Simulation of Malfunctions for the ISS Double-Gimbal Control Moment Gyroscope

Ravi Inampudi[*] and John Gordeuk[†]

This paper presents a simplified approach to simulation of malfunctions of the Control Moment Gyroscope (CMG) on board the International Space Station (ISS). These malfunctions will be used as part of flight training of CMG failure scenarios in the guidance navigation control (GNC) subsystem of the Training Systems for 21st Century (TS21) simulator. 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. In both cases, the ISS GNC flight software will read the telemetry and respond accordingly. The user executes these malfunctions by supplying conditional data which modify internal model states and then elicit a response as seen on the user displays. Ground operators and crew on board the ISS use CMG malfunction procedures to better understand and respond to anomalies observed within the CMG subsystem.

I. Introduction

NASA needs to prepare astronauts for spaceflight training in nominal and emergency scenarios for missions in the post Shuttle era. Training systems for 21st Century will provide a simulation-based training for crew members, instructors, and flight controllers on the operation of Mission Operations Directorate (MOD) supported spacecraft including the 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]

Since its inception in 1998, the ISS has always played a vital role for humans to carry out scientific space research and to study life-support systems in the space environment. ISS remains continuously manned for maintaining its systems and performing various scientific experiments. In order to maintain the ground flight controllers' and on-orbit crewmembers' operational proficiency, they continuously train themselves to be familiar with the flight procedures to conduct nominal operations as well as to effectively handle unexpected system failures during time critical situations. For instance, TS21 malfunction procedures are developed for the on-orbit crew and the ground team to understand and to respond to anomalous situations. These malfunctions are designed to guide the users through a series of system failures, in troubleshooting them, in detecting and isolating such failures, for recovery procedures and for preparing the system for subsequent operations.[1]

[*]Lead Software Engineer, Training systems for the 21st Century (TS21), Lockheed Martin, Houston, Texas, 77058, AIAA Member

[†]Senior Engineer, Training systems for the 21st Century (TS21), GHG Corporation, Webster, Texas, 77598.

In the past, several researchers have studied the application of malfunctions to different types of systems [2-3]. Reference 2 presents an approach to increase the system reliability by integrating an on-board computer system (OCS) with the environmental and thermal control systems as well as the life-support systems. The OCS periodically monitored key subsystem performance parameters, and in the event of a failure, it identifies the malfunctioning unit, performs the necessary switching functions to place the failed unit in a safe conflagration, and finally activates standby equipment. It is shown that immediate failure detection and restoration of critical functions provided by the on-board checkout system, enhances crew safety and reduces stress imposed on the crew. Reference 3 addressed the evolution process of the malfunction procedures of the mobile servicing system (Canadian robotic arm on the ISS) based on lessons learned from ISS on-orbit experience. On-orbit operations revealed that the inherent complexities in the hardware and software architectures lead to unexpected behaviors as a result of failures. Recovery is often complicated by software issues, coupling between the subsystems and timing dependent behavior. A solution methodology is given where the system state and the failure operation type are the primary cues in determining the recovery.

Another interesting application of malfunctions is the simulation trajectory analysis on various vehicle failure modes which is performed to support the preliminary flight plan approval for the Ares-I-X vehicle.[5] Several vehicle failure modes are identified which could cause the vehicle trajectory to deviate from its nominal flightpath. The failure modes included loss of thrust vector control (TVC), solid rocket motor burnout, nozzle case breach failures, and loss of roll control from inconsistent thruster firings. The analysis indicate that the TVC and nozzle burn malfunctions cause the largest project dispersions during the early part of the vehicle trajectory whereas the case breach and roll control failures cause the largest tragical dispersions during the middle part of the trajectory. Such simulation analysis ensures range safety analysis and presents an estimate of the overall risk to personnel and facilities in the vicinity of the launch area.

