Proceedings of the Canadian Society for Mechanical Engineering International Congress 2021
CSME Congress 2021
June 27-30, 2021, Charlottetown, PE, Canada

Attitude Determination and Control System Flight Software Development and Design

William Travis, Mike Alger, Ijaz Qureshi, Emily Shepherdson, Dr. Anton de Ruiter
Department of Aerospace Engineering
Ryerson University
Toronto, Canada
wtravis@ryerson.ca, malger@ryerson.ca, ijaz.qureshi@ryerson.ca, eshepherdson@ryerson.ca, aderuiter@ryerson.ca

***Abstract*—In this paper, the flight software development and design of the attitude determination and control system for the ESSENCE CubeSat mission is documented and discussed. The majority of the software is reused from the DESCENT mission, which was previously written by the RACS team, with strong inspiration from NGC Aerospace's work on the PROBA satellite program. The paper highlights the benefits of reusable software modules and methodologies between projects. Finally, it is encouraged that similar processes be considered by other CCP teams developing ADCS systems for future missions.**

***Index Terms*—ADCS, Attitude Control, Attitude Filter, CCP, CubeSat, CSA Flight Code, RACS, Real Time Operating System, Reaction Wheels, Requirements, Software Architecture .**

I. Introduction

The Educational Space Science and ENgineering Cubesat Experiment (ESSENCE) Mission is aligned with the Canadian CubeSat Project (CCP) sponsored by the Canadian Space Agency (CSA). The primary motivation for the ESSENCE mission is to train Highly Qualified Personnel (HQP) through the experience of designing, building, launching and operating a CubeSat. ESSENCE will be launched from the International Space Station (ISS) via the NanoRacks launching service and operated by the project team through the ground station located at York University.

One mission objective is to fly and operate a 3U-CubeSat to observe the impact of climate change and solar storms on the Earth's environment. Simultaneously, the spacecraft will be used to demonstrate novel attitude navigation and control algorithms for an under actuated system on-orbit. The details of the Attitude Determination and Control System (ADCS) specific mission requirements will be discussed in Section II. This work expands upon ADCS software design and testing methodologies used in a previous mission, as outlined in Section III, validating that these tools are replicable for varying mission objectives. Section IV will focus on the design, development and testing of the ADCS flight software for the ESSENCE mission and how flight code will be generated and implemented in the onboard computer's Real Time Operating System (RTOS). Preliminary results for key function modules are shown in Section V while future work and concluding remarks can be found in Section VI and VII respectively.

II. Mission Requirements

The ADCS specific mission requirements were developed to satisfy the main mission objectives as well as known good practices used by the Ryerson Attitude Control Systems (RACS) team. The naming convention can be defined as: **M** – mission requirement, **ADCS-SW** – ADCS software requirement and **ADCS-HW** – ADCS hardware requirement, **PER** – performance, **FUN** – function and **INT** - integration.

A. Applicable Mission Requirements

ADCS applicable mission requirements are as follows:

REQ-M-PER-001
- The spacecraft shall de-tumble within a timespan of 24 hours, it shall then autonomously switch to coarse pointing mode with magnetic torquers once the orbital estimator is operational and confirmed to be valid.

REQ-M-PER-002
- The system shall have attitude knowledge with a tolerance of 0.5 degrees RMS reported in terms of a quaternion.

REQ-M-PER-003
- The system shall have a pointing accuracy with tolerances of 0.5 degrees RMS and 0.01 degrees/s RMS.

REQ-M-FUN-001
- Attitude experiments shall be programmed as modes within the ADCS system.

B. ADCS Software and Hardware Requirements

From the requirements outlined in Section II-A, the ADCS software and hardware requirements were produced:

REQ-ADCS-SW-FUN-001
- Attitude experiments shall have a back up safety mode to intervene in maintaining safety limits and redundancy.

REQ-ADCS-SW-FUN-002
- The ADCS shall be configured to operate in a hybrid magnetic attitude control mode. Control torques shall be provided by continuous use of the magnetorquers and impulsive operations of the reaction wheels.

REQ-ADCS-SW-FUN-003
- The spacecraft shall report GPS data to the ground station, while using raw GPS measurements for orbital determination.

