ISS Double-Gimbaled CMG Subsystem Simulation using the Agile Development Method

Ravi Inampudi*

This paper presents an evolutionary approach in simulating a cluster of 4 Control Moment Gyros (CMG) on the International Space Station (ISS) using a common sense approach (the agile development method) for concurrent mathematical modeling and simulation of the CMG subsystem. This simulation is part of Training systems for the 21st Century simulator which will provide training for crew members, instructors, and flight controllers. The basic idea of how the CMGs on the space station are used for its non-propulsive attitude control is briefly explained to set up the context for simulating a CMG subsystem. Next different reference frames and the detailed equations of motion (EOM) for multiple double-gimbal variable-speed control moment gyroscopes (DGVs) are presented. Fixing some of the terms in the EOM becomes the special case EOM for ISS's double-gimbaled fixed speed CMGs. CMG simulation development using the agile development method is presented in which customer's requirements and solutions evolve through iterative analysis, design, coding, unit testing and acceptance testing. At the end of the iteration a set of features implemented in that iteration are demonstrated to the flight controllers thus creating a short feedback loop and helping in creating adaptive development cycles. The unified modeling language (UML) tool is used in illustrating the user stories, class designs and sequence diagrams. This incremental development approach of mathematical modeling and simulating the CMG subsystem involved the development team and the customer early on, thus improving the quality of the working CMG system in each iteration and helping the team to accurately predict the cost, schedule and delivery of the software.

I. Introduction

Training systems for the 21st Century (TS21) at JSC will provide a simulation-based training for crew members, instructors, and flight controllers on the operation of FOD (Flight Operations Directorate) supported spacecraft including the International Space Station (ISS), Robotics, ISS Visiting Vehicles and other future NASA owned crew transport like MPCV (Multi-Purpose Crew Vehicle). TS21 products include the simulation architecture and math models for the space environment, robotics, and vehicle subsystems as well as an integrated image generation system for out-the-window and camera views.[1] The 6-dof state (position and attitude) of the space station is one of the most fundamental components for all ISS operations. The ISS Attitude Determination and Control Officer (ADCO) has overall responsibility for the integration of

*Lead Software Engineer, Training systems for the 21st Century (TS21), Lockheed Martin, Houston, Texas, 77058, AIAA Member all Guidance, Navigation and Control (GNC), including propulsive and CMG attitude control. The ADCO works in partnership with Russian controllers to manage the station's orientation, controlled by the on-board Motion Control Systems. They also plan and calculate future orientations and maneuvers for the station.[2]

The ISS GNC subsystem for TS21 has two sensor models and an actuator model. The sensors are the Rate-Gyro Assembly (RGA) and the Space Integrated Global Positioning System/Inertial Navigation System (SIGI). The SIGI is broken down into a GPS model and an accelerometer model. The actuator model needed for ISS attitude control is a cluster of four CMGs. The Russian segment (RS) is considered an external model and interacts with the TS21 GNC models through the 1553 telemetry buses. The CMGs operate as momentum storage devices that exchange momentum with the ISS, through induced gyroscopic torques. The CMGs are non-propulsive actuators that provide continuous attitude control in the microgravity environment. Control moment gyros create moments through gimbaling of constant rotation rate momentum wheels. Each wheel is gimbaled to the commanded pointing direction using inner and outer gimbals. The gimbaling of all four wheels is done symmetrically, until all are aligned, at which point desaturation is required to reinitialize the CMGs to their initial state. A minimum of two CMGs are required for attitude control. The magnitude of the induced torque is proportional to the rate of change of the direction of angular momentum vector.[4]

TS21 goal for building ISS simulation is to create a working, reliable software incrementally that is flexible, maintainable and reusable. As TS21 is a developmental and non-sequential project with multiple subsystems, *agile* development process is used to quickly and reliably develop the CMG subsystem simulation. This natural process also facilitates to concurrently work with engineers, managers and the customer. Agile development is a set of software development methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change.[5,6] A few ways of developing better software using the *agile* method are

1. Have a self-organized and motivated team.
2. Working software is a more useful measure.
3. Involve customer feedback on a regular and frequent basis.
4. Respond to change and continuous development.

The basic idea is to start with something simple and small that works, and build the software incrementally by implementing a set of high priority features in a set period of time. For instance, pick a simple feature, describe the feature in a sentence, understand the underlying model (business, engineering or mathematical), and write short code to implement the feature. To write production code and test it at the same time, one widely used practice in *agile* development methodologies is the test driven development (TDD) strategy. The key rule of TDD is summarized as "test twice, code once," which refers to this three-step procedure involved in any code change:[7]

1. Write a test of the new code and see it fail.
2. Write the new code, which does the simplest thing that could possibly work.
3. Verify that the test succeed, and re-factor the code.

These three basic steps form the TDD cycle. The production code and the test cases independently evolve together and this practice decouples the test cases as well as the production code modules from each other. The key element of TDD are the unit test frameworks and such frameworks can contribute to every stage of software design, development, testing and debugging. Furthermore, the principles of object-oriented design play an important and powerful role in writing flexible (decoupled), testable, maintainable and reusable software.

This paper presents the mathematical modeling, software design and implementation of a cluster of 4 control moment gyros (CMG) on the space station (ISS). The introduction section briefly explained the CMG subsystem as well as the agile development method. How the actual CMGs on the space station are used is then briefly explained to set up the context for simulating a CMG simulation. Next, different reference frames and the detailed equations of motion (EOM) for multiple double-gimbal variable-speed control moment gyroscopes (DGVs) are presented. The software simulation aspect is discussed in detail from a user requirements, user stories and use case analysis perspective. UML class diagrams and sequence diagrams are freely used to illustrate the underlying abstractions and design patterns. Finally, the implementation details from a TDD perspective are discussed that supports and verifies the designs.

II. ISS Non-propulsive Attitude Control

A. CMG

There are four CMGs on the ISS nominally rotating at 6600 rpm and are used for non-propulsive attitude control. The CMGs are used more often than the rocket engines to control the ISS attitude because the gyroscopes do not require propellant, which is expensive to launch to the ISS. Instead, the CMGs use the power generated by the solar panels. Similar to a toy gyroscope, each CMG contains a wheel which spins very fast. By pointing the wheels in different directions, the CMG's can either rotate the ISS, or prevent it from rotating. Since an ISS CMG has two axes of rotation (an outer gimbal and an inner gimbal), the momentum vector has three degrees of freedom (3-DOF) i.e, it can be pointed in any direction.[2] A simple way to view the relationship between CMG momentum and external torques is to consider that external torque disturbances can either change the vehicle attitude or the CMG system must counterbalance those disturbances, resulting in an effective absorption of the external torque disturbances. This "absorption" prevents the vehicle's attitude from changing since there is no net torque on the vehicle. For a system of multiple gyroscopes, such as that used by the US GNC, the system's momentum vector is the vector sum of the individual momentum vectors. Thus, the torque applied by the system is caused by the time rate of change (i.e., rotation) of the system's momentum vector.

B. Environmental Torques

To maintain non-propulsive attitude hold control, the controller causes the CMGs to rotate such that the CMG torque is equal to the external torques on the ISS. The primary source of these torques are the environmental forces gravity (F_g)and aerodynamic drag (F_d). To simplify many problems, gravity is assumed to act directly upon an object's center of mass (C_m). However, this simplification would introduce unacceptable error into the attitude control algorithm for the ISS; more precision is required in modeling the effects of gravity. ISS Environmental Torques can vary at different locations on the vehicle, depending on its orientation with respect to Earth's gravitational field. This gravity gradient creates a resultant torque about the vehicle C_m. F_d is proportional to the product of the atmospheric density and the frontal area of the vehicle. Unlike gravity, drag may be assumed to act through the aerodynamic center of pressure (C_p) with no loss of precision. F_d, then, causes a corresponding torque on the vehicle when the C_p is not co-located with the C_m. Because F_d is dependent on atmospheric conditions, the aerodynamic torque will vary over the course of an orbit even if the vehicle's orientation and frontal area remain the same due to diurnal bulge causing density changes and articulation of the solar arrays reducing the frontal area.[2]