Before introducing various CMG malfunctions, it is imperative to consider the normal operations of a CMG. The CMG cluster mounted on the ISS is a set of 4 spinning wheels that are used to provide long-term attitude control as well as to store energy. A typical CMG is an electromechanical momentum wheel assembly with supporting gimbals (inner/outer) and electronic equipment. And a spinning wheel assembly consists of a flywheel, a spin motor (SM) rotor and a hall resolver. For spacecraft attitude control, the gimbals respond to digital angular rate commands sent from the GNC flight software (FSW) resulting in reaction torque on the spacecraft structure. Electric motors called spin motors in the CMG are used to control the spinning wheel speed which as an example of electromechanical motion control through the transfer of energy between an electrical system (spin motor winding) and a mechanical wheel system (motor load). Such a motion control helps the CMG cluster to manage the angular momentum and to precisely position and/or point the spacecraft. A CMG is a momentum exchange device consisting of mechanical and electrical assemblies. For instance, a double-gimbaled momentum wheel assembly contains the mounting ring, an outer gimbal (OG), and inner gimbal (IG) assemblies, and some electronic equipment. The electrical assembly (EA) provides control and monitoring capabilities of the IG, OG and wheel spin functions of the mechanical assembly. The EA also communicates with the GNC FSW through commands and telemetry. For instance, the EA receives and executes commands from the GNC FSW and the EA also relays to the GNC FSW, feedback and sensor information, including Built-In-Test (BIT) data, temperatures, motor currents, rotor unbalances, and rotational speeds. Reference 6 presents a thorough treatment of CMG mechanical assembly, Reference 8 discusses in detail the CMG dynamics and control and Reference 7 specifically covers the dynamics and control of a double gimbaled variable speed CMG.

This paper studies various CMG malfunctions and presents a simplified approach to simulate them. The user executes these malfunctions by supplying conditional data which modify internal model states and then elicit a response as seen on the user displays. Ground operators and crew on board the ISS use CMG malfunction procedures to better understand and respond to anomalies observed within the CMG subsystem.[6] Unlike an engineering simulator, TS21 is a training simulator which doesn't require high fidelity simulation of the CMG firmware. To begin with, an overview of the TS21 CMG malfunctions and their structure is discussed. The CMG malfunctions are grouped under mechanical, thermal and electrical categories. Next, various aspects of each major malfunction are discussed in detail: malfunction responsibility, its signature, whether it affects telemetry only or changes the state and behavior of the CMG model, or if it indirectly interacts with the flight software and the desired end response. Where applicable, one or more of these perspectives are presented: mathematical models used in simulating a malfunction, present plots with and without a malfunction applied, real CMG failures and how they are simulated, procedures to test the malfunctions and how a dependency might exist among some malfunctions. Finally, the simulation results of several important malfunctions are discussed.

II. Overview of CMG Malfunctions

This section presents an overview of all the malfunctions in the CMG subsystem. Table 1 lists all the malfunctions and are grouped under three categories (electrical, thermal and mechanical). For the users to set and fire them, each malfunction has an activation flag at minimum and additional flags and parameters depending on the malfunction type and its signature. A malfunction may be a simple on/off type to enable or disable the respective malfunction, or a malfunction may include scale and bias to be applied to a model variable, or a malfunction is capable of adding different noise signatures such as random, ramp or sine functions to the model variables.

A few helper function types which are used by some of the malfunctions in Table 1 are

1. The scale and bias function
$$x_{\text{new}} = sx + b \tag{1}$$
where the user supplies the scale s and bias b to be applied to the variable x. The default values for s is 1 and for b it is 0. It depends on the type of malfunction if the x_{new} is the updated model variable or if it only changes the telemetry.

2. The ramp function which is defined as
$$x_{\text{new}} = x_0 + \frac{\Delta x}{t_{\text{span}}}\text{d}t \tag{2}$$
where x_0 is the initial value, Δx is the rise over the time span, t_{span}, and dt is the simulation time step. The user supplies the initial and final values and the t_{span}.

3. The sine wave function and is defined as
$$x = a\sin(\frac{2\pi}{T}(t - t_0) + \phi) + b \tag{3}$$
with a is the amplitude, T is the period, $t - t_0$ is the elapsed duration of the malfunction, ϕ is the phase and b is the DC offset. The user sets these variables before activating the sine wave function and at each simulation time step, the function returns a sample value x of the sine wave. For example, a wheel vibration can be simulated using this function.

4. The random function is used to simulate noise and is defined as
$$x = x_{\text{min}} + (x_{\text{max}} - x_{\text{min}}) * (\frac{\text{rand}()\%10000}{10000}) \tag{4}$$
where the user inputs x_{min} and x_{max}. The rand()%10000 generates an integer in the range 0 to 9999 and then dividing by 10000 generates a random decimal in the range 0 to .9999 which then scales the range $x_{\text{max}} - x_{\text{min}}$ and adds the initial value of the range x_{min}.

