle in SPARTAN consists of a Reaction Control System (RCS) and a main engine. Since the main engine is not used after completion of phasing operations, the main engine model remains very basic. Actuator model development has instead been focused on the RCS model, which obviously is used extensively during proximity operations. The current RCS model assumes a thruster constellation consisting of a number of thruster banks at various locations on the vehicle. Each bank consists of a set of thrusters, with each thruster having a specified direction. The thrusters can have different sizes (thrust levels) at different banks. The RCS model responds to individual thruster on-time commands from the controller, calculating the resultant forces and moments by the following expression:

$$\begin{bmatrix} \mathbf{F}_B^{RCS} \\ \mathbf{M}_B^{RCS} \end{bmatrix} = \mathbf{A}^{RCS}\mathbf{u}^{RCS} \tag{32}$$

where $\mathbf{u}^{RCS}$ is an n x 1 vector of on-times, with n representing the number of thrusters. Here, the control effectiveness matrix of the RCS matrix is represented as $\mathbf{A}^{RCS}$. It is constructed from the thruster constellation specifications provided the chaser vehicle, while also taking into account the effect of modeled RCS errors and failures. This model tracks propellant usage by summing the on-times of all of the thrusters at each time step and then calculating how much mass was expelled as a result of the firings. For increased accuracy, the effect of thruster impingement on the target vehicle during the docking phase can be modeled. However, such a model has not yet been implemented in SPARTAN.

As mentioned above, thrust errors are implemented in the RCS model to evaluate the overall GN&C performance in the presence of such errors. These errors include off-time quantization, thrust level variation, thruster location/orientation errors, and individual thruster failures. Quantization of the thruster turn-off time models the digital nature of the thruster controller's clock. The implementation of this error only allows the thrusters to be turned off at certain time intervals corresponding to the cycle times of the thruster controller. This quantizes the time that a thruster can be on, thus quantizing its output impulse, assuming a constant thrust. Thrust level variations model the effects of small perturbations in the propellant flow rate or turbulence in the thruster itself by the addition of random perturbations to the nominal thrust level. The thruster misplacement and misalignment errors represent deficiencies in the measured parameters of the thruster constellation. The thruster position errors incorporate the effects of both C.G. estimation error and direct measurement error. Finally, any combination of thrusters can be failed. Currently, the RCS model supports thruster-closed failures, but thruster-open failures could be added.

F. Sensors

The target's sensor model is idealized and passes perfect state information to the target flight computer. The chaser sensor model is more realistic and provides a separate model for each sensor. The primary focus has been on relative sensors used for proximity operations. SPARTAN models a radio-frequency interrogator (RFI) and laser range finder (LRF) for long-range navigation, as well as a Video-Based Guidance Sensor (VBGS) and a digital correlator for navigation at closer ranges. The implemented sensor models calculate the true measured parameters from state information and then add typical errors (bias, scale factor, and noise errors) to provide realistic measurements. Currently, these error parameters are compiled from hardware requirements for the sensors, but will be replaced with measured parameters from laboratory testing of specific sensors. Several absolute navigation sensors have also been implemented in SPARTAN: and inertial measurement unit (IMU), star tracker, and a Global Positioning System (GPS) receiver. The inclusion of these sensors allows for realistic errors to be added to the navigation estimate of the inertial states used by guidance and control when the distance to the target prevents the use of relative navigation. The addition of these sensors also provides a framework for the implementation of a dual-state Kalman filter that tracks both the chaser relative and absolute states. It also enables analysis of the effect of absolute sensor errors on the initial acquisition of the target by the chaser's long-range relative sensors. Finally, a telemetry stream from the target to the chaser vehicle which provides the target vehicle's absolute states has been implemented as well.

G. Navigation

The navigation system in SPARTAN currently includes 4 separate Kalman filter implementations: a linear and an extended filter for translational states and a linear and an extended filter for rotational states. The inclusion of several different Kalman filter implemetations allows for trade studies on the performance of various filter algorithms and their effect on mission performance. The extended Kalman filters are able to perform sensor fusion on information from different types of sensors arriving at different rates. They are also able to estimate sensor error parameters, such as scale factors and biases. In addition, these filters are not influenced by dropped or missing sensor measurements. The filters have measurement acceptance testing implemented to recognize and discard faulty sensor measurements.

H. Mission Manager

The chaser's mission manager uses estimated state information and the current time to determine the current phase and its parameters. The entire rendezvous mission is divided into phases, where a phase is simply defined as a

period of time where a certain constant set of parameters are used by the simulation models. The use of multiple phases allows the chaser vehicle to utilize different guidance algorithms, guidance settings, Kalman filter settings, etc. at different segments of the rendezvous process. The mission manager also features basic functionality for adjusting mission goals based on its evaluation of current progress. Finally, the mission manager is responsible for determining when the mission has ended and the simulation is complete.

