Modular Software Architecture for Fully Coupled Spacecraft Simulations

Cody Allard,* Manuel Diaz Ramos,† Hanspeter Schaub,‡ Patrick Kenneally,* and Scott Piggott§
University of Colorado, Boulder, Boulder, Colorado 80309-0431

DOI: 10.2514/1.I010653

Computer simulations of spacecraft dynamics are widely used in industry and academia to predict how spacecraft will behave during proposed mission concepts. Current technology and performance requirements have placed pressure on simulations to be increasingly more representative of the environment and the physics that spacecraft will encounter. This results in increasingly complex computer simulations. Designing the software architecture in a modular way is a crucial step to allow for ease of testing, maintaining, and scaling of the software code base. However, for complex spacecraft modeling including flexible or multibody dynamics, modularizing the software is not a trivial task because the resulting equations of motion are fully coupled nonlinear equations. In this paper, a software architecture is presented for creating complex fully coupled spacecraft simulations with a modular framework. The architecture provides a solution to these common issues seen in dynamics modeling. The modularization of the fully coupled equations of motion is completed by solving the complex equations analytically such that the spacecraft rigid-body translational and rotational accelerations are solved for first and the other second-order state derivatives are found later. This architecture is implemented in the Basilisk astrodynamics software package and is a fully tested example of the proposed software architecture.

Nomenclature

$\boldsymbol{a}_\alpha, \boldsymbol{b}_\alpha, c_{\alpha_i}$	=	vectors and variable required for backsubstitution effector equation
B_c, E_c	=	rigid hub center-of-mass location and effector center-of-mass location
$\{\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2, \hat{\boldsymbol{b}}_3\}$	=	body frame basis vectors
$\boldsymbol{c}$	=	vector from point B to center of mass of the spacecraft C
$\{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{e}}_3\}$	=	effector frame basis vectors
$\boldsymbol{F}_{\text{ext}}$	=	vector sum of external forces on spacecraft
$[I_{\text{eff},E_c}]$	=	inertia matrix of effector about point E_c
$[I_{\text{sc},B}]$	=	inertia matrix of spacecraft about point B
$\boldsymbol{L}_B$	=	vector of sum of external torques of spacecraft about point B
$m_{\text{sc}}, m_{\text{hub}}, m_{\text{eff}}$	=	mass of spacecraft, hub, and effector, respectively
$\mathcal{N}, \mathcal{B}, \mathcal{E}$	=	inertial, body-fixed, and effector reference frames
N, B, E	=	inertial frame origin, body frame origin, and effector frame origin
$N_{\text{eff}}, N_{\text{DOF}}$	=	number of effectors and number of degrees of freedom of an effector
$\boldsymbol{r}_{B/N}$	=	position vector of B with respect to N
α	=	effector state variable
$\boldsymbol{v}_{\text{Rot,LHS}}, \boldsymbol{v}_{\text{Rot,RHS}}$	=	vectors required for backsubstitution rotational equation
$\boldsymbol{v}_{\text{Trans,LHS}}, \boldsymbol{v}_{\text{Trans,RHS}}$	=	vectors required for backsubstitution translational equation
$\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$	=	modified Rodrigues parameters representing $\mathcal{B}$ with respect to $\mathcal{N}$ frame
$\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$	=	angular velocity vector of $\mathcal{B}$ frame with respect to $\mathcal{N}$ frame

I. Introduction

IMPORTANT aspects when considering software design are the scalability, maintainability, and testability of the software [1]. If the software is not designed well, adding complexity (scalability), maintaining functionality amid a changing code base (maintainability), and the ease of verifying functionality (testability) can become extremely laborsome [2]. For complex simulations of spacecraft, this methodology needs to be considered to avoid these complications. However, multibody dynamics poses a difficult problem because of the coupled nature of the system through the nondiagonal system mass matrix [3]. This mass matrix relates the dynamical effect of the second-order state variables between all of the interconnected bodies.

Although multibody dynamics is a complex challenge, not only from an equation of motion (EOM) development perspective but from a software implementation perspective, there is an abundant amount of open-source software packages simulating multibody dynamics. Bullet [4] is an open-source multibody dynamics software package that uses the Gauss–Seidel method to solve the system mass inverse for diagonally dominant matrices [5]. Project CHRONO is an open-source multiphysics software package that uses parallel computing to solve multibody

Received 30 March 2018; revision received 31 August 2018; accepted for publication 24 September 2018; published online 22 October 2018. Copyright © 2018 by Cody Allard. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the ISSN 2327-3097 (online) to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Graduate Research Assistant, Department of Aerospace Engineering Sciences, 431 UCB, Colorado Center for Astrodynamics Research. Student Member AIAA.

†Graduate Research Assistant, Department of Aerospace Engineering Sciences, 431 UCB, Colorado Center for Astrodynamics Research.

‡Professor, Glenn L. Murphy Endowed Chair, Department of Aerospace Engineering Sciences, 431 UCB, Colorado Center for Astrodynamics Research. Associate Fellow AIAA.

§ADCS Integrated Simulation Software Lead, Laboratory for Atmospheric and Space Physics.

dynamics with a large number of degrees of freedom [6]. The Rigid Body Dynamics Library is an open-source multibody dynamics software package that uses the articulated body algorithm and the composite rigid-body algorithm for solving the dynamics [7]. Moby is a multibody dynamics software package that uses an interior point quadratic solver to solve for constraints [8]. Although these software packages are powerful for simulating a large number of bodies at a time, adding spacecraft-specific environmental factors and incorporating flight software into these open-source packages can be laborsome and is not the intended use of these software packages. Additionally, validation and verification of the simulation are important for spacecraft missions, and the capability to provide the necessary information for that process is not always a key feature of these open-source packages.

In contrast to the open-source packages, there are commercial software packages that are solving multibody dynamics problems. COMSOL is a multiphysics software package that has a multibody dynamics module for simulating multibody dynamics [9]. As this is a commercial software, the details of the software architecture and the method for solving the complex multibody dynamics are not readily available. Similarly, Adams is a multibody commercial software package that can simulate flexible and multibody dynamics. It allows for the input of CAD models to simulate the dynamics of complex systems [10]. MathWork's Simscape Multibody [11] can generate EOMs to be integrated and can output simulation code to MATLAB or C code. MotionGenesis uses Kane and Levinson's method [12,13] to output simulation code to MATLAB, C or Fortran and includes energy and momentum verification [14]. One downfall of these equation-of-motion generators is that the equations are specific to the system, which introduces scalability, maintainability, and testability issues for software architecture.

Computer graphics also has a strong influence in physics engine software, even though being visually realistic typically takes precedence over the dynamics accuracy. For example, Interactive Computer Graphics has a library called Position Based Dynamics Library [15] that uses position-based dynamics. This method, which integrates the position and velocity using kinematics, avoids physical constraints but is focused on being visually realistic. This results in the dynamics not being as accurate but the computations being extremely fast. Indeed, the software can simulate a very large number of degrees of freedom and is visually appealing [16,17]. Additionally, Interactive Computer Graphics has another physics engine called IBDS: Physics Library, which uses both forward dynamics and position-based dynamics [18,19].

In contrast to general multibody dynamics software, there are software packages that focus only on spacecraft simulation because of the unique environment that spacecraft encounter, as well as the specific challenges that modeling spacecraft dynamics entails. STK SOLIS is a software package for modeling spacecraft with both translational and attitude dynamics, but it does not model disturbances that can change the center of mass of the spacecraft: for example, flexing solar arrays [20]. The Jet Propulsion Laboratory has a software package called Dynamics Algorithms for Real-Time Simulation (or DARTS) [21]. This simulation software package uses spatial operator algebra for the development of the multibody dynamics [22] to create the system mass matrix in a form that can be solved efficiently with a recursive algorithm [23]. NASA's open-source software package named 42 [24] allows for spacecraft composed of multiple rigid or flexible bodies using tree topology [25] to formulate the dynamics, resulting in a system mass matrix inversion solution. OreKit is an open-source software package for spacecraft simulations and flight software; it models the spacecraft as a rigid body, and the dynamics are primarily focused on defining perturbations as external forces and torques [26].

The spacecraft-specific software packages described that involve multibody dynamics have to populate a system mass matrix and either have to find the inverse of the matrix or use other linear algebra techniques [22,23], assuming explicit integration techniques are being used. Populating the system mass matrix while retaining a modular software architecture is difficult because the system needs to know the locations of the states in the system mass matrix and know the relative locations of other coupled states. Additionally, inverting the system mass matrix can be computationally expensive because the calculation can scale with N^3, depending on the form of the system mass matrix and the method used to invert the matrix, with N being the number of states.

To combat these common problems, this paper introduces a method to generalize the EOMs that is applicable to a wide range of spacecraft configurations, uses a backsubstitution method to modularize the EOMs, and develops a software architecture that leverages the modularized equations. Although the prior methods allow for general multibody setups, this method is specifically developed for common spacecraft configurations in which there is a single rigid spacecraft hub onto which additional bodies (both rigid and flexible) are attached. This assumption is a key enabler that leads to an elegant modular framework that can be implemented in numerical simulations without dropping any dynamical coupling between the components. This allows for the underlying physics to be retained, which enables energy and momentum conservation checks to be completed. The resulting dynamics software is a turnkey solution that allows the user to rapidly configure a broad range of spacecraft configurations without having to derive equations of motion or integrate autocoded equations. The modular form allows for new types of dynamic forces and torques to be added without having to rederive all the other spacecraft equations of motion, enabling a layered approach to increase the simulation modeling capabilities.

