A Basilisk-Based Benchmark Analysis Of Different Constrained Attitude Dynamics Planners

Riccardo Calaon*
University of Colorado, Boulder, CO, 80303, United States

Michael A. Trowbridge†
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, United States

Hanspeter Schaub‡
University of Colorado, Boulder, CO, 80303, United States

Maneuvering a spacecraft subject to positional constraints is a nontrivial problem. Several attitude guidance solutions are found in literature that use different approaches to provide guidance algorithms to safely reorient a spacecraft while ensuring that all constraints and requirements are satisfied. However, such algorithms are often problem-specific, and/or not immediately comparable with one another as they have slightly different objectives. The purpose of this paper is to show how a range of constrained attitude guidance solutions can be compared using the Basilisk Astrodynamics Simulation Framework. The solution allows users to easily set up a customizable simulation environment that can be used as a benchmarking tool to test the performances of different guidance algorithms. Performance metrics are defined and implemented within a Basilisk module to provide a numerical comparison between the different guidance strategies being tested.

I. Introduction

Highly constrained spacecraft attitude planning often represents a challenge in space missions. The positional constraints that spacecraft are subject to are often modeled as conical regions of the spacecraft-centered inertial frame that the body-fixed boresight axis should either keep in or out during the maneuver. Keep-out constraints consist in the need of maneuvering the spacecraft while avoiding orientations that could cause damage to some sensitive payload, and potentially result in mission failure. For example, in the presence of a telescope for scientific data acquisition or a star tracker for attitude determination, it is vital to keep these instruments at a sufficiently large angular distance from any bright celestial object such as Sun, Moon, or Earth's albedo. Another class of constraints is defined as keep-in constraints, which consists in the need of maneuvering while keeping a certain celestial object within the field of view of a body-mounted instrument. One example is maintaining the Sun in the field of view of at least one of the spacecraft's sun sensors to ensure continuous attitude determination capabilities over time.

Constrained attitude maneuvering continues to be investigated as the problem remains challenging and open. Solutions exist in literature that use Lyapunov potential functions to drive the spacecraft to the desired final attitude, combined with potential barrier function that steer the spacecraft away from the obstacle boundary [1–3]. Such solutions are often computationally easy to implement, but are often applicable to a limited set of constraints. Other, more recent approaches apply path planning algorithms to compute a constraint-compliant reference trajectory in attitude parameter space, which is to be tracked by the spacecraft [4–6]. Such algorithms are generally more versatile with respect to the type and quantity of constraints that they can deal with, but usually rely on the discretizaion of the attitude workspace and provide reference trajectories only in terms of sequences of attitude waypoints, rather than elaborating a full kinodynamic plan that includes the spacecraft rates, accelerations and inertia properties. Space mission operators such as JPL have great interest in constraint-compliant path planning, and therefore are developing novel algorithms [7, 8].

*Graduate Student, Ann and H.J. Smead Aerospace Engineering Sciences, Colorado Center for Astrodynamics Research, 3775 Discovery Drive, 429 UCB - CCAR, Boulder, CO 80303. AIAA Student Member

†Member of Technical Staff, Artificial Intelligence, Observation Planning And Analysis, M/S 301-260, 4800 Oak Grove Drive, Pasadena, CA 91109.

‡Professor, Glenn L. Murphy Chair in Engineering, Ann and H.J. Smead Aerospace Engineering Sciences, Colorado Center for Astrodynamics Research, 3775 Discovery Drive, 429 UCB - CCAR, Boulder, CO 80303. AIAA and AAS Fellow.

Many other algorithms exist that can be qualitatively compared in literature, but not in how they perform with respect to the same sets of constraints. Moreover, some of these algorithms were derived for mission-specific requirements, therefore might not perform as well in different scenarios. For the above reasons, the absence of direct comparison studies makes it hard to choose the best algorithm for a desired application. The purpose of this work is to demonstrate the feasibility of using the modular Basilisk Astrodynamics Simulation Framework * discussed in Ref. [9] as a benchmarking tool to test the performances of different algorithms against a standardized scenario, which can easily be modified by the user according to their needs.

This work simulates the a sample cube satellite using Bevo-2 satellite parameters with its solar orientation constraints, described in Reference 4. This paper starts with a description of Basilisk, laying out key modules used in the attitude simulation (Section II). Next, it steps through the spacecraft mass and inertia properties, as well as the specific attitude constraints and their mathematical formulation (Section III). The performance metrics used in Basilisk are then identified (Section IV), followed by the four different path planners those metrics are used to evaluate (Section V). Finally, conclusions about the path planners based on the results from the comparison metrics are offered (Section VI).

II. Basilisk

Basilisk is an open-source software framework that can simulate complex spacecraft systems in the space environment. The dual nature of Basilisk consists in its C/C++ core software modules, which ensure speed of execution, combined with a Python interface, to allow for easy scriptability and reconfigurability. Basilisk's main strength relies in its modular structure, which allows for minimal coupling between different segments of code that simulate different spacecraft behaviors. The minimal coupling between modules is enabled by Basilisk's messaging system: each module reads in input messages from other modules and outputs its own message(s), thus decoupling the data flow between modules and removing explicit intermodule dependency [9].

In the following notation, left superscripts indicate the frames with respect to which vectors are expresses, whereas right subscripts describes the two frames between which the vector properties are expressed [10]. The modules used in the following sections are briefly described. This discussion on the Basilisk capabilities is not meant to be comprehensive, but illustrative of how complex benchmark simulation scenarios can be setup to provide comparative analysis of constrained attitude guidance solutions.