The SPARTAN mission manager can also perform automatic collision avoidance maneuvers (CAMs). This logic provides active trajectory protection; passive trajectory protection is provided by judicious selection of hold points and transfer maneuvers when creating nominal phases in the case parameter file. The automatic CAM functionality supports two types of aborts: a "basic" abort and a TPZ abort. The "basic" abort relocates the chaser vehicle to a previous nominal phase that corresponds to a hold point. The TPZ abort relocates the chaser vehicle to the boundary of the target proximity zone (TPZ). A basic abort can be triggered manually by setting an abort time for any of the nominal phases declared in the case parameter file. It can also be triggered automatically if the chaser vehicle decides that it has violated the keep-out zone (KOZ) approach corridor for the docking port. When the abort is triggered, the CAM logic searches the previous phases to find an acceptable hold point, and the vehicle uses predictive Clohessy-Wiltshire guidance to move to that point. The vehicle spends a period of time at the hold point in standby mode, corresponding to a diagnostic period during which the problem which caused the abort could be diagnosed. Once this standby period is completed, the vehicle continues again with the nominal mission.

If a basic abort occurs and there is no previous phase corresponding to a hold point, the vehicle will instead execute a TPZ abort. A TPZ abort can also be triggered if the chaser vehicle suffers 3 aborts in the same phase, signifying that it is unable to complete that phase satisfactorily. Additionally, if the vehicle is out of position at the end of a grace period, a TPZ abort is triggered. When a TPZ abort is triggered, the vehicle decides whether is better to move to a TPZ boundary hold point on the +V-bar or –V-bar side of the target. Once the desired hold point is determined, the chaser moves there using predictive Clohessy-Wiltshire guidance. Once the vehicle achieves the TPZ hold point, the simulation ends. The TPZ abort, which is mainly used when the vehicle cannot complete the nominal mission, signifies that the chaser has suffered a more serious problem. Therefore, it is moved to a hold point at a safe distance from the target where that problem can be addressed.

I. Guidance

The chaser's guidance system provides both translational guidance and attitude guidance. The translational guidance takes the current settings from the mission manager and the navigation data from the Kalman filters and determines commands designed to bring the chaser to a desired location with a desired final velocity. The commands are issued as either desired accelerations or desired delta-V's expressed in the body frame. The user can select how often these commands are issued, as well as which algorithm is used to generate the commands. Several guidance algorithms are currently implemented: zero-effort miss, predictive-proportional, predictive Clohessy-Wiltshire, glideslope, position-hold, relative Lambert transfers, and inertial guidance. Representative guidance algorithms that are used in representative docking missions are described in greater detail below.

The SPARTAN implementation of the predictive Clohessy-Wiltshire algorithm is typical used to transfer between hold points. It currently provides guidance to the desired location, but does not enforce the desired arrival velocity. This algorithm uses state transition matrices and the time remaining to complete the maneuver (desired maneuver duration minus elapsed time) to predict the relative state at the end of the maneuver from the current relative state. The state transition matrices used are the ones derived in Section II, but rearranged into in-plane and out-of-plane matrices. The algorithm then calculates the error in the predicted position and uses it to calculate the delta-V needed to drive that error to zero. The algorithm equations are reproduced below:

$$\mathbf{x}_{\mathrm{L}}^{\mathrm{IN}} = \begin{bmatrix} 1 & 6(\omega\Delta t - \sin\omega t) & \frac{4}{\omega}\sin\omega\Delta t - 3\Delta t & \frac{2}{\omega}(1-\cos\omega\Delta t) \\ 0 & (4-3\cos\omega\Delta t) & \frac{2}{\omega}\cos\omega\Delta t - 1 & \frac{\sin\omega\Delta t}{\omega} \\ 0 & 6\omega(1-\cos\omega\Delta t) & 4\cos\omega\Delta t - 3 & 2\sin\omega\Delta t \\ 0 & 3\omega\sin\omega\Delta t & -2\sin\omega\Delta t & \cos\omega\Delta t \end{bmatrix} \begin{bmatrix} x \\ z \\ \dot{x} \\ \dot{z} \end{bmatrix} \tag{33}$$

$$\mathbf{x}_{\mathrm{L}}^{\mathrm{OUT}} = \begin{bmatrix} \cos\omega\Delta t & \frac{\sin\omega\Delta t}{\omega} \\ -\omega\sin\omega\Delta t & \cos\omega\Delta t \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \tag{34}$$

$$\Delta\mathbf{V}_{\mathrm{L}} = \boldsymbol{\Phi}_{\mathrm{R\dot{R}}}^{-1}(\Delta t)\left(\begin{bmatrix} \mathbf{x}_{\mathrm{L}}^{\mathrm{IN}}(1) \\ \mathbf{x}_{\mathrm{L}}^{\mathrm{OUT}}(1) \\ \mathbf{x}_{\mathrm{L}}^{\mathrm{IN}}(2) \end{bmatrix} - \mathbf{R}_{\mathrm{T}} \right) \tag{35}$$

The glideslope algorithm implemented in SPARTAN enforces the desired final velocity constraint in addition to the position constraint; therefore, it is often used in SPARTAN for final approach and docking. The decelerating glideslope algorithm provides a linear decrease in velocity with respect to the distance to the target vehicle. This decrease is then discretized by the application of impulsive thrust at evenly spaced intervals. Like the predictive Clohessy-Wiltshire algorithm, state transition matrices are used to predict future positions and determine the delta-V needed to the correct position and velocity errors at the future time. However, instead of trying to reduce errors at the end of the maneuver like the predictive Clohessy-Wiltshire algorithm, the glideslope algorithm attempts to prevent errors at the next interval point. The basic equations of the glideslope algorithm are provided below, with greater detail provided in Reference.

