IWSCFF 19-52

SEQUENTIALLY DISTRIBUTED ATTITUDE GUIDANCE SOFTWARE ACROSS A SPACECRAFT CONSTELLATION

Mar Cols Margenet[*] and Hanspeter Schaub[†]

A distributed attitude guidance strategy is proposed that generates a complex attitude reference by adding sequential contributions from multiple spacecraft. Without loss in generality, two spacecraft are going to be considered: the chief spacecraft computes an attitude reference pointing towards a planet, and this time-varying reference is communicated to a deputy spacecraft that aligns with it and superimposes dynamic motion relative to the chief orientation. The final guidance reference of the deputy is a specific scanning pattern on the surroundings of the chief pointing target. Simulating a distributed spacecraft guidance-control solution with hardware-in-the-loop components is challenging due to the complexity of the software-hardware interactions and the need for synchronization. The proposed guidance concept is numerically demonstrated using a novel software-hardware framework that takes advantage of the Basilisk flight software testbed and the Black Lion communication architecture. This framework allows simulating the coupled behavior of a constellation of satellites in a realistic, disaggregated and multi-platform fashion. This novel framework has been built up under the principles of reusability and scalability and, as matter of fact, its applications extend far beyond the sample mission scenario discussed here. Performing distributed guidance within a constellation of satellites provides an interesting case-scenario that is implemented and demonstrated within this framework.

INTRODUCTION

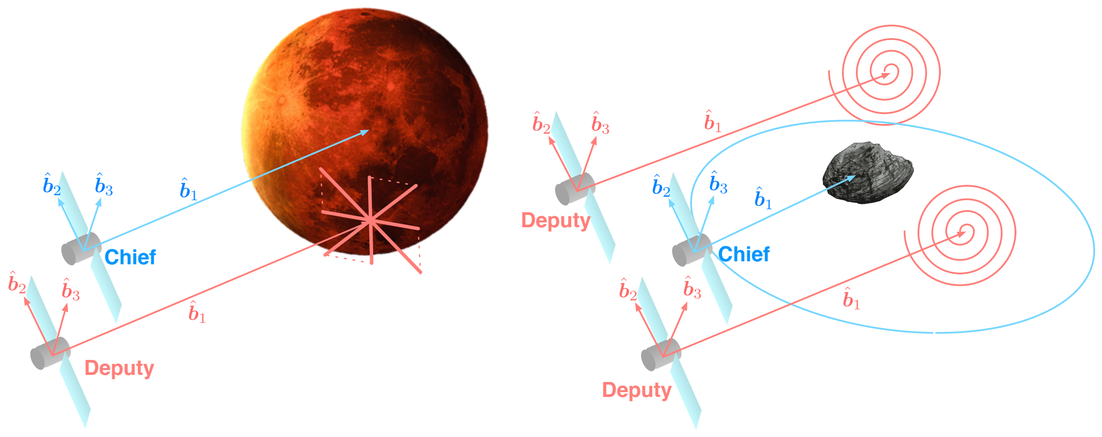
Space missions rely highly on the efficiency and reliability of the on-board flight software (FSW) in order to perform autonomous attitude control or orbit corrections. These critical software functions undergo a stringent review and validation process prior to flight, which can be both costly and time consuming. The complete engineering process to develop an aerospace FSW system encompasses an involved path starting from a preliminary desktop design and analysis all the way to testing on the flight hardware. The testing challenges of a flight system increase even further when there is not only a single spacecraft involved but multiple ones that need to cooperate together in order to accomplish a common mission objective. This is indeed the case of spacecraft constellation scenarios, where each member in the group is actually a flight system in itself. Yet, it is still required that all these individual flight systems, which jointly conform the constellation, are tested together in an integrated and holistic manner. The present manuscript proposes a novel concept for designing relative attitude guidance maneuvers within a constellation of satellites and for testing them in realistic, distributed, closed-loop simulations.

In the proposed concept, the guidance reference for attitude maneuvering is built onboard by contribution from multiple spacecraft. Architecting the design of attitude guidance trajectories/maneuvers

[*]Graduate Student, Aerospace Engineering Sciences, University of Colorado Boulder.

[†]Professor, Glenn L. Murphy Chair, Department of Aerospace Engineering Sciences, University of Colorado, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO 80309-0431. AAS Fellow.

is specially interesting because guidance algorithms tend to be mission-specific and, often, these algorithms are not built under the principles of scalability nor reusability.[1] Very so often, the lack of software architecture yields to a lack in documentation and reported literature as well. For small-sat constellations in particular, there is very little reported on algorithms for controlled pointing and distributed target observation.[2] In contrast, there is active investigation on: relative navigation techniques for satellite constellations;[3–5] on control laws for formation flying missions;[6] and even on ground scheduling algorithms for constellations.[2] This paper aims to contribute to the field of onboard guidance generation in the context of space exploration through small-sat constellations.



(a) Mars Atmosphere Exploration (b) Asteroid Surroundings Exploration

Figure 1. Sample Mission Concepts with Distributed Guidance

Figure 1 shows two sample mission concepts that could take advantage of distributed guidance through small satellites: exploring the martian atmosphere and the surroundings of an asteroid system. Both cases in Fig. 1 present a chief spacecraft performing nominal science and small-sat deputies that perform scanning maneuvers relative to the chief's attitude reference. In the lines of Fig. 1(a), mapping Mars' atmosphere is the science goal of an ongoing single-spacecraft mission*, but extending such mission with small-satellites could increase the science return. In turn, scanning the surroundings of an asteroid, as in Fig. 1(b), is specially interesting because these celestial objects are usually not fully characterized –specially if it is the first time they are being explored.[7–11] Further, a considerable number of asteroids are part of binary or multiple systems[12–15] and the proposed constellation could use the deputy satellites to look for the presence of other small bodies around the chief's main target. The recent Osiris-Rex mission, for example, found more asteroid debris-ejecta in the neighborhood of the asteroid Bennu than expected.

From a guidance point of view, the general idea is that the guidance reference for the chief is a time-varying pointing reference, such as pointing at the center of the orbited celestial body. The pointed celestial object could be any target for which the chief has onboard ephemeris: a planet, moon, asteroid, etc. For the deputies, the goal is to scan the surroundings of the chief's target, in search of additional science opportunities or events. As the chief keeps orbiting, the base pointing reference will change, and the deputies will realign to match this time-varying base, on top of which the scanning pattern will continue. This strategy provides constrained autonomy to the deputies, which rely on the base reference generated by the chief but not on the ground. In the numerical

*`http://emiratesmarsmission.ae`

simulations that will be presented, the chief is autonomous in the sense that it does not follow a predefined attitude reference, but rather computes its final reference onboard, based on a celestial body pointing request. An interesting aspect of this concept is that the deputy satellites do not require any kind of knowledge about the nature of the target –they simply scan the surrounding area of the chief's pointing reference. Note that this concept of operation is valid for any relative distribution between chief and deputies (which could certainly be on different orbits).

In order to advance in the development of new concepts for spacecraft formation and constellation, it is critical to have flexible testbeds that allow realistic hardware-in-the-loop (HWIL) simulations as well as pure-software ones. Pure-software simulations are convenient because they provide emulated substitutions for expensive and limited pieces of hardware –therefore allowing simultaneous testing among different mission groups.[16–18] In turn, HWIL simulations are a common industry practice for reducing in-flight risk in experimental technologies as well as for achieving/proving Flight Software Technology Readiness Levels (TRL).[19] As an example of experimental technology, an emerging new trend in some small-sat constellation missions is to use commercial processors in redundant configurations instead of a single radiation hardened processor.[20] On these lines, small-satellite missions or start-up companies with limited resources and without extensive flight software legacy, would highly benefit from having available an easily deployable, easily customizable HWIL testbed. Currently, there is no multi-purpose, scalable constellation simulator capable of testing the interactions between members of a constellation, interactions between constellations, constellation interactions with ground operations, or the response of a constellation to intentional interference.[19]

The present work proposes a novel testbed framework that allows both real-time HWIL simulations as well as faster-than-real-time pure-software simulations. The proposed framework takes advantage of several modern software tools and hardware components to provide the space community with a flight system and test environment that is highly customizable, user-friendly and cost-effective, such that it can potentially suite a wide range of constellation mission profiles. The software and hardware tools that are combined together are: the Basilisk astrodynamics software,[1,21–24] the Black Lion communication architecture,[16]and the Raspberry Pi hardware.[*] As a matter of fact, the Basilisk desktop testbed and the Black Lion communication architecture are currently being used to support an interplanetary single-spacecraft mission. The work presented in this manuscript uses Basilisk-developed flight algorithms in the extended scenario of a constellation of satellites. In turn, Black Lion is being used for emulated flat-sat testing with the purpose of testing the interactions between the embedded FSW, the ground system model and the spacecraft physical behavior.[16] In this paper it will be shown how these tools, which have been built under the principles of scalability and reusability, can effectively be used in different mission concepts: for instance, in a spacecraft constellation.