- **`Simulation Task`**: a task in Basilisk is a grouping of modules, which are updated with a fixed integration rate. Multiple tasks can be used within a simulation, especially when different integration steps are required to capture the behavior of certain high-frequency phenomena. Tasks can be switched on and off according to the necessity [9]. For this work, only a single task with a constant integration rate is used for every simulation.
- **`spacecraft()`**: this module contains information such as spacecraft mass and inertia, and outputs a message containing information about the inertial position of the spacecraft and its center of mass, together with attitude, angular rates and accelerations of the body-fixed frame with respect to the inertial frame.
- **`gravityEffector()`**: this module is used to create gravity bodies such as Earth and Sun. Earth is used as the primary center of gravity around which the spacecraft is orbiting. The Sun is generated to play the role of the bright celestial body about which the constraints are defined.
- **`simpleNav()`**: this module adds error on top of the message that it receives from the `spacecraft()` module. The motivation for this module is to provide a realistic navigation signal, in order to test the guidance and control modules in presence of signal errors. Such error is modelled as a Gauss-Markov process.
- **`RW()`**: this module creates a list of reaction wheels (RWs). These can be generated from a database of existing wheels, or they can be customized by the user. When generating a RW, the user must specify the body-frame direction of the spin axis of the wheel ${}^{\mathcal{B}}\hat{g}_s$. Optional parameters can be provided such as initial wheel speed, maximum speed and/or maximum momentum.
- **`inertial3D()`**: this module is used to set a fixed inertial attitude that the spacecraft must converge to. This module provides a message containing the Modified Rodrigues Parameter (MRP) attitude[10] of the reference frame $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}}$ with respect to the inertial frame, together with zeroed reference angular rates and accelerations ${}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}} = {}^{\mathcal{R}}\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} = [0,0,0]^T$.
- **`waypointReference()`**: this new module reads the reference trajectory of a reorientation maneuver from a text file. This allows this Basilisk-based benchmarking tool to be used with constrained attitude guidance solutions created outside the Basilisk environment. Such reference trajectory should be provided as a ordered list of time-tagged attitude waypoints, together with the associated angular rates and accelerations. The module outputs a

*`https://hanspeterschaub.info/basilisk`

reference message based on the reference trajectory that is to be tracked. The attitude can be provided in terms of Modified Rodrigues Parameters (MRPs) $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}}$ or quaternions $\boldsymbol{\beta}_{\mathcal{R}/\mathcal{N}}$. The angular rates and accelerations can be specified either in the reference frame $\mathcal{R}$ or in the inertial frame $\mathcal{N}$. This module reads the attitude waypoints and compares the given time-tags with the current simulation time: when the current simulation time falls between the time-tags of two attitude waypoints, the reference message computed is obtained via linear interpolation of attitude, angular rates and angular accelerations between the two waypoints right before and after the current simulation time. It should be noted that, when the attitude is specified using quaternions, the module always converts it to MRP set: interpolation in MRP space always return sets that represent valid attitudes, thus avoiding the unity constraint required by quaternion representation.

- **`attTrackingError()`**: computes and outputs the relative attitude $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ of the spacecraft with respect to the reference, the relative angular rates ${}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{R}}$ and accelerations ${}^{\mathcal{B}}\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{R}}$ in body-frame components.
- **`mrpFeedback()`**: this module computes the required control torque on the spacecraft according to a MRP-based Lyapunov feedback control law. It receives the messages containing the relative attitude between spacecraft and reference frames, mass and inertia properties of the spacecraft, and RW states, and computes a commanded torque in body-frame components.
- **`rwMotorTorque()`**: this module maps the required torque into individual reaction wheel motor torques, according to the RW configuration and availability of the wheels.
- **`boreAngCalc()`**: this module is used to compute the angular distance between a certain user-defined body-fixed direction ${}^{\mathcal{B}}\hat{b}$ and a celestial object, in this case the Sun.
- **`reactionWheelPower()`**: this module computes the power required to spin the reaction wheel(s) at the desired angular rate Ω.
- **`pathScorer()`**: this module receives information from the `boreAngCalc()` modules, `attTrackingError()` module and `reactionWheelPower()` and computes the performance metrics that are used to compare the different path planners.

The following analysis will feature the `inertial3D()` module for a constraint-naive approach, and the `waypointReference()` module to simulate constraint-aware maneuvers obtained using path planning algorithms computed outside of Basilisk. A module for performing attitude maneuvers while enforcing keep-out and keep-in constraints is currently being developed after the work presented in Reference 11, and it will be implemented in Basilisk soon. The modular, open source nature of Basilisk allows the reader to implement their own attitude guidance algorithm into a new module and connect it to the simulation in place of `inertial3D()` or `waypointReference()`.

A scheme of the modular structure of the Basilisk simulation is represented in Figure 1.

III. Spacecraft model

The spacecraft and its constraints are modeled after the Bevo-2 satellite as described in [4]. A sensitive star tracker with a field of view of 20 degrees is aligned with the ${}^{\mathcal{B}}\boldsymbol{b}_x = [1,0,0]^T$ direction, while two sun sensors with a field of view of 70 degrees each are aligned with the ${}^{\mathcal{B}}\boldsymbol{b}_y = [0,1,0]^T$ and ${}^{\mathcal{B}}\boldsymbol{b}_z = [0,0,1]^T$ directions. While performing the maneuver, the star tracker must avoid pointing at the Sun. On the other hand, at least one of the sun sensors must always be able to see the Sun at all times, therefore the keep-in constraint is violated when the Sun is outside of the field of view of both sensors simultaneously. In each of the following simulations, the spacecraft is maneuvering between two at-rest, constraint-compliant configurations.

The mass and inertia properties of the spacecraft are modeled according to those of a 3-unit cubesat with a uniform mass distribution [12]:

$$m = 4.0 \text{ kg} \qquad {}^{\mathcal{B}}[I] = \begin{bmatrix} 6.67 & 0 & 0 \\ 0 & 41.87 & 0 \\ 0 & 0 & 41.87 \end{bmatrix} \cdot 10^{-3} \text{ kg} \cdot \text{m}^2. \tag{1}$$

The actuation is provided by a set of three reaction wheels aligned with the principal inertia axes ${}^{\mathcal{B}}\boldsymbol{b}_x$, ${}^{\mathcal{B}}\boldsymbol{b}_y$ and ${}^{\mathcal{B}}\boldsymbol{b}_z$. The reaction wheels can provide a control torque up to 1 mNm each. The mass of the reaction wheels is accounted for in the total spacecraft mass m and inertia tensor ${}^{\mathcal{B}}[I]$. The wheels are assumed to be perfectly balanced, and with the center of mass perfectly aligned along the principal inertia axes.