C. Torque Equilibrium Attitude

While both the aerodynamic and gravity gradient torques are understandably small and the exact magnitude of each will change as the vehicle orientation and/or configuration changes, it is important to note their relative size. The general rule of thumb is that the gravity gradient torque will be about one order of magnitude greater than the aerodynamic torque. Even with the relatively large difference in magnitude, it is possible to orient the vehicle such that the gravity gradient and aerodynamic torques balance each other out. This orientation is known as a Torque Equilibrium Attitude (TEA). The TEA is the foundation for the *momentum management* controllers used by the US GNC. However, CMGs alone cannot be used to hold a single attitude. Simply put, one cannot simply pick one single attitude to hold over the orbit and expect the CMGs to be able to control as it is impossible to exactly model the aerodynamic and gravity-gradient torques that will be exerted on the vehicle over the course of an orbit. As a consequence, even very small attitude errors will result in residual torques, eventually enough momentum will accumulate to saturate the CMG system. One way to avoid saturation is to continuously manipulate vehicle attitude in order to encounter external torques that tend to reduce the magnitude of the CMG momentum vector required for attitude control. The outcome of doing this is to track, on average, the TEA. Since you could never hold the average TEA perfectly the FSW controllers will always require desaturations in order to effectively maintain attitude control with a system of CMGs. The US GNC does this desaturation by requesting thruster firings from the Russian Segment (RS) GNC. These thruster firings are coordinated by the FSW to coincide with the CMGs moving away from the saturated state.[2]

III. Double-Gimbal Variable-Speed Control Moment Gyroscope Equations Of Motion

This section derives the equations of motion for a cluster of 4 Double-Gimbal Variable-Speed Control Moment Gyroscopes. An overview of these EOM for one DGV is presented in Reference 3. The EOM are for the cluster are rearranged and formulated in terms of the external torques which will be sent to the GNC FSW. Figure 1 shows a DGV representation depicting different frames and various gimbal angles.

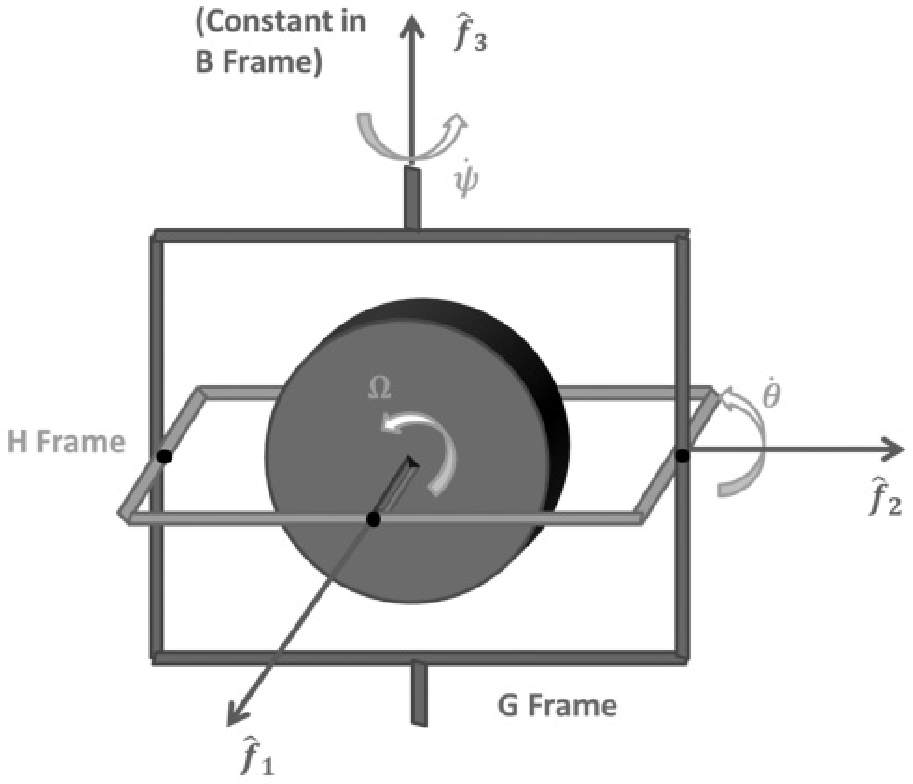


Figure 1: Depiction of a single DGV with coordinates and gimbal angles labeled.

The $\mathcal{F}$ frame defines the orientation of a single DGV device in the spacecraft body but stays fixed with respect to the craft. If the $\mathcal{F}$ frame orientation is related to the $\mathcal{B}$ frame orientation through a direction cosine matrix $[BF]$ then it can be expressed in terms of three orthogonal $\mathcal{F}$ frame unit vectors $\hat{\boldsymbol{f}}_1$, $\hat{\boldsymbol{f}}_2$ and $\hat{\boldsymbol{f}}_3$ given by

$$[BF] = \left[\hat{\boldsymbol{f}}_1, \hat{\boldsymbol{f}}_2, \hat{\boldsymbol{f}}_3\right] \tag{1}$$

The rotation matrix $[BF]$ maps a vector with components taken in the $\mathcal{F}$ frame into a vector with components taken in the $\mathcal{B}$ frame. Similarly, coordinate frame rotations to the outer and inner gimbal frames, $\mathcal{G}$ and $\mathcal{H}$, are given by Euler angle rotations through angles ψ and θ. The direction cosine matrix $[GF(\psi)]$ that relates the $\mathcal{F}$ frame to $\mathcal{G}$ frame is given by

$$[GF] = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Similarly, the other DCMs $[HG(\theta)]$ and $[HF(\psi,\theta)]$ are expressed as

$$[HG] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3}$$

$$[HF] = [HG]\,[GF] = \begin{bmatrix} \cos\psi\,\cos\theta & \sin\psi\,\cos\theta & -\sin\theta \\ -\sin\psi & \cos\psi & 0 \\ \cos\psi\,\sin\theta & \sin\psi\,\sin\theta & \cos\theta \end{bmatrix} \tag{4}$$

The angular velocity vectors of each frame with respect to the one outside it can be written in terms of these unit vectors and various angular rates:

$$\boldsymbol{\omega}_{B/N} = \boldsymbol{\omega} \tag{5a}$$

$$\boldsymbol{\omega}_{G/B} = \dot{\psi}\hat{\boldsymbol{f}}_3 \tag{5b}$$

$$\boldsymbol{\omega}_{H/G} = \dot{\theta}\hat{\boldsymbol{g}}_2 \tag{5c}$$

$$\boldsymbol{\omega}_{W/H} = \Omega\hat{\boldsymbol{h}}_1 \tag{5d}$$

where Ω is the spin rate of the fly wheel about $\hat{\boldsymbol{h}}_1$.

The total angular momentum of the spacecraft and the DGV about the spacecraft center of mass is given by[4]

$$\boldsymbol{H} = \boldsymbol{H}_B + \boldsymbol{H}_G + \boldsymbol{H}_H + \boldsymbol{H}_W \tag{6}$$

where $\boldsymbol{H}_B$ is the angular momentum component of the spacecraft, $\boldsymbol{H}_G$ is the angular momentum of the outer gimbal frame, $\boldsymbol{H}_H$ is the angular momentum of the inner gimbal frame, and $\boldsymbol{H}_W$ is the angular momentum of the spin wheel. Assuming that that all moments are taken about the center of mass, the equations of motion of a system of rigid bodies follow from Euler's equation

$$\dot{\boldsymbol{H}} = \boldsymbol{L} \tag{7}$$

The vector $\boldsymbol{L}$ represents sum of all of the external torques experienced by the spacecraft.

Then $\boldsymbol{H}_B$ and $\dot{\boldsymbol{H}}_B$ for the spacecraft body are written as

$$\boldsymbol{H}_B = [I_s]\,\boldsymbol{\omega} \Rightarrow \dot{\boldsymbol{H}}_B = [I_s]\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times [I_s]\,\boldsymbol{\omega} \tag{8}$$

Because the DGV center of mass is not usually located at the spacecraft center of mass, the ${}^{\mathcal{B}}[I_s]$ matrix incorporates the spacecraft inertia terms and the off-center DGV inertia components using the parallel-axis theorem. With the off-center DGV inertia added, ${}^{\mathcal{B}}[I_s]$ is a constant matrix as seen from the $\mathcal{B}$ frame.

Application of Euler's equation for the other components requires usage of transport theorem that relates a vector derivative between two arbitrarily moving reference frames as

$$\frac{{}^{\mathcal{N}}d}{dt}(\boldsymbol{r}) = \frac{{}^{\mathcal{B}}d}{dt}(\boldsymbol{r}) + \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \times (\boldsymbol{r}) \tag{9}$$

where $\boldsymbol{r}$ is a generic vector.

