Journal of Aerospace Information Systems
Vol. 20, No. 2, February 2023

Basilisk-Based Benchmark Analysis of Different Constrained Attitude Dynamics Planners

Riccardo Calaon* and Hanspeter Schaub†
University of Colorado, Boulder, Colorado 80303
and
Michael A. Trowbridge‡
California Institute of Technology, Pasadena, California 91109

https://doi.org/10.2514/1.I011109

Maneuvering a spacecraft subject to rotational constraints is a nontrivial challenge. Several attitude guidance solutions are found in literature that use a range of 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 open-source Basilisk Astrodynamics Simulation Framework. New Basilisk modules are introduced to allow 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. The simulation allows for guidance solutions to be implemented within Basilisk or read in from a file generated by an outside attitude path planner. Sample performance metrics are defined and implemented within a Basilisk module to illustrate this approach performing a numerical comparison between the different guidance strategies.

Nomenclature

$\mathcal{B}$	=	spacecraft body-fixed frame
E	=	reaction wheel power consumption over the maneuver time, J
$\boldsymbol{L}_r$	=	commanded torque to the spacecraft, N
$\mathcal{N}$	=	inertial frame
$\mathcal{R}$	=	reference frame to be tracked by the spacecraft
T_{KI}	=	total teep-in violation time, s
T_{KO}	=	total teep-out violation time, s
U	=	integral of the control torque norm over the maneuver time, N·s
$\boldsymbol{\beta}_{\mathcal{R}/\mathcal{N}}$	=	quaternion set describing the orientation of the frame $\mathcal{R}$ with respect to the frame $\mathcal{N}$
$\Theta_{\mathcal{B}/\mathcal{R}}$	=	integral of the principal rotation angle error between body frame $\mathcal{B}$ and reference frame $\mathcal{R}$
$\theta_{\mathcal{R}/\mathcal{N}}$	=	quaternion set describing the orientation of the frame $\mathcal{R}$ with respect to the frame $\mathcal{N}$, rad
$\theta_{\mathcal{R}/\mathcal{B}}$	=	principal rotation angle between the frame $\mathcal{R}$ and the frame $\mathcal{B}$
$\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}}$	=	modified Rodrigues parameter set describing the orientation of the frame $\mathcal{R}$ with respect to the frame $\mathcal{N}$
${}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}}$	=	angular velocity of the frame $\mathcal{R}$ with respect to the frame $\mathcal{N}$, expressed in the $\mathcal{R}$ frame, rad/s
${}^{\mathcal{R}}\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}}$	=	angular acceleration of the frame $\mathcal{R}$ with respect to the frame $\mathcal{N}$, expressed in the $\mathcal{R}$ frame, rad/s^2

Subscript

$\mathcal{R}/\mathcal{N}$	=	differential quantity between frame $\mathcal{R}$ and frame $\mathcal{N}$

Superscripts

$\mathcal{N}$	=	frame with respect to which the vector is expressed
T	=	transpose of a vector or matrix

I. Introduction

HIGHLY constrained spacecraft attitude planning often represents a challenge in space missions. The rotational 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 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 typically 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 discretization of the attitude

Presented as Paper 2022-2277 at the 2022 AIAA SciTech Forum, San Diego, CA, January 3–7, 2022; received 8 February 2022; revision received 18 July 2022; accepted for publication 21 September 2022; published online 28 October 2022. Copyright © 2022 by The Authors. 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 eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

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

†Professor and Department Chair, Schaden Leadership Chair, Ann and H. J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. Fellow AIAA.

‡Member of Technical Staff, Artificial Intelligence, Observation Planning and Analysis, Jet Propulsion Laboratory, M/S 301-260, 4800 Oak Grove Drive.

workspace and provide reference trajectories only in terms of sequences of attitude waypoints, rather than elaborating a full kinematic plan that includes the spacecraft rates and accelerations. Space mission operators such as NASA, ESA, and industry operators have great interest in constraint-compliant path planning, and therefore are developing novel algorithms [7,8].

Many other algorithms exist that can be qualitatively compared in literature, but it is often challenging or impossible to do a direct comparison with respect to the same sets of constraints. Moreover, some of these algorithms were derived for mission-specific requirements and, 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 an open-source simulation tool such as 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.

Other open-source software tools exist that can simulate either a full space mission or specific subsegments, such as orbital motion and attitude dynamics. Some of these tools, such as the NASA General Mission Analysis Tool (GMAT) [10,11], Java Astrodynamics Toolkit (JAT) [12], and OreKit [13], were initially developed to provide accurate orbit propagation and estimation. Although these tools primarily focus on space mission orbit simulation, a few have grown to include attitude simulation capabilities [14]. However, of these tools GMAT is focused on optimal orbit trajectories and does not have a modular architecture to include attitude flight algorithms. The JAVA-based simulation tools will not have the speed of a C-based simulation solution. Other off-the-shelf tools include NASA Trick to simulate complex space physics [15]; FreeFlyer[¶]; Analytical Graphics, Inc. (AGI), Systems Tool Kit (STK)[**]; and JPL's Dynamics Algorithms for Real-Time Simulation (DARTS) [16] for complex spacecraft dynamics and behavior. These tools are not open source and require a license to use.

This paper presents new Basilisk functionalities that allow externally generated attitude guidance solutions to be incorporated and analyzed. The goal is to provide a framework where the only difference in the attitude maneuvering performance is the guidance solution. The use of this setup is illustrated by simulating a sample cube satellite using the Bevo-2 satellite parameters with its solar orientation constraints, described in Ref. [4]. This paper starts with a description of Basilisk, laying out key modules used in the attitude simulation (Sec. II). Next, it steps through the spacecraft mass and inertia properties, as well as the specific attitude constraints and their mathematical formulation (Sec. III). The performance metrics used in Basilisk are then identified (Sec. IV), followed by the four different path planners those metrics are used to evaluate (Sec. V). Finally, conclusions about the path planners based on the results from the comparison metrics are offered (Sec. VI).

II. Basilisk Framework Overview

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 [9,17].

The modules used in the following sections are briefly described. Figure 1 illustrates a sample attitude maneuver of a spacecraft using three reaction wheels (RWs) to control the orientation, and the translational motion is subject to both Earth and Sun gravity. This overview of Basilisk capabilities is not meant to be comprehensive, but illustrative of how complex benchmark simulation scenarios can be set up 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 solves the position and attitude differential equations of the spacecraft hub. It contains information such as spacecraft mass and inertia, and it 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.

RW(): This module creates a list of RWs. These can be generated from a database of existing wheels, or they can be customized by the user. When generating an 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.