The keep-out and keep-in constraints are modelled as hard constraints, and the following equations must be satisfied

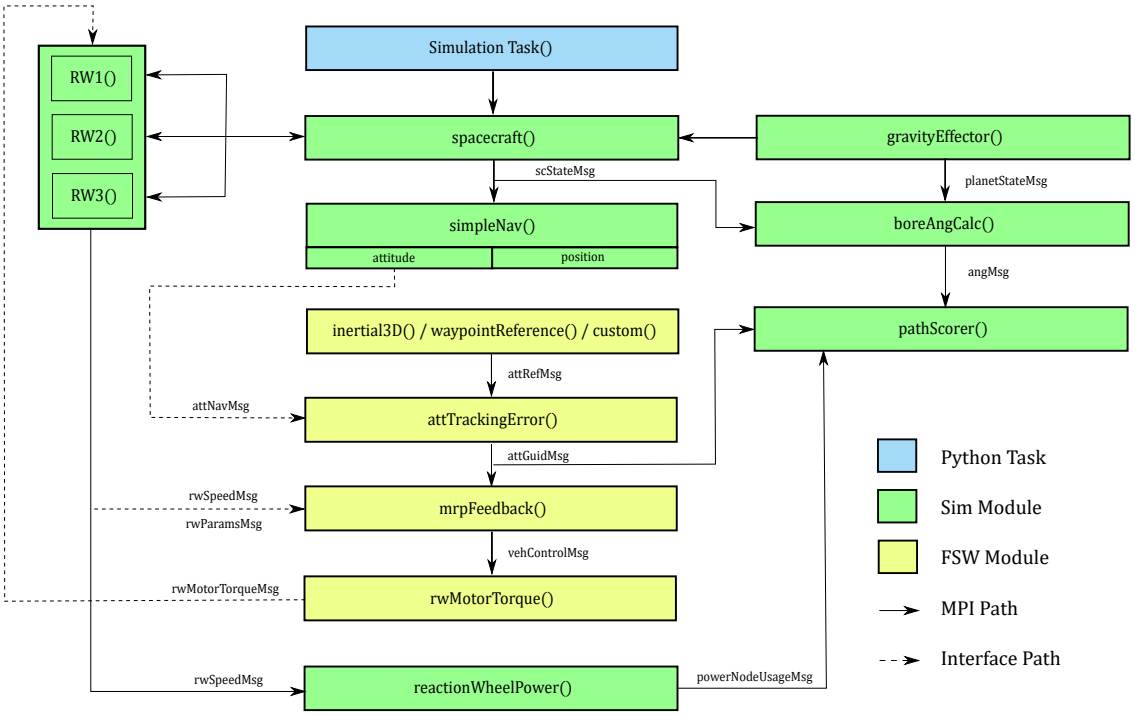


Fig. 1 Basilisk modular structure for the current simulation

at all times:

$$ {}^{\mathcal{B}}\hat{\boldsymbol{b}}_x \cdot [\mathcal{BN}]^{\mathcal{N}}\hat{\boldsymbol{s}} < \cos(20^\circ) \tag{2} $$

$$ {}^{\mathcal{B}}\hat{\boldsymbol{b}}_y \cdot [\mathcal{BN}]^{\mathcal{N}}\hat{\boldsymbol{s}} \geq \cos(70^\circ) \;\; || \;\; {}^{\mathcal{B}}\hat{\boldsymbol{b}}_z \cdot [\mathcal{BN}]^{\mathcal{N}}\hat{\boldsymbol{s}} \geq \cos(70^\circ) \tag{3} $$

where $[\mathcal{BN}]$ is the direction cosine matrix that maps vectors from the inertial frame $[\mathcal{N}]$ to the body frame $[\mathcal{B}]$ and ${}^{\mathcal{N}}\hat{\boldsymbol{s}}$ is the inertial direction of the Sun.

IV. Performance metrics

Five performance metrics are defined to test the suitability of the difference planners.

- **Total keep-out violation time**:

$$ T_{\text{KO}} = \int_0^T \delta_{\text{KO}} \mathrm{d}t \qquad \delta_{KO} = \begin{cases} 1 & \text{if} \quad {}^{\mathcal{N}}\boldsymbol{b}_x \cdot {}^{\mathcal{N}}\boldsymbol{s} \geq \cos(20^\circ) \\ 0 & \text{otherwise} \end{cases} \tag{4} $$

- **Total keep-in violation time**:

$$ T_{\text{KI}} = \int_0^T \delta_{\text{KI}} \mathrm{d}t \qquad \delta_{KI} = \begin{cases} 1 & \text{if} \quad {}^{\mathcal{N}}\boldsymbol{b}_y \cdot {}^{\mathcal{N}}\boldsymbol{s} < \cos(70^\circ) \quad \text{and} \quad {}^{\mathcal{N}}\boldsymbol{b}_z \cdot {}^{\mathcal{N}}\boldsymbol{s} < \cos(70^\circ) \\ 0 & \text{otherwise} \end{cases} \tag{5} $$

- **Attitude error integral**:
  Defining $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ as the attitude error between the body frame $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ and the target reference frame $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}}$, this metric computes the integral of the principal rotation angle error $\theta_{\mathcal{B}/\mathcal{R}}$ over the maneuver time, and is meant to provide an estimate of how accurately the reference can be tracked. Large tracking errors can lead to constraint violations even when the reference trajectory is constraint compliant.