II. Modularization of the Spacecraft Equations of Motion

A. Spacecraft Specific Compact Equations of Motion Form

Important considerations when first developing the EOMs are the associated assumptions because they will ultimately dictate how applicable the mathematical structure is to different dynamical systems. Figure 1 shows an example spacecraft with flexing solar arrays and lumped mass fuel slosh, and it will be the reference when discussing the assumptions [27]. Because both of these types of physical phenomena change the center of

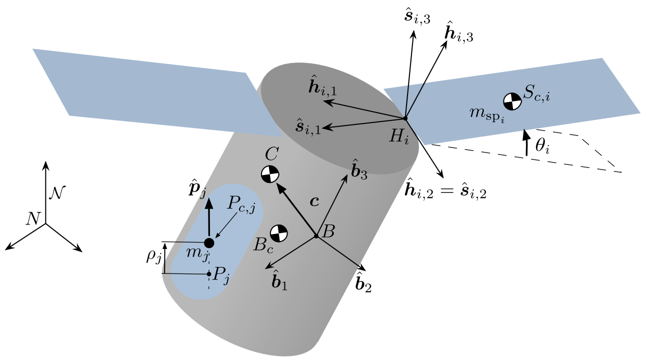


Fig. 1 Complex spacecraft with multiple degrees of freedom.

mass of the spacecraft, they are good examples for the multibody spacecraft problems. The common aspect that the majority of spacecraft share is that at least a small portion can be assumed to be rigid. In Fig. 1, the rigid portion of the spacecraft is the gray cylinder. This portion is called the rigid-body hub. The hub is assumed to have a nonzero mass m_{hub}, a center of mass location B_c, and an inertia matrix defined about its center of mass $[I_{\text{hub},B_c}]$.

The most important aspect of the rigid-body hub is that it is the object that the body frame $\mathcal{B}$: $\{\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2, \hat{\boldsymbol{b}}_3\}$ is attached to. To keep the formulation as general as possible, the body frame origin (point B) does not have to be coincident with the hub's center of mass (point B_c). It is very common to make the assumption that these two points are coincident, and it makes the derivation of the EOMs simpler [12,28]; however, allowing point B to be located at any location fixed with respect to the rigid hub gives more generality. It is very common in spacecraft missions that a structure frame is defined by the structural engineering team, where its origin is not coincident with the rigid-body hub's center of mass. Therefore, it gives flexibility in where the body frame origin can be defined. An additional assumption that keeps the formulation as general as possible is the inertia matrix $[I_{\text{hub},B_c}]$, which does not need to be diagonal when defined in body frame components. The formulation would be simpler but less general if a diagonal matrix were used [28–30].

Now that the rigid-body hub is defined, the state variables that define the state of the hub at any given time are the position of point B with respect to the origin of the inertial frame N ($\boldsymbol{r}_{B/N}$), the inertial velocity of point B with respect to point N ($\dot{\boldsymbol{r}}_{B/N}$), the modified Rodrigues parameters (MRPs) representation of the body frame $\mathcal{B}$ with respect to the inertial frame $\mathcal{N}$ ($\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$), and the inertial angular velocity vector of the body frame $\mathcal{B}$ with respect to the inertial frame $\mathcal{N}$ ($\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$). The MRPs are the chosen attitude parameterization set because it is a minimal set of three parameters with elegant nonsingular implementations [29]. However, the dynamics are independent of the chosen attitude parameterization; therefore, any attitude description can be used. These four variables represent the six degrees of freedom that the rigid-body hub exhibits and represent the 12 state variables that are needed to implement a second-order differential equation in software. These, at a minimum, are the states required for the system. All of the additional degrees of freedom on the system will be referenced to the body frame $\mathcal{B}$.

Now that the important parameters have been defined for the rigid-body hub, other degrees of freedom need to be introduced and generalized. Figure 1 shows an example with the flexing solar panels and lumped mass fuel slosh as additional degrees of freedom as an example system. Each of these models are labeled as "effectors." Each effector is assumed to have a mass m_{eff}, a center mass location E_c, and a position vector from point B to E_c ($\boldsymbol{r}_{E_c/B}$). If the effector has inertia properties, it also has a frame $\mathcal{E}$: $\{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{e}}_3\}$ and an inertia matrix $[I_{\text{eff},E_c}]$ that is defined about its center of mass E_c. Each effector is assumed to be attached to the rigid-body hub, and therefore not interconnected between other effectors. This a key assumption that enables the modular form introduced in this paper. However, interconnected effectors can be modeled as a single effector with multiple degrees of freedom attached to the rigid-body hub.

With the hub parameters and the effector parameters defined, the general form proposed in this research for the hub's EOMs is shown in Eqs. (1) and (2). This general form is formalized by using a systematic approach for multiple dynamics problem formulations including flexible solar arrays [31], spring mass damper-based fuel slosh [27], pendulum-based fuel slosh [32], imbalanced reaction wheels [33], fully coupled mass depletion [34], and imbalanced variable-speed control moment gyroscopes [35,36]. These references explain the derivations in detail and result in a familiar form. The first equation proposed for this general form is the translational motion equation:

$$\begin{aligned} m_{\text{sc}}\ddot{\boldsymbol{r}}_{B/N} - m_{\text{sc}}\boldsymbol{c}\times\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{\text{eff}}}\sum_{j=1}^{N_{\text{DOF},i}} \boldsymbol{v}_{\text{Trans,LHS}_{ij}}\ddot{\alpha}_{ij} &= \boldsymbol{F}_{\text{ext}} - 2m_{\text{sc}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}\times\boldsymbol{c}' \\ &- m_{\text{sc}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}\times(\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}\times\boldsymbol{c}) + \sum_{i=1}^{N_{\text{eff}}}\boldsymbol{v}_{\text{Trans,RHS}_i} \end{aligned} \tag{1}$$

The system parameters in Eq. (1) include the total mass of the spacecraft m_{sc}, the vector from point B to the instantaneous center of mass of the entire spacecraft $\boldsymbol{c}$, and the body frame relative time derivative with respect to the body frame of $\boldsymbol{c}$ ($\boldsymbol{c}'$). N_{eff} is the number of effectors, $N_{\text{DOF},i}$ is the ith effector's degrees of freedom, $\boldsymbol{v}_{\text{Trans,LHS}_{ij}}$ is a vector for the translational equation that corresponds with the jth degree of freedom of the ith effector's second-order derivative of its state α_{ij}, and $\boldsymbol{v}_{\text{Trans,RHS}_i}$ is the ith effector's vector contribution to the forces on the right-hand side (RHS) of Eq. (1). This proposed equation-of-motion form is general and common for any effector attached to a spacecraft. Later in this paper, specific formulations are illustrated for a select set of effectors.

The rotational EOMs' forms are proposed to be of the following form:

$$\begin{aligned} m_{\text{sc}}\boldsymbol{c}\times\ddot{\boldsymbol{r}}_{B/N} + [I_{\text{sc},B}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{\text{eff}}}\sum_{j=1}^{N_{\text{DOF},i}} \boldsymbol{v}_{\text{Rot,LHS}_{ij}}\ddot{\alpha}_{ij} &= \boldsymbol{L}_B - \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}\times([I_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}) \\ &- [I'_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{\text{eff}}}\boldsymbol{v}_{\text{Rot,RHS}_i} \end{aligned} \tag{2}$$

where $[I_{\text{sc},B}]$ is the inertia matrix of the total spacecraft (hub and effectors) about point B, $[I'_{\text{sc},B}]$ is the time derivative with respect to the body frame of $[I_{\text{sc},B}]$, $\boldsymbol{v}_{\text{Rot,LHS}_{ij}}$ is a vector for the rotational equation that corresponds with the jth degree of freedom of the ith effector's second-order derivative of its state α_{ij}, and $\boldsymbol{v}_{\text{Rot,RHS}_i}$ is the ith effector's vector contribution to the torques on the right-hand side of Eq. (2).

Finally, the individual effector degree of freedom EOMs are proposed to fit the following form:

$$a_{jj_i}\ddot{\alpha}_{ij} + \sum_{k=1;k\neq j}^{N_{\text{DOF},i}} a_{jk_i}\ddot{\alpha}_{ik} = \boldsymbol{a}_{\alpha_{ij}}\cdot\ddot{\boldsymbol{r}}_{B/N} + \boldsymbol{b}_{\alpha_{ij}}\cdot\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + c_{\alpha_{ij}} \tag{3}$$

where each effector has $N_{\text{DOF},i}$ EOMs needed to fully describe the motion of that effector. If $N_{\text{DOF},i} = 1$ for an effector, Eq. (3) simplifies to the following:

$$\ddot{\alpha}_i = \boldsymbol{a}_{\alpha_i}\cdot\ddot{\boldsymbol{r}}_{B/N} + \boldsymbol{b}_{\alpha_i}\cdot\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + c_{\alpha_i} \tag{4}$$

Equations (1–4) are the generalized EOMs that can apply to a wide variety of spacecraft. Using this common form yields consistent EOMs that enable the modular software architecture. While looking at Eqs. (3) and (4), it should be pointed out that the ith effector EOM does not include the second-order state variables from other effectors, but only the individual effectors and their corresponding degrees of freedom. This is a result of the assumption that effectors are not connected to other effectors but rather connected directly to the rigid-body hub. This specific form of the EOMs is a key insight that will allow for modularity between all of the effectors attached to the rigid-body hub.