Therefore $\boldsymbol{H}_G$ and $\dot{\boldsymbol{H}}_G$ for the outer gimbal in $\mathcal{G}$ frame are written as

$$\boldsymbol{H}_G = [I_G]\,\boldsymbol{\omega}_{G/N} = [I_G]\,(\boldsymbol{\omega} + (\dot{\psi}\hat{\boldsymbol{f}}_3) \tag{10}$$

$$\begin{aligned}\dot{\boldsymbol{H}}_G &= \frac{^G d}{dt}([I_G]\,\boldsymbol{\omega}_{G/N}) + \boldsymbol{\omega}_{G/N} \times ([I_G]\,\boldsymbol{\omega}_{G/N}) \\ &= [I_G]\left(\frac{^N d}{dt}(\boldsymbol{\omega} + \dot{\psi}\hat{\boldsymbol{f}}_3)\right) + \boldsymbol{\omega}_{G/N} \times ([I_G]\,\boldsymbol{\omega}_{G/N}) \\ &= [I_G]\,(\dot{\omega} + \ddot{\psi}\hat{\boldsymbol{f}}_3 + \boldsymbol{\omega} \times (\dot{\psi}\hat{\boldsymbol{f}}_3)) + \boldsymbol{\omega}_{G/N} \times ([I_G]\,\boldsymbol{\omega}_{G/N})\end{aligned} \tag{11}$$

Note that $^{\mathcal{G}}\,[I_G]$ is a constant in the $\mathcal{G}$ frame. And transport theorem is used to show that

$$^{\mathcal{N}}\boldsymbol{\omega}_{G/N} =^{\mathcal{G}} \boldsymbol{\omega}_{G/N} \tag{12}$$

$$\frac{^{\mathcal{N}}d}{dt}(\dot{\psi}\hat{\boldsymbol{f}}_3) = \ddot{\psi}\hat{\boldsymbol{f}}_3 + \boldsymbol{\omega} \times (\dot{\psi}\hat{\boldsymbol{f}}_3) \tag{13}$$

Similarly, the respective equations for the inner gimbal and fly-wheel are

$$\boldsymbol{H}_H = [I_H]\,\boldsymbol{\omega}_{H/N} = [I_H]\,(\boldsymbol{\omega} + \dot{\psi}\hat{\boldsymbol{f}}_3 + \dot{\theta}\hat{\boldsymbol{g}}_2) \tag{14}$$

$$\dot{\boldsymbol{H}}_H = [I_H]\,(\dot{\boldsymbol{\omega}} + \ddot{\psi}\hat{\boldsymbol{f}}_3 + \dot{\theta}\hat{\boldsymbol{g}}_2 + \omega \times (\dot{\psi}\hat{\boldsymbol{f}}_3 + \dot{\theta}\hat{\boldsymbol{g}}_2) + \dot{\psi}\hat{\boldsymbol{f}}_3) \times (\dot{\theta}\hat{\boldsymbol{g}}_2)) + \boldsymbol{\omega}_{H/N} \times ([I_H]\,\boldsymbol{\omega}_{H/N}) \tag{15}$$

$$\boldsymbol{H}_W = [I_W]\,\boldsymbol{\omega}_{W/N} = [I_W]\,(\boldsymbol{\omega} \times \dot{\psi}\hat{\boldsymbol{f}}_3 + \dot{\theta}\hat{\boldsymbol{g}}_2 + \Omega\hat{\boldsymbol{h}}_1) \tag{16}$$

$$\dot{\boldsymbol{H}}_W = [I_W]\,(\dot{\boldsymbol{\omega}} + \ddot{\psi}\hat{\boldsymbol{f}}_3 + \ddot{\theta}\hat{\boldsymbol{g}}_2 + \Omega\hat{\boldsymbol{h}}_1) + \boldsymbol{\omega} \times (\dot{\psi}\hat{\boldsymbol{f}}_3 + \dot{\theta}\hat{\boldsymbol{g}}_2 + \Omega\hat{\boldsymbol{h}}_1) \tag{17}$$

$$+ (\dot{\psi}\hat{\boldsymbol{f}}_3) \times (\dot{\theta}\hat{\boldsymbol{g}}_2 + \Omega\hat{\boldsymbol{h}}_1) + (\dot{\theta}\hat{\boldsymbol{g}}_2) \times (\Omega\hat{\boldsymbol{h}}_1)) + \boldsymbol{\omega}_{W/N} \times ([I_W]\,\boldsymbol{\omega}_{W/N}) \tag{18}$$

Since these equations all need to be expressed in body coordinates, the moment of inertia matrices need to be modified as follows from their diagonal measured form. The unit vectors are also to be expressed in body coordinates as discussed earlier:

$$^B\,[I_G] = [BF]\,[FG]^G\,[I_G]\,[GF]\,[FB] \tag{19}$$

$$^B\,[I_H] = [BF]\,[FH]^H\,[I_H]\,[HF]\,[FB] \tag{20}$$

$$^B\,[I_W] = [BF]\,[FH]^H\,[I_W]\,[HF]\,[FB] \tag{21}$$

If the total spacecraft inertia matrix $[I]$ is defined as

$$[I] = [I_S] + \sum_{i=1}^{N}[I_{G_i}] + [I_{H_i}] + [I_{W_i}] \tag{22}$$

The full EOM of a DGV then become

$$[I]\,\dot{\omega} = -\omega \times [I]\,\omega - \sum_{i=1}^{N}\left(\dot{\boldsymbol{H}}'_{G_i} - \omega \times [I_G]\,\omega + \dot{\boldsymbol{H}}'_{H_i} - \omega \times [I_H]\,\omega + \dot{\boldsymbol{H}}'_{W_i} - \omega \times [I_W]\,\omega\right) + \boldsymbol{L} \tag{23}$$

The induced torque due to the four CMGs is needed by the ISS dynamics model. Consequently, the expression for CMG induced torque on ISS is

$$\boldsymbol{L} = [I]\,\dot{\omega} + \omega \times [I]\,\omega + \sum_{i=1}^{N}\left(\dot{\boldsymbol{H}}'_{G_i} - \omega \times [I_G]\,\omega + \dot{\boldsymbol{H}}'_{H_i} - \omega \times [I_H]\,\omega + \dot{\boldsymbol{H}}'_{W_i} - \omega \times [I_W]\,\omega\right) \tag{24}$$

This induced torque L is the external torque applied on to the ISS, and, in the TS21 simulation, this external torque is sent to the rigid body dynamics subsystem. The rigid body dynamics propagates and calculates the rotational dynamics variables of the ISS.

IV. CMG Simulation Design

This section describes the first iteration in the development of a simplified CMG subsystem. The problem statement is picked up from the customers' request for proposal document. The customer has selected several user stories as part of the first iteration where basic functionality of the CMG subsystem is clearly defined. Improving the fidelity of the CMG model, interfacing with thermal and electrical subsystems, and the malfunction design are scheduled for later iterations.

A. Preliminary Design

The objective is to design and implement a simplified CMG subsystem simulation that helps the flight controllers to train on it to guide the ISS to the desired attitude (orientation) for various activities, such as stationkeeping, the docking and undocking of visiting vehicles. Attitude control keeps the ISS pointing in the desired direction and maintains the microgravity environment needed for scientific research. An external GNC interface to the CMG subsystem supports three attitude control modes and are defined here.

1. *Free-Drift*: This mode requires to maintain station attitude by providing active attitude control. In this mode the CMG gimbal rate is commanded in the FSW such that the CMG system momentum is constant relative to the user-specified frame (LVLH, J2000 or body). For the body reference frame, the gimbals are locked such that the system momentum is fixed with respect to the body. Optionally, a target momentum can be specified which is different from the current momentum value.

2. *CMG-Only*: This mode is useful for full attitude control of the station required for proximity operations. No thrusters are fired during the *CMG-only* control mode. The user specifies an attitude hold command where the station attitude is to be maintained fixed relative to the specified reference frame. During docking, un-docking and capture scenarios, the mode is transitioned to free drift.

3. *CMG Thruster Assist*: The station is in full attitude control according to the CMG-Thruster Assist (TA) mode. During TA mode, the user specifies a desired attitude command where the CMGs perform full attitude control of the station (either attitude hold or attitude maneuvers), with RS thruster assistance for CMG desaturation only.

Therefore, to support these three modes the four CMGs provide the main attitude control of the ISS using the dynamical model discussed in section (III.). Each CMG consists of an outer gimbal, an inner gimbal and a heavy spinning wheel. The CMG subsystem accepts the FSW commands and the FSW dictate the commanded wheel speeds as well as the inner and outer gimbal rates. The FSW generates the commanded rates using the Kennel's steering law.[11,12] Therefore, each CMG accepts such commands, processes them and sends the inputs to the CMG dynamics in Eq. (24). For wheel speed control, a CMG compares the current speed and the commanded speed coming from the GNC flight software and a voltage controller generates the required voltage to reach the desired speed. Reference 13 presents more details on the spin motor controller used in this CMG model. The CMG system keeps track of its state, and, to the FSW, each CMG provides the telemetry that contains various states updated at a specified rate. CMG command and telemetry data is communicated through the 1553 Bus interface with the FSW and the FSW interacts with the PCS to display the telemetry in real time. If necessary, the CMG system has the capability to log the data for testing and debugging purposes. A preliminary CMG model development plan identified the ISS GNC subsystem's requirements with respect to the flight controller issued commands and telemetry signature expected.