$$\Theta_{\mathcal{B}/\mathcal{R}} = \int_0^T 4 \arctan\left(||\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}||\right) \mathrm{d}t = \int_0^T \theta_{\mathcal{B}/\mathcal{R}} \mathrm{d}t \tag{6}$$

- **Commanded torque integral**:
  The commanded torque $\boldsymbol{L}_r$ is the torque provided to the spacecraft to track the reference trajectory. Its expression is derived according to a nonlinear control law based on the relative MRP attitude $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ and angular rates $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{R}}$ [10]:

$$\boldsymbol{L}_r = -K\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}} - P\boldsymbol{\omega}_{\mathcal{B}/\mathcal{R}} + {}^{\mathcal{B}}[I]\left(\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} - [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}}\right) + [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}]\left({}^{\mathcal{B}}[I]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + [G_s]\boldsymbol{h}_s\right) \tag{7}$$

  where K and P are Proportional-Derivative-like control gains, $[G_s]$ is the $3 \times n$ matrix containing the body-frame directions of the spinning axes of the wheels (it coincides with the identity matrix for the current reaction wheel configuration), and $\boldsymbol{h}_s$ is the vector containing the angular momenta of each reaction wheel about its spinning axis. The commanded torque integral is defined as

$$U = \int_0^T ||\boldsymbol{L}_r|| \mathrm{d}t \tag{8}$$

- **Total energy consumption**:
  The total energy consumption is the integral over maneuver time of the power requirements of all the reaction wheels combined. For each reaction wheel, the power required is obtained as the product between the torque applied to the wheel u_s and the wheel speed relative to the spacecraft Ω. The total required energy is:

$$E = \int_0^T \left(\sum_{i=1}^{3} u_{s_i}\Omega_i\right) \mathrm{d}t \tag{9}$$

V. Path Planners

This section describes the four path planners that are compared in the following sections.

- **Planner #0**:
  Planner #0 is, effectively, a constraint-naive planner. The slew maneuver is performed implementing a nonlinear feedback control law that drives the spacecraft from an initial attitude $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_i}$ to a final attitude $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_f}$ achieving a final rest state. Since this planner is entirely based on the desired final attitude, constraint avoidance is not enforced. This planner is presented as an example of how constraints can easily be violated if not accounted for when performing a slew maneuver.

- **Planner #1**:
  Planner #1 is based on a sequence of constraint-compliant reference attitude points $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_j}$ with $j = 0, ..., N$, with zero associated angular rates and accelerations $\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}_j} = \dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}_j} = 0$, where $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_0} = \boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_i}$ and $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_N} = \boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_f}$. The sequence of constraint-compliant waypoints is obtained searching a uniform 3D grid in MRP space as described in [11]. The path obtained for this planner is searched using the simpler implementation of the A* algorithm described in [11], where the cost function is given by the total cumulative Cartesian distance between waypoints in MRP space. The information given to the `MRPFeedback()` module is, therefore, a list of time-tagged attitude waypoints, that the spacecraft must try to target one after another, as simulation time

increases. The time-spacing between waypoints is proportional to the mutual distance between waypoints in MRP space.

- **Planner #2**:
  Planner #2 improves on the results of Planner #1. The same sequence of waypoints is used as by Planner #1. However, instead of feeding a list of time-tagged waypoints to the `MRPFeedback()` module, the list of attitude waypoints is interpolated in MRP space to obtain a smooth trajectory as a twice-differentiable C^2 function of time. Angular rates and accelerations are computed to ensure rest states (zero angular rates) at the endpoints and a constant angular rate norm of $||\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}}|| = 0.04$ rad/s along the trajectory. The angular rate norm is ramped up and down smoothly from the zero initial and final condition to the desired constant angular rate norm.

- **Planner #3**:
  Planner #3 coincides with the effort-minimizing A* algorithm described in [11]. A path is searched such that the interpolated trajectory is the optimal in terms of the integral of the required control torque U. The information passed to the `MRPFeedback()` module is, as for Planner #2, a smooth trajectory that has attitude, angular rates and accelerations as functions of time, again with zero angular rates at the endpoints and a constant angular rate norm of $||\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}}|| = 0.04$ rad/s along the trajectory.

VI. Benchmark analysis

A. Planner comparison

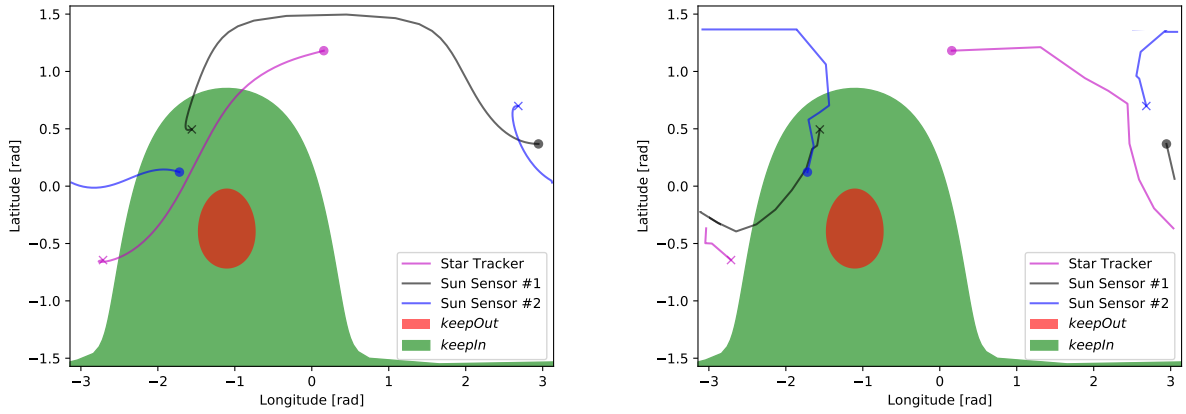
This section shows the performance of the different planners based on a set of common evaluation criteria and the performance metrics described above. The scenario presented here features a slew maneuver between the attitudes $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_i} = [0.522, -0.065, 0.539]^T$ and $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_f} = [0.342, 0.223, -0.432]^T$. The inertial position of the Sun is obtained from the SPICE database for the date January 15, 2021, at 00:30:30 UTC, which gives ${}^{\mathcal{N}}\boldsymbol{s} = [0.419, -0.833, -0.361]^T$. The spacecraft is assumed to be orbiting the Earth, in a position along its orbit where the Earth does not cause an eclipse.