REQ-ADCS-SW-PER-001

- Actuators shall have the ability to be selectively enabled and disabled to support experiments and ADCS requirements.
  REQ-ADCS-SW-INT-001
- Required orbital knowledge shall be no better than USSTRATCOM TLEs.
  REQ-ADCS-HW-FUN-001
- The on-board computer shall be capable of executing the ADCS SW at a rate of 10 Hz.

III. Design Experience and Approach

Prior to the ESSENCE mission, the RACS team were key contributors to the De-orbiting Spacecraft using Electrodynamic Tethers (DESCENT) mission[1] that launched in November 2020. The process followed while developing DESCENT was based largely on the ideas conceived and published by NGC Aerospace for the PROBA program of autonomous satellites outlined in [2] and [3]. In order to test the ADCS flight software for this mission, the team developed an ADCS simulator within the MATLAB/Simulink environment. Additionally, working in Simulink allowed the team to convert flight code into C-code for direct implementation on the RTOS. With the same team leading the ADCS for ESSENCE, the focus can be on the development and testing of novel ADCS software by reusing a significant portion of the DESCENT simulator.

While certain modules require slight modifications, much of the simulator and ADCS SW can transfer from DESCENT seamlessly, such as: the International Geomagnetic Reference Field (IGRF) magnetic field model [4] [5] and the Sun [section 5.1.1 of 6] and Earth ephemeris models [section 5 of 7]. The major difference between the ESSENCE and DESCENT simulation environments are the actuators that are being used for attitude control. DESCENT operated using only magnetorquers while ESSENCE has included reaction wheels. The addition of these reaction wheels requires new actuator models and slight modifications to the simulation. Additional modifications needed for ESSENCE include upgrading the navigation filter from a fixed rigid body and developing new sensor models, i.e. two-axis Sun sensors, a fibre-optic gyroscope and Global Positioning System (GPS).

The development process of the ESSENCE on-board software proves the reusable design concept is feasible, efficient and reliable. Maintaining libraries of functions and unit tests allows for consistency in the development and implementation of software for future missions. This work is being shared for others who may be considering developing ADCS systems of their own.

IV. ADCS Architecture and Design Philosophy

The ADCS SW is developed within the MATLAB/Simulink environment, as it provides a high-level graphical language designed for the development and testing of control systems. It also allows for the creation of stand alone library blocks, which are developed and tested separately before being integrated with one another to create a more complex system. Simulink also provides means of generating portable and readable C/C++ code from these large models via the embedded coder tools. This conversion capability enables the ADCS SW to be easily implemented on most modern RTOS.

The ADCS simulator is divided into two primary functions, as seen in Figure 1. First, the Real World Software (RWSW), which consists of models for actuators, spacecraft dynamics, and sensors. Then the Onboard Software (OBSW), which is the flight software written in Simulink library blocks. To make the flight software simpler to develop and test, it is split into three key functions: Navigation (NAV), Guidance (GDC), and Control (CTL). NAV is the process used to determine where the spacecraft is located in orbit, it's orientation and all desired target orientations and is a combination of estimation theory and propagation of astronmical models. GDC is the process to calculate a trajectory or difference from the current attitude to a desired attitude. Finally, CTL is the process of firing actuators to accomplish the desired trajectory or orientation computed from guidance. Again as this is computed onboard a computer the analog to digital and digital conversion processes and their delays needs to be accounted for as well in the design process.

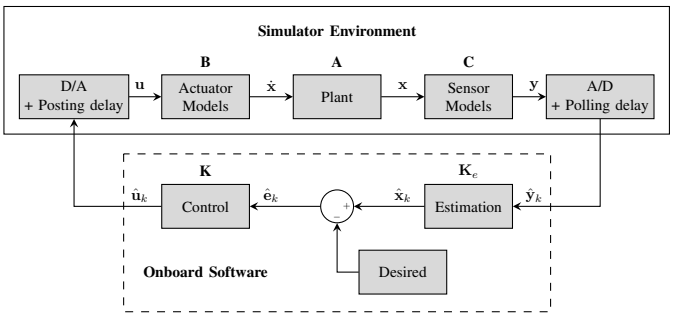

Fig. 1. Abstraction used to model simulation environment and on-board software.