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 modeled as a Gauss–Markov process.

inertial3D(): This module is used to set a fixed inertial attitude that the spacecraft must converge to. The provided message contains the modified Rodrigues parameter (MRP) attitude [18] of the reference frame $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}}$ with respect to the inertial frame, together with zeroed reference angular rates and accelerations ${}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}} = {}^{\mathcal{R}}\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} = [0, 0, 0]^T$.

attTrackingError(): It 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 an MRP-based Lyapunov feedback control law [18]. It receives the messages containing the relative attitude between spacecraft and reference frames, mass and inertia properties of the spacecraft, and RW states, and it computes a commanded torque in body-frame components.

rwMotorTorque(): This module maps the required torque into individual RW 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 RW(s) at the desired angular rate Ω.

The modular, open-source nature of Basilisk allows the user to implement complex attitude reference trajectory tracking thanks to the subdivision of the attitude tracking segment of code into modular components. These guidance modules consist in a base pointing reference, an attitude offset, and a dynamic reference trajectory relative to the base that is to be tracked [19]. This enables rapidly creating a custom attitude guidance algorithm and connecting it to the simulation in place of inertial3D().

[§]https://hanspeterschaub.info/basilisk.

[¶]https://www.agi.com/products/stk.

[**]https://ai-solutions.com/freeflyer-astrodynamic-software/.

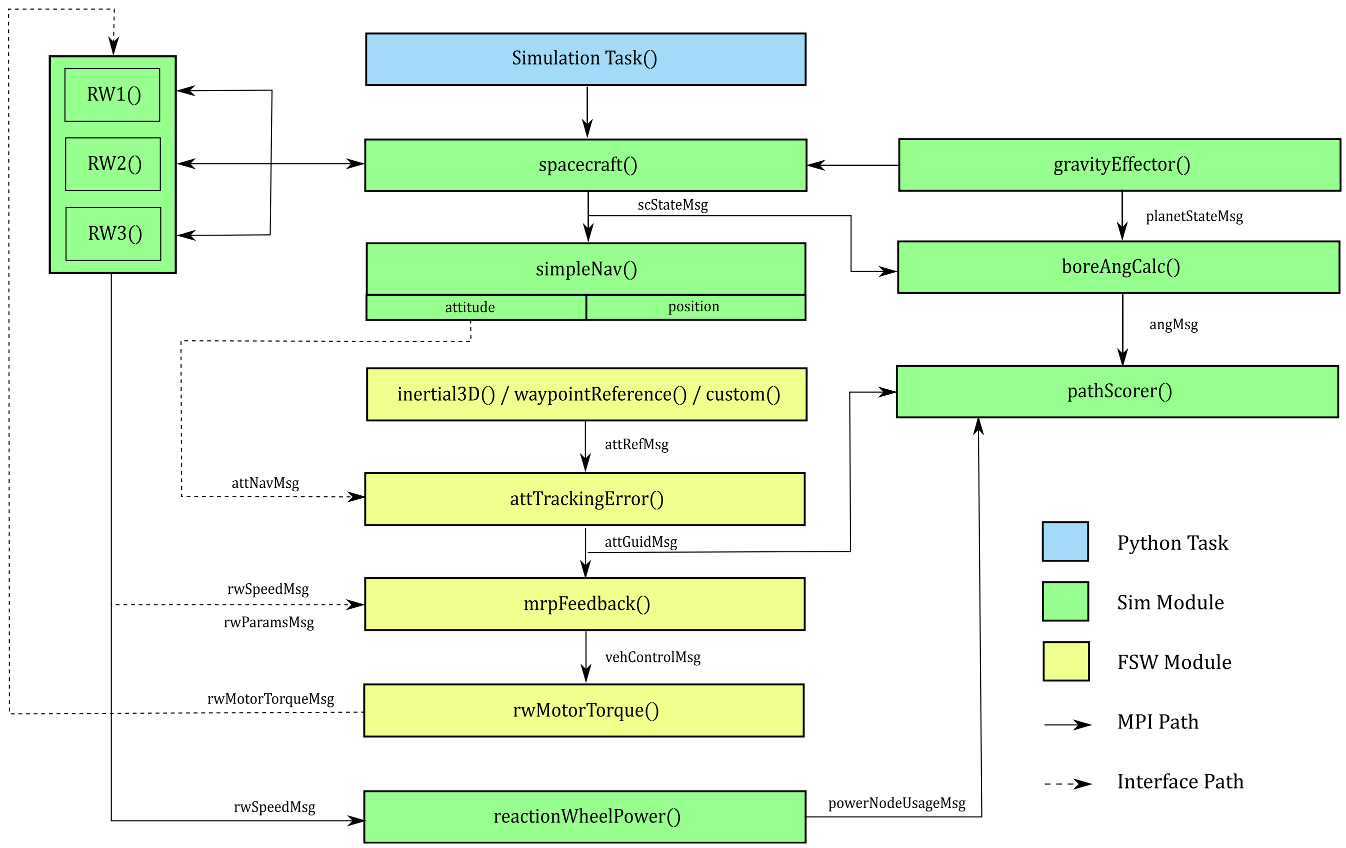


Fig. 1 Basilisk modular structure for the current simulation.

III. New Basilisk Capabilities to Analyze Constrained Attitude Solutions

A scheme of the modular structure of the Basilisk simulation is illustrated in Fig. 1. This section discusses two new capabilities to incorporate externally generated attitude guidance solutions, as well as to readily compare the performance of a constrained attitude maneuver.

A. Incorporating Externally Generated Attitude Guidance

A new module called waypointReference() imports the reference trajectory of a reorientation maneuver from a text file. This enables Basilisk to be used as a benchmarking tool to test constrained attitude guidance solutions computed outside the Basilisk environment. The reference trajectory in the text file should be provided as an ordered list of time-tagged attitude waypoints, together with the associated angular rates and accelerations, all separated by a delimiter. The attitude can be provided in terms of 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 $\mathcal{N}$. Defining $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}} = [\sigma_1, \sigma_2, \sigma_3]^T$, ${}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}} = [\omega_1, \omega_2, \omega_3]^T$, and ${}^{\mathcal{R}}\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} = [\dot{\omega}_1, \dot{\omega}_2, \dot{\omega}_3]^T$, the syntax for each line of the text file, relative to one attitude waypoint, is

$$t, \sigma_1, \sigma_2, \sigma_3, \omega_1, \omega_2, \omega_3, \dot{\omega}_1, \dot{\omega}_2, \dot{\omega}_3$$