$$\Delta\mathbf{V}_{\mathrm{L}} = \boldsymbol{\Phi}_{\mathrm{R\dot{R}}}^{-1}(t_{m+1} - t_m)\left(\mathbf{R}_{\mathrm{m+1}} - \boldsymbol{\Phi}_{\mathrm{RR}}(t_{m+1} - t_m)\mathbf{R}_{\mathrm{m}}\right) - \dot{\mathbf{R}}_{\mathrm{m}} \tag{36}$$

$$\mathbf{R}_{\mathrm{m+1}} = \mathbf{R}_{\mathrm{T}} + \left| \boldsymbol{\rho}_0 e^{at_m} + \frac{1}{a}\left(e^{at_m} - 1\right)\mathbf{V}_{\mathrm{T}} \right| \mathbf{u}_{\rho} \tag{37}$$

The attitude guidance provides several set attitude modes. These include alignment with any of the six directions along the LVLH axes, tracking of the target's attitude (with or without reversed heading), and pointing of the chaser body x-axis towards the target vehicle. The reversed heading mode is used when the chaser has its docking port on its nose (+ body x-axis) and it is trying to dock on the "front", or +V-bar side, of the target vehicle. In this case, the chaser must assume a heading of 180° relative to the LVLH frame to successfully dock.

J. Controller

The chaser's controller consists of three subsystems: a main thruster controller, an RCS controller, and an attitude controller. The main thruster controller issues open-loop fire commands to the main thruster based on on/off times that can be provided by the user in the case parameter file. The chaser's attitude controller is similar to the target's attitude controller, except that it generates the quaternion error signal differently to provide better performance during large reorientations.

The target controller's performance is determined by the gains applied to the quaternion and angular rate error feedback. These gains are loaded from the vehicle parameter file; the ISS values are chosen to allow for a ±3° oscillation with an angular rate equal to half of the orbital rate.

The algorithm that calculates the thruster on-times is simple to implement, near-optimal, and readily lends itself to changing thruster configurations, either from actual thruster constellation changes, or from impingement considerations during docking. The thruster on-time calculation algorithm takes in a thruster "A" matrix, which is a

6 by n matrix, where n is the number of thrusters. The A matrix holds each thruster's reaction capabilities in torque and force about and along each axis. The algorithm takes the pseudoinverse of this matrix and then multiplies it by the force and torque commands coming from the guidance and navigation subsystems, resulting in an n by 1 vector that can be interpreted as thruster on-times. Unfortunately, some of these on-times will probably have negative values, which are physically meaningless. To correct for this, the thruster A matrix is recalculated, putting zeros in the columns corresponding to the thrusters that were commanded to fire for a negative time. The pseudoinverse of this new A matrix is then calculated, and it is multiplied by the same force and torque commands as previously. This generates a new set of thruster on-times. Though these are not guaranteed to be positive, for common 20-thruster and 24-thruster configurations, they have been observed to generate only positive on-times; however, to prevent any physically meaningless numbers from propagating through the system, these new thruster on-times are then constrained to be positive and scaled linearly such that the longest thruster on-time is equal to the allotted thruster firing time. This preserves the direction of the force and moment generated by that command, as well as their relative magnitudes but not their absolute magnitude. If all thruster on-times were less than the allotted firing time, then the on-time is sent as-is to the actuator model. As was mentioned earlier, this algorithm is near-optimal but is not intended to be flight software. It was chosen chiefly for its grace when handling changing A matrices. As soon as an actual thruster configuration and firing logic is available, this algorithm is modular and can easily be upgraded.

There is an algorithm surrounding the thruster on-time calculator that makes sure guidance commands are handled properly. Since attitude control currently is run at a higher rate than translational control, there is the chance that a translational command cannot be carried out all in one attitude control timestep. If the guidance subsystem issues a translational command that will take multiple attitude control timesteps to execute, the guidance command is re-issued with the subsequent attitude commands until the full command has been achieved. If a new guidance command arrives before the previous guidance command has been fully executed, the unexecuted portion is dropped and replaced by the new command. Currently, this is done in body frame, but a planned upgrade is to make these calculations in the LVLH frame to make the force outputs more insensitive to attitude errors.

The minimum firing time of the thruster is related to the thruster's Minimum Impulse Bit (Min Bit), which is the smallest non-zero amount of impulse that the thruster can generate. This is implemented in the flight controller and not in the actuator model because it is assumed that the software will be aware of the Min Bit and not command thrusts less than that value from any thruster.

K. Miscellaneous

The post-processing function includes the capability to calculate the geodetic latitude, longitude, and altitude using the user-specified date and time. This allows for the creation of accurate ground tracks for both LEO and LLO missions and provides a simulation framework that can be extended to include analysis of the effect of ground station or satellite observability on a given mission. Accurate tracking of the simulation date and time also allows for eventual inclusion of the Sun's position and sun angle calculations in the simulation for examination of lighting concerns.

IV. Results

Results are provided for two nominal rendezvous missions: one in low Earth orbit (LEO) and the other in lunar orbit. In both simulation cases, the J4 gravity model is used and gravity-gradient torques are included. Aerodynamic effects are also included in the LEO case. In both cases, the chaser vehicles utilize perfect sensed information (no sensor errors are injected) and do not use their Kalman filters. The simulation time step is 0.1 seconds.