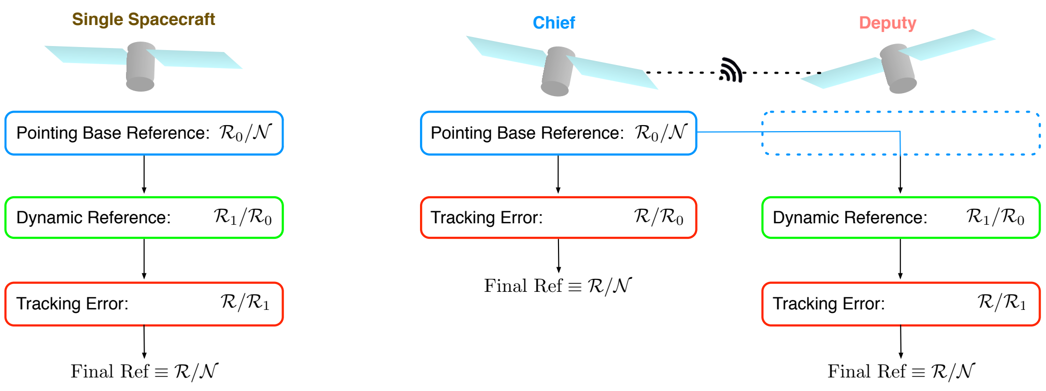
The paper is outlined as follows. First the modular guidance solution of prior work on a single spacecraft is reviewed and then expanded to the concept of a distributed guidance solution within a constellation. Next, the software-hardware framework used to test the holistic behavior of the constellation, in a realistic and disaggregated fashion, is presented. After introducing the tools, the particular setup for numerically simulating a distributed maneuver with one chief and one deputy spacecraft is described; numerical simulation results are shown and conclusions are finally drawn.

[*]`https://www.raspberrypi.org`

FROM MODULAR GUIDANCE TO DISTRIBUTED GUIDANCE

Reference 1 presents a modular scheme for generating attitude reference motions, where the term *generating* implies computing the desired motion autonomously onboard. Without loss in generality, any attitude reference motion is modularized into three atomic parts: a base pointing reference, a dynamic reference that is relative to the base, and an attitude offset that allows controlling any spacecraft fixed frame that is not the principal body frame. The final, desired reference motion is a result of cascading any combination of these attitude reference parts. The work in Ref. 1 encompassed the implementation of several guidance modules and their integration to the Basilisk software framework. The numerical simulations in Ref. 1 show how the guidance modules can be plugged-and-played as lego-like pieces in order to achieve very different rotational patterns.

Within the scope of Ref. 1, modular guidance maneuvers were performed on a single spacecraft. This layered, modular approach of building an attitude reference was initially created to promote reusability of code throughout distinct mission profiles. Now, it is shown how this layering strategy holds for relative ADCS guidance as well, preserving the same full kinematic properties. The transition from the modular guidance strategy described in Ref. 1 to the distributed guidance approach presented here is depicted in Fig. 2


Single Spacecraft
Pointing Base Reference: $\mathcal{R}_0/\mathcal{N}$
Dynamic Reference: $\mathcal{R}_1/\mathcal{R}_0$
Tracking Error: $\mathcal{R}/\mathcal{R}_1$
Final Ref $\equiv \mathcal{R}/\mathcal{N}$
Chief
Deputy
Pointing Base Reference: $\mathcal{R}_0/\mathcal{N}$
Tracking Error: $\mathcal{R}/\mathcal{R}_0$
Final Ref $\equiv \mathcal{R}/\mathcal{N}$
Dynamic Reference: $\mathcal{R}_1/\mathcal{R}_0$
Tracking Error: $\mathcal{R}/\mathcal{R}_1$
Final Ref $\equiv \mathcal{R}/\mathcal{N}$

(a) Single-Spacecraft Modular Guidance (b) Multi-Spacecraft Distributed Guidance

Figure 2. Attitude Guidance Reference Generation: Modular vs. Distributed

Figure 2(b) shows the generation of two final references belonging to two different spacecraft. The idea here is that a secondary spacecraft –let us say the deputy– builds a compounded attitude reference by, first, taking a base pointing reference from a primary spacecraft –let us say the chief– and, then, defining a relative rotational motion on top.

For a given spacecraft, the goal of the onboard GN&C flight software is to estimate the current state of the spacecraft body $\mathcal{B}$ (navigation task), generate a reference state $\mathcal{R}$ that can be time-varying or not (guidance task), derive the attitude tracking error between the current state $\mathcal{B}$ and the desired one $\mathcal{R}$ (also guidance task), and apply the required control torque to align $\mathcal{B}$ with $\mathcal{R}$ (control task). In the proposed distributed guidance strategy, the final reference of the chief, $\mathcal{R}_{\text{chief}}$, is used by the deputy to build up a compounded rotational reference, $\mathcal{R}_{\text{deputy}}$.

On each spacecraft, the spacecraft-body state $\mathcal{B}$ and the reference state $\mathcal{R}$ computed onboard are expressed with respect to an inertial frame $\mathcal{N}$. For generating a guidance reference in a distributed fashion, this inertial $\mathcal{N}$ frame must be consistent throughout all the members in the constellation.

Note that $\mathcal{N}$ is kept as a generic inertial frame to be chosen (it could be J2000, Earth fixed-frame, Mars centered frame, etc.) since all that matters is consistency on the picked inertial frame.

The computed reference state $\mathcal{R}$ is, in this paper, composed of three parameters:

$$\mathcal{R} = [\boldsymbol{\sigma}_{R/N}, {}^{\mathcal{N}}\boldsymbol{\omega}_{R/N}, {}^{\mathcal{N}}\dot{\boldsymbol{\omega}}_{R/N}] \tag{1}$$

These parameters are: an inertial attitude measure, denoted through the Modified Rodrigues Parameters (MRP) set $\boldsymbol{\sigma}_{R/N}$,[25,26] an inertial angular rate vector ${}^{\mathcal{N}}\boldsymbol{\omega}_{R/N}$ expressed in inertial frame $\mathcal{N}$ components, and an inertial angular acceleration vector ${}^{\mathcal{N}}\dot{\boldsymbol{\omega}}_{R/N}$ also in $\mathcal{N}$-frame components. The left-superscript denotes the frame with respect to which the vector components are taken. Note that the nomenclature of the rotational state in Eq. (1) is taken from Ref. 26.