The module outputs a reference message based on the reference trajectory described by the text file. When the simulation time is smaller than the first time tag, the output reference message contains the first reference attitude waypoint and zero angular rates and accelerations. When the simulation time exceeds the last time tag, the output reference message contains the last reference attitude waypoint with zero angular rates and accelerations. If the simulation time falls between two time tags $t_i \leq t_{\text{sim}} < t_{i+1}$, the reference message is obtained by means of linear interpolation of attitude, rates, and accelerations between waypoints i and $i+1$:

$$\hat{\boldsymbol{\sigma}}_{\mathcal{R}/\mathcal{N}} = \boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N},i} + \frac{\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N},i+1} - \boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N},i}}{t_{i+1} - t_i}(t_{\text{sim}} - t_i) \tag{1a}$$

$$\hat{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} = \boldsymbol{\omega}_{\mathcal{R}/\mathcal{N},i} + \frac{\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N},i+1} - \boldsymbol{\omega}_{\mathcal{R}/\mathcal{N},i}}{t_{i+1} - t_i}(t_{\text{sim}} - t_i) \tag{1b}$$

$$\hat{\dot{\boldsymbol{\omega}}}_{\mathcal{R}/\mathcal{N}} = \dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N},i} + \frac{\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N},i+1} - \dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N},i}}{t_{i+1} - t_i}(t_{\text{sim}} - t_i) \tag{1c}$$

where the hat indicates the quantities contained in the reference message. The linear interpolation between waypoints is an approximation, and it is therefore not correct from a kinematic standpoint. The three pieces of information contained in the reference message are, in fact, mutually interconnected by the relations [18]

$$\dot{\boldsymbol{\sigma}}_{\mathcal{R}/\mathcal{N}} = \frac{1}{4}[B(\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}})]^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}} \qquad {}^{\mathcal{R}}\dot{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} = \frac{\mathrm{d}}{\mathrm{d}t}[{}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{R}/\mathcal{N}}] \tag{2}$$

where

$$[B(\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}})] = (1 - \sigma^2_{\mathcal{R}/\mathcal{N}})[\boldsymbol{I}_{3\times3}] + 2[\tilde{\boldsymbol{\sigma}}_{\mathcal{R}/\mathcal{N}}] + 2\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}}\boldsymbol{\sigma}^T_{\mathcal{R}/\mathcal{N}} \tag{3}$$

and the tilde operator indicates the cross-product matrix $[\tilde{a}]b = a \times b$. This problem can cause the tracking error over time to appear irregular, because the reference trajectory that is being tracked is not kinematically consistent. However, this phenomenon becomes negligible as the density of the waypoints increases and consequently the spacing between each pair of consecutive waypoints is reduced.

It should be noted that, when the attitude is specified using quaternions, the module always converts it to the equivalent MRP set, because a linear interpolation in MRP space always returns a set that represents a valid attitude description. The output of a linear attitude interpolation in quaternion space would produce a set that does not respect the unit-norm constraint on the quaternion at all times. Interpolation in quaternion space can be done via spherical linear interpolation (SLERP), which can, however, be convoluted and computationally demanding [20]. For these reasons, the choice of MRPs as the attitude representation set seems obvious in order to simplify the approach.

B. Evaluating the Constrained Attitude Maneuver Performance

A new module called pathScorer() provides more insight on the performances of different path planners. The main goal is obviously to ensure constraint compliance at all times during a slew maneuver. However, in the presence of multiple algorithms, it can be useful to compare other metrics in order to assess performances also with respect to other physical quantities. Five performance metrics are defined to test the difference planners.

Total keep-out violation time: The total keep-out violation time is defined as the cumulative time during which any of the keep-out constraints are violated by the sensitive instrument's boresight direction. This study focuses on hard positional constraints. However, this metric also becomes particularly interesting when integral constraints are considered, where the boresight is allowed to violate the keep-out zone for a certain amount of time.

$$T_{\mathrm{KO}} = \int_0^T \delta_{\mathrm{KO}}\,\mathrm{d}t \qquad \delta_{\mathrm{KO}} = \begin{cases} 1 & \text{if keep-out is violated} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Total keep-in violation time: In this keep-in constraints are considered satisfied when there is at least one constraint-compliant sensor at all times. This is to ensure that the keep-in object is visible at all times, which does not require for it to be visible by all the sensors at all times.

$$T_{\mathrm{KI}} = \int_0^T \delta_{\mathrm{KI}}\,\mathrm{d}t \qquad \delta_{\mathrm{KI}} = \begin{cases} 1 & \text{if keep-in is violated} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Attitude error integral: Defining $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ as the MRP 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 it 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; therefore this quantity must ideally remain small to ensure that the trajectory is tracked successfully.

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

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 $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{R}}$ [18]:

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

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 RW configuration), and $\boldsymbol{h}_s$ is the vector containing the angular momenta of each RW about its spinning axis. The commanded torque integral is defined as

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

Total RW energy consumption: The total energy consumption is the integral over maneuver time of the power requirements of all the RWs combined. For each RW, 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

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

IV. Attitude Path Planners

To illustrate the use of these new capabilities, four different attitude guidance solutions are compared in this Basilisk simulation framework. The goal is to have an unconstrained attitude guidance solution as a baseline, and then compare this to different constrained attitude guidance solutions.

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. In this case the reference message remains constant over time, with $\hat{\boldsymbol{\sigma}}_{\mathcal{R}/\mathcal{N}}$ being the final target attitude, and $\hat{\boldsymbol{\omega}}_{\mathcal{R}/\mathcal{N}} = \hat{\dot{\boldsymbol{\sigma}}}_{\mathcal{R}/\mathcal{N}} = 0$. Because 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 $j = 0, \ldots, N$, 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 $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_N} = \boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_f}$. This approach is often found in literature involving attitude path planning [4,21]: in the absence of any knowledge of the required angular rates and accelerations at the intermediate waypoints, these are set to zero. This choice yields a path planning algorithm that tries to perform rest-to-rest maneuvers between the intermediate waypoints. The sequence of waypoints is obtained by applying the A* algorithm to an undirected graph, whose nodes correspond to MRP sets that represent constraint-compliant attitudes for the spacecraft. Such grid is obtained sampling the unit sphere centered at the origin in MRP space, which is a subset of the $\boldsymbol{R}^3$ MRP space that, however, contains all the possible attitudes for a spacecraft that rotates in $SO(3)$ [22]. The unit sphere is sampled with equally spaced node points, whose associated attitudes are tested for constraint compliance. Only the constraint compliant nodes become part of the search graph, together with the initial and final attitude nodes. More details on how the search graph is set up can be found in [23]. The cost function used by the A* algorithm, for this planner, is equivalent to a Cartesian distance in MRP space between the nodes: $d(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$. The time-spacing between waypoints is proportional to the mutual distance between waypoints in MRP space according to the law $\Delta t = [d(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)/\omega^*]$. For this planner, the time-tagged waypoints contained in the text file processed by the waypointReference() module are relatively sparse: with respect to [23], the grid density corresponds to a level $N = 13$. In conclusion, this planner tries to target a sequence of constraint-compliant waypoints, in a rest-to-rest style, because the angular rates and accelerations contained in the reference message are always zeroed.