B. Backsubstitution Method

A product of multibody dynamics is the dynamic coupling between the second-order state variables that results in a nondiagonal system mass matrix [28]. This can be troublesome in attempting to integrate the EOMs in software. When integrating EOMs, the form that is beneficial is $\dot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X}, t)$, where $\boldsymbol{X}$ is the state vector, $\dot{\boldsymbol{X}}$ is the time derivative of $\boldsymbol{X}$, and $\boldsymbol{f}(\boldsymbol{X}, t)$ is a function of the current state and time t. This form is convenient for numerical integration when using explicit integration techniques and, in this paper, implicit integration is not being considered. When there is a system mass matrix $[M]$ present, the form changes to $[M]\dot{\boldsymbol{X}} = \boldsymbol{g}(\boldsymbol{X}, t)$. Therefore, a system mass matrix needs to be inverted to solve this complex problem. This results in two problems with inverting a system mass matrix. First, inverting a matrix can be computationally inefficient because the calculation can scale with the cube of the number of states, depending on the method used. Second, the modularization of the dynamics from a software implementation perspective is a difficult task because the system needs to know the location of each effector within the system mass matrix and locations relative to the other effectors. A backsubstitution method is developed to solve this problem.

To visualize the impact of the EOMs' generalized form, the spacecraft seen in Fig. 2 is used as an example. The spacecraft has panels modeled as two interconnected rigid bodies with a single rotational degree of freedom each. Figure 2 only shows two sets of dual-connected solar panels, but the example is generalized to N_s number of sets. If the EOMs were put into the generalized form from the previous section, the dynamical coupling of this complex system would be visualized in the following schematic of the resulting coupled differential equations:

$$\begin{bmatrix} 3\times3 & 3\times3 & 3\times1 & 3\times1 & 3\times1 & 3\times1 & . & 3\times1 & 3\times1 \\ 3\times3 & 3\times3 & 3\times1 & 3\times1 & 3\times1 & 3\times1 & . & 3\times1 & 3\times1 \\ 1\times3 & 1\times3 & 1\times1 & 1\times1 & 0 & 0 & . & 0 & 0 \\ 1\times3 & 1\times3 & 1\times1 & 1\times1 & 0 & 0 & . & 0 & 0 \\ 1\times3 & 1\times3 & 0 & 0 & 1\times1 & 1\times1 & . & 0 & 0 \\ 1\times3 & 1\times3 & 0 & 0 & 1\times1 & 1\times1 & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . \\ 1\times3 & 1\times3 & 0 & 0 & 0 & 0 & . & 1\times1 & 1\times1 \\ 1\times3 & 1\times3 & 0 & 0 & 0 & 0 & . & 1\times1 & 1\times1 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}}_{B/N} \\ \dot{\boldsymbol{\omega}}_{B/N} \\ \ddot{\theta}_{11} \\ \ddot{\theta}_{12} \\ \ddot{\theta}_{21} \\ \ddot{\theta}_{22} \\ . \\ \ddot{\theta}_{N_s 1} \\ \ddot{\theta}_{N_s 2} \end{bmatrix} = \begin{bmatrix} 3\times1 \\ 3\times1 \\ 1\times1 \\ 1\times1 \\ 1\times1 \\ 1\times1 \\ . \\ 1\times1 \\ 1\times1 \end{bmatrix} \tag{5}$$

Equation (5) shows the form of the second-order state variable coupling that results from this configuration. This equation is included as a schematic to show the form and sparsity of the system mass matrix on the left-hand side (LHS) of the equation. Each element in the matrix is showing the size of the corresponding submatrix. For example, a 3×3 element is indicating that a submatrix of size 3×3 is present at that location. The dashed rows and columns are to indicate that this matrix has been generalized for N_s sets of panels and shows that the pattern repeats throughout the matrix. Equation (5) confirms that the individual degrees of freedom for the sets of solar panels are coupled with each other, but they do not directly couple through second-order state derivatives with the other sets of panels. This is a key insight and is exploited in the following backsubstitution method.

Looking further into Eq. (5), all of the solar panel second-order state derivatives are present in the hub translational and rotational equations. On the other hand, the hub translational and rotational second-order state variables are present in the individual solar panel EOMs. This dynamic coupling through the hub is another key insight that the backsubstitution method will use to modularize the EOMs.

This section of the paper expresses the vector equations shown in the past section as matrix equations. These equations do not specify a frame in which the matrix components are expressed with respect to but when implementing the equations in software, a single reference frame must be used. A common frame to in which to express the equations would be the body frame $\mathcal{B}$. Since these equations are matrix equations the following notation is used: the cross product is expressed as $[\tilde{\boldsymbol{a}}]\boldsymbol{b}$, the dot product is expressed as $\boldsymbol{a}^T\boldsymbol{b}$, and the outer product is expressed as $\boldsymbol{a}\boldsymbol{b}^T$.

The backsubstitution method is presented for effectors that have $N_{\mathrm{DOF},i} = 1$ for this paper but is extended to effectors with multiple degrees of freedom in Ref. [36]. The first step in the backsubstitution method is to substitute Eq. (4) into both the translational and rotational EOMs for the rigid-body hub. First, the substitution into the translational motion for $N_{\mathrm{DOF},i} = 1$ is shown in the following equation:

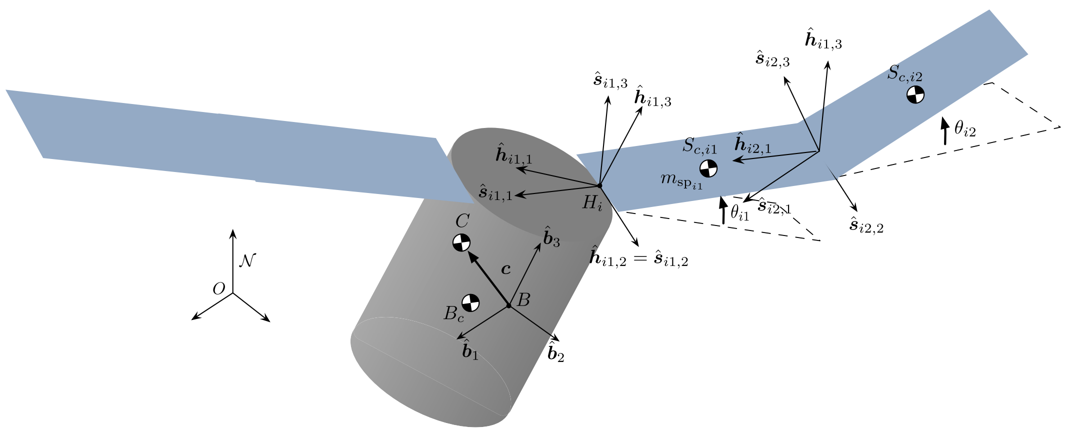

$\hat{\boldsymbol{h}}_{i1,3}$
$\hat{\boldsymbol{s}}_{i2,3}$
$\hat{\boldsymbol{s}}_{i1,3}$
$\hat{\boldsymbol{h}}_{i1,3}$
$S_{c,i2}$
θ_{i2}
$S_{c,i1}$
$\hat{\boldsymbol{h}}_{i2,1}$
$\hat{\boldsymbol{h}}_{i1,1}$
$m_{\mathrm{sp}_{i1}}$
$\hat{\boldsymbol{s}}_{i1,1}$
H_i
θ_{i1}
$\hat{\boldsymbol{s}}_{i2,1}$
$\hat{\boldsymbol{s}}_{i2,2}$
C
$\hat{\boldsymbol{b}}_3$
$\hat{\boldsymbol{h}}_{i1,2} = \hat{\boldsymbol{s}}_{i1,2}$
$\mathcal{N}$
O
$\boldsymbol{c}$
B
B_c
$\hat{\boldsymbol{b}}_1$
$\hat{\boldsymbol{b}}_2$

Fig. 2 Dual-hinged rigid-bodies frame and variable definitions.

$$
\begin{aligned}
m_{\text{sc}}\ddot{\boldsymbol{r}}_{B/N} - m_{\text{sc}}[\tilde{\boldsymbol{c}}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Trans,LHS}_i}[\boldsymbol{a}_{\alpha_i}^T\ddot{\boldsymbol{r}}_{B/N} + \boldsymbol{b}_{\alpha_i}^T\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + c_{\alpha_i}] = \boldsymbol{F}_{\text{ext}} \\
- 2m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c}' - m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c} + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Trans,RHS}_i}
\end{aligned}
\tag{6}
$$

Simplifying and combining like terms yield the translational EOM that has been decoupled from the other effector accelerations:

$$
\begin{aligned}
\left[m_{\text{sc}}[I_{3\times3}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Trans,LHS}_i}\boldsymbol{a}_{\alpha_i}^T\right]\ddot{\boldsymbol{r}}_{B/N} + \left[-m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Trans,LHS}_i}\boldsymbol{b}_{\alpha_i}^T\right]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} = \boldsymbol{F}_{\text{ext}} \\
- 2m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c}' - m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c} + \sum_{i=1}^{N_{\text{eff}}}[\boldsymbol{v}_{\text{Trans,RHS}_i} - \boldsymbol{v}_{\text{Trans,LHS}_i}\boldsymbol{c}_{\alpha_i}]
\end{aligned}
\tag{7}
$$

Following the same pattern for the rotational hub EOM [Eq. (2)] yields the following:

$$
\begin{aligned}
\left[m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Rot,LHS}_i}\boldsymbol{a}_{\alpha_i}^T\right]\ddot{\boldsymbol{r}}_{B/N} + \left[[I_{\text{sc},B}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Rot,LHS}_i}\boldsymbol{b}_{\alpha_i}^T\right]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} = \boldsymbol{L}_B \\
- [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - [I'_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{\text{eff}}}[\boldsymbol{v}_{\text{Rot,RHS}_i} - \boldsymbol{v}_{\text{Rot,LHS}_i}c_{\alpha_i}]
\end{aligned}
\tag{8}
$$