A. Development Strategy and Testing Philosophy

Testing is planned to be done with respect to the requirements outlined in Section II and in ESSENCE's supporting technical notes. The standard Vee philosophy from software development practices fits well with the testing plan and it is shown in Figure 2.

Starting at the top left of the Vee and working down, it is expected that *User Requirements* have been delivered and digested into top level software requirements by the Preliminary Design Review (PDR). By the end of Critical Design Review (CDR), the *Architectural Design* and *Conceptual Design* tasks are complete. Also, by this time *Implementation* and *Unit Testing* of functions are to be well under way. Finally working back up the right side of the Vee, by the Flight Readiness Review (FDR), *Integration Tests*, *System Level Tests* and *System Acceptance Tests* shall have formal test plans outlined. This is to capture scenarios that need to be demonstrated in order to ensure that top level user software requirements and user requirements are met.

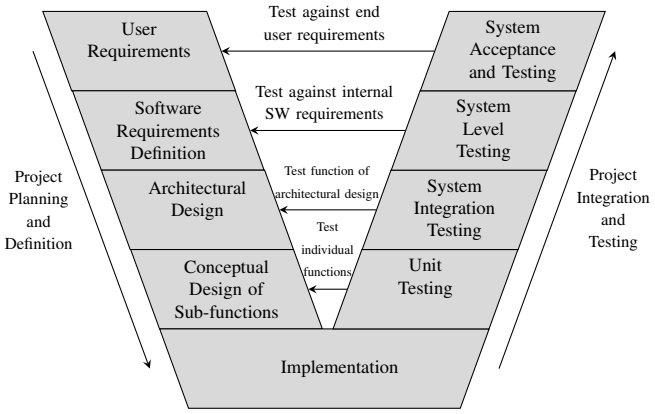

Fig. 2. Software development philosophy.

B. ADCS Control Modes

ADCS systems are seldom required to meet all of the mission requirements at one time, due to conflicting objectives. Therefore, one of the first design tasks is to plan out the modes of operation and how they will be toggled based on the mission and system requirements. For the ESSENCE mission, requirements were distilled into the modes shown in Figure 3 along with how the modes interface with eachother. A detailed description of each mode is as follows:

BDOT Mode

- Used to dampen the angular rates of the spacecraft to provide an acceptable starting point for the magnetic torquer nadir pointing mode.

BDOT with Bias Mode

- Used to align the spacecraft with the local magnetic field by providing a magnetic moment that will tend to align with the local magnetic field.

Quiescent Mode

- Used to provide a mode that will output no control signal to actuators, this is to allow for testing and tuning of the on-board Kalman filters or help identify sources of unexpected disturbance torques.

MTQ Nadir Pointing Mode

- Used to orient the spacecraft's long axis with the nadir vector through magnetorquer actuation only.

Angular Momentum Management Mode

- Used to prevent the reaction wheels from either saturation or constant zero crossing through magnetorquer actuation. This mode is typically augmented with an *active wheel* mode, but has the ability to be turned off or on separately.

3 Axis ORB Pointing Mode

- Used to point the Body Orientation Frame (BOF) axes relative to the instantaneous Orbital (ORB) axes through reaction wheel actuation only.

3 Axis WGS Pointing Mode

- Used to point the BOF axes relative to the instantaneous World Geodetic System (WGS) axes through reaction wheel actuation only.

3 Axis SUN Pointing Mode

- Used to orient the spacecraft's long axis with the relative Sun position vector through reaction wheel actuation only.

3 Axis ECI Pointing Mode

- Used to point the BOF axes relative to the instantaneous Earth Centered Inertial (ECI) axes through reaction wheel actuation only.

Experiment One Mode

- Used to perform the University of Toronto Institute for Aerospace Studies (UTIAS) experiment, both magnetic torquers and reaction wheels will be active.

Experiment Two Mode

- Used to perform the Ryerson University experiment, both magnetic torquers and reaction wheels will be active.