Table 1: CMG Malfunction List

Malfunction	Description	Simulation Variable
Electrical		
IG Power Supply Failure	Fails to power the IG	`igPowerSupplyFail`
OG Power Supply Failure	Fails to power the OG	`ogPowerSupplyFail`
SM Power Supply Failure	Fails to power the SM	`smPowerSupplyFail`
RPC Power Supply Failure	Fails to power the RPC	`rpcPowerSupplyFail`
CPU Power Supply Failure	Fails to power the CPU	`cpuPowerSupplyFail`
Loss of Communication	Communication loss	`lossOfCommEnabled`
IG 5V Power Supply (PS) Bias	Scale and bias the IG 5V PS	`igVoltsPlus5Fail`
OG 5V Power Supply (PS) Bias	Scale and bias the OG 5V PS	`ogVoltsPlus5Fail`
SM 5V Power Supply Bias	Scale and bias the SM 5V PS	`smVoltsPlus5Fail`
SM Drive Control Card Failure	Sine/cosine drive failure and EA sensor failure	`smDriveCardFailure`
Wheel Speed Hall Sensor Fail	Cannot measure wheel speed	`wheelSpeedHallSensorFailure`
SM Command Current	Commanded current to spin the motor	`userSmccNoiseValue` `smccNoise` `smccRamp` `smccBias` `smccSpike`
Thermal		
Gimbal Temperature Sensor Fail	Sensor failure	`igTemperatureFail` `ogTemperatureFail`
Spin Bearing Heater Failed	Heater failure	`sbHeaterFailEnabled` `sbHeaterFailOn`
EA Temperature Increase	Temperature rises	`eaTemperatureIncrease`
EA Temperature Sensor Fail	Sensor failure	`eaTemperatureSensorFail`
Mechanical		
Gimbal Failure	Gimbal fails	`igFail/ogFail`
Gimbal Rate Failure	Gimbal rate fails	`igRateFail/ogRateFail`
Wheel Vibration	Wheel vibrates	`userwheelVibrationNoiseValue` `wheelVibrationNoise` `wheelVibrationRamp` `wheelVibrationSpike`
Bearing Seize	Wheel under drag	`bearingSeize`

III. CMG Malfunction Scenarios

This section discusses detailed scenarios of each major malfunction grouped under electrical, thermal and mechanical categories; specifically it discusses each malfunction's responsibility, its signature, a procedure to test and the desired end response, dependency with other malfunctions, and, if it affects telemetry only or if it changes the state and behavior of the CMG model. Furthermore, where applicable, simpler mathematical models or helper functions used in simulating a malfunction and their results are presented.

A. Electrical

1. *IG Power Supply Failure*

A flight controller (FC) applies this malfunction by setting the boolean variable `igPowerSupplyFail` to true. CMG is past initialization and the CMG simulation model detects a power supply failure in a CMG gimbal after the CMG has been powered up. Various voltages related to the inner gimbal are set to zero which are shown on the user displays and sent as telemetry to the GNC flight software (GNC FSW) as well. At this point the CMG will be removed from the steering law by the GNC flight software causing the online status to go from ”online” to ”off-line”. Depending on how many other CMGs are left, this could cause a loss of attitude control. Despite the gimbal power supply failure, the CMG will continue to spin normally. Note that this is a failure of the IG power supply, and not of the relay to the gimbals or the gimbals themselves.

2. *OG Power Supply Failure*

The behavior of the malfunction is the same as that of the inner gimbal power supply failure but applied to the outer gimbal by setting the boolean variable `ogPowerSupplyFail`.

3. *SM Power Supply Failure*

This malfunction, `smPowerSupplyFail`, when active causes a trip in the CMG spin motor, all telemetry upstream from the spin motor is lost, including bearing temperature data and wheelspeed. The GNC FSW removes the CMG from the steering law right away. The CMG mode is set to *braking* mode if the wheel speed is less than the critical speed and the CMG uses its own power generated from the back EMF; otherwise, the CMG mode is set to *coasting*. Therefore when `smPowerSupplyFail` is active, it disables other dependent malfunctions such as `smVoltsPlus5Fail`, `smVoltsPlus10Fail`, `smVoltsPlus15Fail` and `smVoltsNeg15Fail`. It is verified that the spin motor data remains static on the user displays such as various SM voltages, bearing temperature and wheel speed.

4. *RPC Power Supply Failure*

When the malfunction `rpcPowerSupplyFail` is enabled, the RPCM voltage is set to 0.0 shutting off the external power supply to the electrical assembly. RPC power supply failure causes a loss of the electrical assembly causing a loss of attitude control with the entire CMG. Both the inner gimbal and outer gimbal rates are set to zero and the CMG starts spinning down and the telemetry shows that the speed is going down. The GNC FSW detects a loss of the CMG and the CMG is immediately removed from the steering law. The CMG puts itself into active *braking* mode and the inner gimbal, outer gimbal and the spin motor are internally powered by the back EMF voltage in the spin motor as long as the wheel speed is above 1300 RPM. During, *braking* mode, the 1553 data communication is enabled with the GNC FSW, the FSW is able to command the wheel and the inner and outer gimbal torquers are enabled. If wheel speed is below 1300 RPM, the wheel enters the *Coasting* mode, if the wheel is spinning down without any power neither from an external power source (RPCMs) nor an internal power source (back EMF), then the 1553 data communication is disabled.