Planner #2: Planner #2 improves on the results of planner #1. The same sequence of baseline waypoints is used as by planner #1. However, the constraint-compliant waypoints are 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. The value of 0.04 rad/s is chosen arbitrarily, and it corresponds to 1.15 deg/s, which is an acceptably small angular rate for a spacecraft performing a slew maneuver. The output text file, for this planner, contains a list of time-tagged attitude waypoints, angular rates, and accelerations that are sampled from the aforementioned interpolated trajectory. Because such trajectory is a continuous function of time, the density of the waypoints in the output text file can be chosen arbitrarily, as the waypoints and associated rates and accelerations can be evaluated at whatever time is desired. The denser the sampling along the trajectory, the smaller the approximation error introduced by the linear interpolation in Eq. (1) performed by the waypointReference() module. As an example of this, Fig. 2 shows the attitude and rate tracking errors for different numbers of waypoints sampled from the interpolated trajectory, where the smallest, $M = 13$, corresponds to the same number of baseline waypoints in the solution provided by planner #0. It is clear how, for an increasing number of samples, the attitude error significantly diminishes until becoming negligible. Therefore, knowing the reference trajectory, it makes sense to always sample the waypoints with a high density. For planner #2, in total 500 waypoints are sampled from the interpolated trajectory and processed by the waypointReference() module.

Planner #3: Planner #3 coincides with the effort-minimizing A* algorithm described in [23]. Such algorithm searches the graph for a path that yields an interpolated trajectory such that the control torque integral, as defined in Eq. (8), is the smallest. The expectation is, therefore, that planner #3 always outperforms planner #2 with respect to the control torque integral performance metric. The output for planner #3 is therefore, 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. Once again, the text file produced by the planner and processed by the waypointReference() module contains 500 time-tagged attitude waypoints, angular rates, and accelerations sampled from the effort-optimal interpolated trajectory. Figure 3 shows the same study, regarding waypoint sampling density from the interpolated trajectory, as for planner #2, and the same considerations apply.

V. 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 deg 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 deg 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

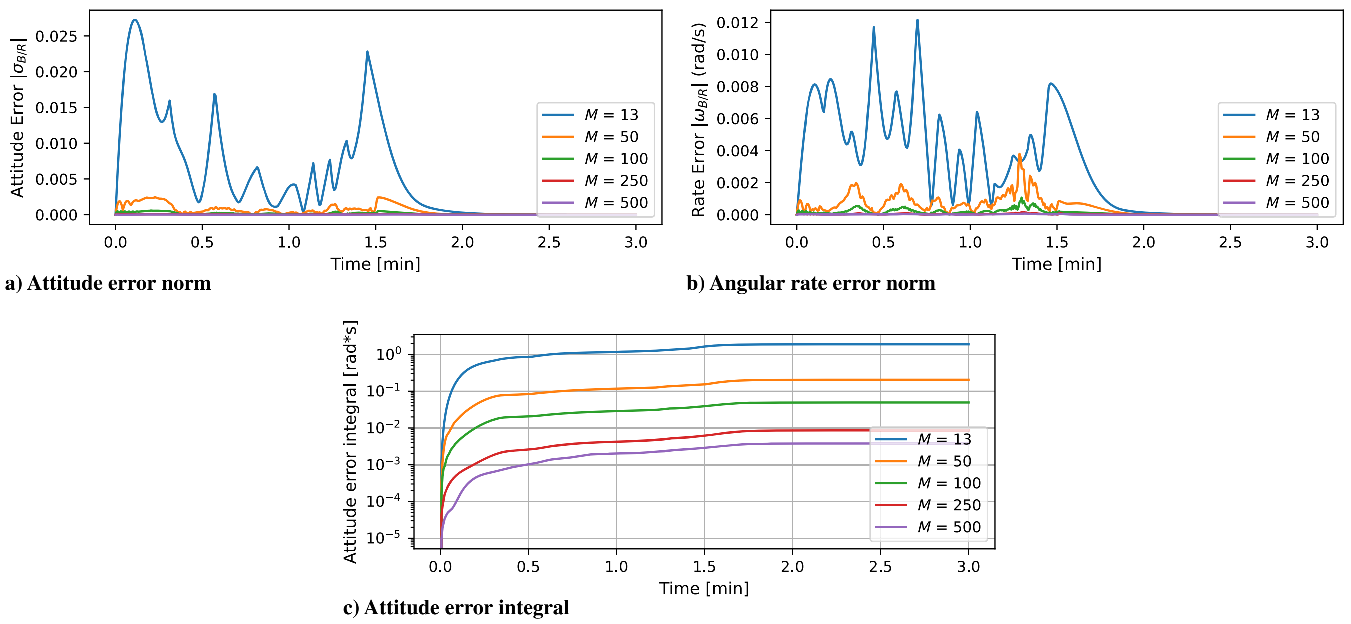


Fig. 2 Tracking error for planner #2 for different density of points *M* sampled from the interpolated trajectory.

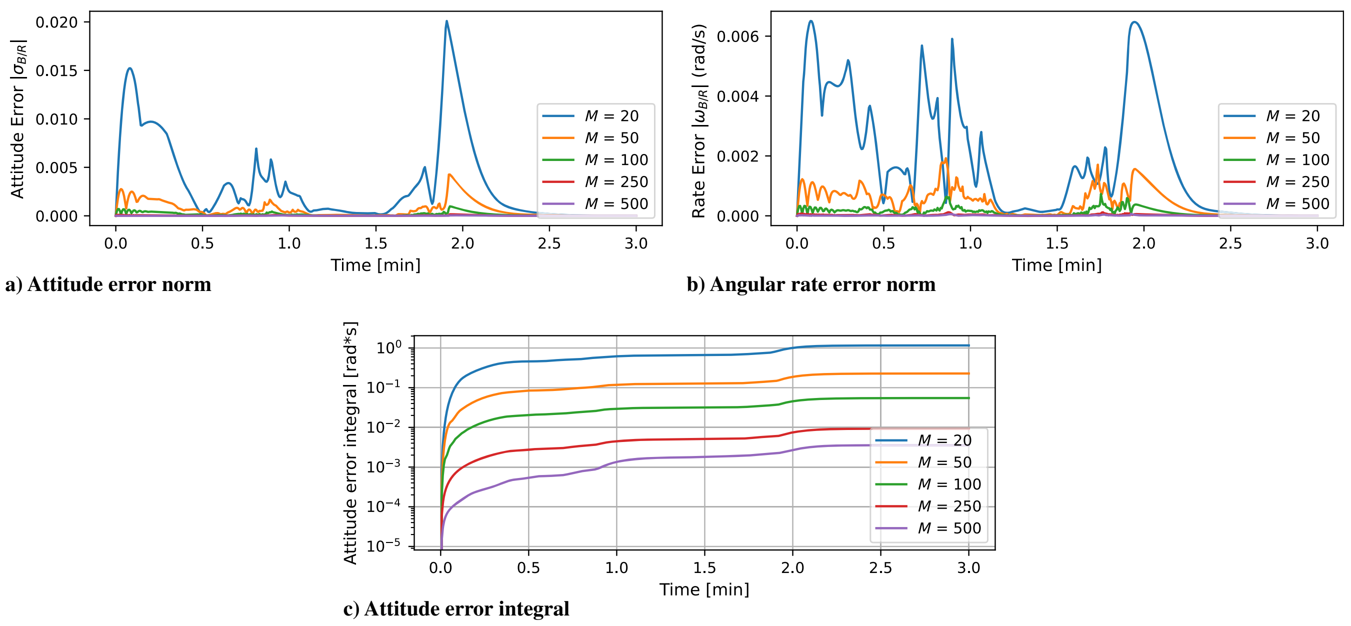


Fig. 3 Tracking error for planner #3 for different density of points *M* sampled from the interpolated trajectory.