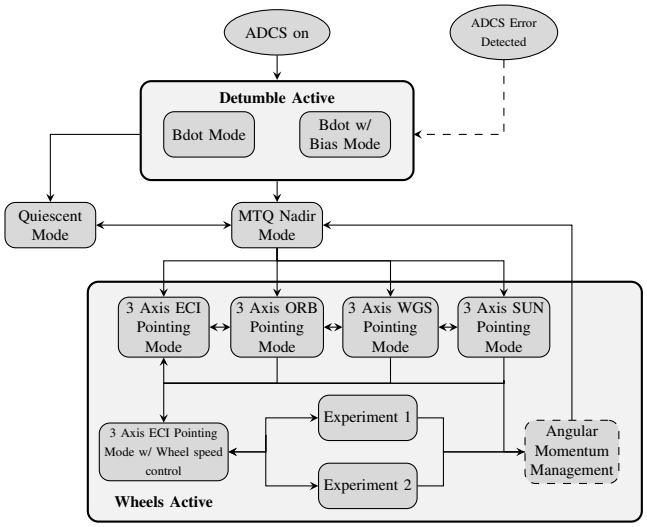

Fig. 3. ESSENCE mission ADCS mode diagram.

C. Real Time Operating System Implementation

The Flight code for the ADCS needs to be executed with a constant and repeatable interval in order to ensure controllability of the spacecraft and to capture flight experiment data at a rate fast enough to evaluate the performance of the experimental modes. The desired cycle time of 10Hz for the ADCS SW will allow for rapid control of the reaction wheels and estimation of the attitude state. The GOMspace NanoMind A3200 [8] was chosen as the On-board Computer (OBC) for the ESSENCE mission since it meets the ADCS mission requirements, has a user friendly development environment and a free RTOS environment.

Real time operating systems differ from regular operating systems since they are designed to allow for multitasking while ensuring tasks are executed by a given deadline. Software designed to operate in RTOS environments need to keep track of ideas such as reentrency (e.g. no dependency on unprotected global variables shared between tasks, keeping calls to external hardware short, etc.) and avoid dynamic memory creation or manipulation (e.g. malloc, free, new, delete).

For ESSENCE, the ADCS SW will run separate tasks from the sensor tasks to prevent potential sensor communication failure from interrupting the ADCS SW. Communication between the ADCS SW tasks and the sensors and actuators tasks will be handled by mutex protected getters and setters. At a high level, the model for these threads can be described by figure 4.

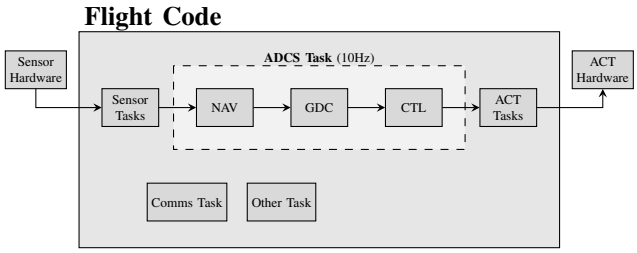

Fig. 4. Flight code threading model.

The use of MATLAB/Simulink to develop and test the ADCS SW allowed the team to leverage many high level benefits of Simulink mentioned earlier and this choice also helped to ease the learning curve for new team members. Additionally, with the built in capability for Simulink to convert well crafted models into C-code, the models are easily implemented on the RTOS. The C-code that is generated has a simple interface consisting of three functions: initialize, terminate and execute. The variable types that are generated for this code include: inputs (e.g. sensor data in), outputs (e.g. commands out), parameters (e.g. unchanging spacecraft mass), and states (e.g. the current state estimate).

In general, figure 5 outlines the approach to the code generation. First, the Simulink tested algorithm for the ADCS SW is placed into a code generation harness to keep track of the compilation settings. Since the ADCS SW exists as a library block in Simulink, any changes to the software will propagate to this harness if any changes are made. The embedded coder[1] in Simulink is used to generate the C-code for the model. In general parameters of the software were configured to be tunable to ensure it can be modified in orbit. The sole exception for this are parameters related to execution time which need to be fixed to generate the code. Another parameter managed by the code generation harness is the stack size of the generated code, managing this parameter correctly is considered an improvement from the previous mission (DESCENT) . Setting this as a parameter allows Simulink to automatically identify the larger variables and create a structure that can be added to the heap systematically, removing the need for manual intervention if the code generation tools decided to place large variables in the stack.