B. User Stories

A user story is a simple, concise description of the end user requirements which can be written on a small note card. For the first iteration, here are the most important CMG user stories that were selected after discussions with the customer (project leads and instructors).

1. Simulate four identical CMGs
2. Provide CMG angular momentum vector and magnitude
3. Provide torques generated by the CMGs to the ISS dynamics
4. Model CMG wheel characteristics
5. Model CMG gimbal characteristics
6. Accept FSW commands
7. Send telemetry to the FSW (e.g., state, status, and health data)
8. Model each CMG's electronic assembly (EA)
9. Model CMG power characteristics
10. Simulate the CMG startup and shutdown sequences

The next step is to analyze the user stories in detail to figure out how to design and implement them.

C. Use Case Analysis

This section presents the analysis by *use cases* considering the expected behavior of the CMG system. A use case is an elaborated user story with details related to requirements, expected behavior, assumptions and underlying abstractions. The use cases listed below are chosen for the CMG system simulation:

1. *Simulate Four Identical CMGs*: Use case 1 indicates that the CMG cluster consists of four identical CMGs. Figure 2 shows the class diagram that represents the GNC simulation interaction with `ISSCmgCluster` that has four CMG instances of `GenericCmg` class type. The `ISSCmgCluster` object maintains the total angular momentum and torque generated from all four CMGs in the body frame reference. The cluster is initialized from an initial configuration data and this data consists of initial gimbal angles, rates, wheel speeds, inertia of the ISS in the body frame, inertias of the gimbals and wheel in different reference frames, and rotation matrices from each CMG to the spacecraft body frame. Within the cluster, each CMG specific data is pushed down to the respective `GenericCmg` initializers.

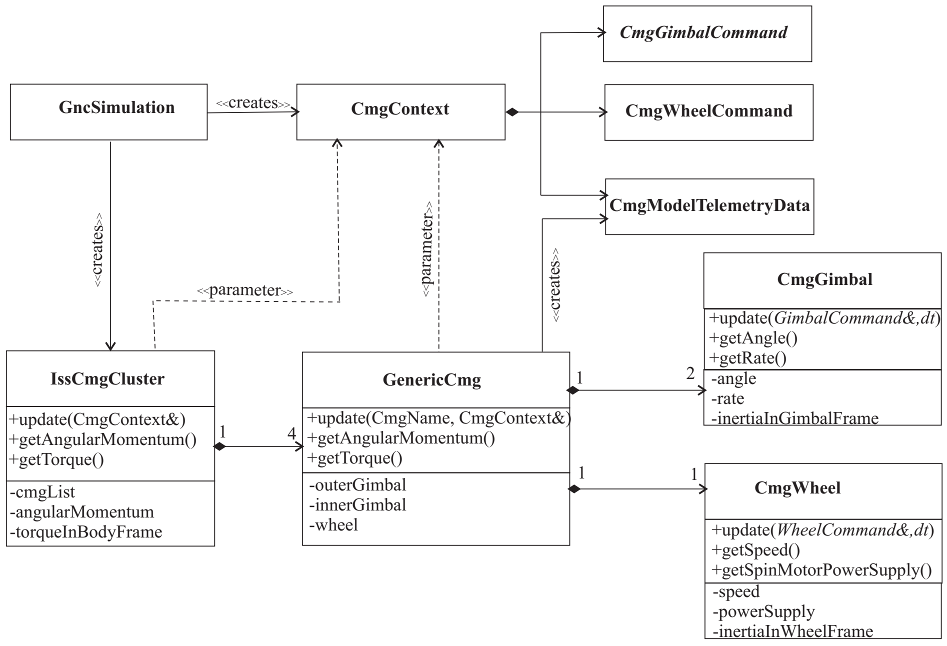


Figure 2: ISS Cmg cluster with four GenericCmg instances.

The `GenericCmg` abstraction has an outer gimbal, an inner gimbal and a heavy spinning wheel. Both the inner and outer gimbals rotate at a very slow rate and are identical from a functional perspective, therefore, there exists a single `Gimbal` abstraction. A concrete instance of `GenericCmg` *has* two instances of `Gimbal` representing an outer and an inner gimbal and one instance of `SpinWheel` representing the wheel. The `SpinWheel` instance supports operations such as spinup or spindown or spin at a specific speed. Each `GenericCmg` object has the required reference frames needed for the dynamics (Eqs. 24) and has its own angular momentum and torque vector states. Each CMG is identified with a *string* type and `GenericCmg`'s `update(CmgName, CmgContext&)` method is invoked at each time step dt. It updates the gimbal and wheel dynamics, the transformation matrices, various frame vector directions, the inertias in body frame, and calculates the angular momentum, torque and power for a CMG. Common data used across the CMG model are stored and passed in the `CmgContext` object which will keep the CMG model class methods' signature length to a minimum.

The class `GenericCmg` is named by prefixing "*Generic*" to "*Cmg*" as it is noticed that its behavior is generic enough that by controlling the gimbal and wheel speeds the Cmg behaves like a reaction wheel, or single-gimbaled fixed or variable speed CMG, or as a double-gimbaled fixed or variable speed CMG. A `GenericCmg` interface could be reused to simulate different types of Cmgs used on different space vehicle types.

2. *Provide CMG Angular Momentum Vector and Magnitude*: A CMG consists of a steel flywheel to produce its angular momentum and a CMG also has gimbal motors that allow the spin axis of the wheel to be repositioned in order to change direction of the angular momentum vector. The total angular momentum of each `GenericCmg` is a simple summation of the individual angular momentums of the two gimbals and the wheel. The angular momentum of the outer gimbal is calculated from Eq. (10); for the inner gimbal Eq. (14) is used; and the wheel's angular momentum comes from Eq. (16).

3. *Provide Torques Generated by the CMGs to the ISS Dynamics*: Torques generated by the CMGs can result in disturbances to ISS attitude that have an impact on properly controlling the station. The torque of each `GenericCmg` is a summation of the torques of the two gimbals and the wheel. These torques are calculated from $\dot{\boldsymbol{H}}'_{G_i}$, $\dot{\boldsymbol{H}}'_{H_i}$ and $\dot{\boldsymbol{H}}'_{W_i}$ expressions which are derived in Section III.. For this simulation, the ISS dynamics treats each CMG torque as an external torque and hence the sign of this torque is flipped before it is sent to the space station dynamics.

4. *Model CMG Wheel Characteristics*: This use case clearly indicates a wheel abstraction. The wheel model should be able to parse a wheel command, handle wheel speed changes depending on the current mode, provide wheel overspeed/underspeed checks, keep track of wheel bearing temperatures and simulate random wheel vibrations. There are five modes of operation for a CMG wheel. *Nominal* mode for normal operating speeds or changing speeds chosen from a set 16 predefined speeds. The *Spin-up* mode for spinning up the wheel from rest. The *Braking* mode for spinning down from a nominal speed and the CMGs are powered from the back EMF as long as the wheel speed is above 1300 RPM. Finally, the wheel can be in *Coasting* mode, if the wheel is spinning down without any power neither from an external power source (RPCMs) and an internal power source (back EMF). As shown in Figure 2, the `SpinWheel` class provides an interface to handle the CMG wheel functionality. A `GenericCmg` instance *has-a* `SpinWheel` instance. A valid `SpinWheel` instance is initialized from its initial speed as well as its inertia given in the wheel frame.

5. *Model CMG Gimbal Characteristics*: This use case indicates a gimbal abstraction to represent a CMG's outer and inner gimbal frames. The gimbal model should be able to parse a gimbal command, store the current states such as the gimbal angles, rates, accelerations, inertias and currents. Also, the model should be able to filter the rates, adjust the angles and rates if they are not in the commanded limits. The desired gimbal acceleration for its motor servo is determined using a simple proportional control with feedforward term. As shown in Figure 2, the `Gimbal` class provides an interface to handle the gimbal functionality. A `GenericCmg` instance *has-a* `Gimbal` instance. A valid `Gimbal` object is initialized from its initial angle and rate, and from its inertia given in gimbal frame.