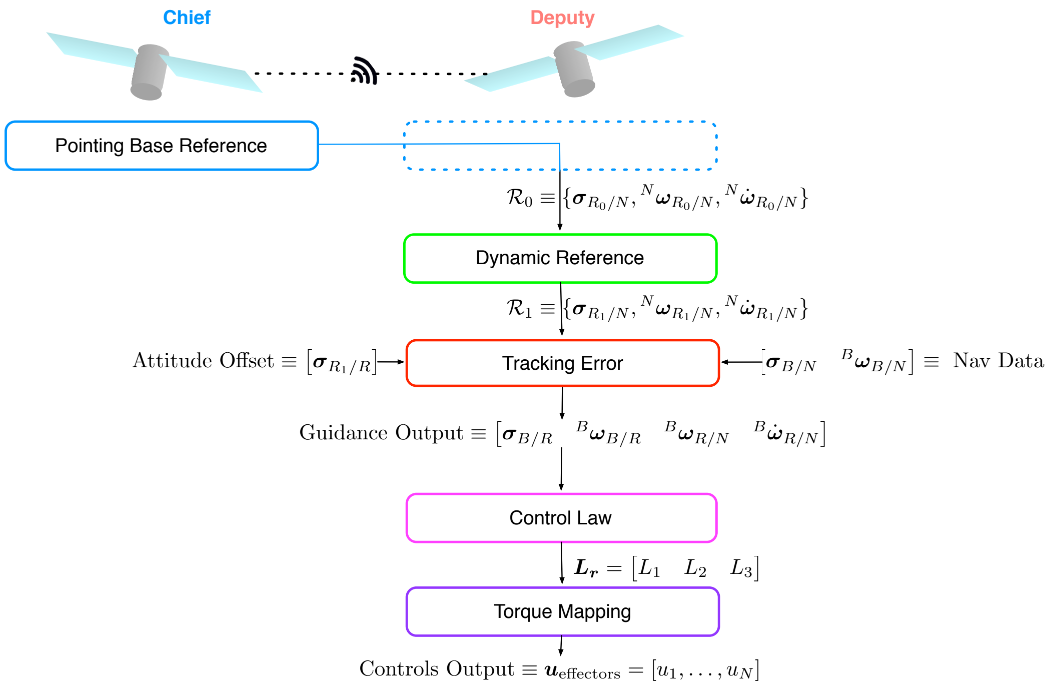

Chief
Deputy
Pointing Base Reference
$\mathcal{R}_0 \equiv \{\boldsymbol{\sigma}_{R_0/N}, {}^{N}\boldsymbol{\omega}_{R_0/N}, {}^{N}\dot{\boldsymbol{\omega}}_{R_0/N}\}$
Dynamic Reference
$\mathcal{R}_1 \equiv \{\boldsymbol{\sigma}_{R_1/N}, {}^{N}\boldsymbol{\omega}_{R_1/N}, {}^{N}\dot{\boldsymbol{\omega}}_{R_1/N}\}$
Attitude Offset $\equiv [\boldsymbol{\sigma}_{R_1/R}]$
Tracking Error
$[\boldsymbol{\sigma}_{B/N} \quad {}^{B}\boldsymbol{\omega}_{B/N}] \equiv$ Nav Data
Guidance Output $\equiv [\boldsymbol{\sigma}_{B/R} \quad {}^{B}\boldsymbol{\omega}_{B/R} \quad {}^{B}\boldsymbol{\omega}_{R/N} \quad {}^{B}\dot{\boldsymbol{\omega}}_{R/N}]$
Control Law
$\boldsymbol{L}_r = [L_1 \quad L_2 \quad L_3]$
Torque Mapping
Controls Output $\equiv \boldsymbol{u}_{\text{effectors}} = [u_1, \dots, u_N]$

Figure 3. Guidance and Control Stacks for the Deputy.

Figure 3 illustrates the guidance and control stacks for the deputy spacecraft, which is the one taking contributions from the chief to ensemble the compounded final reference $\mathcal{R}$. The final deputy reference is computed through addition of a base pointing reference (i.e. definition of $\mathcal{R}_0/\mathcal{N}$, which is received from the chief), a dynamic reference motion relative to the base (i.e. definition of $\mathcal{R}_1/\mathcal{R}_0$, which results in the output of $\mathcal{R}_1/\mathcal{N}$), and an attitude offset (i.e. definition of $\mathcal{R}_1/\mathcal{R}$). The final reference state is then $\mathcal{R}/\mathcal{N}$. The tracking error module is always the last component in the guidance chain. This module reads the output of the latest reference module (which is $\mathcal{R}_1/\mathcal{N}$ for the case of Fig. 3), and adds an attitude offset if applicable (which is $\mathcal{R}/\mathcal{R}_1$ in Fig. 3). An attitude offset is added when the generated attitude reference is meant for a spacecraft fixed frame that is not the main one. The tracking error module also reads the current state of the main spacecraft body frame (spacecraft attitude $\boldsymbol{\sigma}_{B/N}$ and rate $\boldsymbol{\omega}_{B/N}$), as estimated from navigation. With the estimated navigation state $\mathcal{B}$ and the generated guidance state $\mathcal{R}$, both derived relative to a common inertial frame $\mathcal{N}$, tracking errors can be readily computed. The output of the tracking error module is also the final output of the whole guidance block and it is used to feed the control block next.

The deputy dynamic reference modules can be linear scanning motions, conic spiral motions, or other types of dynamic motions which are kinematically super-imposed on the current chief pointing solution. The mathematical details of such dynamic motions are discussed in Ref. 1 and are not repeated here. Note that any number of dynamic references modules could be sequentially chained to create increasingly complex rotational patterns. The key to this scalability is that, as shown in Fig. 3, all the reference generation modules (both pointing base and dynamic) output the same message structure defined in Eq. (1). The beauty of this modular approach is that each guidance module can be individually validated, verified and then integrated into a flight software suite for next phase of integrated testing. This makes it possible to readily update and enhance flight guidance solutions, without extensive re-testing, in a safe and systematic manner. Scaling capability in a controlled and robust way is of particular importance with the distributed guidance development discussed here, as the complex software implementation could otherwise become a mission risk.

SOFTWARE-HARDWARE TESTBED

The concept of performing distributed guidance maneuvers is demonstrated through a flexible and modular hardware-in-the-loop testbed. This testbed takes advantage of several modern software tools and hardware components to provide the space community with a flight system and test environment that is highly customizable, user-friendly and cost-effective, such that it can potentially suite a wide range of formation flying mission profiles. The software and hardware tools that are combined together are: the Basilisk astrodynamics software,[1, 21–24] the Black Lion communication architecture,[16] and the commercial processor of a Raspberry Pi. This section describes briefly each of these three components and their contribution to the proposed constellation scenarios.

Basilisk

Basilisk is an open-source, cross-platform, desktop testbed for designing flight algorithms and testing them in close-loop dynamics simulations. The Basilisk testbed is currently being built by the Autonomous Vehicle Systems (AVS) laboratory at the University of Colorado Boulder and the Laboratory for Atmospheric and Space Physics (LASP) in order to support an interplanetary spacecraft mission.

Basilisk seeks to capitalize on the potential of using Python as a user-interface language for prototyping and testing flight algorithm code that is actually written in C/C++. Figure 4 illustrates the nominal setup – but not necessary required – of a Basilisk desktop simulation. In this setup, C++ modules perform spacecraft physical simulation tasks (*SC Models* in Fig. 4) while C modules perform mission-specific guidance, navigation and control tasks (*FSW Algs* in Fig. 4). The C/C++ modules are then wrapped in Python for setup, desktop execution and post-processing.

During a simulation run, the different modules communicate with each other through a custom message passing interface (*MPI* in Fig. 4), which is also written in C/C++ and it is based on a publish-subscribe pattern. The beauty of using an MPI is that it allows a clear separation between the different processes – dynamics simulation process and FSW process for the setup in Fig. 4. This separation facilitates, later on, the migration of the FSW application into a different processor. As a matter of fact, Basilisk-developed flight algorithms have already been ported to different kind of

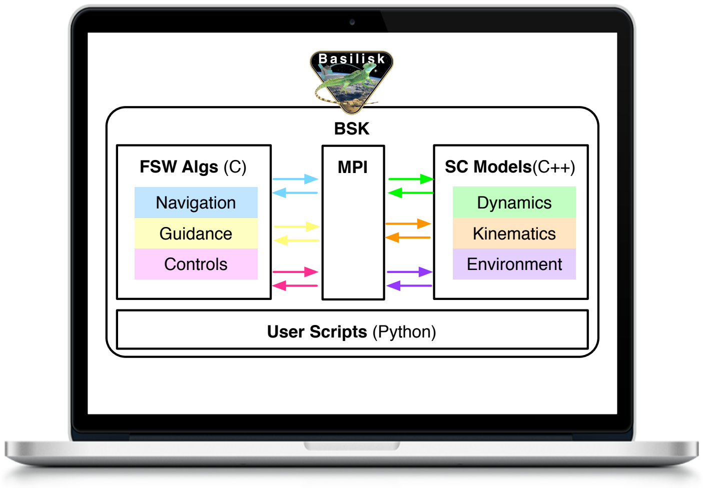


Figure 4. Basilisk (BSK) Desktop Environment

processor targets: commercial-off-the-shelf processors like the Raspberry Pi as well as radiation-hardened flight processors (i.e. embedded systems).

In this paper, Basilisk is used to test the validity of distributed guidance maneuvers within a formation. Several Basilisk processes are instantiated to simulate the physical behavior of each spacecraft as well as to execute each FSW suite. These processes are completely independent; indeed some of them run on different computing platforms and close the loop through TCP connection. The cross-platform nature of Basilisk is exploited in the numerical simulations to include commercial flight processors in the loop. It is important to clarify that once the validity of distributed guidance maneuvers is proved numerically, this proof still holds out of the Basilisk environment. The use of Basilisk has been the author's choice for convenience.

Black Lion

Black Lion (BL) is a purely software-based communication architecture for integrated testing of independent mission models or applications. Black Lion is architected to be reconfigurable and scalable, allowing for any number of heterogeneous software models, across one or multiple computing platforms, to be integrated into a single spacecraft simulation. This architecture enables, for instance, seamless integration of legacy software models that in principle were never designed to work together.