All the simulations run in this subsection use the gains $K = 6 \cdot 10^{-3}$ N and $P = 1.256 \cdot 10^{-2}$ Ns in Equation (7) to compute the commanded torque to the spacecraft. Such gains are chosen primarily for Planner #0, to ensure a near-to-critical response of the system which would converge to the desired target in about one minute. Figure 2 shows the projection on the 2D latitude-longitude plane of the boresight directions in inertial space, with respect to the keep-out constraint (in red) and the keep-in constraint (in green). Figure 2 (a) shows the actual trajectory of the boresights when the Planner #0 is used, whereas Figure 2 (b), (c) and (d) show the boresight directions for the reference waypoints $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_j}$ provided to the `attTrackingError()` module. For Planner #0, this would correspond to just the initial and final reference attitudes. For Planner #1, the discrete reference waypoints produce a sequence of discrete target boresight inertial directions. Lastly, for Planners #2 and #3, full reference trajectories are obtained for the boresight directions as functions of time.

Figure 2 shows that the keep-out constraint is respected with all four planners. As far as the keep-in constraint, it is possible to see that Sensor #2 sees the Sun in the initial attitude, whereas Sensor #1 sees the Sun once the target attitude is reached. Figure 3 shows the angle between the Star Tracker and the Sun and the angles between the Sun Sensors and the Sun, together with the respective fields of view (f.o.v.) for each instrument. With planners #0, #1 and #2 the keep-in constraint is violated for a certain amount of time. This happens when the two Sun Sensors 'exchange' roles. Leaving aside Planner #0, which is constraint-naive, what happens for Planners #1 and #2 is more interesting. For Planner #1, a sequence of constraint-compliant waypoints is provided. However, the path that connects such waypoints is not constraint compliant in all its parts, and this is evident in the keep-in constraint violation. A similar phenomenon occurs with Planner #2, where the interpolated trajectory uses the same reference waypoints as Planner #1: although the interpolated trajectory violates the constraint for a shorter time, it still does, since the interpolating function used maintains the trajectory within the convex hull described by the interpolated attitude waypoints [13]. Planner #3, on the other hand, does not violate any of the constraints. This is not due to a refined sampling of the attitude space, but rather to the fact that the different nature of the cost function used by Planner #3 makes it converge to a trajectory that stays farther away from the boundary of the constraint-compliant space, thus avoiding the issue described for the previous two planners.

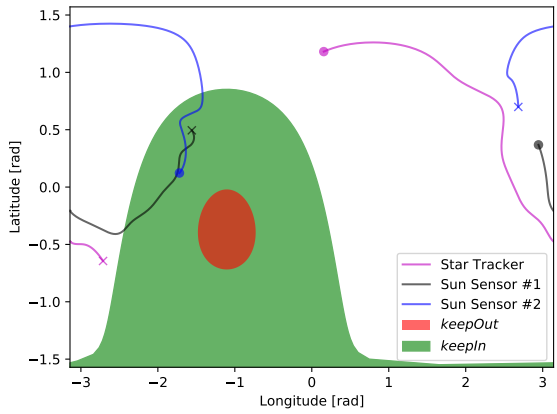
Figure 4 shows the performance metrics described above, and offers a direct comparison between the four planners. Subfigures (a) and (b) show the constraint violation times, where the same information can be observed as in Figure 2,

with more details on how long the constraint violations last for each planner. Subfigures (c), (d) and (e) offer more insights on the performance of the different planners other than just constraint compliance. Subfigure (c) shows that Planner #3 outperforms Planner #2 in terms of required commanded torque, as expected. Given the reaction wheel configuration, with one wheel along each principal body axis, this property transfers also to the subfigure (d), where Planner #3 is shown to outperform Planner #2 also in terms of energy consumption: this is due to the fact that each torque component is mapped directly to the corresponding reaction wheel. It is interesting to observe, however, how Planner #1 outperforms both Planners #2 and #3 in terms of total commanded torque and energy consumption. This was unexpected, since Planner #1 does not try to optimize for torque and/or power requirements. This unexpected behavior can be explained looking at subfigure (e), a gap of 3 orders of magnitude is observed between the attitude error integrals of Planners #0 and #1, and #2 and #3. As explained above, Planners #2 and #3 feed to the `attTrackingError()` module a time-dependent reference trajectory along with the required reference angular rates and accelerations: this allows the `mrpTracking()` module to accurately track the full desired state of the spacecraft along such reference trajectory. In contrast, Planner #1 only provides target attitude waypoints, therefore the `mrpTracking()` module tries to constantly steer the spacecraft towards the next attitude waypoint with zeroed final angular velocity. However, the target waypoint changes faster than the actuators can track, thus causing the spacecraft to be constantly chasing a moving target, until such target eventually settles at the final target attitude. This inefficient guidance strategy causes the attitude errors along the trajectory to be comparatively large, potentially resulting in constraint violations even when a