6. *Accept FSW Commands*: A CMG command frame from the FSW is communicated through the 1553

Bus interface. Most of the commands in the command frame are directed to the wheel, the outer gimbal or the inner gimbal. Both the gimbals have commands that *turn-on* the torquer, the power supplies and the rate over-ride capability. The wheel has commands to *turn-on* the spin motor, the power supply and the temperature over-ride capability. Also, the wheel can be commanded to change to a different speed. Furthermore, there are a few generic commands such as enable fault isolation, reset telemetry bit history and enable loss of 1553 communication over-ride. As shown in Figure 3, the `CmgCommand` class hierarchy provides an interface to support different kinds of commands. The generic commands go in the base class `CmgCommand` and wheel commands are handled by the class `CmgWheelCommand`. In spite of having a single `Gimbal` class, there are two types of the gimbal commands in the hierarchy because each is unique to a gimbal type (inner gimbal or outer gimbal).

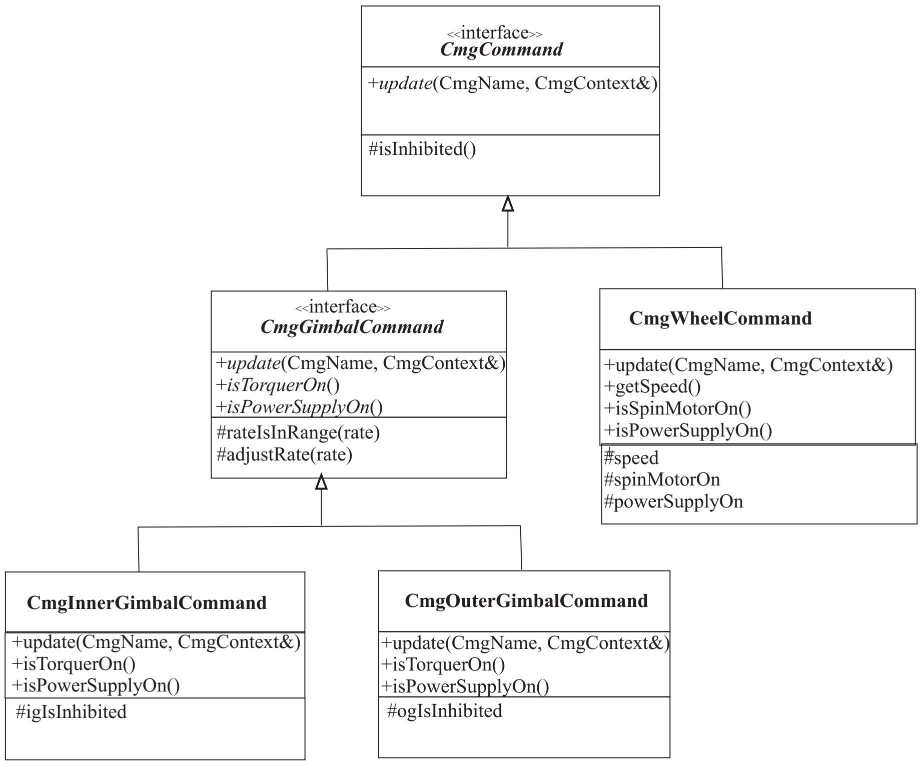


Figure 3: Cmg Command class hierarchy.

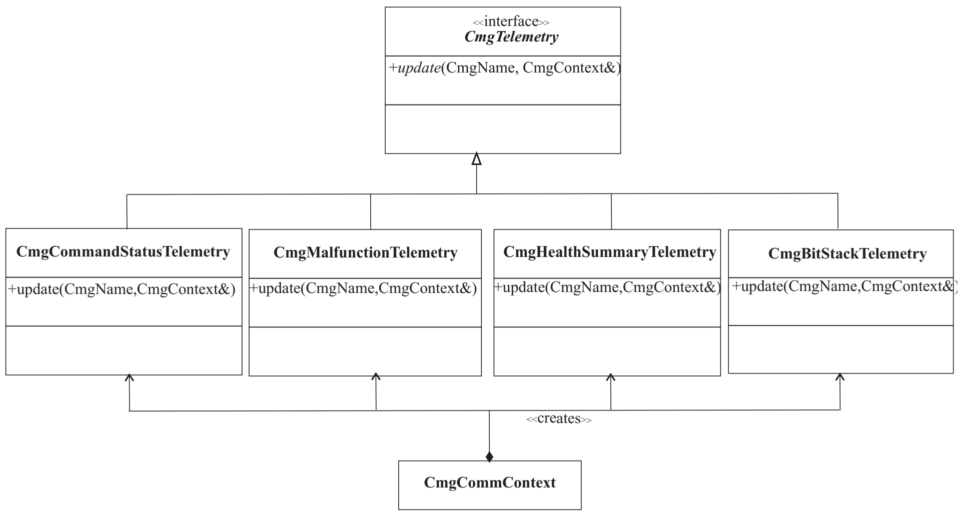
7. *Send Telemetry to the FSW*: A set of 4 CMG telemetry frames are sent to the FSW via the 1553 Bus interface. The telemetry sent captures the CMG data characteristics (e.g., state, status, and health data) which are necessary for training the crew/flight controllers in correct CMG signatures, parameters and limits. The four types of CMG telemetry sent to the FSW are

(a) *Command Status Information*: This is the normal status of each CMG's response to a command frame. Basic information about a CMG is sent as part of the telemetry, i.e. is the CMG powered, are the gimbals and the spin motor powered, what are the current gimbal locations and rates, is a CMG thermally at the right temperature, what about the electrical assembly temperature, and more information such as the commanded vs current wheel speeds, the total system current usage, spin motor heater status, wheel vibration information etc. The class `CmgCommandStatusTelemetry` encapsulates all this information.

(b) *Malfunction Status*: The class `CmgMalfunctionTelemetry` captures the results of a series of bit tests performed on the CMG states to verify if the states are within the tolerance range. *Startup* and *periodic* bit functions perform test of functional paths of both mechanical and electrical assembly models. The bit status words indicate if a CMG detected a fault. For the most part, they are warnings to the end user that a parameter is out of range. Most of these bits toggle, meaning if they return to within the tolerance range the bit will reset automatically. There are a few situations in the CMG model when bits will remain latched and will take action on the bit set. For instance, if the voltage is greater than the upper limit the *high* bit will be set and if the voltage is lesser then the lower limit the *low* bit will be set. The *bit* summary telemetry is eventually displayed on the user displays for users to monitor the fault states of each CMG.

(c) *Health Summary*: The class `CmgHealthSummaryTelemetry` captures the current state of each CMG which includes time tag, various power supply values of the SM, IG, and OG (5 V, 10 V, +15 V and -15 V), analog wheel speed in RPM updated every 15 seconds, SM sine and cosine currents, and, IG and OG information (rates, voltages, current commands, 6.2 V power supply values, torquer currents and temperatures). Health monitoring telemetry is eventually displayed on the user displays for users to monitor the health of each CMG and to troubleshoot in case of malfunctions.

(d) *Bit Stack History*: The contents of the bit history are used to track the last five reported bit changes. Each bit change is reported with five pieces of information about it; time tag at the time the change was reported, the bit number and to which word it belongs to, and identify whether the fault has appeared or disappeared. A `BitEvent` object that stores five of these fields is pushed onto a bit queue that can hold up to five `BitEvent` objects. The class `CmgBitStackTelemetry` encapsulates this bit history telemetry.



<<interface>>
CmgTelemetry
+update(CmgName, CmgContext&)
CmgCommandStatusTelemetry
+update(CmgName,CmgContext&)
CmgMalfunctionTelemetry
+update(CmgName,CmgContext&)
CmgHealthSummaryTelemetry
+update(CmgName,CmgContext&)
CmgBitStackTelemetry
+update(CmgName,CmgContext&)
<<creates>>
CmgCommContext

Figure 4: Cmg Telemetry class hierarchy.

Figure 4 shows the hierarchy of 4 Cmg telemetry classes where each telemetry type fits into its own class. Any telemetry common to the four classes is abstracted into the base `CmgTelemetry` class.