5. *CPU Power Supply Failure*

When the user sets the malfunction `cpuPowerSupplyFail` to true, the CPU power supply failure causes a loss of the electrical assembly causing a loss of attitude control with the entire CMG. And the

CMG model behaves similar to applying `rpcPowerSupplyFail` malfunction which is discussed in detail earlier. However, CPU power supply failure also disables the 1553 data communication with the GNC FSW. The FSW is unable to command the wheel and the inner and outer gimbal torquers are disabled. It is verified that the data for the offending CMG remains static on the user displays until the malfunction is disabled.

6. Loss of Communication

Loss of communication malfunction simulates intermittent 1553 communication failures between the CMG and the GNC FSW. On orbit a CMG can experience such a failure due to CMG power supply failure, 1553 chipboard interactions, or firmware timing issues. When the malfunction, `lossOfCommEnabled`, is enabled, and, if ”Retry” is enabled, and the GNC CMG mode is ”Drift”, ”Thruster Assist”, or ”CMG Only”, the GNC FSW automatically cycles power to the CMG, thus putting it back online with a total of 5 consecutive retries. During the retries, the FSW control system compensates for the temporary loss of the CMG. In real-time and in the simulation, this is a latched bit and so a *reset* command has to be sent to clear the advisory bit. When the *latched* bit is set to *false*, the software will reset the bit when the condition no longer exists. When *true*, the software will not reset the bit, and a reset command must be sent by the FSW to reset the flag to its default status. This malfunction, `lossOfCommEnabled`, is sometimes used in combination with others in order to produce different scenarios. For example, when applying a `cpuPowerSupplyFail` malfunction, the `lossOfCommEnabled` malfunction is activated in order to get the appropriate signatures. As part of the integration testing, activating the `lossOfCommEnabled` malfunction on a CMG, a loss of 1553 communication between the CMG and GNC MDM is simulated by disabling the process of receiving and transmitting the data over the remote terminal.

7. Power Supply Bias

This malfunction fails, *high* or *low*, the voltages on the power supply of your choice; it can be applied to IG (`igVoltsPlus5Fail`) or OG (`ogVoltsPlus5Fail`) or SM (`smVoltsPlus5Fail`). For instance, to apply a 5 V bias malfunction, `igVoltsPlus5Fail`, first set a scale and bias on the 5V voltage supply for the IG of a CMG and then enable the malfunction. Application of this malfunction effects telemetry only and the CMG model is not affected. Also, when `igVoltsPlus5Fail` is enabled, the assumption is that the `igPowerSupplyFail` is inactive. Similar malfunctions exist for 10 V, +15 V and -15 V with respective behaviors. A unit test and an integration test verified an application of this malfunction to produce the desired results.

8. SM Drive Card Fail

In orbit when this malfunction happens it fails the entire SM Current Drive Card including the failure of the sine/cosine drives as well as the EA temperature sensor. The result is that the CMG will spin down and eventually be taken out of the steering law. To simulate this, the malfunction `smDriveCardFailure` takes in a new speed value from the user and overrides the current speed command with the new speed. The speed input could be greater than the current speed and the wheel mode is changed to *MalfCmd*. And the corresponding bit tests indicate a wheel overspeed/underspeed and the telemetry is affected. The total drive card failure is not simulated where the sine or the cosine drive currents can be 0 and EA temperature sensor is failed. Figure 1 shows the results of an integration test where in a `smDriveCardFailure` is simulated by defining a new user speed of 6780 RPM and then enabling the malfunction. It clearly shows that the set speed of 6780 RPM is reached in 1000 s from a nominal speed 6600 RPM. The malfunction is disabled at 1000 s and the CMG model has slowed down the wheel speed back to the nominal speed 6600 RPM.