After validating that the code compiles as expected, a list of all variables, along with their sizes and data types is generated.

[1]Our team has chosen to use Simulink's "Embedded coder" tool over the "Simulink coder" tool for a few aesthetic reasons. Namely, we found the code generation report valuable for evaluating if a change increased or decreased memory usage or made the code either simpler or more complex by checking the code complexity estimate for the step function

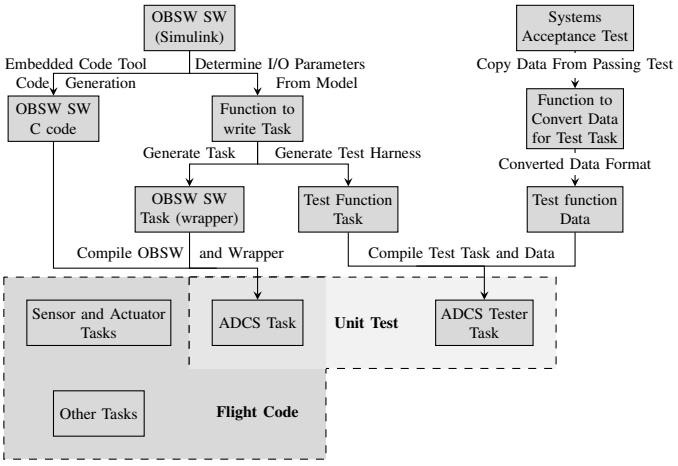

Fig. 5. Procedure for code generation.

This list is used to create wrapper code for a freeRTOS task to run the ADCS SW. Similarly, a tester task is used to run input data through the system and save the output data in order to verify that the code is running correctly and that the interfaces work as intended. These two tasks and code are shared with the rest of the ESSENCE team for integration of the ADCS SW with the Flight Software (FSW), for additional testing. This method allows for easy modifications to be made to the ADCS code should an issue arise.

V. Early Unit and Systems Testing

This section contains preliminary results for some early unit and integration testing to highlight the testing philosophy discussed previously. Beginning with the unit test results for the on-board magnetic field model, then the integration test used to validate the performance of the attitude filter, and lastly, the early integration test results of the nadir pointing guidance and control modes.

A. Magnetic Field Navigation Model Results

The validation of the on board magnetic field model is to show that the model is able to accurately compute the local magnetic field at a given point. This type of model may be inspected against other implementations to ensure its accuracy. Fortunately, there are several on-line calculators available to compare results as well as the magnetic field intensity map. The generated map shown in Figure 6 is comparable to similar sources, such as the world magnetic model in [9].

B. Attitude Filter Design and Results

As previously mentioned, one of the primary objectives of the ESSENCE mission is to educate engineering students with real space mission application. With ESSENCE being a learning experience for many students, it allowed the attitude filter to be designed in a non-conventional manner. The attitude filter at its core level contains four separate functions: the TRIAD algorithm, two different attitude filters, and a selection logic function. The breakdown for this model is shown in

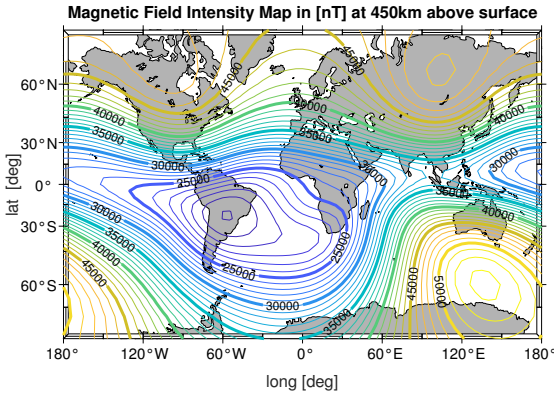

Fig. 6. Magnetic field intensity plot generated from NAV_MAG.

Figure 7, where the dashed lines provide the interaction with outside of the system and the bold lines show interaction within the system. Note, the SELECT function provides reset conditions for the filter models if the ground station commands it to.