8. *Model Each CMG's Electronic Assembly (EA)*: This use case points out an abstraction to represent a CMG's electrical assembly (EA). The EA contains the majority of the electronics needed to operate the CMG. A real CMG contains circuit card assemblies, a microprocessor and analog controller electronics. An EA box determines the source of the power: is it the external power supply from the RPCMs or the internal power from the spin wheel when it is in breaking mode. This power is used in powering up the various EA cards of a CMG's IG, OG and SM. All three components support 5 V, 10 V, +15 V and -15 V power supplies and the gimbals support an additional 6.2 V power supply. Apart from the fixed power supplies, all three components to run their motors consume power dynamically. This variable power is further divided into electrical and thermal power sinks. The circuit boards driving each of the motors is inside the EA but each gimbal and wheel calculate their respective power consumptions. Assuming that only little heat goes in to the EA circuit board (efficiency factor), the total EA power (electrical or thermal) is calculated from the fixed and variable power consumptions. The heat generated ends up in the thermal control subsystem

(TCS) and the electrical power consumed goes to the electrical power subsystem (EPS) for effective (average) calculations of CMG power consumption. Essentially, the total power consumed in the motor is turned into a simple resistor and its value sent to the EPS system. The EPS solves for current and powers in the CMG as part of the solution for the whole ISS system. The class `CmgElectricalAssembly` encapsulates the various fixed power supply data and an update interface receives the variable power supplies from the three instances (IG, OG and SM). The total electrical and thermal power are calculated within the `CmgElectricalAssembly::update` method. If the wheel is on internal power i.e., in braking mode, then the resistive load is not calculated as there is no external power to the CMG.

9. *Model CMG Power Characteristics*: This use case discusses the total power consumption by the wheel and the two gimbals. The total power is obtained from the mechanical and thermal power consumption (electrical losses/drag). The mechanical power is calculated from $J\omega$, where J is the inertia in kg m^2 and ω in rad/s. The thermal power is calculated from the familiar expression, I^2R, where I is the current in Amp and R is the resistance in Ohms. If the drag component of the thermal power becomes dominant then the $B\omega^2$ expression is used, where B is the viscous drag constant. Detailed mathematical models involved in power calculations for the spin-motor are presented in Reference 13. The parameters of this use case are used by the EA abstraction for power calculations. Furthermore, spin bearing heater power calculations are also to be considered as part of the CMG power characteristics. This behavior is specific to `Gimbal` and `SpinWheel` classes and should be part of the respective classes discussed in use cases 4 and 5. Finally, a resistive load (resistance) for each CMG is calculated from the power consumed and the external power source voltage. This resistive load goes to the electrical power subsystem (EPS) so that the EPS solves for current and powers in the CMG as part of the solution for the whole ISS system.

10. *Simulate the CMG Startup and Shutdown Sequences*: The startup sequence clearly indicates the construction and initialization of a valid `GenericCMG` instance which is typically part of an object's constructor or an initialization method. Similarly, the shutdown sequence indicates the destruction and clean-up of a valid `GenericCMG` instance which is typically part of an object's destructor. A list of operations to consider for each sequence are discussed below.
(a) *Startup Sequence*: A CMG needs to self-test at start-up: set first pass flags, initialize all the constants used in the simulation, the wheel and gimbal parameters and states, default power power characteristics i.e., low-power standby, drive standby power, start-up power (IG, OG and SM) and heater power, and finally set up the closed-loop control gains and any filter coefficients.
(b) *Shutdown Sequence*. Once the command for CMG shutdown is given the FSW will send a command bits to the gimbals and the wheel. The gimbals' torquer is turned off and the rates and accelerations are set to 0 (angles are freezed). The wheel command changes the wheel mode from the *nominal* mode to the *braking* mode. Then the wheel starts to spin-down and when the speed falls below a specified speed, the mode changes to *coasting*. The EA starts to drive on internal power generated from the spinning wheel when it is in *braking* mode. The wheel takes more than 10 hours for a complete spin down.
The behaviors discussed in this use case should be pushed out to the classes discussed in use cases 4 through 8.

D. Underlying Abstractions and Design Patterns

A use case analysis expresses the roles of the underlying abstractions, defines various design patterns, principles and insights gained from the CMG simulation design. The use cases also define the interaction of some of the abstractions with the GNC FSW. The 10 use cases clearly define that for each CMG in the cluster there exists an outer gimbal, an inner gimbal, a spin wheel and an electrical assembly box.

For FSW communication, each CMG's interaction with a remote terminal (RT) is abstracted in `CmgComm` as shown in Figure 5. The `CmgComm` interface receives the incoming commands and passes them along to the `CmgContext` to distribute the incoming commands for individual processing. All the commands have a common interface (an update method) and they're all invoked in the same way. The `CmgContext` processes the command information and sets the necessary command flags for the subsequent model abstractions to decipher. For instance, the `Gimbal` and `SpinWheel` classes receive the respective concrete commands to access the processed command information. The *Command* pattern[9] is used to create an abstract `CmgCommand` with two derivatives: `CmgGimbalCommand` and `CmgWheelCommand`. Though both outer and inner gimbals share the same behavior, the commands from the FSWs are unique to each gimbal. Therefore, the commonality is abstracted in `CmgGimbalCommand` and the specific behavior is pushed into the two derivatives: `CmgOuterGimbalCommand` and `CmgInnerGimbalCommand`. Figure 5 display each of these derivative classes wherein each class is responsible for its own job conforming to the *Single-Responsibility Principle* (SRP).[8] SRP states that a class should have only one reason to change which also means that each abstraction should have one responsibility. If a class has coupled responsibilities, then there'll be more than one reason for it to change.

Furthermore, once the CMG model processes the commands from the FSWs, the model's response is captured in the telemetry and `CmgComm` sends back this information to the FSWs. Therefore, the *Command* pattern is used once again to create an abstract base class `CmgTelemetry` with four derivatives: `CmgCommandStatusTelemetry`, `CmgMalfunctionTelemetry`, `CmgHealthSummaryTelemetry` and `CmgBitStackTelemetry`. Figure 4 shows a simple hierarchy of `CmgTelemetry` classes. The Figure also shows that the `CmgContext` not only creates these telemetry classes but also the `CmgContext` type is used as an argument to the telemetry update methods so that the methods have access to all the states needed to pack the telemetry.

Figure 5 shows the GNC simulation interaction with `IssCmgCluster`, `CmgComm` and `CmgContext` classes. `IssCmgCluster` and `CmgComm` instances invoke their respective update methods and CmgContext instance is passed as an argument to `CmgComm` update method. `CmgContext` class abstraction is designed based on *Monostate* and *Encapsulated Context* patterns.[8] From Monostate pattern perspective, all data members of `CmgContext` are static, so all instances of the `CmgContext` use the same (static) data. Any of the Cmg instances using `CmgContext` do not create multiple instances of it as each `CmgContext` instance uses the same data all through the application. From an Encapsulated Context pattern perspective, divergent parts of the Cmg system can access necessary common data contained in the `CmgContext` eliminating long parameter lists and global data. Therefore, any access to common data is restricted through `CmgContext` object which is passed as an argument from one object type to the other. Enforcing *const* correctness for `CmgContext` object within other object types prevents from inadvertent modification of `CmgContext`'s state.

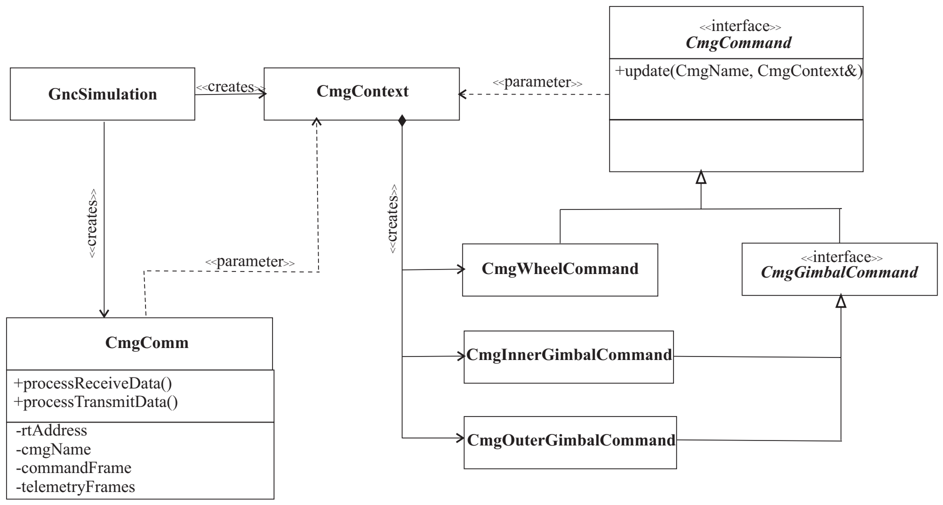


Figure 5: GNC simulation interaction with CmgComm and CmgContext classes.

V. CMG Simulation Development

This section discusses how the design and development together evolved incrementally from the mathematical models, various TDD scenarios, and the simulation results and analysis. One of the key aspects of the CMG simulation is to thoroughly understand the mathematical models developed in section III. The next challenge was to incrementally start writing the code that supports and verifies the designs discussed in section IV. The unit tests are developed concurrently with the production code but there exists a dependency of the unit tests on the production code. The production code doesn't know anything about the unit tests and the tests are a powerful mechanism to develop, re-factor and maintain the production code.