The following matrices are defined to yield a more compact notation:

$$
[A] = m_{\text{sc}}[I_{3\times3}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Trans,LHS}_i}\boldsymbol{a}_{\alpha_i}^T \tag{9}
$$

$$
[B] = -m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Trans,LHS}_i}\boldsymbol{b}_{\alpha_i}^T \tag{10}
$$

$$
[C] = m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Rot,LHS}_i}\boldsymbol{a}_{\alpha_i}^T \tag{11}
$$

$$
[D] = [I_{\text{sc},B}] + \sum_{i=1}^{N_{\text{eff}}} \boldsymbol{v}_{\text{Rot,LHS}_i}\boldsymbol{b}_{\alpha_i}^T \tag{12}
$$

$$
\boldsymbol{v}_{\text{Trans}} = \boldsymbol{F}_{\text{ext}} - 2m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c}' - m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c} + \sum_{i=1}^{N_{\text{eff}}}[\boldsymbol{v}_{\text{Trans,RHS}_i} - \boldsymbol{v}_{\text{Trans,LHS}_i}c_{\alpha_i}] \tag{13}
$$

$$
\boldsymbol{v}_{\text{Rot}} = \boldsymbol{L}_B - [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - [I'_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{\text{eff}}}[\boldsymbol{v}_{\text{Rot,RHS}_i} - \boldsymbol{v}_{\text{Rot,LHS}_i}c_{\alpha_i}] \tag{14}
$$

Using these definitions, the coupled translation and rotation hub EOMs are written compactly as

$$
\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}\begin{bmatrix} \ddot{\boldsymbol{r}}_{B/N} \\ \dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{\text{Trans}} \\ \boldsymbol{v}_{\text{Rot}} \end{bmatrix} \tag{15}
$$

Equation (15) represents a system of six equations that can be solved using the Schur complement matrix formulation for the partitioned form of the hub system mass matrix:

$$
\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} = ([D] - [C][A]^{-1}[B])^{-1}(\boldsymbol{v}_{\text{Rot}} - [C][A]^{-1}\boldsymbol{v}_{\text{Trans}}) \tag{16}
$$

$$
\ddot{\boldsymbol{r}}_{B/N} = [A]^{-1}(\boldsymbol{v}_{\text{Trans}} - [B]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}) \tag{17}
$$

This shows that the backsubstitution method only requires two 3×3 matrix inverses. The additional degree-of-freedom second-order state derivatives are found by backsubstituting these solutions into Eqs. (3) and (4).

C. Modularization of Energy and Momentum

A key part of EOM development is expressing the total energy and momentum of the spacecraft for verification purposes. This section describes the proposed method for how the energy and momentum are calculated and modularized for each effector to add their contributions to the overall total energy. It is advantageous to define the energy and momentum of the center of mass of the spacecraft (orbital) and about its center of mass (rotational). This is because the orbital and rotational energies typically have different orders of magnitude, and so separating these terms will avoid numerical issues in the verification process, and both quantities should be conserved in applicable scenarios.

First, the orbital energy is analytically expressed to be in terms of the state variables. The total orbital kinetic energy (i.e., kinetic energy of the center of mass) of the spacecraft is

$$T_{\text{orb}} = \frac{1}{2} m_{\text{sc}} \dot{\boldsymbol{r}}_{C/N} \cdot \dot{\boldsymbol{r}}_{C/N} \tag{18}$$

Expanding $\dot{\boldsymbol{r}}_{C/N}$ to be in terms of $\dot{\boldsymbol{r}}_{B/N}$ and $\dot{\boldsymbol{c}}$ results in

$$T_{\text{orb}} = \frac{1}{2} m_{\text{sc}} (\dot{\boldsymbol{r}}_{B/N} + \dot{\boldsymbol{c}}) \cdot (\dot{\boldsymbol{r}}_{B/N} + \dot{\boldsymbol{c}}) \tag{19}$$

This simplifies to the final desired equation:

$$T_{\text{orb}} = \frac{1}{2} m_{\text{sc}} (\dot{\boldsymbol{r}}_{B/N} \cdot \dot{\boldsymbol{r}}_{B/N} + 2\dot{\boldsymbol{r}}_{B/N} \cdot \dot{\boldsymbol{c}} + \dot{\boldsymbol{c}} \cdot \dot{\boldsymbol{c}}) \tag{20}$$

Each effector contributes to $\boldsymbol{c}$ and $\dot{\boldsymbol{c}}$, but it does not have direct individual contributions. Additionally, in this form, each effector does not need to know about the center-of-mass location of the spacecraft, which is advantageous from a modularity perspective.

The total orbital potential energy depends on what type of gravity model is being used or if other conservative external forces are acting on the spacecraft. For simplicity, the orbital potential energy due to point gravity is included here, but spherical harmonics and other higher-order effects could be included:

$$V_{\text{orb}} = -\frac{\mu}{|\boldsymbol{r}_{C/N}|} \tag{21}$$

It is convenient to combine the kinetic and potential energies into one term E_{orb} because the total orbital energy of the spacecraft must be conserved when there are no nonconservative external forces and torques acting on the spacecraft. This is shown in the following equation:

$$E_{\text{orb}} = T_{\text{orb}} + V_{\text{orb}} \tag{22}$$

Next, there is the expression of the rotational energy. The total rotational and deformational kinetic energy (i.e., kinetic energy about the center of mass) of the spacecraft is

$$\begin{aligned} T_{\text{rot}} &= \frac{1}{2} \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \cdot [I_{\text{hub},B_c}] \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \frac{1}{2} m_{\text{hub}} \dot{\boldsymbol{r}}_{B_c/C} \cdot \dot{\boldsymbol{r}}_{B_c/C} \\ &+ \sum_{i=1}^{N_{\text{eff}}} \left(\frac{1}{2} \boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} \cdot [I_{\text{eff},E_{c,i}}] \boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} + \frac{1}{2} m_{\text{eff}} \dot{\boldsymbol{r}}_{E_{c,i}/C} \cdot \dot{\boldsymbol{r}}_{E_{c,i}/C} \right) \end{aligned} \tag{23}$$

Expanding and combining like terms results in

$$\begin{aligned} T_{\text{rot}} = \frac{1}{2} \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \cdot [I_{\text{hub},B_c}] \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \frac{1}{2} m_{\text{hub}} \dot{\boldsymbol{r}}_{B_c/B} \cdot \dot{\boldsymbol{r}}_{B_c/B} + \sum_{i=1}^{N_{\text{eff}}} \bigg[\frac{1}{2} \boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} \cdot [I_{\text{eff},E_{c,i}}] \boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} \\ + \frac{1}{2} m_{\text{eff}} \dot{\boldsymbol{r}}_{E_{c,i}/B} \cdot \dot{\boldsymbol{r}}_{E_{c,i}/B} \bigg] - \left[m_{\text{hub}} \dot{\boldsymbol{r}}_{B_c/B} + \sum_{i=1}^{N_{\text{eff}}} m_{\text{eff}} \dot{\boldsymbol{r}}_{E_{c,i}/B} \right] \cdot \dot{\boldsymbol{c}} + \frac{1}{2} \left[m_{\text{hub}} + \sum_{i=1}^{N_{\text{eff}}} m_{\text{eff}} \right] \dot{\boldsymbol{c}} \cdot \dot{\boldsymbol{c}} \end{aligned} \tag{24}$$

Performing a final simplification yields the desired result for which each effector adds its contributions to the rotational energy:

$$\begin{aligned} T_{\text{rot}} &= \frac{1}{2} \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \cdot [I_{\text{hub},B_c}] \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \frac{1}{2} m_{\text{hub}} \dot{\boldsymbol{r}}_{B_c/B} \cdot \dot{\boldsymbol{r}}_{B_c/B} \\ &+ \sum_{i=1}^{N_{\text{eff}}} \left[\frac{1}{2} \boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} \cdot [I_{\text{eff},E_{c,i}}] \boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} + \frac{1}{2} m_{\text{eff}} \dot{\boldsymbol{r}}_{E_{c,i}/B} \cdot \dot{\boldsymbol{r}}_{E_{c,i}/B} \right] - \frac{1}{2} m_{\text{sc}} \dot{\boldsymbol{c}} \cdot \dot{\boldsymbol{c}} \end{aligned} \tag{25}$$

This form is advantageous because each effector does not need to know the location of the center of mass of the spacecraft, but rather they define their contributions with respect to point B. From a software architecture standpoint, this form will prove to be desirable.