9. Wheel Speed Hall Sensor Fail

Hall sensor failure indicates that the sensor is not able to detect the wheel speed. Enabling the malfunction `wheelSpeedHallSensorFailure`, overrides the sensed wheel speed with a user set speed and it only affects telemetry. The actual wheel speed is not affected by this malfunction and the displays show

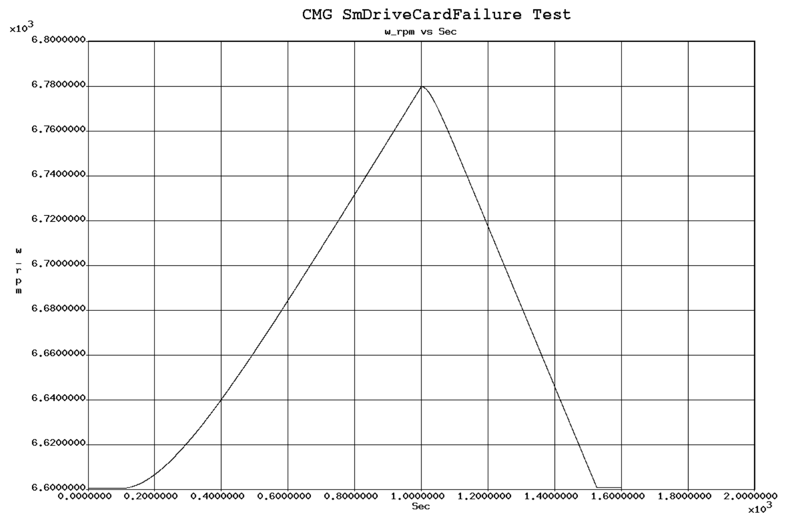

Figure 1: Spin Motor Drive Card Failure

the correct speed, whereas, the sensed speed shown on the displays is the user set speed. For example, if the nominal speed is 6600 RPM and the malfunctioned speed is 7000 RPM, on the user displays the digital speed shows 6600 RPM and the analog speed shows 7000 RPM.

10. SM Command Current

This malfunction simulates the flight software generated spin motor command current (smcc). The user can add noise, ramp, bias and/or a spike currents to the smcc current signature. The `smccNoise` object generates a noise value based on the user defined upper bound value set in `userSmccNoiseValue`. The `smccRamp` creates a ramp for a duration based on the input current range and `smccBias` sets the bias. The `smccSpike` object generates a positive spike using the sine wave function like a half-wave rectified smcc current. Each of the components are then added to create the total smccMalfCurrent as

$$\text{smccMalfCurrent} = \text{smccNoise} + \text{smccRamp} + \text{smccBias} + \text{smccSpike} \tag{5}$$

The resultant smccMalfCurrent is then added to the actual smcc current before packing it as part of the telemetry. The smcc can be positive or negative, and in real CMGs the smcc comes from the SM controller card which then drives the PWM and the motor, such that sine and cosine currents (I_s and I_c) are measured quantities in the SM controller card. In-orbit, it is a current controller which uses speed and current feedback to drive the current the spin motors. Whereas, in the simulation, a voltage controller is used based on speed and torque feedback. Reference 9 present details on a simplified electromechanical spin motor and voltage controller models used in this simulation.

B. Thermal

1. Gimbal Temperature Sensor Fail

Relevant malfunctions are (`igTemperatureFail`) for IG and (`ogTemperatureFail`) for OG. The user inputs a new temperature in Fahrenheit and the CMG model sets the gimbal temperature to the new temperature (within the transducer limits which are provided in Table 4-15 in Reference 6). The IG/OG telemetry sees this temperature change and the CMG model telemetry bit tests set the temperature fail bits to *high/low*. The telemetry bit states are sent to the FSW over the 1553 communication and the bit status is shown on the user displays.

2. *Spin Bearing Heater Failed*

If the temperature of the wheel gets cold (is below the *low* bearing temperature) then the spin bearing heaters are turned on to heat up the wheel. The heaters are enabled or disabled depending on violations of bearing temperature low conditions. There are two dependent malfunctions `sbHeaterFailEnabled` and `sbHeaterFailOn` related to the spin bearing heater. If the `sbHeaterFailEnabled` malfunction is activated it either fails the heater always *on* or fails always *off*. Once the malfunction `sbHeaterFailEnabled` is activated a second flag `sbHeaterFailOn` needs to be set to keep the spin bearing heater failure *on/off*. Based on the wheel temperatures, `sbHeaterFailOn` overrides the spin bearing heater command status conditions whether failure is always *on* or always *off*. This code logic shows how these two variables are used to simulate such a behavior

```
if (malf.sbHeaterFailEnabled) {
  if (malf.sbHeaterFailOn == true) {
    spinBearingHeaterCmd_ = true;
  } else {
    spinBearingHeaterCmd_ = false;
  }
}
```

In each simulation run, the default value of the `spinBearingHeaterCmd_` which is either *true* or *false* is set based on the FSW command status.