The communication aspects that BL is responsible for are: translation and transport of data between models as well as synchronization of all the applications to ensure they remain in lock-step. Furthermore, the aforementioned communication goals are achieved while being as transparent as possible to the internals of the models; an abstracted communication layer is achieved within BL by means of a unique central controller and two generic APIs attached to each of the applications/models. These APIs are currently implemented for software applications whose heterogeneity spans from: multithreaded vs. single-threaded, asynchronous vs. synchronous, little-endian vs. big endian, as well as for a variety of programming languages: Python, C, C++ and C#. While the BL development is currently motivated for an interplanetary mission to emulate a flat-sat configuration, the system is being built under the principles of reusability and scalability, and the BL applications

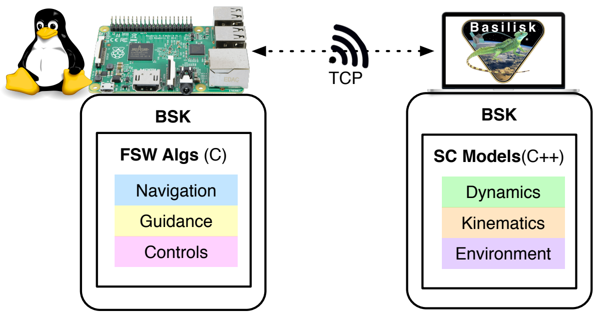


Figure 5. Basilisk FSW on the Raspberry Pi

extend beyond the flat-sat scenario. Examples of potential BL applications include formation flying scenarios with large clusters of spacecraft communicating with each other.

In the present work, the simulation setup encompasses independent Basilisk processes running on separate computing platforms and communicating with each other via TCP connections. The transfer of data and synchronization between applications is handled through the Black Lion communication architecture.

Commercial Processors

Of particular interest is the use of the Raspberry Pi as a flight target for FSW suites; being small, powerful, affordable and with large community support, the Raspberry Pi is a very attractive candidate for low-cost space exploration provided that it can operate reliably in space. Figure 5 illustrates the migration of Basilisk-developed flight algorithms into the Raspberry Pi.

The Pi platform has a built-in ARM processor and comes with the Linux OS out- of-the-box. Since Basilisk is cross-platform in nature, a regular Basilisk FSW process runs readily on the Raspberry Pi. The idea of targeting Basilisk-developed flight algorithms into the Raspberry Pi was already assessed in Ref. 24. However, the distributed architecture used in Ref. 24 was not scalable to a constellation of multiple satellites –it was a straight peer-to-peer communication. Now, by incorporating the Black Lion communication architecture into the scheme, formation flying scenarios like the one conceived in the present work can be addressed.

DISTRIBUTED GUIDANCE: SIMULATION SETUP

Figure 6 depicts the setup that will be used to test distributed, relative guidance on a spacecraft formation. The formation in Fig. 6 involves, for the sake of simplicity, one chief spacecraft and one deputy spacecraft. Let us remark upfront that there is no reason why this setup could not be scaled into a bigger constellation. Having said that, a chief-and-deputy constellation is already sufficient to prove the concept of distributed guidance. In the current setup, each spacecraft has a separate FSW process and spacecraft (SC) physical simulation process. For the chief (highlighted in blue color in Fig. 6), both processes run on the same computing platform. For the deputy (highlighted in orange color in Fig. 6), the SC simulation process runs on a desktop computer while the FSW process runs on a separate commercial processor, the Raspberry Pi. Note that each SC physical simulation is conformed by dynamic, kinematic and environment (DKE) models as well as spacecraft hardware models like sensors and actuators. In turn, each FSW process is conformed by navigation tasks

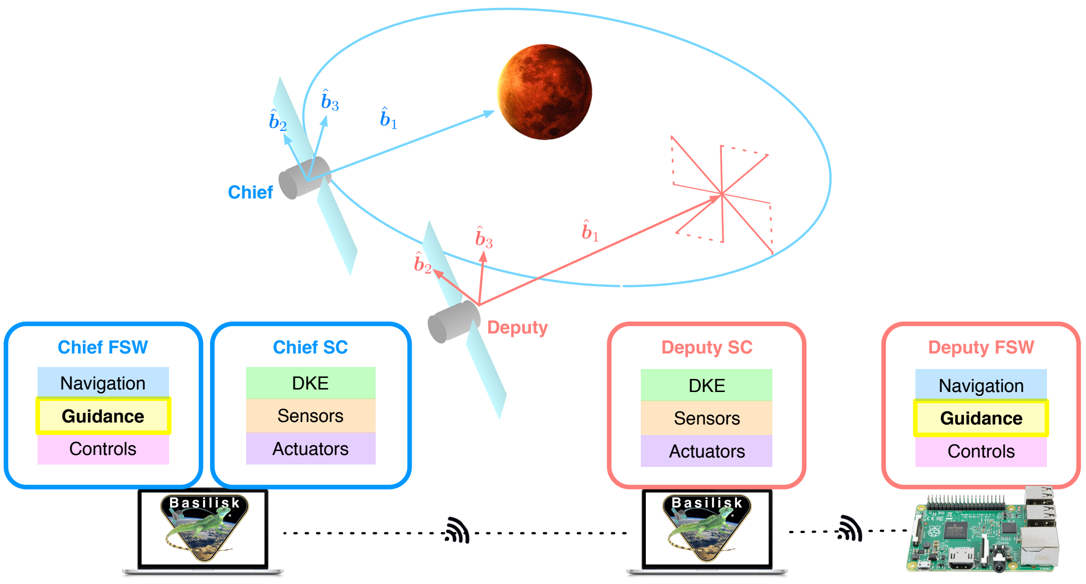


Figure 6. Distributed Guidance: Concept and Setup

(sensor processing, estimation, etc.), guidance tasks (reference generation, tracking error, etc.) and control tasks (control law, torque mapping, etc).

The concept of performing distributed guidance consists of the following: the chief FSW generates a base pointing reference that becomes an input to the deputy spacecraft (through hardware models on the deputy). This incoming base reference, which could be either sensed or received by the deputy spacecraft, is then transferred to the deputy FSW. Within the guidance task of the deputy FSW, the incoming reference is the base on top of which dynamic relative motions are superimposed. In the orbit of Fig. 6, the base reference is seen in the alignment of both spacecrafts while the relative dynamic motion of the deputy is represented by the asterisk scanning pattern drawn in orange.

A distributed guidance maneuver could work in two different modes: it could be either commanded or sensed. In the commanded case, the deputy spacecraft receives the base guidance reference from the chief through a communication receiver using radio frequency. In the sensed case, the deputy spacecraft uses a relative attitude sensor in order to figure out the current attitude and angular rate of the chief spacecraft. This paper focuses on distributed commanded guidance, while the sensed counterpart is considered potential future work.

Numerical Simulation: Commanded Distributed Guidance

The numerical simulation corresponds to a distributed commanded guidance maneuver, involving one chief spacecraft and one deputy spacecraft actively communicating data with each other. This maneuver is to be performed in the context of the Mars' exploration scenario illustrated earlier in Fig. 1(a) . The chief spacecraft keeps a constant Mars-nadir pointing while in orbit. The deputy spacecraft aligns with the current chief's pointing and performs an asterisk scanning relative to the time-varying base. A more detailed version of the simulation setup is illustrated in Fig. 7. Here, the connections between the modules of the different processes/applications are specified.