The total rotational potential energy is specific to each effector. For example, the spring joint potential energy for a hinged rigid body is shown in the following equation:

$$V_{\text{rot}} = \frac{1}{2} k_\theta \theta^2 \tag{26}$$

Each effector might not have a potential energy contribution; however, each effector will have the ability to add their contribution to the total potential energy. Because the total rotational energy of the system is conserved when there are no nonconservative internal or external forces or torques acting on the system, it is convenient to combine the kinetic and potential energies into one term E_{rot}. This is shown in the following equation:

$$E_{\text{rot}} = T_{\text{rot}} + V_{\text{rot}} \tag{27}$$

It should be noted that, if there are nonconservative forces and torques present in the system, the rate of change of the energy expression in Eqs. (22) and (27) can be found using the fundamental power equation [29]. The total orbital angular momentum of the spacecraft about point N is

$$\boldsymbol{H}_{\text{orb},N} = m_{\text{sc}}\boldsymbol{r}_{C/N} \times \dot{\boldsymbol{r}}_{C/N} \tag{28}$$

Expanding in terms of the state variables yields

$$\boldsymbol{H}_{\text{orb},N} = m_{\text{sc}}(\boldsymbol{r}_{B/N} + \boldsymbol{c}) \times (\dot{\boldsymbol{r}}_{B/N} + \dot{\boldsymbol{c}}) \tag{29}$$

The final form of this equation that does not have direct contribution from effectors outside of their contributions to $\boldsymbol{c}$ and $\dot{\boldsymbol{c}}$ is

$$\boldsymbol{H}_{\text{orb},N} = m_{\text{sc}}[\boldsymbol{r}_{B/N} \times \dot{\boldsymbol{r}}_{B/N} + \boldsymbol{r}_{B/N} \times \dot{\boldsymbol{c}} + \boldsymbol{c} \times \dot{\boldsymbol{r}}_{B/N} + \boldsymbol{c} \times \dot{\boldsymbol{c}}] \tag{30}$$

The total rotational angular momentum of the spacecraft about point C is

$$\boldsymbol{H}_{\text{rot},C} = [I_{\text{hub},B_c}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + m_{\text{hub}}\boldsymbol{r}_{B_c/C} \times \dot{\boldsymbol{r}}_{B_c/C} + \sum_{i=1}^{N_{\text{eff}}}[[I_{\text{eff},E_{c,i}}]\boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} + m_{\text{eff}}\boldsymbol{r}_{E_{c,i}/C} \times \dot{\boldsymbol{r}}_{E_{c,i}/C}] \tag{31}$$

Expanding these terms yields

$$\begin{aligned}\boldsymbol{H}_{\text{rot},C} &= [I_{\text{hub},B_c}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + m_{\text{hub}}(\boldsymbol{r}_{B_c/B} - \boldsymbol{c}) \times (\dot{\boldsymbol{r}}_{B_c/B} - \dot{\boldsymbol{c}})\\ &+ \sum_{i=1}^{N_{\text{eff}}}[[I_{\text{eff},E_{c,i}}]\boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} + m_{\text{eff}}(\boldsymbol{r}_{E_{c,i}/B} - \boldsymbol{c}) \times (\dot{\boldsymbol{r}}_{E_{c,i}/B} - \dot{\boldsymbol{c}})]\end{aligned} \tag{32}$$

Distributing this result and simplifying yields the final equation:

$$\begin{aligned}\boldsymbol{H}_{\text{rot},C} &= [I_{\text{hub},B_c}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + m_{\text{hub}}\boldsymbol{r}_{B_c/B} \times \dot{\boldsymbol{r}}_{B_c/B}\\ &+ \sum_{i=1}^{N_{\text{eff}}}[[I_{\text{eff},E_{c,i}}]\boldsymbol{\omega}_{\mathcal{E}_i/\mathcal{N}} + m_{\text{eff}}\boldsymbol{r}_{E_{c,i}/B} \times \dot{\boldsymbol{r}}_{E_{c,i}/B}] - m_{\text{sc}}\boldsymbol{c} \times \dot{\boldsymbol{c}}\end{aligned} \tag{33}$$

Again, this form is desirable because the effectors define their contributions with respect to the body-fixed point B as opposed to the commonly used center-of-mass location point C, which is varying and depends on the other effectors. This will be leveraged in the modular form of the software architecture.

The results seen in Eqs. (22), (27), (30), and (33) are the modularized equations for energy and momentum in which the effectors provide contributions to orbital terms through $\boldsymbol{c}$ and $\dot{\boldsymbol{c}}$ and to rotational terms through direct contributions. This form is vital to retain the modularity of the system, and it validates the software implementation of the dynamics.

III. Example Spacecraft Equations of Motion

To include a meaningful example to apply to this methodology, a first-order approximation to a flexing appended body by using hinged rigid bodies is chosen. A diagram depicting this phenomenon is shown in Fig. 3. The EOMs developed for this system were discussed in detail in Ref. [31]. The model is approximating flexing by attaching a rigid body to the rigid-body hub through a torsional spring with spring constant k_i and damping terms c_i. Other frames and variables are defined in Fig. 3, and a further description can be seen in Ref. [31].

The equations in this section are matrix equations as opposed to vector equations to leverage some linear algebra techniques. This means that the matrix quantities need to be expressed with respect to a common reference frame. The typical frame chosen is the body frame $\mathcal{B}$. The translational EOM for the hub is shown in Eq. (34) [31]:

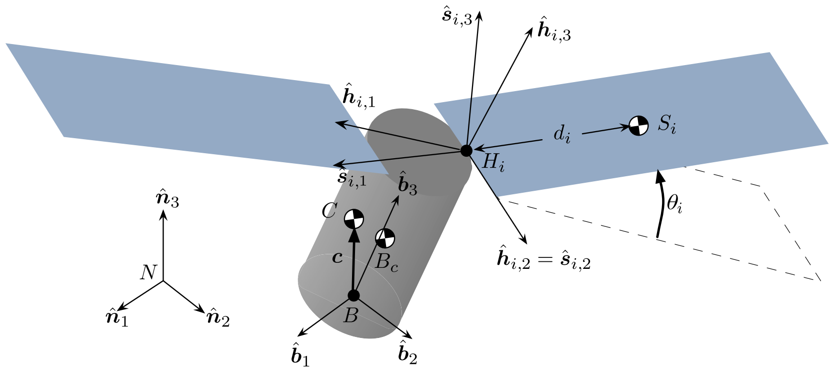


Fig. 3 Hinged rigid-bodies frame and variable definitions.

$$m_{\text{sc}}\ddot{\boldsymbol{r}}_{B/N} - m_{\text{sc}}[\tilde{\boldsymbol{c}}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_s} m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}\ddot{\theta}_i = \boldsymbol{F}_{\text{ext}} - 2m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c}'$$
$$- m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c} - \sum_{i=1}^{N_s} m_{\text{sp}_i}\text{d}_i\dot{\theta}_i^2\hat{\boldsymbol{s}}_{i,1} \tag{34}$$

This form is in agreement with the general form for the hub translational EOM introduced in Eq. (1). Similarly, the hub rotational EOM is, in this case,

$$m_{\text{sc}}[\tilde{\boldsymbol{c}}]\ddot{\boldsymbol{r}}_{B/N} + [I_{\text{sc},B}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + \sum_{i}^{N_s}\{I_{s_i,2}\hat{\boldsymbol{h}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_i/B}]\hat{\boldsymbol{s}}_{i,3}\}\ddot{\theta}_i$$
$$= \boldsymbol{L}_B - [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - [I'_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - \sum_{i}^{N_s}\{\dot{\theta}_i[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}](I_{s_i,2}\hat{\boldsymbol{h}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_i/B}]\hat{\boldsymbol{s}}_{i,3})$$
$$+ m_{\text{sp}_i}\text{d}_i\dot{\theta}_i^2[\tilde{\boldsymbol{r}}_{S_i/B}]\hat{\boldsymbol{s}}_{i,1}\} \tag{35}$$

Lastly, the hinged rigid-body single-degree-of-freedom differential equation is

$$m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T\ddot{\boldsymbol{r}}_{B/N} + [(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)\hat{\boldsymbol{s}}_{i,2}^T - m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T[\tilde{\boldsymbol{r}}_{H_i/B}]]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}$$
$$+ (I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)\ddot{\theta}_i = -k_i\theta_i - c_i\dot{\theta}_i + \hat{\boldsymbol{s}}_{i,2}^T\boldsymbol{\tau}_{\text{ext},H_i} + (I_{s_{i,3}} - I_{s_{i,1}} + m_{\text{sp}_i}\text{d}_i^2)\omega_{s_{i,3}}\omega_{s_{i,1}}$$
$$- m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{r}_{H_i/B} \tag{36}$$

and follows the same form that is introduced in Eq. (4). Equations (34–36) provide the $6 + N_s$ EOMs required to describe the motion of the spacecraft with flexing appended bodies.

Next, the equations of motion need to be placed in the backsubstitution form. The following backsubstitution method is repeated from Ref. [31] for convenience. First, Eq. (36) is solved for the angular accelerations $\ddot{\theta}_i$:

$$\ddot{\theta}_i = \frac{1}{(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)}(-m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T\ddot{\boldsymbol{r}}_{B/N} - [(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)\hat{\boldsymbol{s}}_{i,2}^T - m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T[\tilde{\boldsymbol{r}}_{H_i/B}]]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}$$
$$- k_i\theta_i - c_i\dot{\theta}_i + \hat{\boldsymbol{s}}_{i,2}^T\boldsymbol{\tau}_{\text{ext},H_i} + (I_{s_{i,3}} - I_{s_{i,1}} + m_{\text{sp}_i}\text{d}_i^2)\omega_{s_{i,3}}\omega_{s_{i,1}}$$
$$- m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{r}_{H_i/B}) \tag{37}$$

Equation (37) is rewritten into the following compact form to match Eq. (4):

$$\ddot{\theta}_i = \boldsymbol{a}_{\theta_i}^T\ddot{\boldsymbol{r}}_{B/N} + \boldsymbol{b}_{\theta_i}^T\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + c_{\theta_i} \tag{38}$$

where the terms $\boldsymbol{a}_{\theta_i}$, $\boldsymbol{b}_{\theta_i}$, and c_{θ_i} are defined as

$$\boldsymbol{a}_{\theta_i} = -\frac{m_{\text{sp}_i}\text{d}_i}{(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)}\hat{\boldsymbol{s}}_{i,3} \tag{39a}$$

$$\boldsymbol{b}_{\theta_i} = -\frac{1}{(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)}[(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{H_i/B}]\hat{\boldsymbol{s}}_{i,3}] \tag{39b}$$

$$c_{\theta_i} = \frac{1}{(I_{s_{i,2}} + m_{\text{sp}_i}\text{d}_i^2)}(-k_i\theta_i - c_i\dot{\theta}_i + \hat{\boldsymbol{s}}_{i,2}\cdot\boldsymbol{\tau}_{\text{ext},H_i} + (I_{s_{i,3}} - I_{s_{i,1}} + m_{\text{sp}_i}\text{d}_i^2)\omega_{s_{i,3}}\omega_{s_{i,1}}$$
$$- m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}^T[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{r}_{H_i/B}) \tag{39c}$$