A. Nominal CEV Rendezvous with ISS in LEO

This mission consists of a docking between the CEV and ISS, with the CEV as the chaser vehicle. The ISS orbit is taken to be a perfectly circular orbit inclined at 51.6°, with an altitude of roughly 342 kilometers and a right ascension of the ascending node (RAAN) of 326.1°. The CEV starts 2.5 kilometers behind the ISS and 600 meters below, with an initial relative velocity of 1.029 meters/second in the V-bar direction. Relative to the chaser LVLH frame, the CEV has an initial attitude corresponding to a heading of 15°, a pitch of -10°, and a roll of 25°. The CEV also has an angular rate of 0.001 radians/second along each body axis. The ISS begins aligned with the target LVLH frame with an angular rate of 3.5×10^{-6} radians/second along each body axis. The mission is defined through the parameters set for each phase; these parameters are listed in Table 4. The desired relative position and velocity for each phase is expressed in the target LVLH frame. Desired velocities are not listed for phases where hold

guidance is used because that algorithm only maintains position and does not include any velocity constraints. The intercept time is the desired time for the chaser to reach the desired relative position for the current phase. If that time is reached and the vehicle is in close proximity to the desired position, the mission manager will increment the phase. If the vehicle is not close enough to the desired position when the intercept time is reached, the mission manager can extend the current phase to allow more time for the vehicle to achieve the desired position. The "Zero-X" option, which is activated for Phases 1 and 4, instructs the guidance system to only issue commands in the y and z LVLH directions by setting any commands along x to zero. This allows for a controlled drift in the V-bar direction while maintaining a desired R-bar and H-bar position. Therefore, the desired relative position in the V-bar direction is not used; the final V-bar position at the end of Phase 1 will depend solely on the initial velocity along V-bar, the intercept time, and the effect of errors and disturbances. Phase 4 also features an open-loop thrust applied by the RCS system which accelerates the vehicle from its hold position toward the target. Glideslope guidance is then used in Phase 5 to provide closed-loop control with deceleration effects to ensure the vehicle arrives at the desired location with minimal relative velocity. This open-loop thrust consists of 1000 Newtons applied in the – V-bar direction for 17.6 seconds.

The motion of the chaser relative to the target is shown in Figure 3. Note that the convention for these relative plots has the V-bar direction (x-axis) positive to the left and the R-bar direction (z-axis) positive downwards. The starting point for each phase is also labeled in Figure 3. Figure 4 shows the time response of the chaser's relative position along each LVLH axis, as well as the time response of the relative velocity. The difference between the actual and desired final relative position, expressed in the target LVLH frame, at the end of each phase is provided in Table 5. The propellant usage and applied delta-V for each phase is also listed in Table 5, with separate values listed for the delta-V expended to execute translational maneuvers and maintain the desired attitude.

Figure 3 illustrates the desired approach. The CEV begins on a lower orbit than the ISS, giving the CEV a shorter orbital period and providing a relative velocity in the V-bar direction. After the CEV passes below the ISS, it moves up to a V-bar position in front of ISS, executes a short position hold, and then completes the rendezvous. The results for this mission show very good performance from the guidance algorithms. Ignoring the V-bar position errors for Phases 1 and 4 (resulting from the lack of closed-loop control in that direction due to the "Zero-X" setting), the largest position error is seen to be on the order of centimeters. The glideslope approach provides an exponential decrease in relative velocity versus time. This can be seen in Figure 4, with the exponential velocity decease approximated by the discrete applications of thrust. The effect of the Phase 3 hold maneuver in eliminating the relative velocity and the delta-V applied during the open-loop RCS thrust can also be seen in this figure. The delta-V expenditures listed in Table 5 are reasonable, but may be conservative since perfect sensor information is utilized. Greater delta-V expenditure can be expected when imperfect information is used since the vehicle will have to adjust its position more often due to noise errors. The increase in delta-V expenditure will be most noticeable during long-range maneuvers where the errors in the sensor outputs will be the largest.

B. LSAM Rendezvous with CEV in LLO, with Automatic CAM

This mission consists of a docking between the CEV and LSAM Ascent Stage, with the LSAM as the chaser vehicle. The CEV is at an altitude of 100 kilometers in a perfectly circular orbit inclined at 22.0° with a RAAN of 0°. The LSAM starts 600 meters ahead of the CEV, with no initial relative velocity. Relative to the chaser LVLH frame, the LSAM has an initial attitude corresponding to a heading of -10°, a pitch of 15°, and a roll of -5°. The LSAM also has an angular rate of 0.001 radians/second along each body axis. The CEV begins aligned with the target LVLH frame with an angular rate of 0.001 radians/second along each body axis. The parameters for this mission are defined in Table 6. The open-loop thrust used in Phase 3 consists of 500 Newtons applied in the –V-bar direction for 10 seconds.