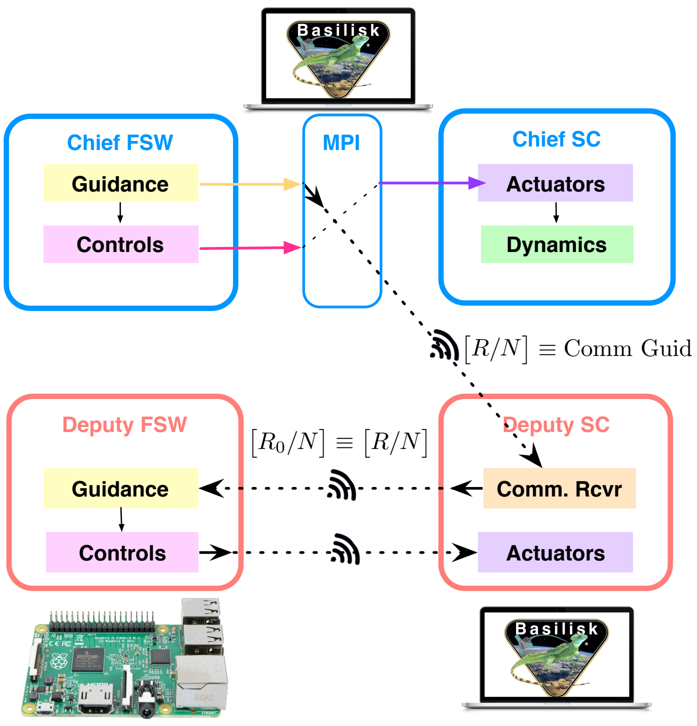


Figure 7. Simulation Setup

Let us first focus on the chief simulation part (corresponding to the blue boxes in Fig. 7). Both the chief FSW and chief SC simulation run on the same computer platform. These two processes communicate with each other through the Basilisk message passing interface (MPI in Fig. 7). Within the chief FSW, the guidance task computes the pointing base reference for Mars-nadir tracking. This base reference, which is the final desired one for the chief, is used downstream in the control task to compute the required control toque. This control torque is stored in the MPI and communicated to the chief SC, where the torque is through the onboard actuators (reaction wheels, in particular). In the presented numerical simulation, the chief spacecraft is initially tumbling. After an initial detumbling phase, the spacecraft is driven to the time-varying Mars pointing reference.

Let us now describe the deputy simulation part (corresponding to the orange boxes in Fig. 7). Here the deputy FSW runs on the Raspberry Pi platform and closes the loop, through TCP connection, with the deputy SC simulation running on a different platform. The deputy SC models include a communication receiver, which reads as input the Mars pointing reference coming from the chief. In Fig. 7, the commanded reference sent by the chief and received by the deputy is labeled as $[R/N]$. The nomenclature R is used because, for the chief, this corresponds to its final desired reference. In the deputy, the received information is transmitted from the the receiver to the onboard FSW for processing. Note that in Fig. 7, this information is labeled as $[R_0/N] \equiv [R/N]$. This nomenclature is used because the chief's final reference $[R/N]$ is, for the deputy, the base reference $[R_0/N]$ on top of which a relative scanning pattern is to be superimposed. Within the deputy FSW, the guidance task is responsible to build up the relative dynamic reference and communicate the final, compounded reference to the controls task. The controls task computes the required control torque to align the deputy SC with the desired reference. This torque is then communicated to the deputy physical SC and applied through the modeled actuators.

Table 1. Chief's Mars Orbit Elements

Parameter	Value	Units
Semi-Major Axis	34895.823	km
Eccentricity	0.332	
Inclination	0.554	rad
Longitude of Ascendant Node	1.599	rad
Argument of Perigee	4.565	rad
Initial true Anomaly	2.303	rad

Numerical Setup for the Chief

In the numerical simulations of this paper, the chief is modelled as a rigid spacecraft with four reaction wheels (RWs). The associated differential equations of motion (EOM) are the following:

$$[I]\dot{\boldsymbol{\omega}}_{B/N} = -[\tilde{\boldsymbol{\omega}}_{B/N}]\left([I]\boldsymbol{\omega}_{B/N} + [G_s]\boldsymbol{h}_s\right) - [G_s]\boldsymbol{u}_s + \boldsymbol{L} \tag{2}$$

where $[I]$ is the spacecraft inertia tensor, $\boldsymbol{L}$ is an external torque and $\boldsymbol{u}_s$ is the set of RW motor torques. Given the EOM in Eq. (2), the control law implemented is a novel MRP steering strategy that is globally asymptotically stabilizing.[27]

The chief's orbit parameters are provided in Table 1. These parameters are needed in order to reproduce the numerical simulations of this paper, since the nadir pointing module uses the relative position between the chief spacecraft and the target planet in order to compute the corresponding base attitude reference. The body attitude and body rate of the chief spacecraft at the beginning of the simulation are provided in Eq. 3. The inertial frame chosen in the simulation is Mars-centered.

$$\boldsymbol{\sigma}_{B/N}(t=0) = \begin{bmatrix}0.4 & 0.2 & 0.1\end{bmatrix} \tag{3a}$$

$${}^{\mathcal{N}}\boldsymbol{\omega}_{B/N}(t=0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix} \text{ rad/s} \tag{3b}$$

The results of the chief numerical simulation will illustrate that the chief is initially not aligned with the desired Mars-nadir pointing reference. Note that the initial attitude conditions for the chief provided in Eq. 3 do not alter the distributed maneuver ensembled and performed by the deputy. Similarly, further configuration details of the chief vehicle and its control parameters are not provided because they are not relevant to the distributed guidance maneuver –the actual pointing of the chief spacecraft does not alter the desired nadir-pointing reference generated in the guidance task. Guidance results for the chief are shown, later on, simply to emphasize on the Mars exploration science scenario that is depicted in Fig. 1(a) and numerically simulated here.

Numerical Setup for the Deputy

The deputy spacecraft is modelled as a rigid-body as well but does not have reaction wheels integrated as actuators. Instead it is assumed the commanded control torque from FSW is readily applied as an external torque $\boldsymbol{L}$. The associated differential equations of motion (EOM) for the deputy are the following:

$$[I]\dot{\boldsymbol{\omega}}_{B/N} = -[\tilde{\boldsymbol{\omega}}_{B/N}][I]\boldsymbol{\omega}_{B/N} + \boldsymbol{L} \tag{4}$$

The deputy spacecraft model is kept minimalistic in order to simplify the message connections between the different applications. As shown in Fig. 7, there are four Basilisk processes communicating with each other (chief FSW, chief spacecraft, deputy FSW and deputy spacecraft). The

Table 2. Deputy Control Parameters

Parameter	Value	Units
Attitude Error Gain K	3.5	$\frac{\text{kg}\cdot\text{m}^2}{\text{s}}$
Rate Error Gain P	30.0	$\frac{\text{kg}\cdot\text{m}^2}{\text{s}}$
I_1, I_2, I_3	600	kg$\cdot$m^2

exchange of messages between applications happens through Black Lion by publications and subscriptions.[16] Hence, it is critical that all the modules in each Basilisk process have unique message names. In this sense, simplifying the deputy spacecraft avoids setup overhead.

The rigid body equations of motion in Eq. (4) are numerically simulated within Basilisk in the deputy spacecraft simulation. Given the EOM in Eq. (4), the control law implemented is an MRP feedback law that is globally asymptotically stabilizing:

$$\boldsymbol{L} = -K\boldsymbol{\sigma}_{B/R} - [P]\boldsymbol{\omega}_{B/R} + \boldsymbol{\omega}_{R/N} \times [I]\boldsymbol{\omega}_{B/N} - [I](\boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{R/N} + \dot{\boldsymbol{\omega}}_{R/N}) \tag{5}$$

The simulated navigation data $\boldsymbol{\sigma}_{B/N}$ and $\boldsymbol{\omega}_{B/N}$ is without any sensor corruptions to better illustrate that the control law does achieve asymptotic tracking of the reference motion. Note that the navigation data is an input to the tracking error module. Table 2 presents the control-related parameters of the deputy FSW. The orbital elements for the deputy are not included because they are not critical to the simulation: they do not effect the attitude maneuver being performed. The deputy is simply inserted in a generic orbit nearby the chief. The body attitude and body rate of the deputy spacecraft at the beginning of the simulation are provided in Eq. 6. The inertial frame is also Mars-centered –as in the chief spacecraft simulation.

$$\boldsymbol{\sigma}_{B/N}(t=0) = \begin{bmatrix} 0.4 & 0.2 & 0.1 \end{bmatrix} \tag{6a}$$

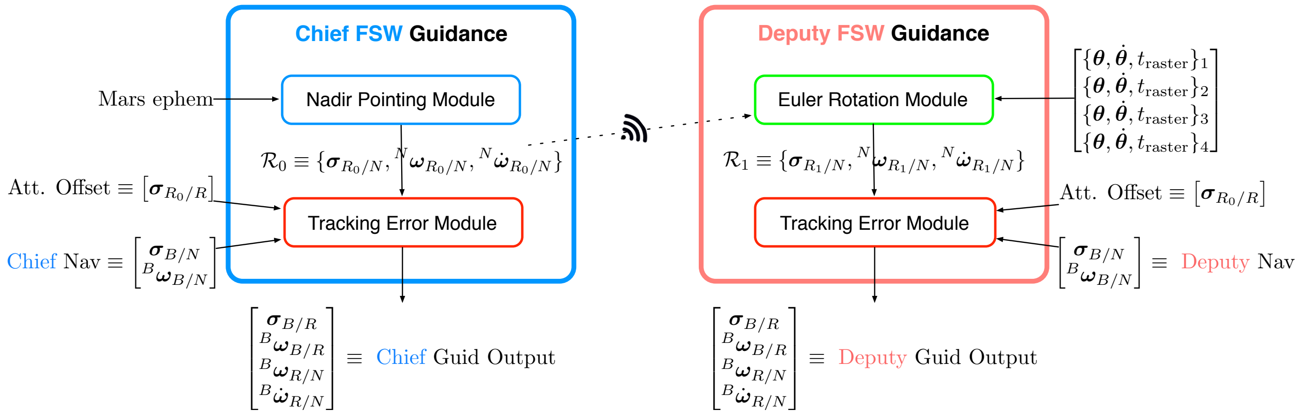
$$^{\mathcal{N}}\boldsymbol{\omega}_{B/N}(t=0) = \begin{bmatrix} 0.0001 & 0.0002 & 0 \end{bmatrix} \text{ rad/s} \tag{6b}$$

Guidance Results

The stack of guidance modules used to generate the chief's final reference (nadir Mars pointing) and the deputy's final reference (asterisk scanning relative to the chief's pointing) are depicted in Fig. 8.