Following the derivation seen in the backsubstitution method, Eq. (38) is substituted into the translational and rotational EOMs. The result of this for the translation EOM is shown in the following equation:

$$\left(m_{\text{sc}}[I_{3\times3}] + \sum_{i=1}^{N} m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}\boldsymbol{a}_{\theta_i}^T\right)\ddot{\boldsymbol{r}}_{B/N} + \left(-m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N} m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}\boldsymbol{b}_{\theta_i}^T\right)\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}$$
$$= m_{\text{sc}}\ddot{\boldsymbol{r}}_{C/N} - 2m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c}' - m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c} - \sum_{i=1}^{N}(m_{\text{sp}_i}\text{d}_i\dot{\theta}_i^2\hat{\boldsymbol{s}}_{i,1} + m_{\text{sp}_i}\text{d}_i c_{\theta_i}\hat{\boldsymbol{s}}_{i,3}) \tag{40}$$

Following the same pattern for the rotational hub EOM yields the following:

$$\begin{aligned}&\left[m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N}(I_{s_{i,2}}\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,3})\boldsymbol{a}_{\theta_i}^T\right]\ddot{\boldsymbol{r}}_{B/N} \\ &+ \left[[I_{\text{sc},B}] + \sum_{i=1}^{N}(I_{s_{i,2}}\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,3})\boldsymbol{b}_{\theta_i}^T\right]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} = -[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - [I'_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \\ &- \sum_{i=1}^{N}\{(\dot{\theta}_i[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}] + c_{\theta_i}[I_{3\times3}])(I_{s_{i,2}}\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,3}) + m_{\text{sp}_i}\text{d}_i\dot{\theta}_i^2[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,1}\} + \boldsymbol{L}_B \qquad (41)\end{aligned}$$

The following matrices are defined to match Eqs. (9–14):

$$[A] = m_{\text{sc}}[I_{3\times3}] + \sum_{i=1}^{N} m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}\boldsymbol{a}_{\theta_i}^T \qquad (42)$$

$$[B] = -m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N} m_{\text{sp}_i}\text{d}_i\hat{\boldsymbol{s}}_{i,3}\boldsymbol{b}_{\theta_i}^T \qquad (43)$$

$$[C] = m_{\text{sc}}[\tilde{\boldsymbol{c}}] + \sum_{i=1}^{N}(I_{s_{i,2}}\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,3})\boldsymbol{a}_{\theta_i}^T \qquad (44)$$

$$[D] = [I_{\text{sc},B}] + \sum_{i=1}^{N}(I_{s_{i,2}}\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,3})\boldsymbol{b}_{\theta_i}^T \qquad (45)$$

$$\begin{aligned}\boldsymbol{v}_{\text{trans}} &= m_{\text{sc}}\ddot{\boldsymbol{r}}_{C/N} - 2m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c}' - m_{\text{sc}}[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{c} \\ &- \sum_{i=1}^{N}(m_{\text{sp}_i}\text{d}_i\dot{\theta}_i^2\hat{\boldsymbol{s}}_{i,1} + m_{\text{sp}_i}\text{d}_i c_{\theta_i}\hat{\boldsymbol{s}}_{i,3}) \qquad (46)\end{aligned}$$

$$\begin{aligned}\boldsymbol{v}_{\text{rot}} &= -\sum_{i=1}^{N}\{(\dot{\theta}_i[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}] + c_{\theta_i}[I_{3\times3}])(I_{s_{i,2}}\hat{\boldsymbol{s}}_{i,2} + m_{\text{sp}_i}\text{d}_i[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,3}) \\ &+ m_{\text{sp}_i}\text{d}_i\dot{\theta}_i^2[\tilde{\boldsymbol{r}}_{S_{c,i}/B}]\hat{\boldsymbol{s}}_{i,1}\} - [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - [I'_{\text{sc},B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \boldsymbol{L}_B \qquad (47)\end{aligned}$$

Equations (42–47) are the final equations needed for implementation into software and will be referenced in the following section when discussing the modular software architecture.

IV. Modular Software Architecture

Figure 4 shows the Unified Modeling Language (UML)-class diagram for the object-oriented computer programming languages proposed in this paper. This is the design that allows complex fully coupled dynamics to be implemented in software while retaining a modular architecture. Additionally, it aims to solve the issues of testability, maintainability, and scalability that fully coupled dynamics problems pose.

The `dynamicObject` seen in Fig. 4 is a parent class or abstract class that defines the base functionality of the object that will control the calculation of the system EOMs and essentially solve for the well-known state derivative vector $\dot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X}, t)$. However, the term state vector is used loosely here because the `stateManager` organizes, stores, and controls all states of the system. The `dynamicObject` is an abstract or parent class because this will allow for different types of systems to be implemented in the future that are not necessarily using the proposed backsubstitution method in this paper. Therefore, the `spacecraftPlus` is an instantiation of the `dynamicObject` and is the class that is implementing the backsubstitution method.

In the generalized EOMs introduced earlier in this paper, the term "effectors" is used to define objects that are attached to the spacecraft and have dynamic states that need to be integrated. Some examples are: reaction wheels, flexing solar arrays, variable speed control moment gyroscopes (VSCMGs), fuel slosh, etc. In this modular software architecture, those effectors are called `stateEffectors` and are illustrated in Fig. 4. In contrast, `dynamicEffectors` are phenomena that result in an external forces or torques being applied to the spacecraft. Examples of these include: gravity, thrusters, solar radiation pressure (SRP), etc.

The `stateEffector` abstract or parent class is the class that defines the necessary methods (and variables) needed for each effector to provide contributions to the spacecraft's mass properties (m_{sc}, $[I_{\text{sc},B}]$, $\boldsymbol{c}$, etc.) using the method `updateEffectorMassProperties` and contributions to the backsubstitution matrices ($[A]$, $[B]\dots\boldsymbol{v}_{\text{Trans}}$, etc.) using the method `updateContributions`. Each effector needs to be able to compute their own state derivatives using the method `computeDerivatives`. Finally, the method `computeEnerMomContributions` is the method that enables effectors to add their contributions to the energy and momentum of the system for verification purposes. Additionally, it should be noted that, in Fig. 4, it shows that both `stateEffectors` and `dynamicEffectors` are aggregated in `spacecraftPlus`. This allows for the modularity of the dynamics because `spacecraftPlus` does not know the type of effectors attached to it, but rather has an array of `stateEffectors` or `dynamicEffectors` that makes it general.

Another important aspect of the software architecture is the `hubEffector` instantiation of `stateEffector`. The `hubEffector` is representing the rigid-body hub defined in the generalized EOM form and has translational and attitude states associated with it. The hubEffector

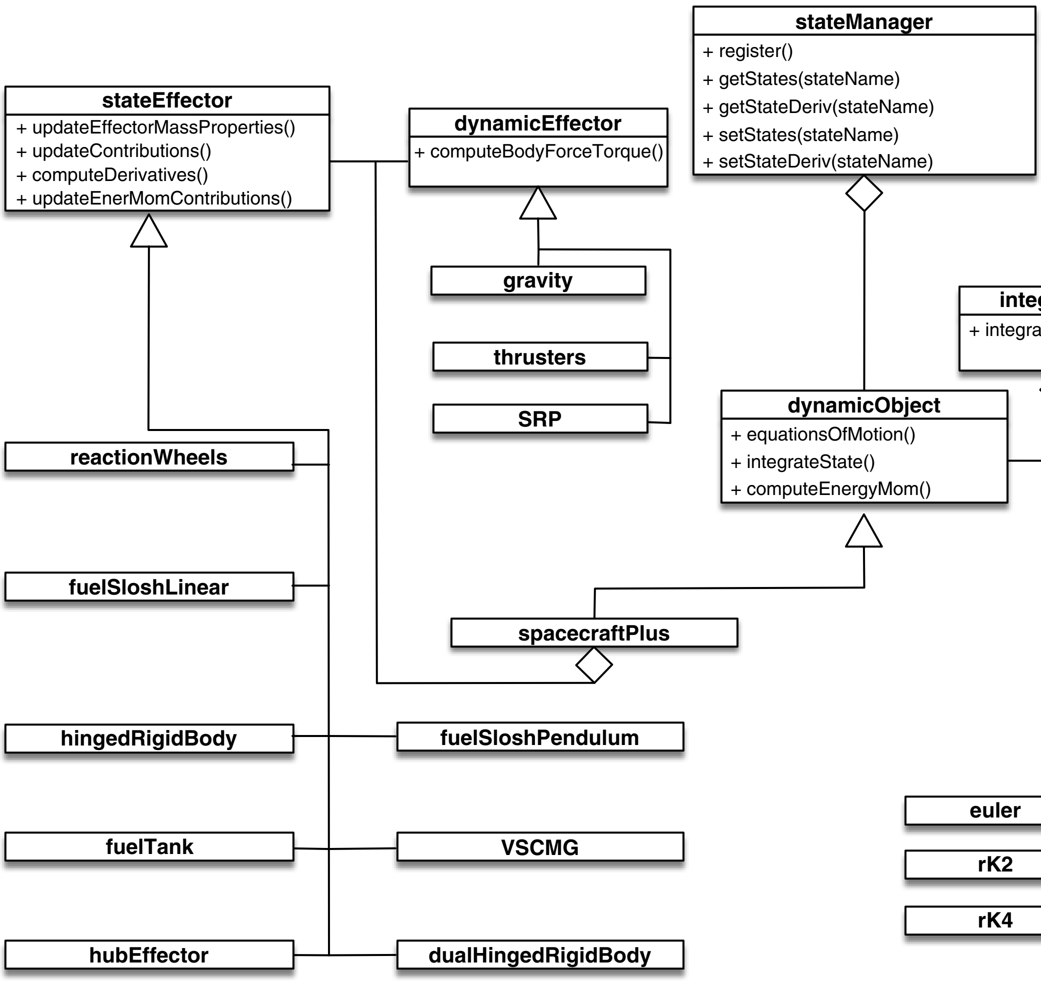


Fig. 4 UML diagram for modular architecture.

is unique to all of the `stateEffectors` because it is not included in the array of `stateEffectors` that are looped over in `spacecraftPlus` but are rather defined as an object in `spacecraftPlus`, and its methods are always called in `equationsOfMotion()`. This is because the assumption for the backsubstitution method, and the generalized EOM form is that the spacecraft will always have a rigid-body hub with a body frame $\mathcal{B}$ attached and with the corresponding states $\boldsymbol{r}_{B/N}$, $\dot{\boldsymbol{r}}_{B/N}$, $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$, and $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$.