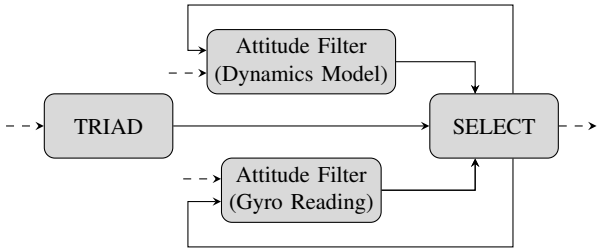

Fig. 7. Inside attitude filter navigation model.

Next, Figure 8 shows the individual functions within the filters, namely: the dynamics and covariance propagators and the individual sensor measurement updates. Individual sensor updates are the preferred method over a single monolithic sensor update as sensors are taking independent observations of the state and separation allows for selective disabling/enabling of sensors as data is made available.

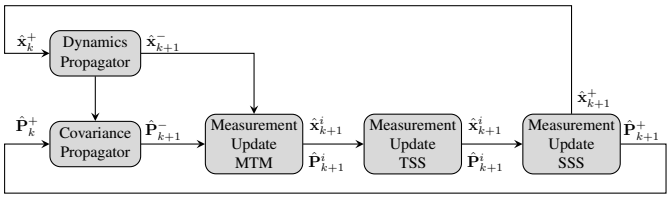

Fig. 8. Attitude Filter block diagram.

The two Multiplicative Extended Kalman Filters (MEKFs) are very similar in structure, but differ in the way that their dynamic propagation is computed. Within Figure 7, the first/top attitude filter models the dynamics fully within the dynamic propagator, while tracking the attitude quaternion and the angular rates as the estimated state vector. The second/bottom filter model uses the angular rate measurement from the fibre-optic gyroscope to compute the propagated dynamics. The estimated state vector for the second filter contains the attitude quaternion and the gyro biases, and uses the gyro reading and estimated bias to estimate the angular velocity. The reason for having both filter models with the attitude navigation model is to compare both methods in flight. With both filters readily available, the ground station is able to choose between filters depending on performance, or switch to one in the case of sensor failure (e.g. single axis sun sensors or gyroscope).

The TRIAD function provides much coarser attitude estimates and thus is not a primary mode of navigation. It is included in this module for early phases of orbit, and for providing initial conditions for the filters to use before attempting to converge. The quaternion from the TRIAD algorithm may also be used as a reset quaternion in the off chance that the filters diverge.

The filters provide similar estimation data, with the full dynamic propagation and direct gyro measurement models providing accuracy of $\pm 1.4^\circ$ (3σ) and $\pm 0.72^\circ$ (3σ) respectively.

C. Orbital Frame Pointing (Nadir) using Reaction Wheels

Control results for the nadir pointing guidance law are summarized in Figures 9 and 10. Figure 9 uses simulated noise to match what is expected from the described attitude filters. Note, this is to try and mimic how the control mode will act with the navigation readings while still preserving the unit test. The attitude control accuracy with the simulated noise is $\pm 1.01^\circ$ (3σ) and $\pm 0.07^\circ$ (3σ) without the simulated noise.

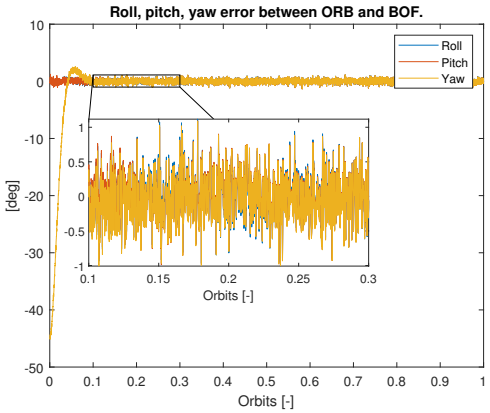

Fig. 9. Attitude errors for nadir pointing control using reaction wheels.

Finally, Figure 10 provides a 3D visualization of how the the nadir pointing control mode is working in tandem with the nadir guidance algorithm. It shows that the x-axis of the spacecraft stays pointing nadir through the entire orbit. Note, the axes have been greatly enlarged for graphing purposes and are not to scale.