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 three-unit CubeSat with a uniform mass distribution [24]:

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

The actuation is provided by a set of three RWs aligned with the principal inertia axes ${}^{\mathcal{B}}\boldsymbol{b}_x$, ${}^{\mathcal{B}}\boldsymbol{b}_y$, and ${}^{\mathcal{B}}\boldsymbol{b}_z$. The RWs can provide a control torque up to 1 mN·m each. The mass of the RWs 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 modeled as hard constraints, and the following equations must be satisfied at all times:

$$ {}^{\mathcal{B}}\hat{\boldsymbol{b}}_x \cdot [\mathcal{BN}]^{\mathcal{N}}\hat{\boldsymbol{s}} < \cos(20^\circ) \tag{11}$$

$$ {}^{\mathcal{B}}\hat{\boldsymbol{b}}_y \cdot [\mathcal{BN}]^{\mathcal{N}}\hat{\boldsymbol{s}} \geq \cos(70^\circ) \| {}^{\mathcal{B}}\hat{\boldsymbol{b}}_z \cdot [\mathcal{BN}]^{\mathcal{N}}\hat{\boldsymbol{s}} \geq \cos(70^\circ) \tag{12}$$

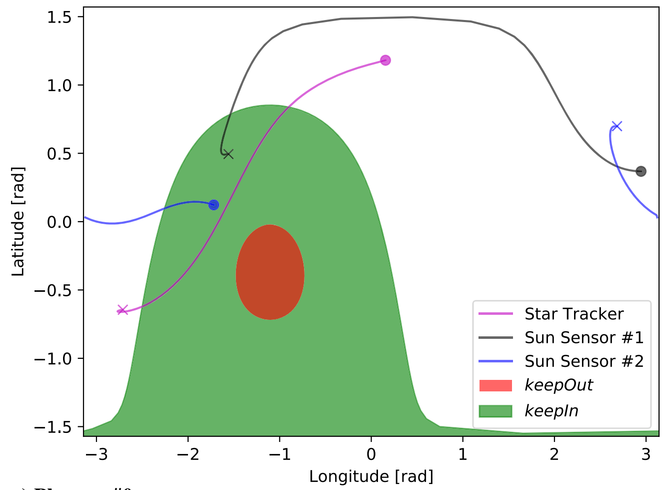
where $[\mathcal{BN}]$ is the direction cosine matrix that maps vectors from the inertial frame $[\mathcal{N}]$ to the body frame $[\mathcal{B}]$ and ${}^{\mathcal{N}}\hat{\boldsymbol{s}}$ is the inertial direction of the Sun.

VI. Benchmark Analysis

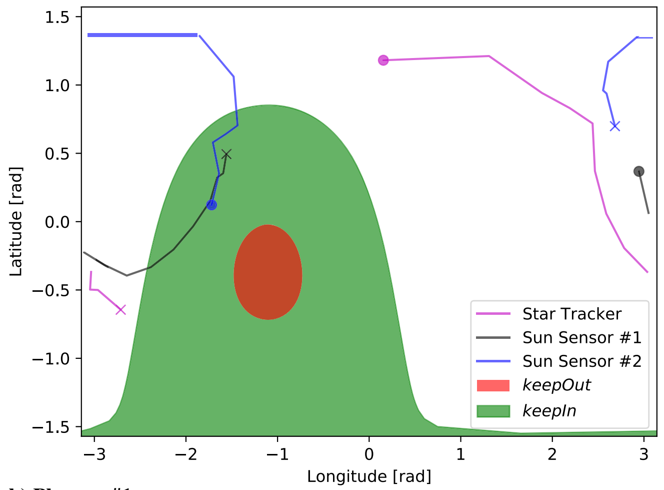
A. Planner Performance Comparison

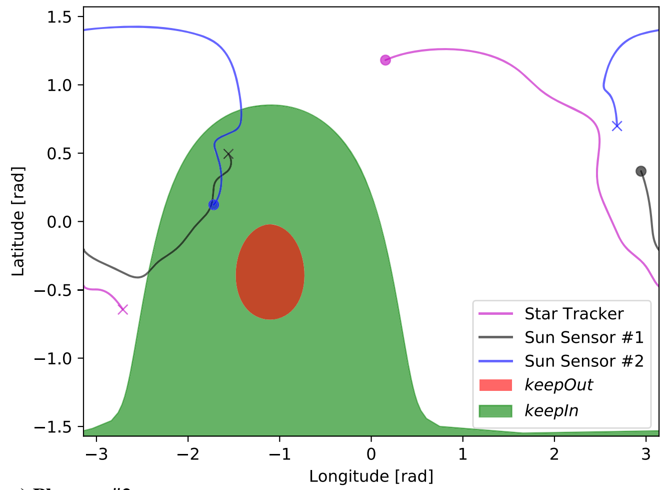
This section shows the performance of the different planners based on a set of common evaluation criteria and the performance metrics described above. The scenario presented here features a slew maneuver between the attitudes $\boldsymbol{\sigma}_{\mathcal{R}/\mathcal{N}_i} = [0.522, -0.065, 0.539]^T$ 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 $P = 1.256 \cdot 10^{-2}$ N · s in Eq. (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 that would converge to the desired target in about 1 minute. Figure 4 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 4a shows the actual trajectory of the boresights in the body frame $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ when planner #0 is used, whereas Figs. 4b–4d 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