Because `spacecraftPlus` is an instantiation of `dynamicObject`, it inherits the methods that are defined in Fig. 4. The method `equationsOfMotion()` is the method that solves for all of the state derivatives of the spacecraft system. To explain this method in more detail, Fig. 5 is included to show the flow in pseudocode. The spacecraft mass properties need to be calculated first because, in Eqs. (1) and (2), the total spacecraft mass m_{sc}, inertia $[I_{\text{sc},B}]$, and other parameters are needed. Next, the `gravityEffector` class is called to compute the gravity acting on the spacecraft. This is done at this location because some `stateEffectors` might need to know the gravitational acceleration. Following this step, the `stateEffectors` are looped over to find their contributions to the backsubstitution matrices and the `dynamicEffectors` are looped over to get their contributions to $\boldsymbol{F}_{\text{ext}}$ and $\boldsymbol{L}_B$. Now, all of the necessary values have been computed for the hub state derivatives to be calculated using Eq. (15), which is computed in the `hubEffector`'s `computeDerivatives`. Finally, the `stateEffectors` are looped over to compute their derivatives using $\ddot{\boldsymbol{r}}_{B/N}$ and $\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}$.

Because the `hubEffector`'s derivative calculation is so vital in this structure, Fig. 6 is shown to explain the calculations needed for this step. Again, this is shown using pseudocode. Additionally, this method shows the interaction between the `stateManager` and the rest of the system. The `stateManager` stores the states of the system in individual objects. These objects can be accessed using a string and, once the object has been accessed, the methods seen in Fig. 4 under the `stateManager` class are available. For example, the `getState` method delivers the current value of the state stored in that state object. In Fig. 6, the hub effector uses those methods to retrieve the desired information from the `stateManager`. Ultimately, setting the derivative values for the `hubEffector` is the goal of the `computeDerivatives` method and does so by using `setStateDeriv` for both $\ddot{\boldsymbol{r}}_{B/N}$ and $\dot{\boldsymbol{\omega}}_{BN}$.

Another important method in this architecture is the `computeDerivatives` method for a generic `stateEffector`. To highlight this method, the hinged rigid bodies example introduced in this paper is used. Figure 7 shows the pseudocode for the computeDerivatives method of a hinged rigid-body effector. When this method is being computed, $\ddot{\boldsymbol{r}}_{B/N}$ and $\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}$ have already been calculated; therefore, the hinged rigid-body effector can use the state manager's method called `getStateDeriv`, which gives access to those precomputed values. Looking at Eq. (38), the hinged rigid-body effector uses $\ddot{\boldsymbol{r}}_{B/N}$ and $\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}$ in its calculation, it uses used saved variables for faster results, and it is a benefit of the backsubstitution method.

The power of this design is that `stateEffectors` can just be attached to the spacecraft in no particular order and the scalability of this design is unconstrained. Adding another effector does not depend on any other effectors, even though the fully coupled nature is still retained. All of the coupling is through the rigid-body hub, and the analytical form of the backsubstitution method allows for this modularity. Additionally, a fixed-size system mass matrix is inverted as opposed to a dynamically allocated matrix of varying size, which is common in fully coupled dynamics simulations.

```
spacecraftPlus

equationsOfMotion()

    hubEffector.updateEffectorMassProperties()
    for(effector in stateEffectors)
        effector.updateEffectorMassProperties()
    end

    gravityEffector.computeGravField()

    for(effector in stateEffectors)
        effector.updateContributions()
    end

    for(effector in dynEffectors)
        effector.computeBodyForceTorque()
    end

    hubEffector.computeDerivatives()
    for(effector in stateEffectors)
        effector.computeDerivatives()
    end

end
```

Fig. 5 Pseudo code for the `equationsOfMotion()` method within `spacecraftPlus`.

```
hubEffector

computeDerivatives()

    rBN_NState = stateManager.getStateObject('hubPosition')
    rBNDot_NState = stateManager.getStateObject('hubVelocity')
    sigmaBN_State = stateManager.getStateObject('hubRotPosition')
    omegaBN_BState = stateManager.getStateObject('hubRotVelocity')

    rBNDot_N = rBNDot_NState.getState()
    rBN_NState.setStateDeriv(rBNDot_N)

    sigmaBNDot = omegaToSigmaDot(omegaBN_BState.getState())
    sigmaBN_State.setStateDeriv(sigmaBNDot)

        $\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} = \Big([D] - [C][A]^{-1}[B]\Big)^{-1}(\boldsymbol{v}_2 - [C][A]^{-1}\boldsymbol{v}_1)$

                $\ddot{\boldsymbol{r}}_{B/N} = [A]^{-1}(\boldsymbol{v}_1 - [B]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}})$

    omegaBN_BState.setStateDeriv(omegaBN_Dot)
    rBNDot_NState.setStateDeriv(rBNDDot_N)

end
```

Fig. 6 : Pseudocode for `hubEffector` `computeDerivatives()` method.

V. Modular Software Architecture Implementation and Verification

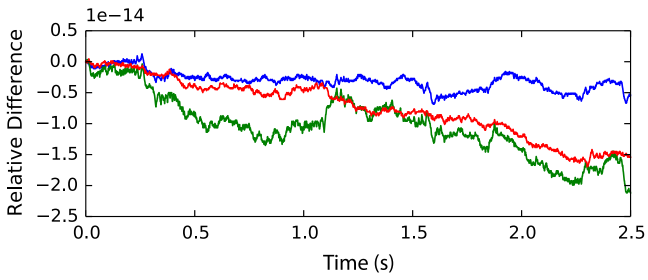
The Basilisk astrodynamics software package is chosen as the implementation code base for the modular dynamics architecture. Basilisk has modularity as a key feature to the software package, and so retaining this functionality for the dynamics is an important influence. Because the backsubstitution method attempts to solve the issue of scalability, maintainability, and testability, effectors can be added to this software package with ease and do not affect the rest of the code base. For example, a software developer can add an effector without changing the `spacecraftPlus` implementation and only needs to add code for the current effector while adhering to rules of the software architecture.

The method in verifying that a specific effector has been implemented correctly following the specific guidelines for effectors and that the effector is in agreement with physics is that the four energy/momentum values of orbital energy, orbital angular momentum, rotational energy, and rotational angular momentum must be conserved when applicable. The derivation for these quantities for effectors are shown in Eqs. (22) and (33). Figures 8a–8d are examples of the verification results for the hinged rigid-body effector and are the desired result: the orbital angular momentum, orbital energy, rotational angular momentum, and rotational energy are conserved down to machine precision. Integrated tests can be used to confirm conservation of these quantities to validate the different models. This gives verification in not only the hinged rigid-body model but also the backsubstitution method and the modular software architecture.

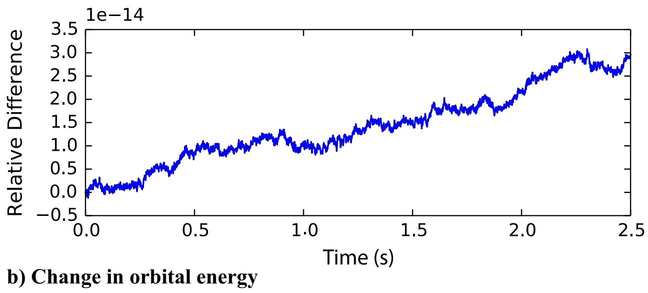
hingedRigidBody

```
computeDerivatives()

    theta_State = stateManager.getStateObject('panelTheta')
    thetaDot_State = stateManager.getStateObject('panelThetaDot')
    hubVelocity_State = stateManager.getStateObject('hubVelocity')
    hubRotVelocity_State = stateManager.getStateObject('hubRotVelocity')

    thetaDot = thetaDot_State.getState()
    theta_State.setStateDeriv(thetaDot)

    rBNDDot_N = hubVelocity_State.getStateDeriv()
    omegaDotBN_B = hubRotVelocity_State.getStateDeriv()

        $\ddot{\theta}_i = \boldsymbol{a}_{\theta_i}^T \ddot{\boldsymbol{r}}_{B/N} + \boldsymbol{b}_{\theta_i}^T \dot{\boldsymbol{\omega}}_{\mathcal{B}/N} + c_{\theta_i}$

    thetaDot_State.setStateDeriv(thetaDDot)

end
```

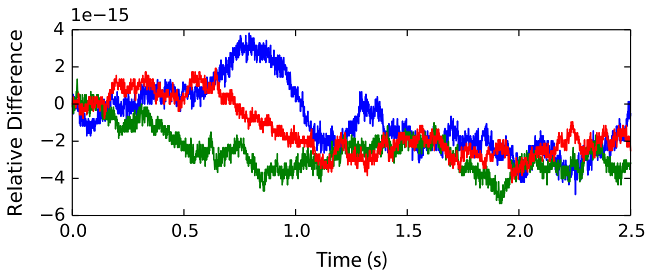
Fig. 7 Pseudocode for `hingedRigidBody` `computeDerivatives()` method.