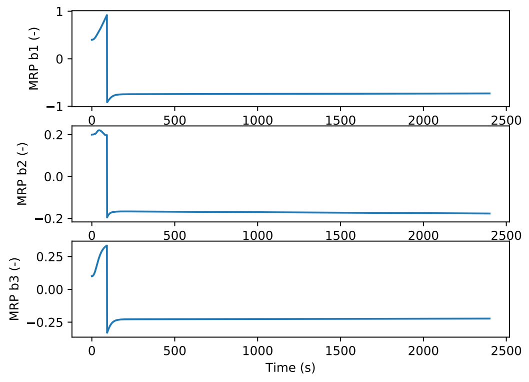
The guidance stack for the chief (highlighted with a blue box) is examined first in detail. The guidance goal for the chief is to keep pointing towards the center of Mars while orbiting about the planet in a generic orbit. The final reference for the chief is a base reference; hence, it is created by simply combining the nadir pointing base module and the attitude tracking error module. Among the modules developed in Ref. 1, there are a couple of modules than can achieve nadir pointing for any planet: either the constrained two-body pointing or the Hill orbit pointing module. In the present manuscript, these modules will simply be referred to as nadir pointing modules. The simulation results for the chief spacecraft are shown in Fig. 9. These results include: the evolution of the spacecraft body attitude using Modified Rodrigues Parameter (MRP) descriptions[26] in Fig. 9(a), and the pointing direction of the onboard Mars instrument in Fig. 9(b). Note that initial attitude and rate for the chief (as provided in Eq. 3) are far apart from the desired nadir reference.

In turn, the guidance goal for the deputy is to perform an asterisk scanning pattern relative to chief's nadir pointing reference. As thoroughly described in Ref. 1, an asterisk scanning pattern

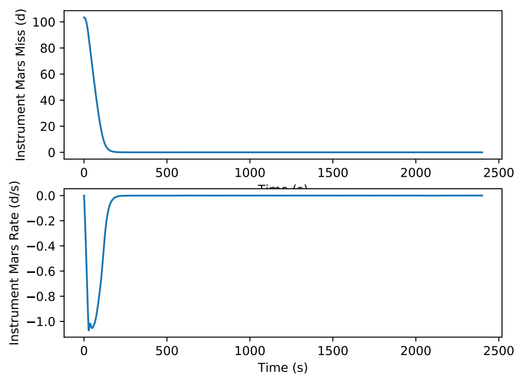


Chief FSW Guidance
Nadir Pointing Module
Tracking Error Module
Mars ephem
$\mathcal{R}_0 \equiv \{\boldsymbol{\sigma}_{R_0/N}, {}^N\boldsymbol{\omega}_{R_0/N}, {}^N\dot{\boldsymbol{\omega}}_{R_0/N}\}$
Att. Offset $\equiv [\boldsymbol{\sigma}_{R_0/R}]$
Chief Nav $\equiv \begin{bmatrix}\boldsymbol{\sigma}_{B/N}\\ {}^B\boldsymbol{\omega}_{B/N}\end{bmatrix}$
$\begin{bmatrix}\boldsymbol{\sigma}_{B/R}\\ {}^B\boldsymbol{\omega}_{B/R}\\ {}^B\boldsymbol{\omega}_{R/N}\\ {}^B\dot{\boldsymbol{\omega}}_{R/N}\end{bmatrix} \equiv$ Chief Guid Output
Deputy FSW Guidance
Euler Rotation Module
Tracking Error Module
$\begin{bmatrix}\{\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, t_{\text{raster}}\}_1\\ \{\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, t_{\text{raster}}\}_2\\ \{\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, t_{\text{raster}}\}_3\\ \{\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, t_{\text{raster}}\}_4\end{bmatrix}$
$\mathcal{R}_1 \equiv \{\boldsymbol{\sigma}_{R_1/N}, {}^N\boldsymbol{\omega}_{R_1/N}, {}^N\dot{\boldsymbol{\omega}}_{R_1/N}\}$
Att. Offset $\equiv [\boldsymbol{\sigma}_{R_0/R}]$
$\begin{bmatrix}\boldsymbol{\sigma}_{B/N}\\ {}^B\boldsymbol{\omega}_{B/N}\end{bmatrix} \equiv$ Deputy Nav
$\begin{bmatrix}\boldsymbol{\sigma}_{B/R}\\ {}^B\boldsymbol{\omega}_{B/R}\\ {}^B\boldsymbol{\omega}_{R/N}\\ {}^B\dot{\boldsymbol{\omega}}_{R/N}\end{bmatrix} \equiv$ Deputy Guid Output

Figure 8. Stack of Guidance Modules for the Chief and Deputy



(a) Spacecraft Main Body-Frame Attitude

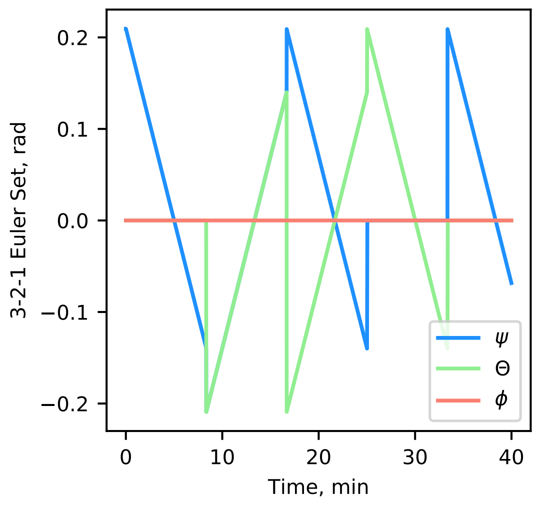


(b) Mars Instrument Pointing

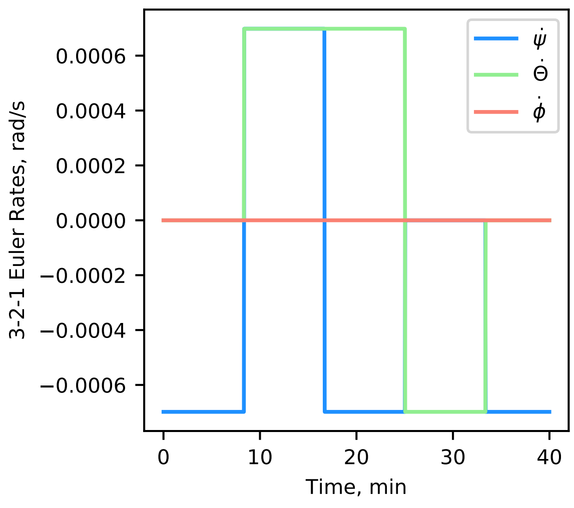
Figure 9. Chief Spacecraft Attitude States

is conformed by a total of four raster lines. This means that, using the Euler rotation module (also described in Ref. 1), the complete pattern can be achieved with only four command steps. Each command is conformed by a 3-2-1 Euler angle set $\boldsymbol{\theta} : \{\psi_0, \theta_0, \phi_0\}$, a 3-2-1 Euler angle rate $\dot{\boldsymbol{\theta}} : \{\dot{\psi}, \dot{\theta}, \dot{\phi}\}$, and the time allocated to complete each the raster. The angle set $\boldsymbol{\theta}$ describes the attitude where the raster line begins (relative to the base), and the angle rates $\dot{\boldsymbol{\theta}}$ define which of these relative angles vary and how rapidly.