a) Planner #0


b) Planner #1


c) Planner #2

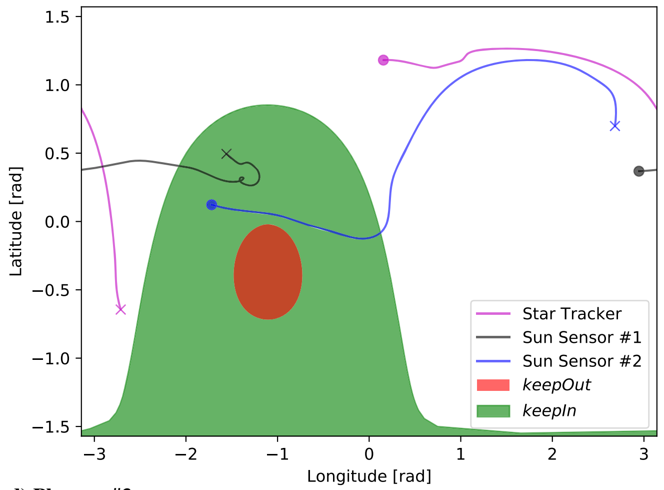
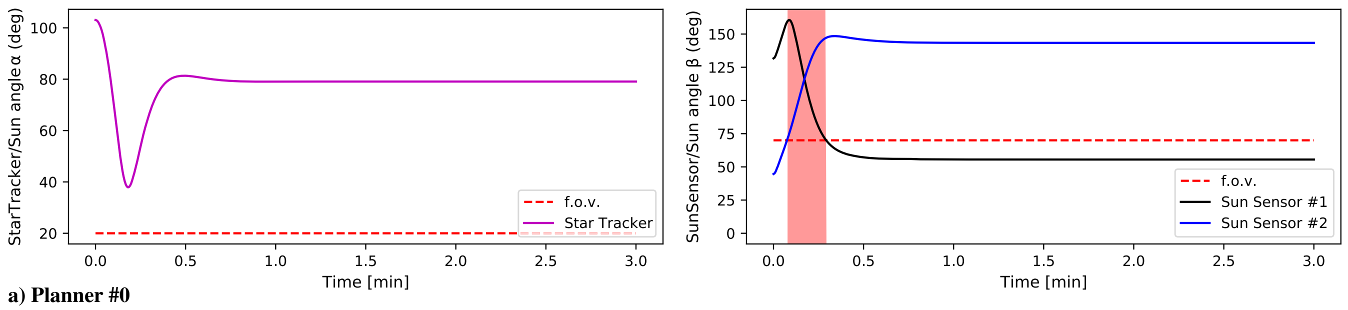

d) Planner #3

Fig. 4 The 2D plots of inertial boresight directions; ●, starting point; ×, endpoint.

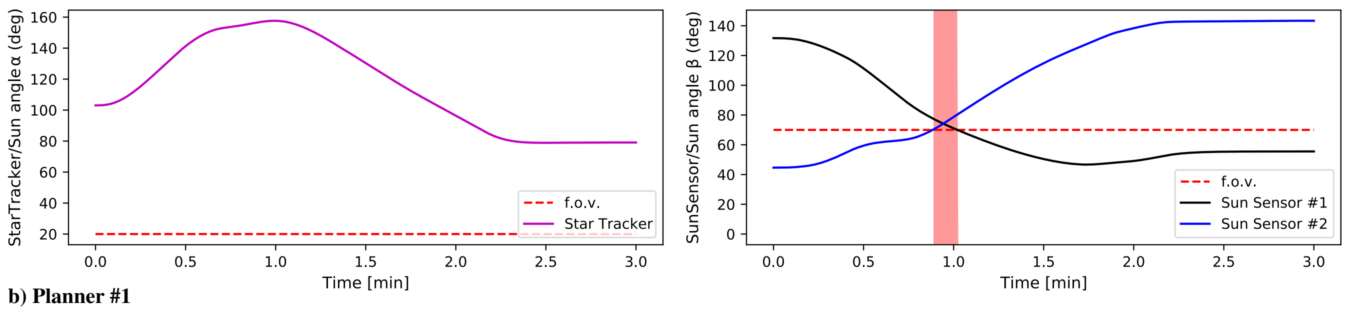
just the initial and final reference attitudes. For planner #1, the discrete reference waypoints produce a sequence of discrete target boresight inertial directions. Lastly, for planners #2 and #3, full reference trajectories are obtained for the boresight directions as functions of time.

Figure 4 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 5 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, because the interpolating function used maintains the trajectory within the convex hull described by the interpolated attitude waypoints [25]. 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.

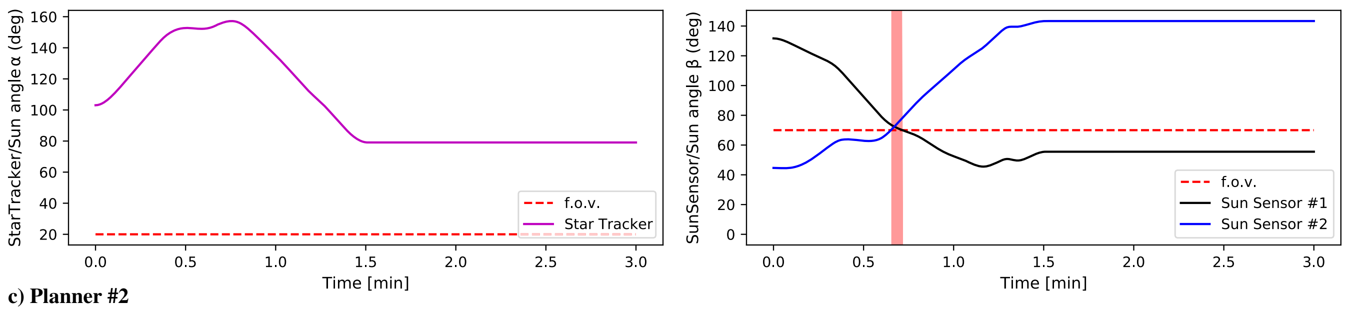
Figure 6 shows the performance metrics described above and offers a direct comparison between the four planners. Subfigures a) and b) show the constraint violation times, where the same information can be observed as in Fig. 4, with more details on how long the constraint violations last for each planner. 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 RW 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



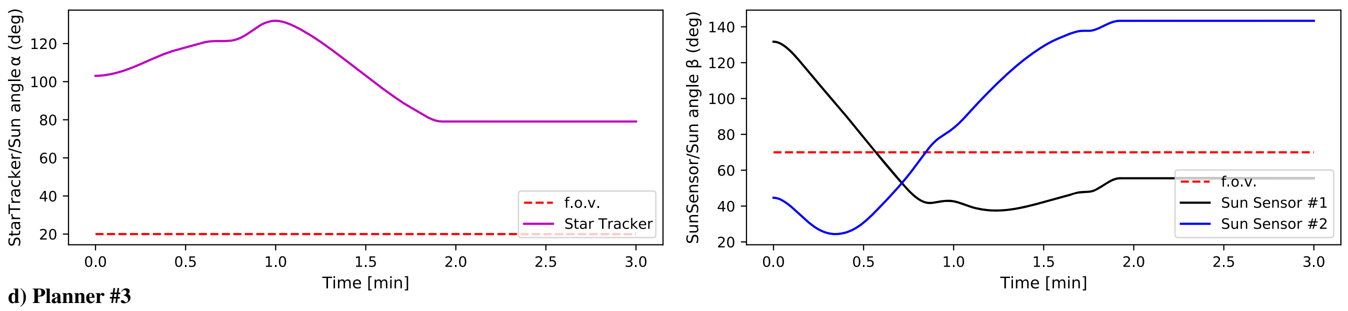
a) Planner #0



b) Planner #1



c) Planner #2



d) Planner #3

Fig. 5 Angles between star tracker and Sun and between sun sensors and Sun.

energy consumption: this is because each torque component is mapped directly to the corresponding RW. 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, because 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 three 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 toward 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 constraint-compliant sequence of waypoints is used. On the other hand, planners #2 and #3 are based on a nominal reference trajectory that interpolates the waypoints. In such cases, the interpolating spline can present wiggles between the waypoints. Such wiggles can appear as a consequence of having the spline curve pass through the waypoints precisely. This phenomenon becomes more significant as the waypoints are denser (denser grid). Because the reference trajectory is tracked accurately by the spacecraft, the torque integral is ultimately larger due to the effort required to track such undesired wiggles between the waypoints presented by the reference trajectory. 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.