The CMG unit test framework consists of more than hundred independent test cases that are created at every stage of software design and development, testing and debugging, re-factoring and performance optimization. In this paper only the most important scenarios are presented starting with tests of the most basic functionality, then gradually building tests of complex behaviors; a few of the test case scenarios are

1. Verify Modeling Parameters
2. A Simple Bicycle Wheel Gyroscope Model
3. Zero Commanded Gimbal Rates - ISS in Free-drift
4. Non-zero Commanded Gimbal Rates
5. Hold ISS Attitude - Regulator Operation
6. Control Station Attitude - Trajectory Tracking Operation
7. Spin-motor Voltage Controller
8. Power Consumption - Electrical, Mechanical and Thermal
9. Malfunctions
10. Integrated GNC Test with FSW

The test case scenarios are elaborated next to provide just enough details that support and fulfill the user stories discussed in section IV.

1. *Verify Modeling Parameters*: The purpose of this test is to verify the constants associated with the CMG models. All units are metric (SI) (unless otherwise noted). This example test case runs at the beginning and end of the unit test runs, and the constants should be logged at the beginning and end of the simulation runs to verify their values and that those values remain constant across the simulation execution.

2. *A Simple Bicycle Wheel Gyroscope Model*: For this test, the simplest gyroscope rotational equations of motion are derived assuming that a frictionless spinning bicycle wheel is suspended from one end of its axis by a rope where the wheel's axle is horizontal and the free end of the axle precessed about the supported end. The gyroscope seems to defy gravity because the torque created by the spinning wheel counteracts the torque due to gravity. Torque expressions for the simplest spinning wheel are compared with the dynamics presented in section III.. With simplified initial conditions, the complicated dynamics collapsed into simple scalar expressions consistent with the bicycle wheel gyroscope's dynamics. This test gave insight on verifying several states states such as wheel speed and inertia, angular momentum and torque, and also on applying transport theorem to develop the expressions for torque.

3. *Zero Commanded Gimbal Rates - ISS in Free-drift*: The wheel is spinning at a fixed RPM but the inner and outer gimbal commanded rates are zero. To keep it simple all the gimbal angles are initialized to zero. This test specifically checks for the system momentum in the body reference frame to remain constant with respect to the ISS body. It also checks for the speed, torque and simple flags whether the CMG is powered. The test results are checked after two time cycles.

4. *Non-zero Commanded Gimbal Rates*: The next logical test is to start with nonzero commanded gimbal rates and the wheel spinning at a constant speed. At this stage, the CMG model is not connected to the FSW; therefore, a realistic FSW command frame is obtained with the standard command bits set in it such as the commanded gimbal rates, the commanded wheel speed, spin motor status, the gimbal torquer information etc. A series of tests are developed further changing only one variable at a time keeping all the

others fixed. Also, the tests are run for one time step, a few time steps and then for several thousand time steps.

5. *Hold ISS Attitude - Regulator Operation*: This test case verifies a regulator operation where the CMGs are utilized to control station attitude for proximity operations. In this mode, each CMG gimbal rate is commanded such that the station attitude is maintained constant relative to inertial frame (J2000). When this test was written, the ISS simulation was not interacting with the FSW and the FSW has the control algorithm to generate the commanded gimbal rates. Therefore, the commanded rates are generated based of a nonlinear control algorithm presented in Reference 3. These commanded rates are packed into a FSW command frame which are parsed by the CMG model. This test checks for the net CMG torque and the ISS attitude states returned from propagating the attitude dynamics to verify that the station is in attitude hold mode. The CMG control torques and the ISS attitude states for a 60 min simulation run are collected and presented to the customer to confirm that the CMG model when hooked to the FSW will perform correctly. The unit tests not only generated the data to plot the results but also verified simulation run's initial conditions, intermediate and final results.

6. *Control Station Attitude - Trajectory Tracking Operation*: In a trajectory tracking test case, each CMG gimbal rate is commanded such that full station attitude control (either attitude hold or attitude maneuver) is performed. Similar to the regulator operation, the commanded rates are generated from a nonlinear control algorithm to track a reference trajectory presented in Reference 3. This test checks for a 60 min simulation run for the net CMG torque and the ISS attitude error states to verify that the station is tracking an attitude maneuver.

7. *Spin-motor Voltage Controller*: This test verifies a pulse-width modulated (PWM) voltage drive model to control the flywheel speed of a CMG. Also, an adaptive proportional voltage controller is tested where in a controller model adjusts the voltages depending on several control modes for speed, current, and torque. The test verifies the CMG model's interaction with the electrical system and the load dynamics and its overall performance of the system. The CMG spin motor model can directly provide electrical power use and thermal power output to spacecraft subsystems for effective (average) calculations of CMG power consumption. Detailed mathematical models involved in simulating the spin motor model and voltage controller are presented in Reference 13.

8. *EA Power Consumption*: The goal of this test is to verify that the behavior of a `CmgElectricalAssembly` abstraction. The most important variable to check for the total power consumption by the electrical assembly. The EA box consumes power to drive the control cards for the IG, OG and SM and the heater components which is typically the *standby* power. The electrical power is due to thermal losses and drag(especially for the wheel) and the mechanical power is used to drive the gimbals and the wheel. The total power consumption is the summation of all these individual powers and the EA model uses an efficiency factor as part of the calculation.

9. *Malfunctions*: There are a series of unit tests to verify the simulation of malfunctions of the Control Moment Gyroscope (CMG). The CMG malfunctions are grouped under mechanical, thermal and electrical categories. A malfunction can be as simple as one which only affects the telemetry or a complex one that changes the state and behavior of the CMG model. Detailed malfunction models and the general simulation results are presented in Reference 14. The results presented in Reference 14 are obtained from running the unit tests.

10. *Integrated GNC Test with FSW*: Once the unit tests are passed, the CMG model code is integrated into the main ISS simulation and then the integration tests are performed where the GNC subsystem and its interaction with other subsystems is tested. Finally, the GNC subsystem code is checked in to a source control repository for the users to perform the final acceptance tests who execute the GNC flight procedures to verify and validate the ISS GNC simulation.

VI. Simulation

This section presents the main results from numerical simulations which illustrate a CMG's behavior under different wheel speed modes i.e, *nominal*, *spin-up*, *braking* and *coasting*. For each mode, how the wheel speed changes with time is shown. The nominal speed mode tests the voltage controller's performance by increasing the speed from 6600 RPM to 6810 RPM. Figure 6 shows the speed increase which the controller achieved in 17 min. The spin-up test is the most critical and sets up the conditions to speed up from rest (0

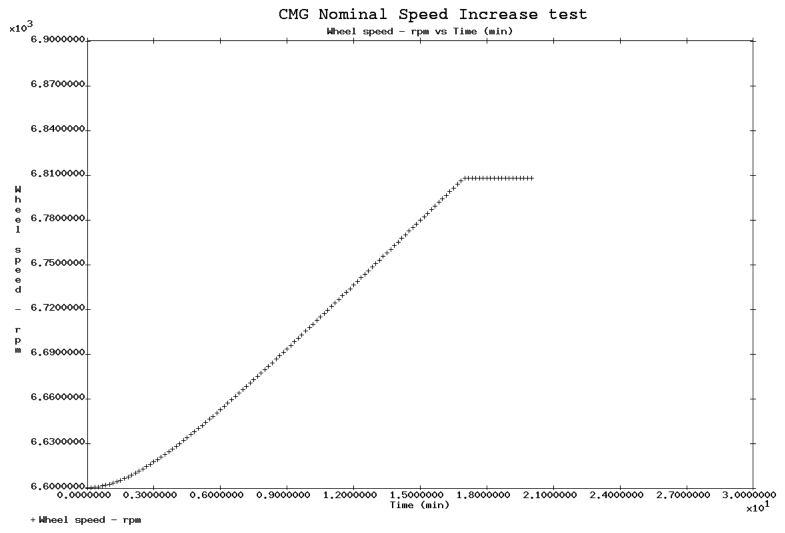


Figure 6: Nominal Speed Increase Test

RPM) to 6600 RPM. As shown in Figure 7, the controller performed well spinning up to 6600 RPM in 7 hrs (the real CMGs spin up in 6-8 hrs), which also implies that the RMS assumption is reasonable. The braking