As illustrated in Figure 8, the deputy spacecraft receives the base attitude states from the chief and superimposes the constant Euler angle rate motions on top, in order to achieve the desired relative scanning. These relative Euler angle commands are shown in Fig. 10. Figure 11(a) illustrates what these commands look like in terms of raster lines. The raster lines of scientific interest are marked in thick, continuous, orange lines, and they have an angular size of $\alpha = 8$ degrees as marked by the orange dots. Note that the raster lines displayed in Fig. 11(a) are also relative to the time-varying base reference –this is why the commanded pattern looks indeed like an asterisk. There is a total of four raster lines (one vertical, one horizontal and two diagonal ones), and they are numbered in the order they are to be scanned. The start and end of each raster line is also indicated. In order to

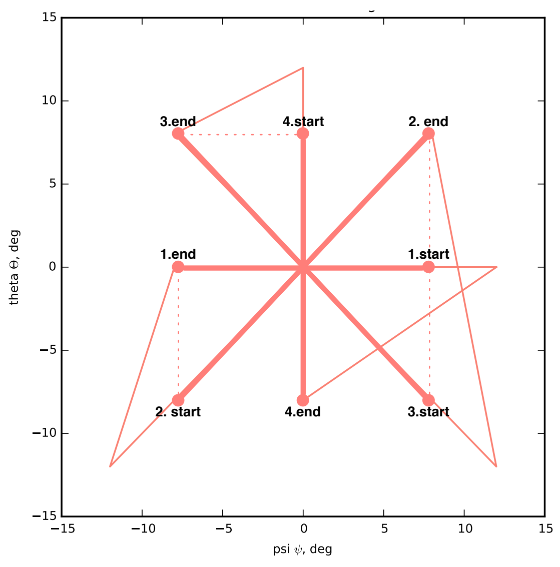


(a) Relative Euler Angle Set

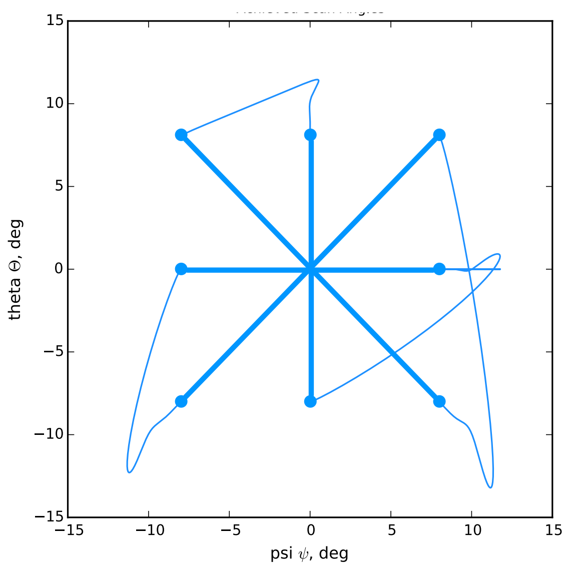


(b) Relative Euler Angle Rates

Figure 10. Deputy FSW: Euler Commands



(a) Commanded Relative Scanned Angles

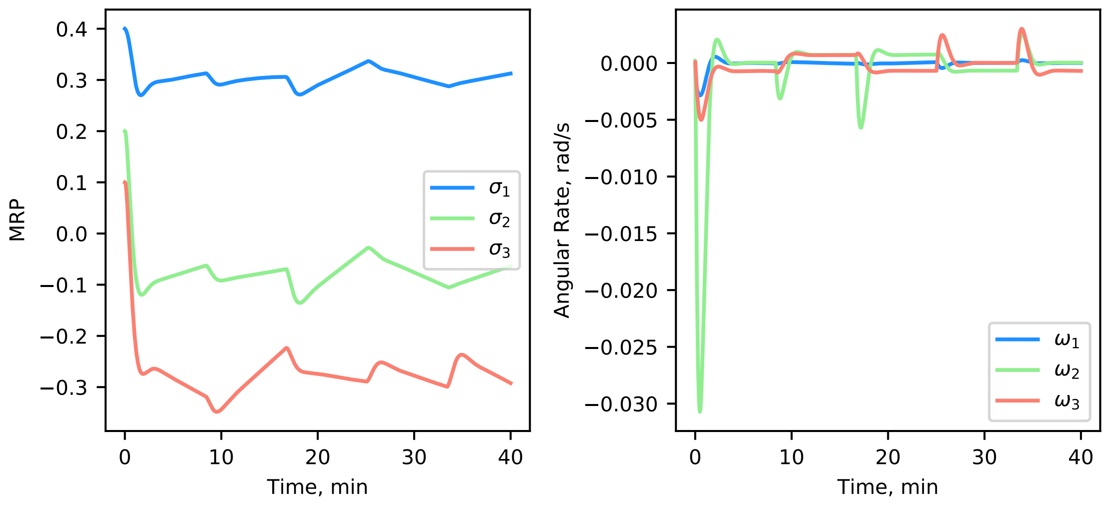


(b) Achieved Relative Scanned Angles

Figure 11. Asterisk Scanning: nominal raster size of 8 degrees.

get into the desired rasters on time, a small angle offset needs to be included in the command. This offset is also depicted in Fig. 11(a), and corresponds to the thin, continuous orange line preceding the start point of each raster. On a side note, the specific values for each Euler command, including the angle offset, for a nominal raster size of $\alpha = 8$ degrees have been taken from Ref. 1. Hence, the reader is referred there if he or she desires to replicate the relative asterisk scanning pattern.

If Fig. 11(a) shows the deputy FSW commands as computed on the Raspberry Pi, then Fig. 11(b) displays the performance of the deputy spacecraft simulation running on a separate platform. Note that the spacecraft converges into the nominal start points on time for each raster (marked with blue



(a) Attitude History (b) Body Angular Velocity Coordinates

Figure 12. Deputy Spacecraft Attitude States.

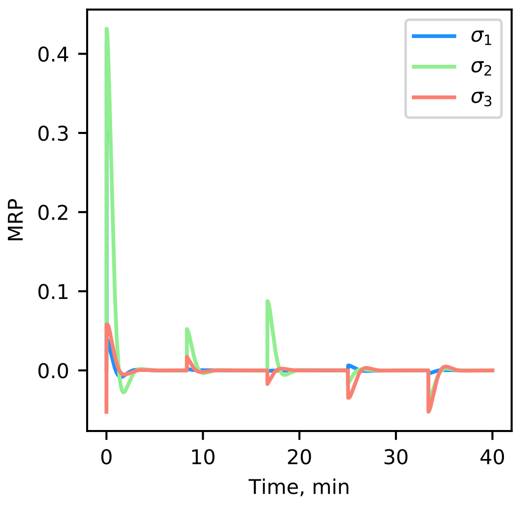
dots). It is important to clarify that the asterisk pattern in Fig. 11(b) only looks like an asterisk because its displayed relative to the time-varying Mars-pointing reference (i.e. as if this pointing reference was inertial). With respect to the true inertial frame used in the simulation, the achieved pattern is barely recognizable, which is expected.

The absolute body attitude and rates for the deputy spacecraft are shown in Fig. 12. The behavior of the deputy spacecraft in terms of body angular rate, as displayed in Fig. 12(b), is very representative of the compounded maneuver being performed. The initial peak is the largest since the deputy spacecraft is initially tumbling when the first raster command is requested. The following peaks correspond to the request of the next raster lines. Note that there is a total of five peaks despite the full asterisk maneuver can be completed with only four commands. This is because at the end of the fourth and last raster, the deputy spacecraft is smoothly driven back to the beginning of the first raster again.

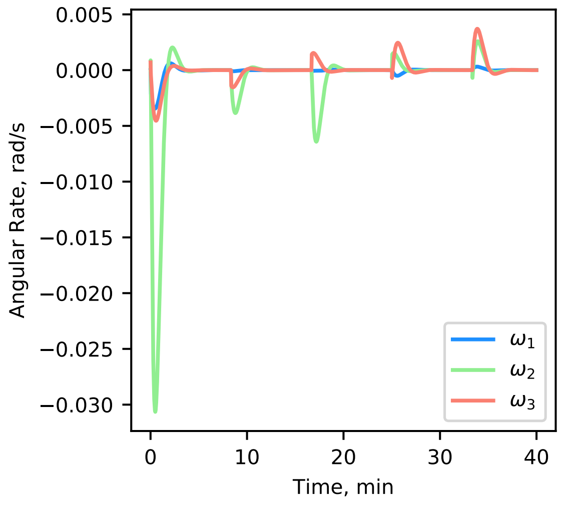
Finally, Fig. 13 displays the control torque computed by the deputy FSW as well as the absolute tracking error. The convergence of the tracking errors to zero in Fig. 13 shows that building a guidance reference in a distributed fashion does not introduce anomalies. As a matter of fact, convergence to zero in the tracking errors would not be eventually achieved if there was mathematical conflict among the attitude, angular rate and angular acceleration that are final output of the deputy's guidance task, or if there was an error in the superposition approach.