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 $P \in [1.256 \cdot 10^{-3}, 1.256 \cdot 10^{-1}]$. 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. With planner #0 being constraint naive, the torque in Eq. (7) is computed using the attitude error $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ computed with respect to the target attitude. This, multiplied by the gain K, can result in a very high initial commanded torque when K is increased, which causes the spacecraft to overshoot the target. This, added to the fact that planner #0 is different in nature to the other planners because it is constraint naive, makes it not interesting for the following gain sensitivity analysis, and it is therefore removed from it.

Figure 7 summarizes the results obtained with varying gains. Solid lines represent the averaged curves for each planner, whereas the color-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

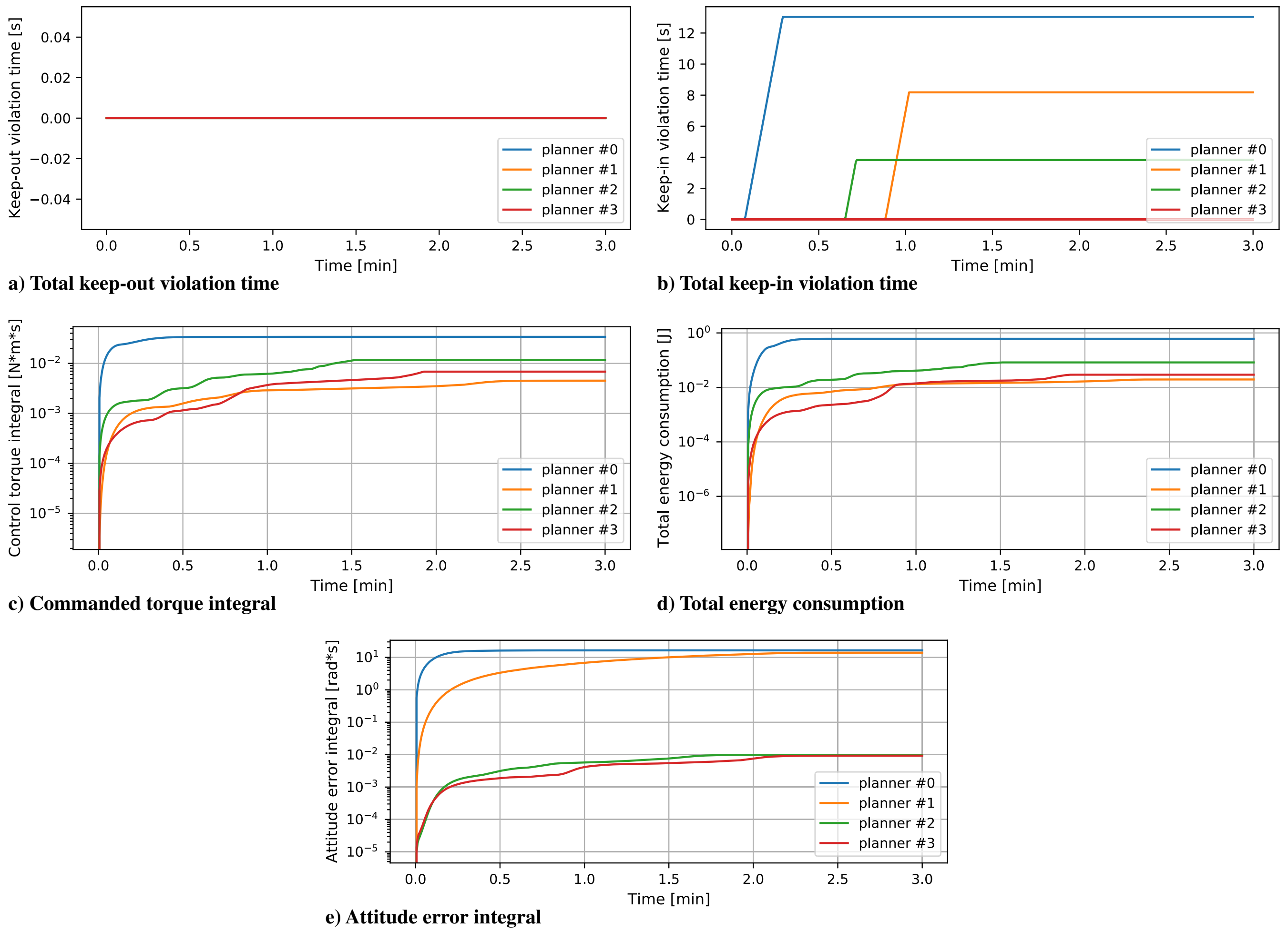
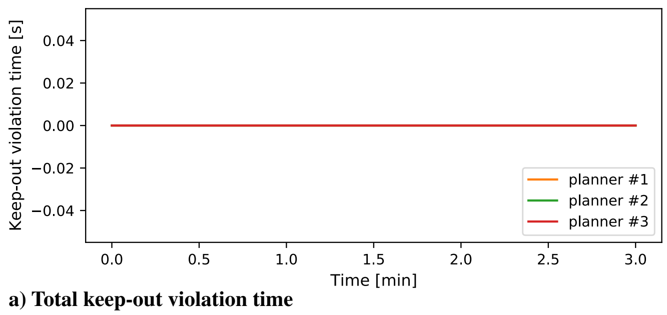
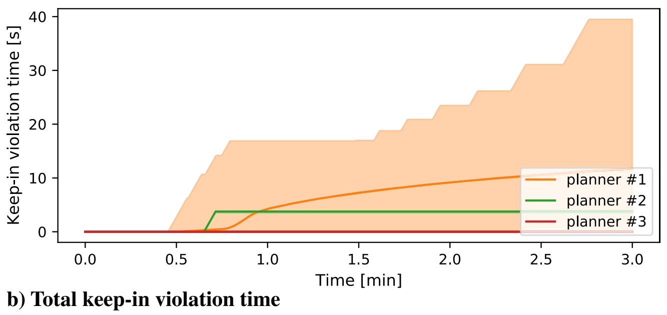


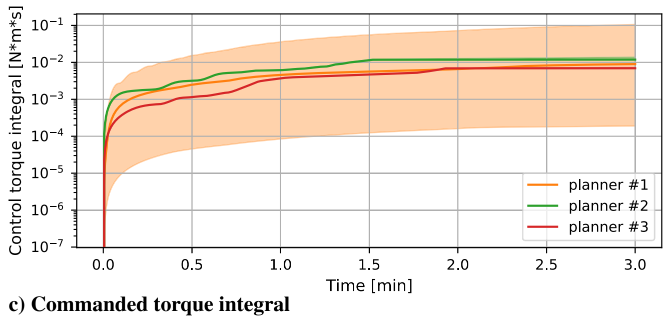
Fig. 6 Performance metrics of the four planners compared.