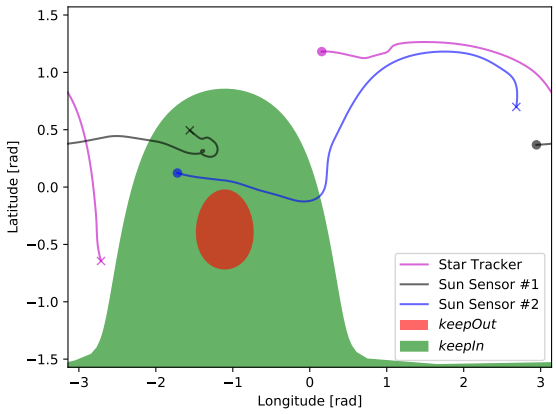


(a) Planner #0

(b) Planner #1

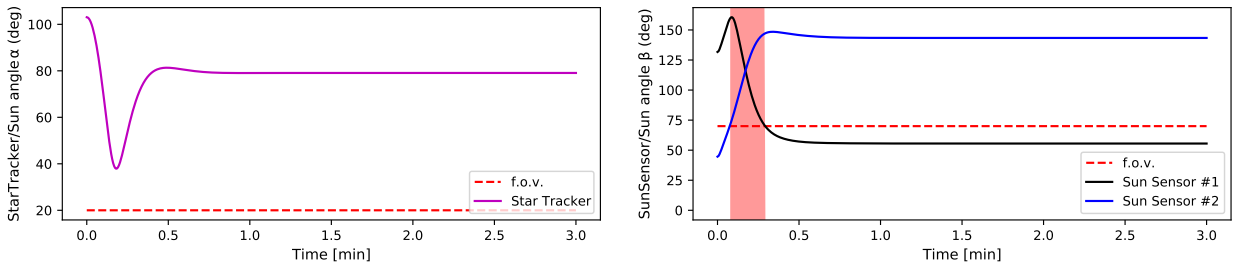


(c) Planner #2

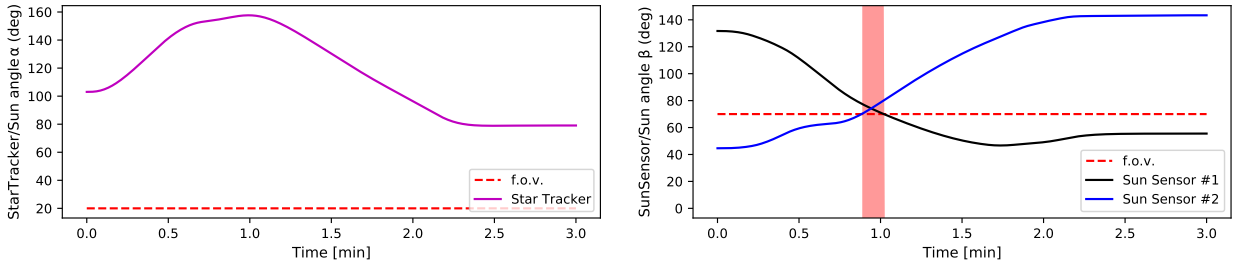


(d) Planner #3

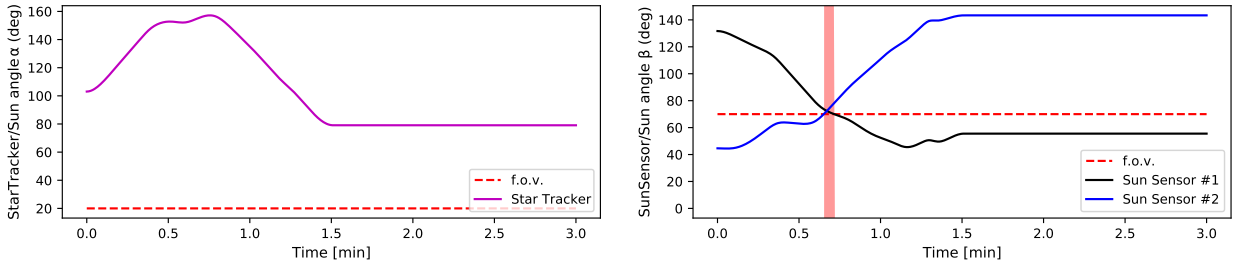
Fig. 2 2D plots of inertial boresight directions; 'o': starting point, 'x' endpoint.