The motion of the chaser relative to the target is shown in Figure 5. The time response of the chaser's relative position along each LVLH axis and the relative speed is shown in Figure 6. The position errors, delta-V expenditure, and propellant usage are tabulated in Table 7. In this mission, as seen in Figure 5, the LSAM begins in front of the CEV. The Lambert guidance algorithm has the LSAM move above the CEV, increasing its orbital period and allowing the distance between the vehicles to reduce. The LSAM moves to a hold point on V-bar, and then completes the rendezvous. Again, these results depict excellent performance from the guidance algorithms. The exponential decrease in velocity through the use of glideslope guidance can be seen again in Figure 6. The glideslope approach approximates a straight-line approach through the use of small hops. Some of these hops can be seen in Figure 5 as the LSAM approaches the CEV.

V. Conclusion

The SPARTAN simulation has been developed in order to simulate realistic AR&D mission profiles. The simulation includes high-fidelity 6DOF representation of both the target and chaser vehicle and can be run for LEO and lunar rendezvous missions. The go-forward plan for simulation refinement and improvement has also been developed and will be implemented in the final six months of the projects. The simulation will satisfy all objectives discussed in the Introduction section by the completion of the project.

Acknowledgments

The authors would like to Marshall Space Flight Center for sponsoring this work. The authors would also like to thank the other engineers at SAIC who supported this program (Christine Dreas, Kevin Geohagan, Walter Hammond, Matthew Johnson, David Stark, Douglas Stranghoener, and Vadim Uchitel) for their contributions to the development of SPARTAN.