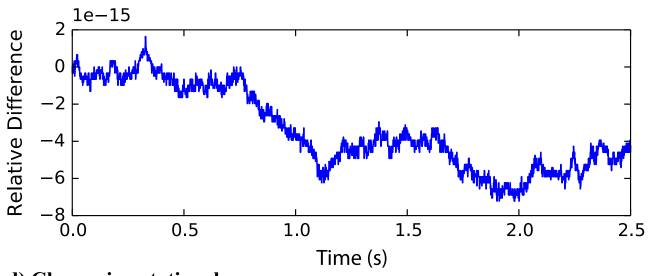
a) Change in orbital angular momentum



b) Change in orbital energy



c) Change in rotational angular momentum



d) Change in rotational energy

Fig. 8 Simulation verification results for a rigid hub with a single hinged panel.

VI. Conclusions

This paper introduces a modular software architecture for fully coupled dynamics and solves the issues of maintainability, scalability, and testability for this problem. The elegant modularity is achieved by considering a specific spacecraft dynamical system in which a range of dynamical subsystems is attached to a central rigid hub. The proposed software architecture is shown to be maintainable, allows for a fixed-size system mass matrix to be inverted, allows effectors to be attached to the spacecraft in no particular order, and does not have scaling limitations. The modular software architecture is verified using energy and momentum conservation, and it is implemented in the Basilisk astrodynamics software package.

References

[1] "IEEE Standard Glossary of Software Engineering Terminology," IEEE STD 610.12-1990, Piscataway, NJ, 1990, pp. 1–84. doi:10.1109/IEEESTD.1990.101064

[2] Filman, R., Elrad, T., Clarke, S., and Akşit, M., *Aspect-Oriented Software Development*, 1st ed., Addison-Wesley Professional, Reading, MA, 2004.

[3] Junkins, J., and Kim, Y., *Introduction to Dynamics and Control of Flexible Structures*, AIAA Education Series, AIAA, New York, 1993. doi:10.2514/4.862076

[4] "Bullet: Real-Time Physics Simulation (online database)," Oct. 2017, http://bulletphysics.org/wordpress/ [retrieved Oct. 2017].

[5] Jeffreys, H., and Jeffreys, B., *Methods of Mathematical Physics*, 3rd ed., Cambridge Univ. Press, New York, 1999, pp. 304–306. doi:10.1002/9783527617210

[6] "PROJECTCHRONO: An Open Source Multi-Physics Simulation Engine," Oct. 2017, http://projectchrono.org [retrieved Oct. 2017].

[7] "RBDL: Rigid Body Dynamics Library (online database)," Oct. 2017, https://rbdl.bitbucket.io/ [retrieved Oct. 2017].

[8] "Moby—Multi-Body Dynamics Simulation Library (online database)," Sept. 2017, http://physsim.sourceforge.net/index.html [retrieved Oct. 2017].

[9] "Multibody Dynamics Module (online database)," COMSOL, Burlington, MA, Sept. 2017, https://www.comsol.com/multibody-dynamics-module [retrieved Oct. 2017].

[10] "Adams: The Multibody Dynamics Simulation Solution (online database)," MSC Software, Newport Beach, CA, Aug. 2018, http://www.mscsoftware.com/product/adams [retrieved Oct. 2017].

[11] "Simscape Multibody: Model and Simulate Multibody Mechanical Systems (online database)," MathWorks, Natick, MA, Oct. 2017, https://www.mathworks.com/products/simmechanics.html [retrieved Oct. 2017].

[12] Kane, T. R., and Levinson, D. A., "Formulation of Equations of Motion for Complex Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 3, No. 2, 1980, pp. 99–112. doi:10.2514/3.55956

[13] Kane, T. R., and Levinson, D. A., "Multibody Dynamics," *Journal of Applied Mechanics*, Vol. 50, No. 4b, 1983, pp. 1071–1078. doi:10.1115/1.3167189

[14] "Motion Genesis: Symbolic Force, Motion, and Code-Generation Tools (online database)," Oct. 2017, www.MotionGenesis.com [retrieved Oct. 2017].

[15] "Position Based Dynamics Library (online database)," Interactive Computer Graphics, Aachen, Germany, Aug. 2017, http://www.interactive-graphics.de/index.php/downloads/92-positionbaseddynamics-library [retrieved Oct. 2017].

[16] Bender, J., Müller, M., and Macklin, M., "Position-Based Simulation Methods in Computer Graphics," *EUROGRAPHICS 2017 Tutorials*, Eurographics Assoc., 2017. doi:10.2312/egt.20171034

[17] Deul, C., Charrier, P., and Bender, J., "Position-Based Rigid Body Dynamics," *Computer Animation and Virtual Worlds*, Vol. 27, No. 2, 2014, pp. 103–112. doi:10.1002/cav.1614

[18] Bender, J., "Impulse-Based Dynamic Simulation in Linear Time," *Computer Animation and Virtual Worlds*, Vol. 18, Nos. 4–5, 2007, pp. 225–233. doi:10.1002/cav.v18:4/5

[19] "IBDS: Physics Library(online database)," Interactive Computer Graphics, Aachen, Germany, Aug. 2017, http://www.interactive-graphics.de/index.php/downloads/12-ibds [retrieved Oct. 2017].

[20] "STK SOLIS: Commercial Plug-In to the Analytical Graphics, Inc (AGI) Systems ToolKit (STK) (online database)," Advanced Solutions, Littleton, CO, April 2017, http://www.go-asi.com/solutions/stk-solis/ [retrieved Oct. 2017].

[21] "DARTS: Dynamics Algorithms for Real-Time Simulation (online database)," NASA, June 2017, https://dartslab.jpl.nasa.gov/DARTS/index.php [retrieved Oct. 2017].

[22] Rodriguez, G., and Jain, A., "Spatial Operator Algebra for Multibody System Dynamics," *Journal of the Astronautical Sciences*, Vol. 40, Jan.–March 1992, pp. 27–50. doi:10.1177/027836499101000406

[23] Jain, A., and Rodriguez, G., "Recursive Flexible Multibody System Dynamics Using Spatial Operators," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 6, 1992, pp. 1453–1466. doi:10.2514/3.11409

[24] "42: A Comprehensive General-Purpose Simulation of Attitude and Trajectory Dynamics and Control of Multiple Spacecraft Composed of Multiple Rigid or Flexible Bodies (online database)," NASA, Sept. 2017, https://software.nasa.gov/software/GSC-16720-1 [retrieved Oct. 2017].

[25] Likins, P. W., "Point-Connected Rigid Bodies in a Topological Tree," *Celestial Mechanics*, Vol. 11, No. 3, 1975, pp. 301–317. doi:10.1007/BF01228809

[26] "OreKit: An Accurate and Efficient Core Layer for Space Flight Dynamics Applications (online database)," Oct. 2017, https://www.orekit.org/ [retrieved Oct. 2017].

[27] Allard, C., Diaz-Ramos, M., and Schaub, H., "Spacecraft Dynamics Integrating Hinged Solar Panels and Lumped-Mass Fuel Slosh Model," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA, Reston, VA, 2016.

[28] Kane, T. R., Likins, P. W., and Levinson, D. A., *Spacecraft Dynamics*, McGraw–Hill Book, New York, 1983.

[29] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 3rd ed., AIAA Education Series, AIAA, Reston, VA, 2014. doi:10.2514/4.102400

[30] Sidi, M., *Spacecraft Dynamics and Control: A Practical Engineering Approach*, Cambridge Aerospace Series, Cambridge Univ. Press, New York, 1997. doi:10.1017/CBO9780511815652

[31] Allard, C., Schaub, H., and Piggott, S., "General Hinged Rigid-Body Dynamics Approximating First-Order Spacecraft Solar Panel Flexing," *Journal of Spacecraft and Rockets*, Vol. 55, No. 5, 2018, pp. 1291–1299. doi:10.2514/1.A34125

[32] Cappuccio, P., Allard, C., and Schaub, H., "Fully-Coupled Spherical Modular Pendulum Model To Simulate Spacecraft Propellant Slosh," *AAS/AIAA Astrodynamics Specialist Conference*," AAS Paper 18-224, Springfield, VA, 2018.

[33] Alcorn, J., Allard, C., and Schaub, H., "Fully Coupled Reaction Wheel Static and Dynamic Imbalance for Spacecraft Jitter Modeling," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 6, 2018, pp. 1380–1388. doi:10.2514/1.G003277

[34] Panicucci, P., Allard, C., and Schaub, H., “Spacecraft Dynamics Employing a General Multi-Tank and Multi-Thruster Mass Depletion Formulation,” *AAS Guidance, Navigation and Control Conference*, AAS Paper 17-011, Springfield, VA, 2017.
[35] Alcorn, J., Allard, C., and Schaub, H., “Fully-Coupled Dynamical Jitter Modeling of Variable-Speed Control Moment Gyroscopes,” *AAS/AIAA Astrodynamics Specialist Conference*, AAS Paper 17-730, Springfield, VA, 2017.
[36] Alcorn, J., “Fully-Coupled Dynamical Jitter Modeling of Momentum Exchange Devices,” M.S. Thesis, Univ. of Colorado Boulder, Univelt, Inc., San Diego, CA, 2017.

J. P. How
Associate Editor