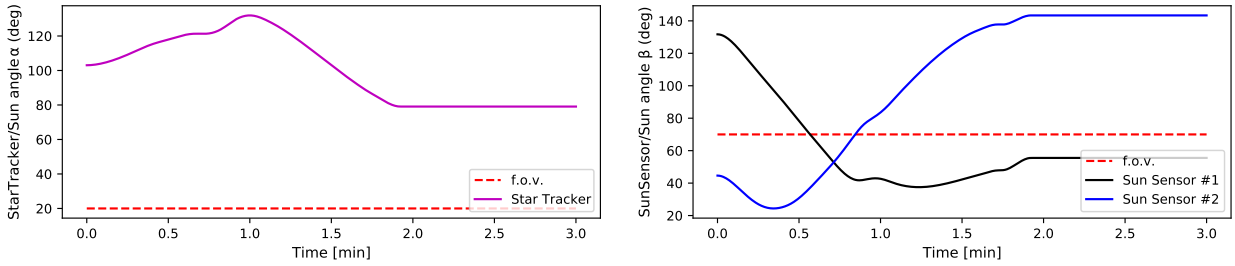
(a) Planner #0



(b) Planner #1



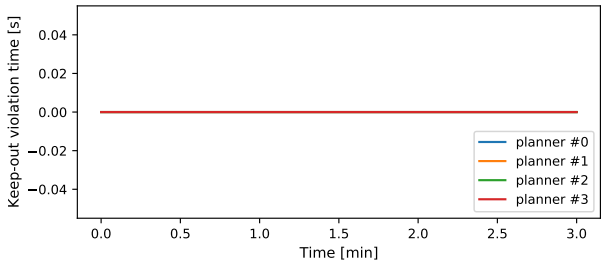
(c) Planner #2



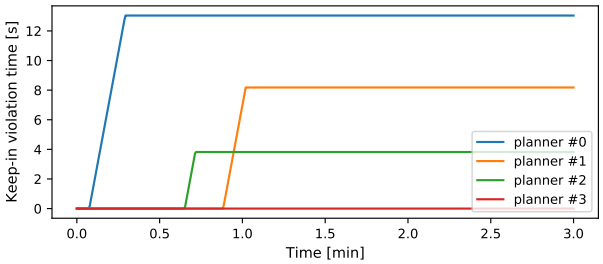
(d) Planner #3

Fig. 3 Angles between Star Tracker and Sun and between Sun Sensors and Sun

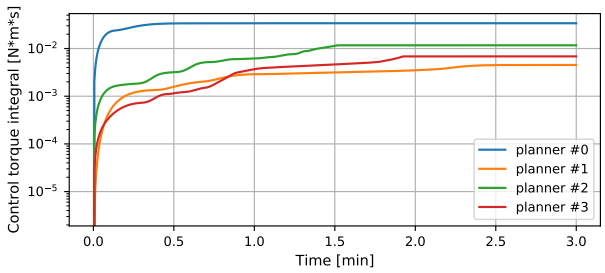
constraint-compliant sequence of waypoints is used. On the other hand, Planners #2 and #3 force the spacecraft to hit all the waypoints precisely: this can cause the interpolating spline to present wiggles where the required torque in unnecessarily large. In contrast, when Planner #1 is used, the simulation automatically smooths the path provided by the waypoints due to its looser tracking capabilities, allowing for a smoother torque profile that ultimately results in a smaller torque integral and energy consumption.



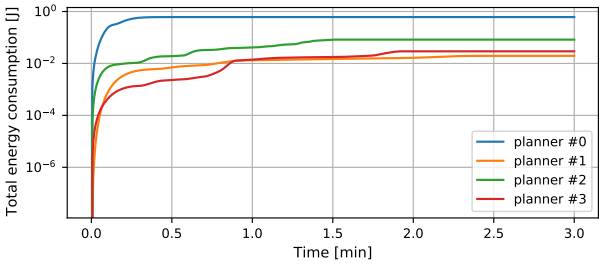
(a) Total keep-out violation time



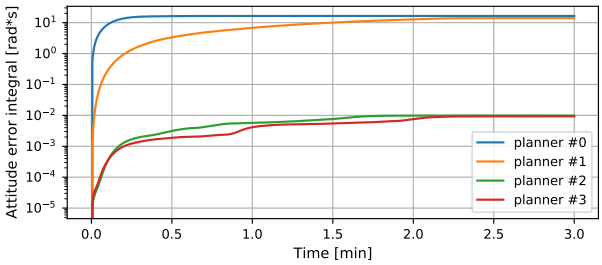
(b) Total keep-in violation time



(c) Commanded torque integral



(d) Total energy consumption



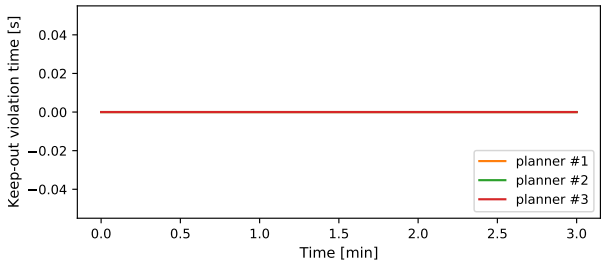
(e) Attitude error integral

Fig. 4 Performance metrics of the four planners compared

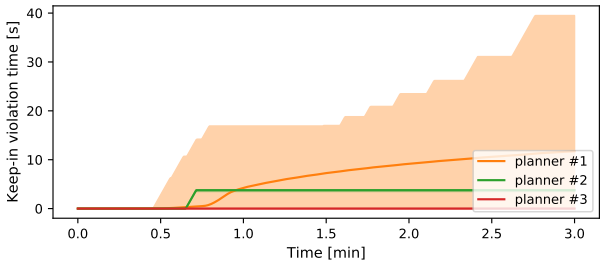
B. Gain sensitivity

Not only do the planners have different performances based on the metrics above, they also have different levels of sensitivity to the gains K and P used in the nonlinear control law in the `mrpFeedback()` module. In the previous subsection the gains were chosen according to the performance of Planner #0. In the following simulations both gains are varied according to an exponential distribution between $K \in [6 \cdot 10^{-4}, 6 \cdot 10^{-2}]$ and $P \in [1.256 \cdot 10^{-3}, 6 \cdot 10^{-1}]$. This is done to highlight the performance of the different planners across a range of gains that spans from one order of magnitude lower to one order of magnitude higher than the values previously tested. Planner #0 is excluded from this analysis due to the very high sensitivity to gains, which often lead to non-comparable results.

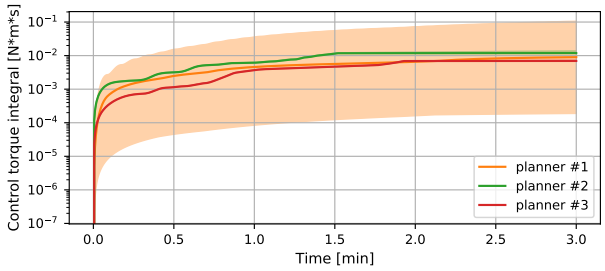
Figure 5 summarizes the results obtained with varying gains. Solid lines represents the averaged curves for each planner, whereas the colored shaded regions represent the bounds between best and worst case scenarios. First of all, subfigure (a) shows that the keep-out constraint is never violated, whereas subfigure (b) shows that Planner #1 can violate the keep-in constraint for very different intervals of time, and also not violate the constraint at all for the right choice of gains. More interestingly, Planners #2 and #3 show the same consistent behavior regardless of the gains: this is emphasized in subfigures (c) and (d) where the upper and lower confidence bounds also coincide with the averaged curve. The same cannot be said about Planner #1, which is much more susceptible to gain changes. This analysis shows the robustness of the interpolated reference trajectories used by Planners #2 and #3 to gain tuning: having a well-defined reference makes the open-loop system track such reference well enough, to the point that the feedback terms $K\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$