References

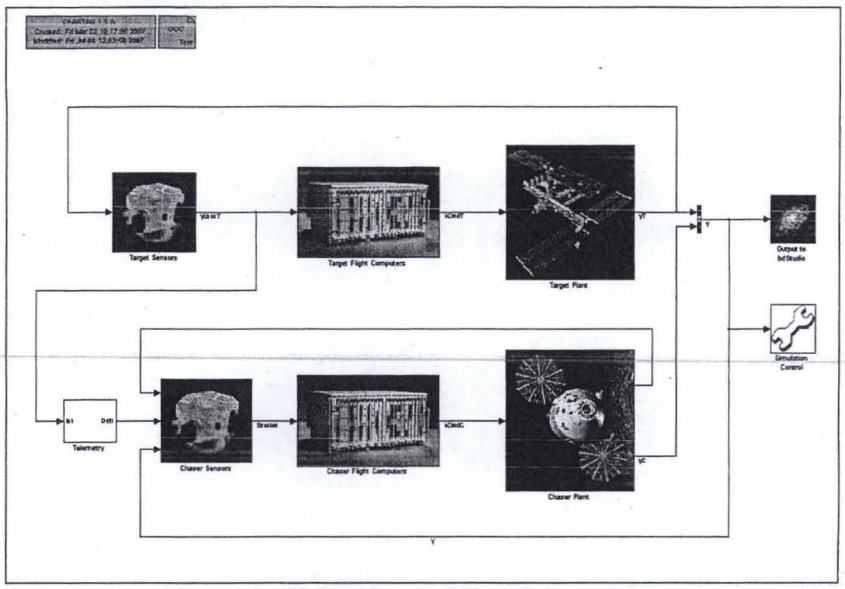

Figure 1: Highest level of Simulink model

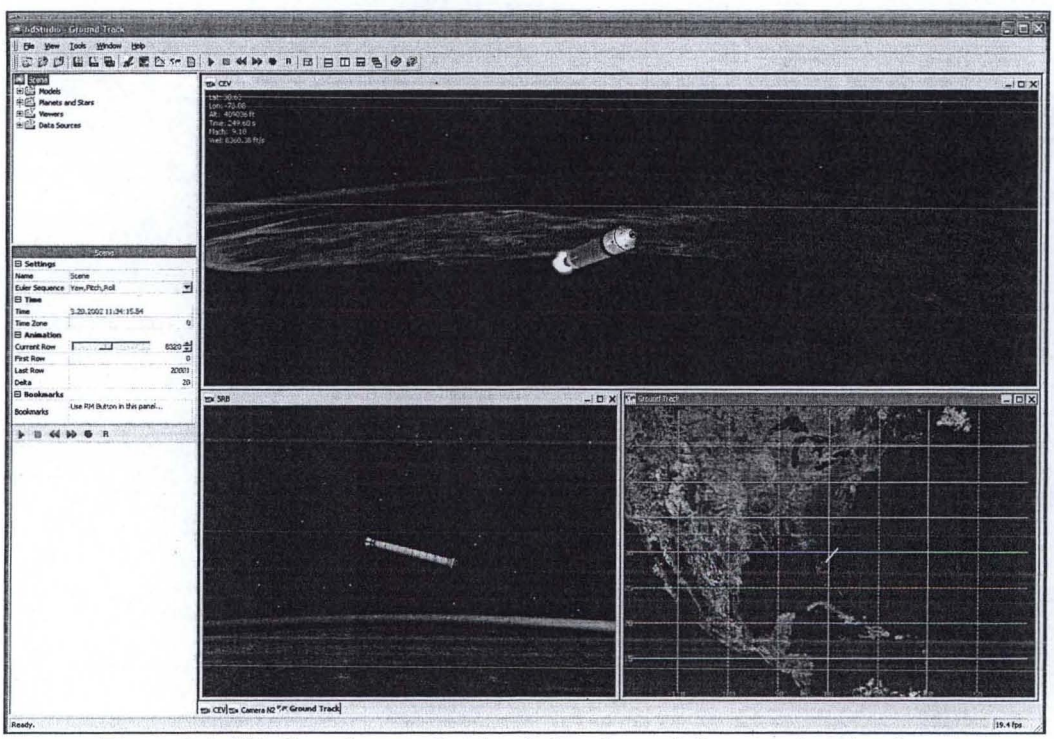

Figure 2: bdStudio enhances the simulation through its extensive visuals

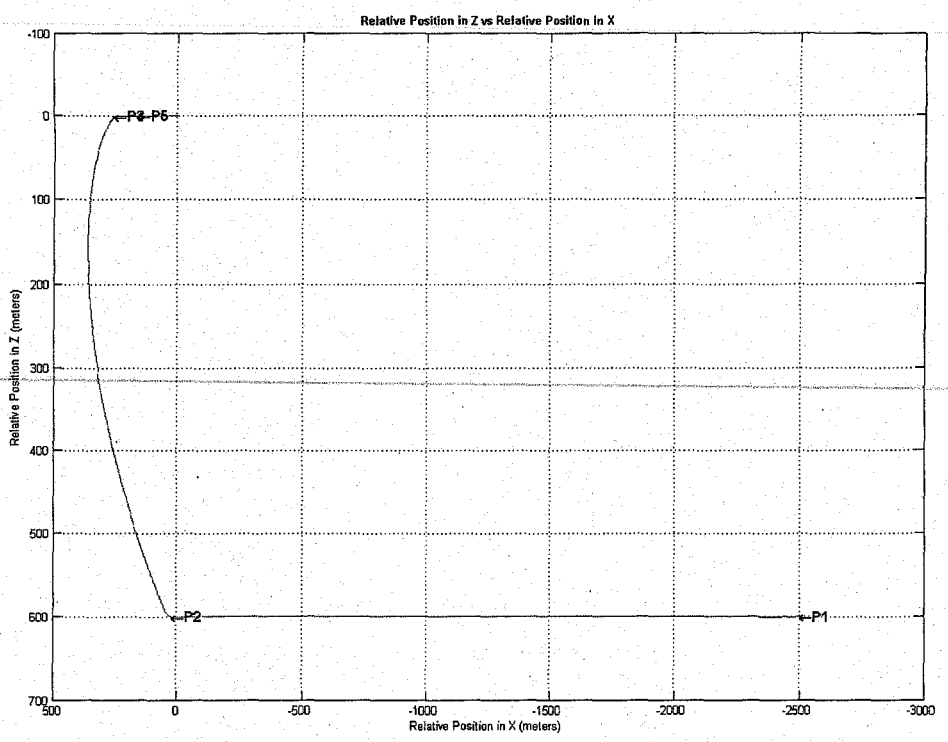

Figure 3. Chaser Relative Trajectory in the LVLH XZ-Plane.

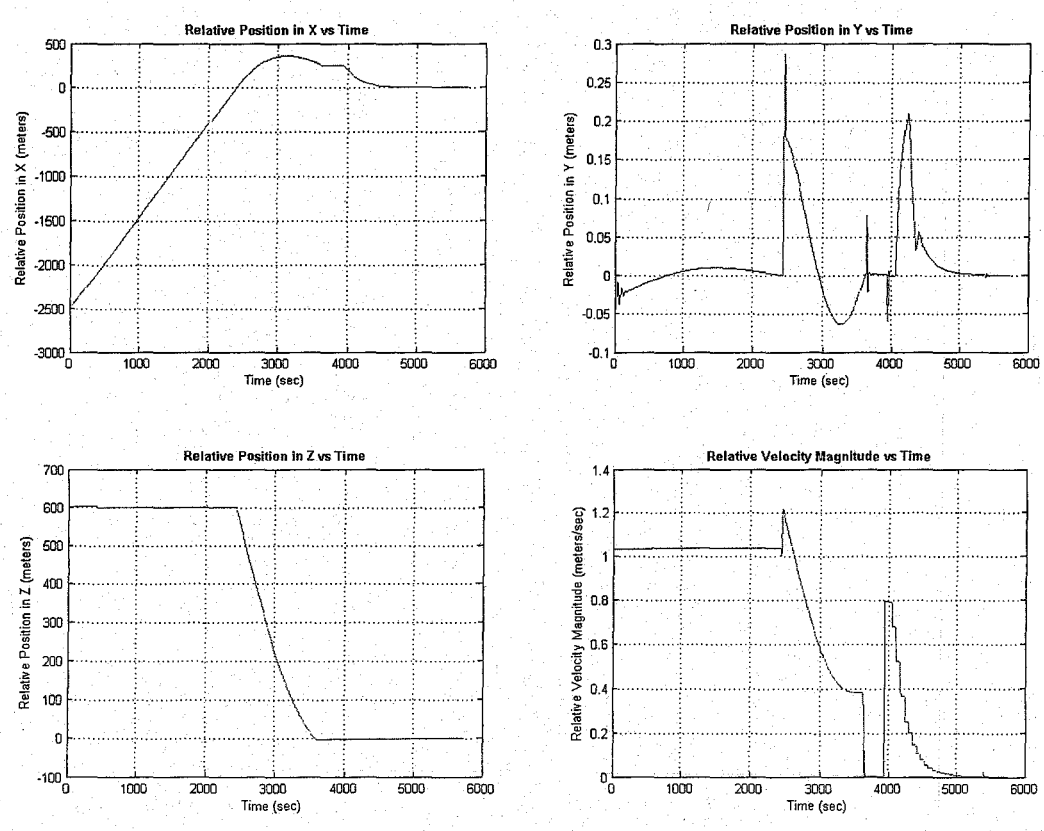

Figure 4. Chaser Relative Position & Velocity Magnitude.