CONCLUSIONS

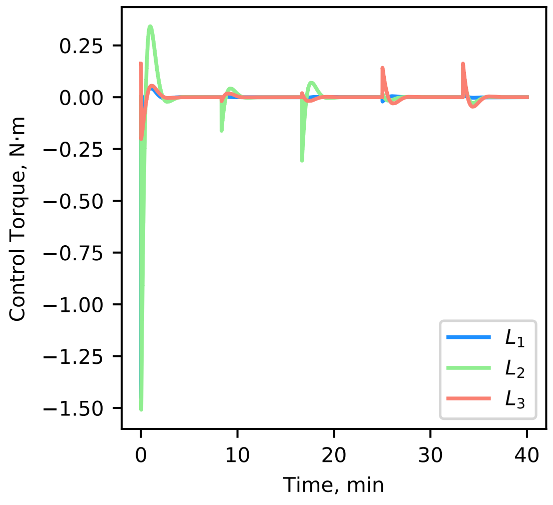
This paper has presented a distributed strategy for 1) performing relative attitude guidance within a constellation of satellites and 2) for testing the constellation behavior in a realistic manner that includes hardware in the loop. The software-hardware framework in which the distributed guidance strategy has been tested takes advantage of the Basilisk software testbed and the Black Lion communication architecture, in order to replicate exactly how such a system would fly –holding to the long-held NASA principle of "test what you fly, fly what you test". Of particular interest is the use



(a) Attitude Tracking Error



(b) Rate Tracking Error



(c) Commanded Control Torque

Figure 13. Deputy FSW: Tacking Error and Computed Torque

of Black Lion, which allows distributed communication among different processes and computing platforms, and also regulates the exchange of messages via publish and subscribe. In the present work, Black Lion has been critical to identify that the message connections between the different members in the constellation were set up properly and to check that there were no undesired connections under the hood. The combined facts of architecting guidance maneuvers and realistically testing them, as shown in this manuscript, offers a robust and flexible manner to extend single-spacecraft missions into constellation of satellites in order to increase the science return. Further, it has been shown that the extension of single-spacecraft missions into constellations does not necessarily require the development of new flight algorithm code. The key to this is modular architecture of flight algorithms even on single-spacecraft missions. As it has been proved for the particular case of attitude guidance, a layered approach of building a reference onboard a single spacecraft still holds for relative guidance among a constellation, preserving the same full kinematic properties.

REFERENCES

[1] M. Cols-Margenet, H. Schaub, and S. Piggott, “Modular Attitude Guidance: Generating Rotational Reference Motions for Distinct Mission Profiles,” *Journal of Aerospace Information Systems*, Vol. 15, No. 6, June 2018.

[2] S. Nag, “Autonomous Scheduling of Agile Spacecraft Constellations for Rapid Response Imaging,” tech. rep., NASA Ames Research Center / Bay Area Environmental Research Institute, June2018.

[3] J. Christian, “Relative Navigation Using Only Intersatellite Range Measurements,” *Journal of Spacecraft and Rockets*, September 2016.

[4] R. Alonso, J. Crassidis, and J. Junkins, “Vision-Based Relative Navigation for Formation Flying of Spacecraft,” *AIAA*, 2000.

[5] M. Marszalek, O. Kurz, M. Drentschew, M. Schmidt, and K. Schilling, “Intersatellite Links and Relative Navigation: Pre-conditions for Formation Flights with Pico- and Nanosatellites,” *18th IFAC World Congress*, Milano, Italy, August 2011.

[6] D. Scharf, F. Hadaegh, and S. Ploen, “A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control,” *American Control Conference*, June 2006.

[7] N. Mastrodemos, D. Kubitschek, and S. Synnott, *Deep Impact Mission: Looking Beneath the Surface of a Cometary Nucleus*, ch. Autonomous Navigation for the Deep Impact Mission Encounter with Comet Tempel 1, pp. 95–121. Springer Netherlands, 2005.

[8] S. Bhaskaran, J. E. Riedel, and S. P. Synnott, “Autonomous target tracking of small bodies during flybys,” *AAS/AIAA Spaceflight Mechanics Meeting*, Maui, Hawaii, United States, Feb 8–12 2004.

[9] T. Kominato, M. Matsuoka, M. Uo, and J. Kawaguchi, “Optical Hybrid Navigation and Station Keeping around Itokawa,” *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, August 2006.

[10] R. Olds, A. May, C. Mario, R. Hamilton, C. Debrunner, and K. Anderson, “The Application of Optical Tracking to OSIRIS-REx Asteroid,” *AAS*, 2015.

[11] a. O. J. Nickolaos Mastrodemos and B. Rush, “Optical Navigation for the Rosetta mission,” *Annual AAS Guidance and Control Conference*, Breckenridge, CO, 2015.

[12] A. C. e. al., “Binary and Multiple Systems,” 2019.

[13] O. S. e. a. w. L. Benner and D. Scheeres), “Radar Imaging of Binary Near-Earth Asteroid (66391) 1999 KW4,” *Science*, 2006.

[14] D. Richardson and K. Walsh, “Binary Minor Planets,” *The Annual Review of Earth and Planetary Science*, 2005.

[15] J. Donnison, “The Hill stability of binary asteroid and binary Kuiper Belt systems,” *Monthly Notices of the Royal Astronomical Society*, July 2011.

[16] M. Cols Margenet, P. W. Kenneally, H. Schaub, and S. Piggott, “Simulation Of Heterogeneous Spacecraft And Mission Components Through The Black Lion Framework,” *John L. Junkins Dynamical Systems Symposium*, No. 7, College Station, TX, May 20–21 2018.

[17] D. Lauretta, *OSIRIS-REx Asteroid Sample-Return Mission*, Vol. Handbook of Cosmic Hazards and Planetary Defense. Springer, 2015.

[18] M. Mangieri and J. Vice, “Kedalion: NASA’s Adaptable and Agile Hardware/Software Integration and Test Lab,” *AIAA SPACE*, 2011.

[19] C. DeGraw and M. Holzinger, “A Massive-Scale Satellite Constellation Hardware-in-the-Loop Simulator and its Applications,” *9th International Workshop on Satellite Constellations and Formation Flight (IWSCFF)*, Boulder, Colorado, June 2017.

[20] S. Busch, P. Bangert, S. Dombrovski, and K. Schilling, “UWE-3, in-orbit performance and lessons learned of a modular and flexible satellite bus for future pico-satellite formations,” *Acta Astronautica*, Vol. 117, December 2015, pp. 73–89.

[21] J. Alcorn, H. Schaub, S. Piggott, and D. Kubitschek, “Simulating Attitude Actuation Options Using the Basilisk Astrodynamics Software Architecture,” *67th International Astronautical Congress*, Guadalajara, Mexico, Sept. 26–30 2016.

[22] S. Piggott, J. Alcorn, M. C. Margenet, P. W. Kenneally, and H. Schaub, “Flight Software Development Through Python,” *2016 Workshop on Spacecraft Flight Software*, JPL, California, Dec. 13–15 2016.

[23] J. Wood, M. Cols-Margenet, P. Kenneally, H. Schaub, and S. Piggott, “Flexible Basilisk Astrodynamics Visualization Software Using the Unity Rendering Engine,” *AAS Guidance and Control Conference*, Breckenridge, CO, February 2–7 2018.

[24] M. Cols Margenet, H. Schaub, and S. Piggott, “Modular Platform for Hardware-in-the-Loop Testing of Autonomous Flight Algorithms,” *International Symposium on Space Flight Dynamics*, Matsuyama-Ehime, Japan, June 3–9 2017.

[25] P. Tsiotras and J. M. Longuski, “A New Parameterization of the Attitude Kinematics,” *Journal of the Astronautical Sciences*, Vol. 43, No. 3, 1996, pp. 243–262.

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

[27] H. Schaub and S. Piggott, “Speed-Constrained Three-Axes Attitude Control Using Kinematic Steering,” *AAS Guidance, Navigation and Control Conference*, Breckenridge, CO, February 2017.