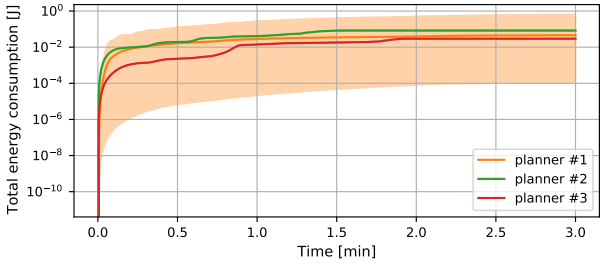
(a) Total keep-out violation time



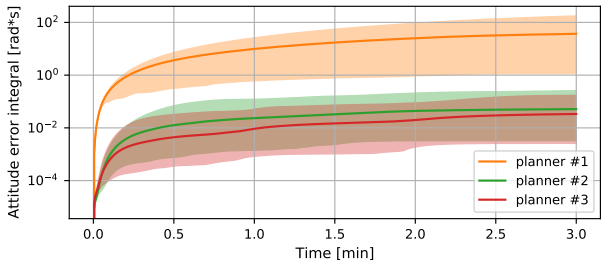
(b) Total keep-in violation time



(c) Commanded torque integral



(d) Total energy consumption



(e) Attitude error integral

Fig. 5 Sensitivity of Planners #1 - #3 to gain variations

and $P\boldsymbol{\omega}_{\mathcal{B}/\mathcal{R}}$ in Equation (7) become irrelevant in the computation of the commanded torque. Lastly, subfigure (e) shows the sensitivity of the integral tracking error to gains. In this case, Planners #2 and #3 do show appreciable variations due to gain selection. Nonetheless, within the gain bounds considered, the worst integral tracking error with Planners #2 and #3 is still one order of magnitude smaller than the best integral error with Planner #1.

VII. Conclusion

This paper shows how to set up a Basilisk simulation to test the performances of different path planning algorithms against a standardized scenario. The strength of Basilisk simulations rely on its scriptability and the ease with which the simulation setup can be modified to match a desired scenario. Basilisk's modular structure allows the user to write their own guidance module to perform constrained attitude maneuvering and incorporate it into Basilisk itself, or alternatively, the reference trajectory can be computed externally to Basilisk and imported from a data file using the `waypointReference()` module. Either way, Basilisk offers the possibility of testing very different approaches against a variety of metrics that provide an apples-to-apples comparison. The `pathScorer()` module that implements such metrics can also be modified and/or enhanced with new metrics according to the user's needs.

The simulations shown in this paper are successful at detecting constraint violations, and offer insights on the ease with which the spacecraft can track a given reference trajectory. Specifically, it was observed that it is not enough to provide a discrete sequence of attitude waypoints for an accurate tracking of a reference trajectory, since angular rates

and acceleration profiles are also required. On the other hand, if the requirement on precise tracking can be relaxed, it is possible to achieve a reorientation maneuver with a reduced control effort and power consumption.

These results become interesting when tested against a broad range of gains for the control law that computes the required torque on the spacecraft. The simulations show that a smooth reference trajectory makes the tracking problem robust to gain selection. On the contrary, a 'breadcrumbs' approach such as that used by Planner #1 is not just less effective at tracking the reference, but also very susceptible to gain design and ultimately less predictable and reliable.

All the above considerations stem from the availability of a standardized benchmarking tool such as Basilisk, which enables the user to make quantitative and qualitative comparisons between path planners that are, due to their very different nature, difficult to compare otherwise. Moreover, the ease of scriptability provided by Basilisk's Python interface, combined with the fast-running C/C++ core, enable the user to easily run multiple simulation with varying design parameters to test the different outcomes. In conclusion, Basilisk proves to be a reliable and accurate simulation environment to simulate spacecraft dynamics in the space environment and highlight nontrivial properties that are not evident a priori.

Acknowledgments

Parts of this research were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Mclnnes, C. R., "Large angle slew maneuvers with autonomous sun vector avoidance," *Journal of guidance, control, and dynamics*, Vol. 17, No. 4, 1994, pp. 875–877.

[2] Diaz Ramos, M., and Schaub, H., "Kinematic steering law for conically constrained torque-limited spacecraft attitude control," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 9, 2018, pp. 1990–2001.

[3] Lee, U., and Mesbahi, M., "Constrained autonomous precision landing via dual quaternions and model predictive control," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 292–308.

[4] Kjellberg, H. C., and Lightsey, E. G., "Discretized constrained attitude pathfinding and control for satellites," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 5, 2013, pp. 1301–1309.

[5] Kjellberg, H. C., and Lightsey, E. G., "Discretized quaternion constrained attitude pathfinding," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 3, 2015, pp. 713–718.

[6] Tanygin, S., "Fast autonomous three-axis constrained attitude pathfinding and visualization for boresight alignment," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 358–370.

[7] Singh, G., Macala, G., Wong, E., Rasmussen, R., Singh, G., Macala, G., Wong, E., and Rasmussen, R., "A constraint monitor algorithm for the Cassini spacecraft," *Guidance, Navigation, and Control Conference*, 1997, p. 3526.

[8] Kim, Y., Mesbahi, M., Singh, G., and Hadaegh, F., "On the constrained attitude control problem," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 5129.

[9] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: a flexible, scalable and modular astrodynamics simulation framework," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 496–507.

[10] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 4$^{\text{th}}$ ed., AIAA, 2018.

[11] Calaon, R., and Schaub, H., "Constrained Attitude Maneuvering Via Modified Rodrigues Parameters - Based Motion Planning Algorithms," *AAS/AIAA Astrodynamics Specialist Conference*, Big Sky, MT, 2021. Paper AAS 21-527.

[12] Mehrparvar, A., Pignatelli, D., Carnahan, J., Munakata, R., Lan, W., Toorian, A., Hutputanasin, A., and Lee, S., "CubeSat design specification (CDS) REV 13," *The CubeSat Project, San Luis Obispo, CA*, 2014, pp. 1–42.

[13] Piegl, L., and Tiller, W., *The NURBS book*, Springer Science & Business Media, 1996.