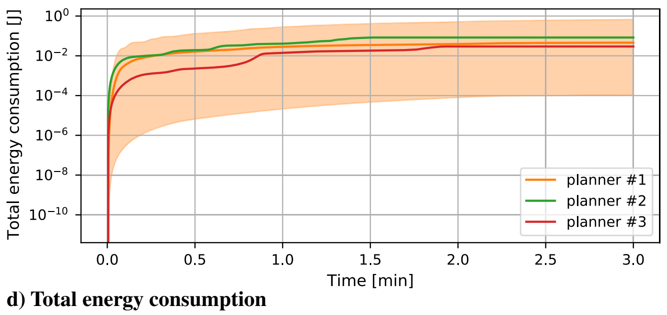
a) Total keep-out violation time



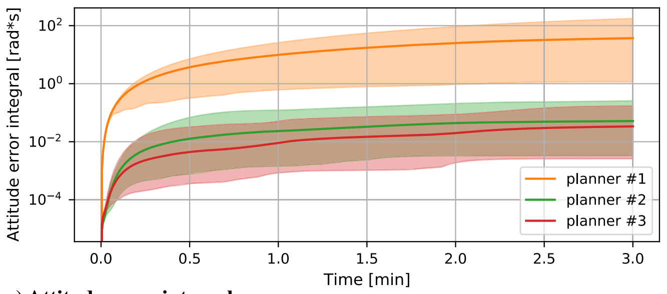
b) Total keep-in violation time



c) Commanded torque integral



d) Total energy consumption



e) Attitude error integral

Fig. 7 Sensitivity of planners #1–#3 to gain variations.

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 $K\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ and $P\boldsymbol{\omega}_{\mathcal{B}/\mathcal{R}}$ in Eq. (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. Conclusions

This paper presents an open-source method to perform comparative studies of constrained attitude guidance solutions. A Basilisk simulation is set up to test the performances of different path planning algorithms against a standardized scenario. The strength of Basilisk simulations relies on its scriptability and the ease with which the simulation setup can be modified to match a desired scenario. Basilisk's modular structure allows users 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 new 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 new 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 is not enough to provide a discrete sequence of attitude waypoints for an accurate tracking of a reference trajectory, because 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 and open-source benchmarking tool, 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.

Acknowledgment

Parts of this research were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA (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.
https://doi.org/10.2514/3.21283

[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.
https://doi.org/10.2514/1.G002873

[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.
https://doi.org/10.2514/1.G001879

[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.
https://doi.org/10.2514/1.60189

[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.
https://doi.org/10.2514/1.G001063

[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.
https://doi.org/10.2514/1.G001801

[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*, AIAA Paper 1997-3526, 1997.

[8] Kim, Y., Mesbahi, M., Singh, G., and Hadaegh, F., "On the Constrained Attitude Control Problem," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA Paper 2004-5129, 2004.

[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.
https://doi.org/10.2514/1.I010762.

[10] Hughes, S. P., Qureshi, R. H., Cooley, S. D., and Parker, J. J., "Verification and Validation of the General Mission Analysis Tool (GMAT)," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2014-4151, 2014.

[11] Hughes, S. P., "General Mission Analysis Tool (GMAT)," *International Conference on Astrodynamics Tools and Techniques (ICATT)*, 2016.

[12] Gaylor, D. E., Berthold, T., and Takada, N., "Java Astrodynamics Toolkit (JAT)," *Advances in the Astronautical Sciences*, Vol. 121, 2005, pp. 263–272.

[13] Maisonobe, L., Pommier, V., and Parraud, P., "Orekit: An Open Source Library for Operational Flight Dynamics Applications," *4th International Conference on Astrodynamics Tools and Techniques*, 2010, pp. 3–6.

[14] Andringa, K. D., and Cuseo, J. A., "A Novel Approach to Spacecraft Modeling and Simulation from Concept Through Operations," *AIAA Modeling and Simulation Technologies (MST) Conference*, AIAA Paper 2013-4831, 2013.

[15] Penn, J., and Lin, A., "The Trick Simulation Toolkit: A NASA/Opensource Framework for Running Time Based Physics Models," *AIAA Modeling and Simulation Technologies Conference*, AIAA Paper 2016-1187, 2016.

[16] Jain, A., "DARTS-Multibody Modeling, Simulation and Analysis Software," *European Congress on Computational Methods in Applied Sciences and Engineering*, Springer, Duisburg, Germany, 2019, pp. 433–441.

[17] Carnahan, S., Piggott, S., and Schaub, H., "A New Messaging System For Basilisk," *AAS Guidance and Control Conference*, AAS Paper 20-134, 2020.

[18] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 4th ed., AIAA Education Series, AIAA, Reston, VA, 2018, Chap. 3.
https://doi.org/10.2514/4.105210

[19] Cols-Margenet, M., Schaub, H., and Piggott, S., "Modular Attitude Guidance: Generating Rotational Reference Motions for Distinct Mission Profiles," *Journal of Aerospace Information Systems*, Vol. 15, No. 6, 2018, pp. 335–352.
https://doi.org/10.2514/1.I010554.

[20] Leeney, M., "Fast Quaternion Slerp," *International Journal of Computer Mathematics*, Vol. 86, No. 1, 2009, pp. 79–84.
https://doi.org/10.1080/00207160801923064

[21] Frazzoli, E., Dahleh, M., Feron, E., and Kornfeld, R., *A Randomized Attitude Slew Planning Algorithm for Autonomous Spacecraft*, Vol. 4155, AIAA, Reston, VA, 2001.
https://doi.org/10.2514/6.2001-4155

[22] Schaub, H., and Junkins, J. L., "Stereographic Orientation Parameters for Attitude Dynamics: A Generalization of the Rodrigues Parameters," *Journal of the Astronautical Sciences*, Vol. 44, No. 1, 1996, pp. 1–19.

[23] Calaon, R., and Schaub, H., "Constrained Attitude Maneuvering via Modified-Rodrigues-Parameter-Based Motion Planning Algorithms," *Journal of Spacecraft and Rockets*, Vol. 60, No. 4, 2022, pp. 1–15.
https://doi.org/10.2514/1.A35294

[24] 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*, California Polytechnic State Univ., San Luis Obispo, 2014, pp. 1–42.

[25] Piegl, L., and Tiller, W., *The NURBS Book*, Springer Science & Business Media, Cham, Switzerland, 1996, Chap. 1.

D. Casbeer
Associate Editor