3. *CMG EA Temperature Increase*

The `eaTemperatureIncrease` malfunction describes a failure within the EA that causes the actual temperature to increase. It is assumed that the affected CMG is powered on and the EA Temperature sensors will detect the temperature increase, but since no FDIR is associated with these sensors, nothing will automatically happen. The user is allowed to input a *scale* and *bias*, and the code logic below shows how the scale and bias are used in the CMG model to simulate such a temperature increase

```
double scale = 1.0;
double bias = 0.0;
double temp = temperature_;
if (malf.eaTemperatureIncrease.isEnabled) {
    scale = malf.eaTemperatureIncrease.scale;
    bias = malf.eaTemperatureIncrease.bias;
    temperature_ = scale*temp + bias;
}
```

Since the temperature is increasing, as opposed to instantaneously jumping, the flight controllers conclude that this signature indicates a true temperature increase in the EA. In-orbit, the FCs also look for other signatures to confirm this conclusion. One example could be an increased current drawn by any of the cards in the EA hardware. However, `eaTemperatureIncrease` malfunction in the simulation model only affects the telemetry for the temperature and does not change the current draw on any of the EA cards. The `bearingSeize` malfunction discussed in section C. simulates the friction in the moving parts of the spin motor which actually increases the temperature which can be confirmed by increased current to the spin motor.

4. *CMG EA Temperature Sensor Failure*

The temperature sensor located on the SM drive card within the CMG electronics assembly fails either *high* or *low*. If the malfunction `eaTemperatureSensorFail` is enabled, the user has to enter a temperature value which updates the CMG model's EA temperature. If the temperature sensor value reads below

-65.0°F, then the CMG model telemetry bit test sends the sensor bit status to the FSW and the telemetry on the user displays will show the value as off-scale *low*. If the sensor reads above 180.0°F, then the telemetry bits indicate that the EA Temp is above 180.0°F. If the sensor reads above 200.00°F, then the user displays will show the value as off-scale *high*. The FCs recognize and realize the impacts to the loss of this sensor. The spin bearing temperature sensors are monitored as usual to make sure the CMG temperature remains above operational values. The loss of this sensor does not indicate any change in the temperature performance of the CMG, just a loss of insight into the temperature of the EA. Also, if this sensor failure malfunction is active it overrides the `eaTemperatureIncrease` malfunction behavior.

C. Mechanical

1. IG/OG Gimbal Fail

Applying the malfunction `igFail` will cause the Inner Gimbal Relay on a CMG to fail; the model sets the gimbal rate to 0.0 and the torquer on command to false. Also, notice the dependency on other malfunctions in this snippet of code from the model

```
if (malf.igPowerSupplyFail || malf.igFail || malf.cpuPowerSupplyFail) {
    rate_ = 0.0;
    torquerOn_ = false;
}
```

This malfunction causes the CMG to be pulled out from the Steering Law. The flight controllers know immediately what the failure is from the signatures, but to make sure, FCs request a dump of the CMG momentum. FCs also monitor the momentum to ensure there will be no impacts to space station maneuvers. There is an equivalent malfunction, `ogFail`, which will cause the outer gimbal relay on a CMG to fail.

2. IG/OG Gimbal Rate Fail

Applying the malfunction `igRateFail` will cause the inner gimbal rate on a CMG to go to zero. There is an equivalent malfunction `ogRateFail` will cause the outer gimbal rate on a CMG to go to zero. For instance, Figure 2 shows the `ogRateFail` results of both the IG and OG rates when the `ogRateFail` malfunction is activated about 2297 seconds in the simulation. The OG rate drops down to zero whereas the IG rate jumps up to 1 rad/s as commanded by the FSW.

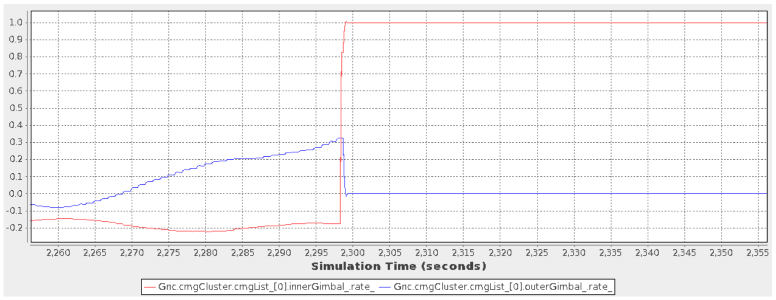

Figure 2: IG Rate Bias Test

3. CMG Vibration

The CMG vibration malfunction is designed to recreate a real-life CMG failure. For a real CMG, at a vibration of 0.2g, the GNC FSW takes the CMG out of the steering law. At 0.5g, the display values (telemetry) will stop updating and will just read off-scale high. The other CMGs will have sympathy vibrations as well, but will not increase as rapidly. If the other CMGs reach the 0.2g threshold because GNC FSW still hasn't taken the original CMG out of the steering law, then these CMGs will also need to be taken out as well. On-orbit data during the CMG 1 failure shows that the other CMGs did not show an associated SMCC increase during the failure.