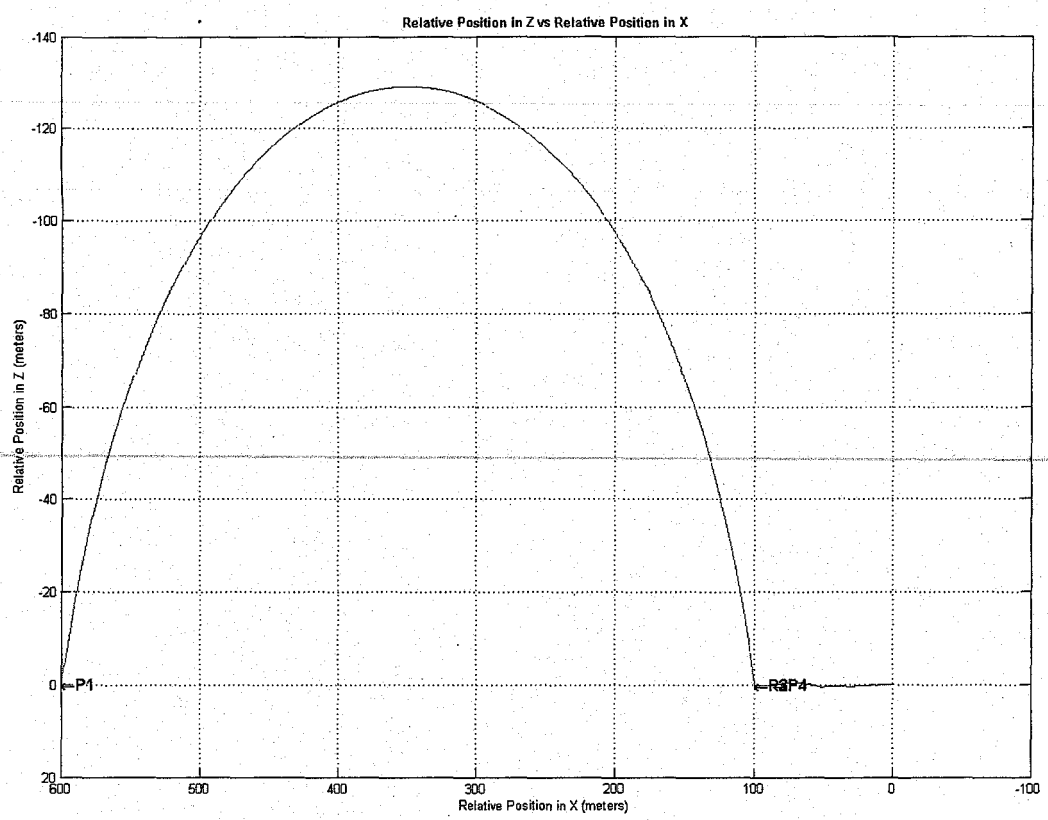

Figure 5. Chaser Relative Trajectory in the LVLH XZ-Plane.