VI. Future Work and Final Steps

Presently, the RACS team is conducting unit tests to confirm the operation of the simulator for final deployment on board ESSENCE. The team is on track to complete its CDR goals and is benefiting greatly from its reuse of the tests and code

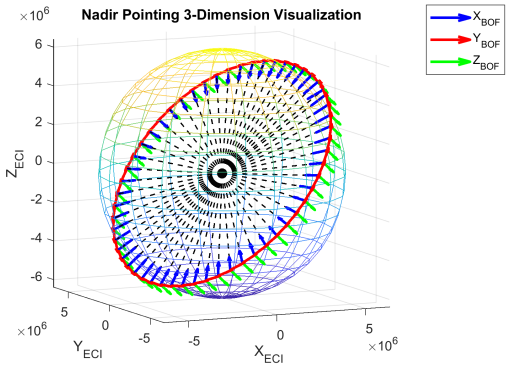

Fig. 10. 3D visualization of nadir pointing control using reaction wheels.

from the DESCENT mission. Future steps to still be completed include: finalizing the sensor models to have accurate data interfaces on the OBSW; augmenting the attitude filter models with estimates of the reaction wheel states to support the experiment, and finalizing the scripts used to generate the freeRTOS tasks to execute and test the software. On a long term basis, the team hopes to integrate power generation and usage models to the simulator to support testing of future power optimization modes. It is also hoped in the future to flesh out the GPS sensor model and to investigate its use in different applications such as formation flight, or possibly terrestrial projects.

VII. Conclusion

The design philosophy employed by the RACS team for the ADCS SW development has proven that a modular and flexible design is reusable between missions, as demonstrated through DESCENT and ESSENCE. The design philosophy in place for both missions would not been possible without the original initiation and development done by NGC Aerospace, which has been validated by their in orbit demonstrations through the PROBA program. This methodology allows for a continuous evolution of the simulator as experimental data is obtained and operational improvements are made. The modularity of the software allows for changes in the system, such as sensors and actuators, to be easily made and the software updates will autonomously expand across the complete system. Overall, this method has increased the efficiency in the software design process and allowed for previous work to be recycled while simultaneously offering a better representation of the real world environment through each iteration.

References

[1] U. Bindra, L. Murugathasan, V. Jain, L. Gangqiang, J. Kang, C. Du, Z. H. Zhu, F. Newland, M. Alger, O. Shonibare, and A. Ruiter, "Descent: Mission architecture and design overview," Sep. 2017. DOI: 10.2514/6.2017-5316.

[2] J. de Lafontaine, J Côté, J. Naudet, A. Kron, and S. Santandrea, "Proba-2: Aocs software validation process and critical results," Jan. 2008.

[3] S. Santandrea, F Teston, K Mellab, P Vuilleumier, D Gerrits, J Naudet, A St-Amour, and J. de Lafontaine, "The PROBA Family: Successful Platforms For The In-Orbit Demonstration of Innovative and Autonomous GNC Techniques," In Proc. 40th Annual AAS Guidance and Control Conference (AAS2017); Breckenridge, Colorado, 2017.

[4] I. D. V.-M. G. F. Modeling, *International Geomagnetic Reference Field*, https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html, Date Accessed: 05-12-2020, 2019.

[5] C. M. Roithmayr, *Contributions of Spherical Harmonics to Magnetic and Gravitational Fields*. NASA Center for AeroSpace Information (CASI), 2004.

[6] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*. The mcGraw Hill Company, INC., 1997, ISBN: 0-07-066829-9.

[7] D. D. McCarthy, "IERS Conventions (1996).," *IERS Technical Note*, vol. 21, pp. 1–95, Jul. 1996.

[8] *Nanomind a3200, data sheet*, 1006901, Rev. 1.16, GOMSpace, Dec. 2019. [Online]. Available: https://gomspace.com/UserFiles/Subsystems/datasheet/gs-ds-nanomind-a3200_1006901.pdf.

[9] NCEI Geomagnetic Modeling Team and British Geological Survey, *World magnetic model 2020*, 2019. DOI: 10.25921/11V3-DA71. [Online]. Available: https://www.ncei.noaa.gov/metadata/geoportal/rest/metadata/item/gov.noaa.ngdc:WMM2020/html.