In the simulation, three kinds of *vibration* malfunctions can be activated: the user can add noise, ramp, or a spike like vibration. The `vibrationNoise` object generates a noise value based on the user defined upper bound value set in `userWheelVibrationNoiseValue`. By default, `vibrationNoise` is active in the simulation which generates a random number (default: 0.0 to 0.04g); if needed, the user can provide a range. The `vibrationRamp` creates a ramp for a duration based on the input vibration range. The `vibrationSpike` object generates a positive spike using the sine wave function much like a half-wave rectified vibration. Each component is added to create the total vibration

$$\text{vibration} = \text{vibrationNoise} + \text{vibrationRamp} + \text{vibrationSpike} \tag{6}$$

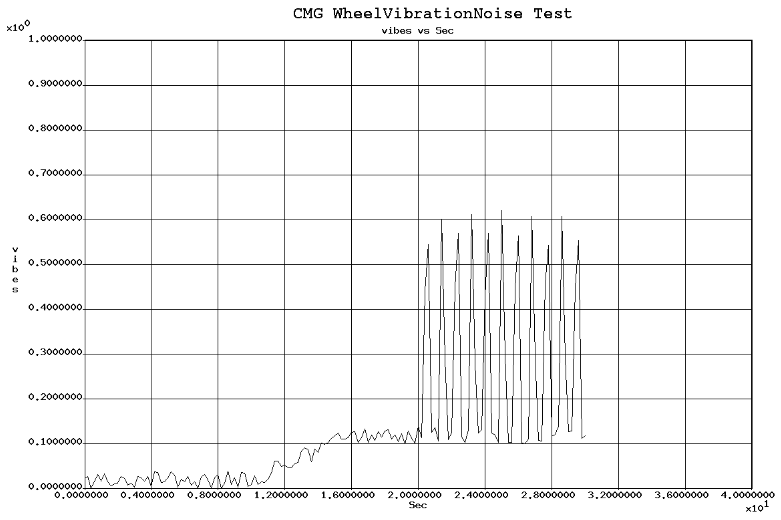

Figure 3: Wheel Vibration Test

The resultant vibration is then packed in to the telemetry. Figure 3 shows an example when all the three types of vibrations are present. For the first 10 seconds, the default `vibrationNoise` is active, and in the next 10 seconds the `vibrationRamp` is activated, and finally at 20 seconds `vibrationSpike`is activated. Also, in orbit, the crew hear an off-balanced washing machine noise but in the simulation this noise has not been simulated yet. An enhancement would be when this malfunction is active to have a noise file sent to the audio.

4. *CMG Bearing Seize*

This is an important malfunction as it affects the model directly and simulates a real failure accurately. The malfunction `bearingSeize` increases drag on the wheel based on the user input and the dynamics which incorporates the drag into its model slows down the wheel simulating the bearing seize. The temperature of the wheel will increase when this malfunction is active. The wheel continues to slow down and the currents used and power generated are directly affected. The drag model related to the spin-motor dynamics is presented in Reference 9. For a bearing seize malfunction, the temperatures increase up to 90 K and the total power going to EPS is around 3000 Watts. Figure 4 shows the results of a simulation when the `bearingSeize` malfunction is active and when the drag force on the wheel is increased 20 times. The malfunction is activated around 5150 s when the wheel is spinning at its nominal speed of 6600 RPM. The drag force slows down the wheel speed continually.

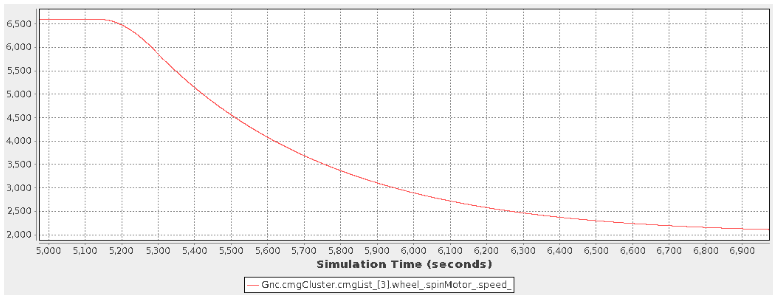

Figure 4: Bearing Seize Test (Applied 200 Times Drag Force)