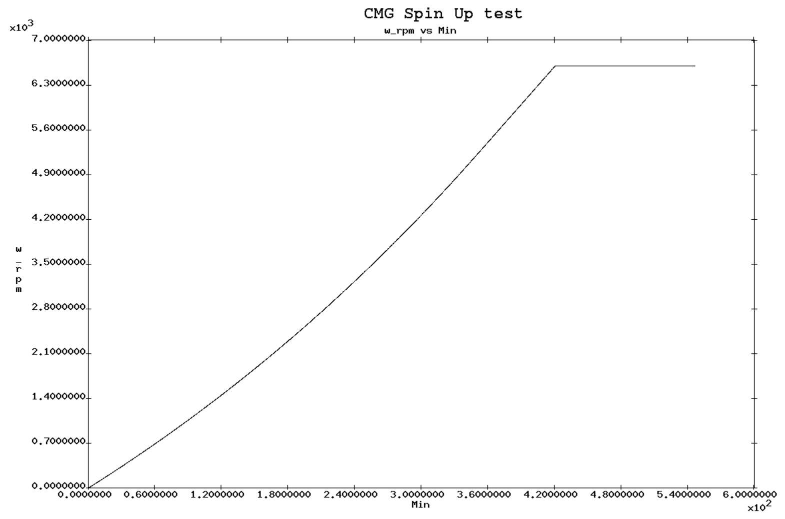


Figure 7: Spin-up Test

test turns off the spin motor power supply which sets the simulation mode to *braking*. Figure 8 shows that the spin-down to 1300 RPM happens in 7.8 hours during which CMGs are also powering the SM, IG and OG control cards using the back EMF. When the speed drops below 1300 RPM, the CMGs transition to *coasting* mode and turn off the power to the SM, IG and OG control cards which results in slower spinning down of the wheel. Figure 9 shows the results for the *coasting* mode only which is set by turning off the voltage supplies to the CMGs and disconnecting the spin motor. Coasting is a slow operation which can

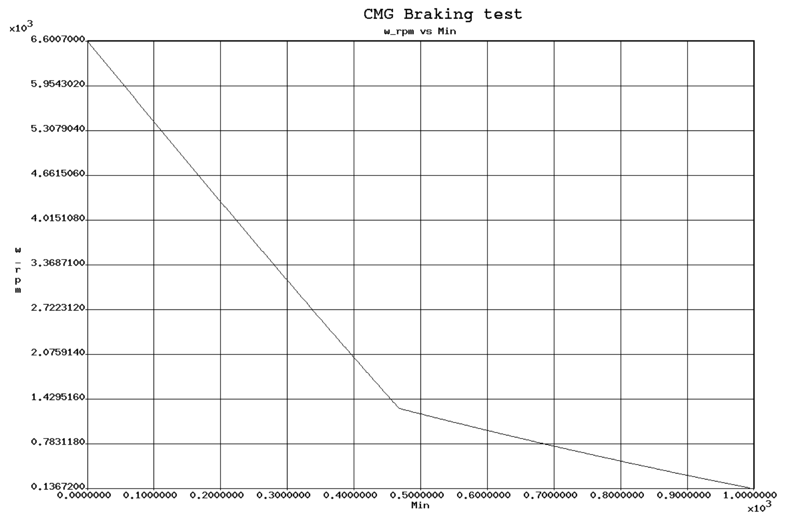


Figure 8: Braking Test

take up to 200 hours for the CMG to completely spin-down to rest.

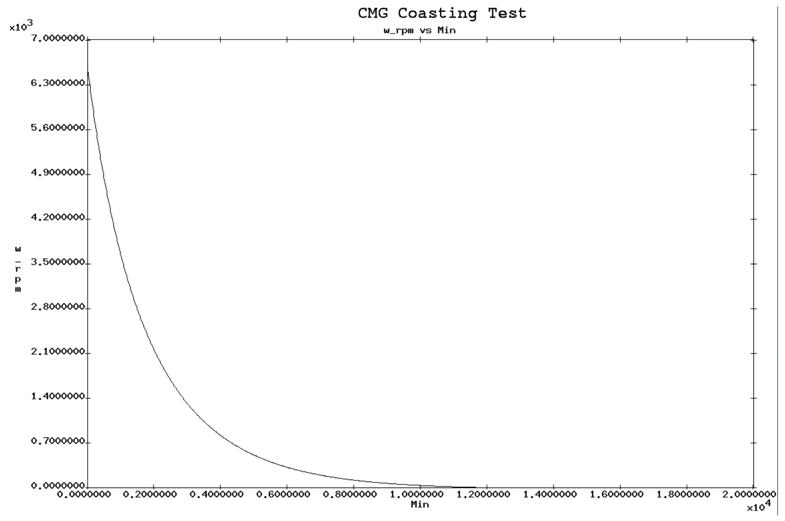


Figure 9: Coasting Test

VII. Conclusion

A new method of simulating a cluster of 4 CMGs on the ISS is used so that the simulation is developed on time and within budget. The agile development method is a natural approach for efficient and effective management of the software development. The CMG simulation work turned out to be enjoyable and fruitful to the development team, and flexible and maintainable for future support and maintenance. All the user stories discussed in this paper are unit tested in a standalone setting and they are verified in the integrated simulation environment. Following the agile principles of development allowed the team to write working software from the beginning of the TS21 project with constant feedback from the customer. An interesting aspect of this work is that object oriented design and analysis, and design principles and patterns greatly enhanced the ability to design and write clean code. For engineering systems, mathematical modeling (dynamics and control) and understanding the underlying principles is of utmost importance before attempting to write code. Several practices of efficient programming are utilized such as discussing the user stories with the customer, keeping short development cycles, pair programming, test driven development, continuous integration and acceptance tests, re-factoring, and constantly strive to keep the design simple. Three technical papers evolved from documentation of the work at each iteration of the development. The CMG model development and testing is completed and is part of the TS21 simulation for the users to verify the customer's user stories. The simple design principles used in this paper can be reused in other engineering applications which use a model-view-controller paradigm with command and telemetry capability. For example, applications that use dynamics and control such as visiting vehicle simulations and flight simulators have similar software architecture.

VIII. Acknowledgments

The author of this paper would like to thank the Johnson Space Center engineering directorate under the Training Systems 21 contract.

References

[1]*The Simulation and Graphics Branch*, ER7, NASA, JSC, Houston, TX, http://er.jsc.nasa.gov/ER7/.

[2]"ADCO Console Handbook," *International Space Station Attitude Determination and Control Officer (ADCO)*, Mission Operations Directorate Systems Division, JSC-36410, Johnson Space Center, NASA, Houston, TX, March 15, 2013.

[3]Stevenson, D., and Schaub, H., "Nonlinear Control Analysis of a Double-Gimbal Variable-Speed Control Moment Gyroscope," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 3, May-June 2012.

[4]Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 2nd ed., AIAA Education Series, AIAA, Reston, VA, Oct. 2009, pp. 178-188, 408-430.

[5]Collier, Ken W., *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing*, Pearson Education, ff. ISBN 9780321669544, pp. 121, 2011.

[6]Kent, B., et al., *Principles behind the Agile Manifesto*, Agile Alliance. Archived from the original on 14 June 2010. Retrieved 6 June 2010.

[7]Hamill, P., *Unit Test Frameworks*, O'Reilly Media, Inc., Sebastopol, CA, 2005, pp. 1-6.

[8]Martin, R. C., *Agile Software Development, Principles, Patterns, and Practices*, Prentice Hall, Upper Saddle River, 2003, pp. 95-98, 193-250.

[9]Gamma, E., R. Helm, R Johnson and J. Vlissides., *Design Patterns*, Addison-Wesley, Reading, MA, Oct. 2009.

[10]*Software Requirements Specification for the Guidance, Navigation and Control Multiplexer/Demultiplexer Computer Software Configuration Item*, International Space Station Program, NASA, JSC, Houston, TX, April 2009.

[11]Kennel, H. F., *A Control Law for Double-Gimbaled Control Moment Gyros Used for Space Vehicle Attitude Control*, NASATM X-64536, 7 Aug. 1970.

[12]Kennel, H. F., *Steering Law for Parallel Mounted Double-Gimbaled Control Moment Gyros*, NASATM X-64930, Feb. 1975.

[13]Inampudi, R., and Gordeuk, J. "Simulation of an Electromechanical Spin Motor System of a Control Moment Gyroscope," *AAS/AIAA SciTech 2016 Conference*, San Diego, CA, 2016.

[14]Inampudi, R., and Gordeuk, J. "Simulation of Malfunctions for the ISS Double-Gimbal Control Moment Gyroscope," *AAS/AIAA SciTech 2016 Conference*, San Diego, CA, 2016.