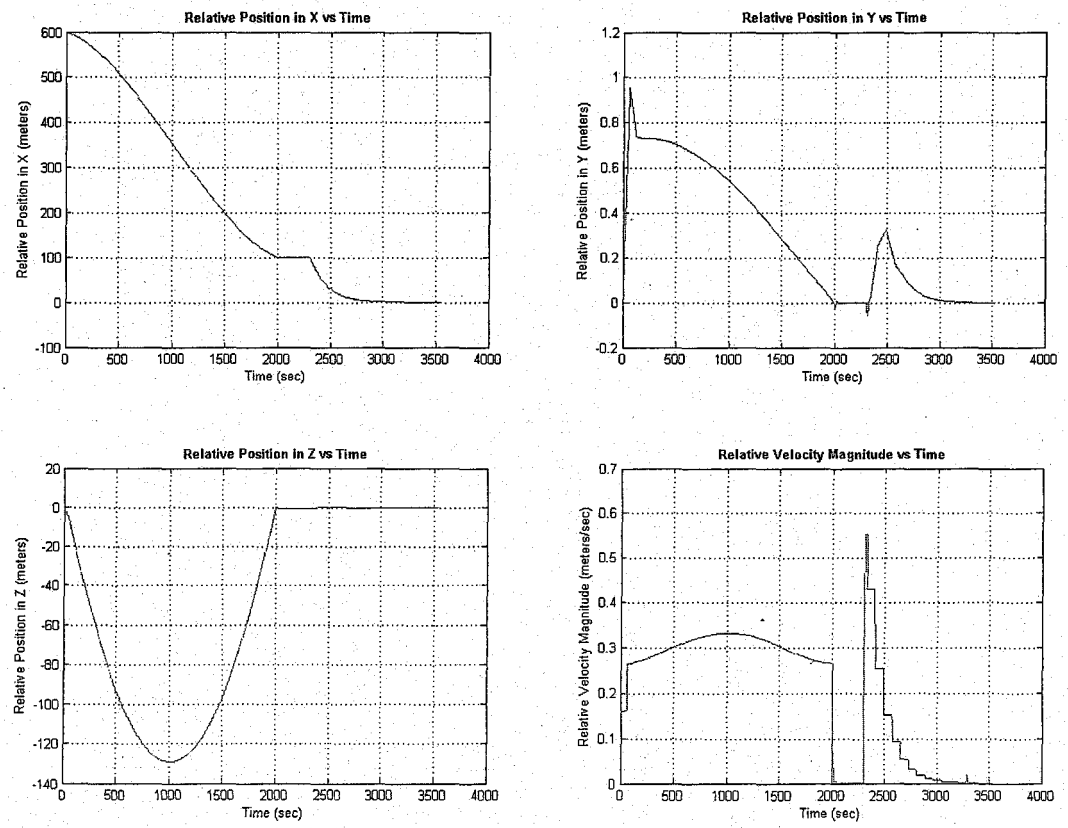

Figure 6. Chaser Relative Position & Velocity Magnitude.

Table 1. Celestial Body Parameters

	Earth	Moon
Rotation Rate (rad/s)	7.292115×10^{-5}	$1.526251526 \times 10^{-4}$
Equatorial Radius (m)	6378137.0	1738140.0
Polar Radius (m)	6356752.0	1735970.0
Mean Radius (m)	6367435.0	1737630.0
Flattening	1 / 298.257223563	1.24846×10^{-3}
Gravitational Parameter (m^3/s^2)	$3.986004995 \times 10^{14}$	4.902794×10^{12}
J2 Gravitational Term	1.08263×10^{-3}	202.7×10^{-6}
J3 Gravitational Term	-2.54×10^{-6}	7.69×10^{-6}
J4 Gravitational Term	-1.61×10^{-6}	0

Table 2. Vehicle Mass Properties

	ISS	CEV	LSAM
Mass (kg)	196028	22000	9000
Inertia ($kg\cdot m^2$)			
I_{xx}	1.28×10^8	59000	25000
I_{yy}	1.07×10^8	138000	60000
I_{zz}	2.01×10^8	138000	60000
I_{xz}	0	0	0
I_{xy}	0	0	0
I_{yz}	0	0	0

Table 3. Chaser Actuator Parameters

	CEV	LSAM
Main Engine		
Thrust (N)	33000	20000
Specific Impulse (s)	270	270
Propellant (kg)	9000	1000
Reaction Control System		
Max Force (N)		
X	1200	1000
Y, Z	800	500
Max Torque (N·m)		
X	2000	1500
Y, Z	800	500
Specific Impulse (s)	250	250
Propellant (kg)	1000	800

Table 4. LEO Mission Parameters

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Attitude Mode	LVLH	Rev Target	Rev Target	Rev Target	Rev Target
Guidance Mode	Hold	Pred CW	Hold	Hold	Glideslope
Guidance Interval (s)	10	5	3	3	60
Intercept Time (s)	2430	1200	300	125	1700
Desired Relative Position (m)					
X	(0)	250	250	(150)	0
Y	0	0	0	0	0
Z	600	0	0	0	0
Desired Relative Velocity (m/s)					
X	--	0	--	--	0

Y	--	0	--	--	0
Z	--	0	--	--	0
Misc Settings	Zero-X On	--	--	Zero-X On OL Thrust	--

Table 5. Selected Results from LEO Mission

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5**	--
Position Error (m)						
X	19.3444	0.0360	0.0000	7.6675	0.0000	--
Y	-0.0002	0.0006	0.0016	-0.0011	0.0000	--
Z	-0.0018	0.0150	0.0047	0.0676	0.0000	--
	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5**	**Totals**
Delta-V (m/s)						
Maneuver	0.249	0.778	0.698	1.008	0.931	3.665
Attitude	0.055	0.257	0.000	0.000	0.001	0.313
Totals	0.304	1.035	0.699	1.008	0.933	3.978
	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Phase 5**	**Totals**
Prop Used (kg)	2.699	9.111	6.127	8.986	8.182	35.105

Table 6. Lunar Mission Parameters

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**
Attitude Mode	Rev Target	Rev Target	Rev Target	Rev Target
Guidance Mode	Lambert	Hold	Hold	Glideslope
Guidance Interval (s)	60	3	3	80
Intercept Time (s)	2000	300	30	1200
Desired Relative Position (m)				
X	100	100	(100)	0
Y	0	0	0	0
Z	0	0	0	0
Desired Relative Velocity (m/s)				
X	0	--	--	0
Y	0	--	--	0
Z	0	--	--	0
Misc Settings	--	--	Zero-X On OL Thrust	--

Table 7. Selected Results from Lunar Mission

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	--
Position Error (m)					
X	0.0100	0.0000	-13.8081	0.0000	--
Y	-0.0003	-0.0001	0.0039	0.0000	--
Z	-0.0202	0.0050	0.0486	0.0000	--
	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Totals**
Delta-V (m/s)					
Maneuver	0.279	0.428	0.589	0.630	1.926
Attitude	0.280	0.000	0.000	0.001	0.281
Totals	0.559	0.429	0.589	0.631	2.207
	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**	**Totals**

Prop Used (kg)	2.046	1.559	2.159	2.268	8.032