5. Speedup Multiplier

A speedup multiplier variable is used to speed up the sim up to 25 times faster. This is not really a malfunction and occasionally a user would like to speed up the sim by some factor to verify whether the CMG wheel reaches the desired speed. The default simulation time step for the CMG model is 5 Hz (0.2 s) and the default multiplier variable is set to 1 for *nominal* speed mode. The speedup multiplier variable can only be changed for speed modes such as *spin-up*, *braking* or *coasting*. The user can set a value between 1 and 25 and any number greater than 25 will be reset to 25. Within the CMG model, the speedup multiplier value is multiplied by the time step dt and the SM voltage controller will generate the corresponding voltages to drive the motor faster. When the speedup multiplier variable is not equal to 1, the telemetry bit tests are skipped and the FSW will not see the speeding up the CMG sim and the large torques generated are not sent to the ISS dynamics. Figure 5 shows the results of a speedup multiplier test where the speedup multiplier value is 20. The wheel is sped up from 0 RPM to 6600 RPM in 20 min instead of 7 hours. Also, once the wheel reaches a speed of 6600 RPM the speedup multiplier value is reset to 1 and the mode becomes *nominal*, and at around 35 min, the wheel is commanded to *braking* mode by turning off the external power supplies. Once again, the speedup multiplier value is set back to 20 by the CMG model and the *braking* operation continues at 30 min for another 25 min before changing to *coasting* mode. Finally, the *coasting* mode ends at around 82 min.

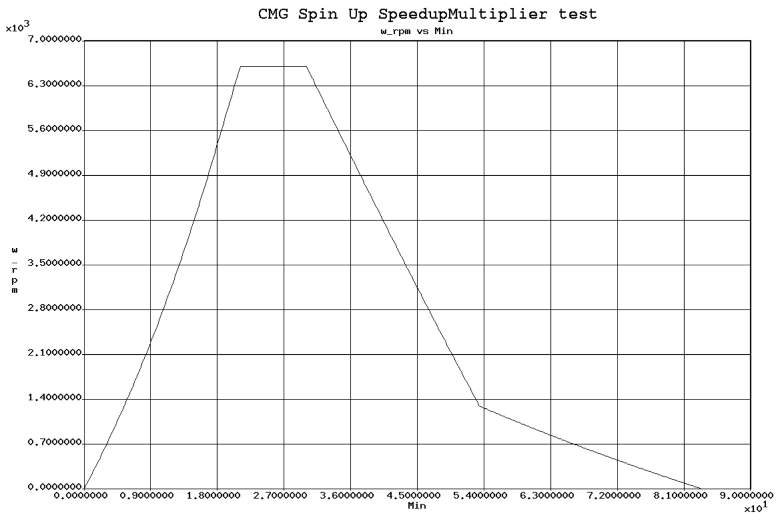

Figure 5: Speedup Multiplier Test

IV. Conclusion

This paper presents a simplified approach to simulation of several types of CMG malfunctions on board the ISS. All the malfunctions discussed in this paper are unit tested in a standalone setting and they are verified in the integrated simulation environment. The CMG model development and testing is completed and is part of the TS21 simulation for the users to verify the malfunction scenarios. The interesting aspect of this work is that the malfunction simulation parameters are fine-tuned to generate realistic data which will help the flight controllers train well on the GNC CMG simulation. For instance, the bearing seize malfunction allows the users to actually change the dynamics of the CMG which will affect the temperatures, currents and power usage in the CMG model. The malfunctions implemented based on the object oriented design facilitated the creation and testing of unit tests and integration tests. The simple malfunction models used in this paper can be reused in other applications which use a model-view-controller paradigm with command and telemetry capability.

V. Acknowledgments

The authors of this paper would like to thank the Johnson Space Center engineering directorate under the Training Systems 21 contract. We also acknowledge the GNC flight controllers and trainers at JSC for providing valuable feedback especially Everett Bolduc.

References

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

[2]Tremblay, P., ”Malfunction detection philosophy for a space station prototype environmental/thermal control and life support system,” *AIAA Paper*, Cocoa Beach, Florida, March 2728, 1972. Paper No. 72-241.

[3]Aziz, S., and Braithwaite, H. T., ”Evolution of the Malfunction Isolation and Recovery Methodology for Canada’s Mobile Servicing System (MSS),” *54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, Bremen, Germany, DOI: 10.2514/MIAF03, 2003.

[4]Gowan, John W, Jr., ”Ascent Trajectory Simulation for the Space Shuttle Launch Area Risk Assessment,” *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, San Francisco, California, Aug 15-18, 2005, Paper. No. AIAA-2005-6505.

[5]Beaty, James R, et al., ”Ares-I-X Vehicle Preliminary Range Safety Malfunction Turn Analysis,” *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 10 January 2008, Paper. No. AIAA-2008-191.

[6]”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.

[7]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.

[8]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.

[9]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.