ASEN 5010: Capstone Project

Jason L Kootsher
MSEE University of Colorado, Denver, Colorado, 80204, United States

The goal of this project is to create attitude and control laws for a low Mars orbit (LMO) craft so that the craft can 1) recharge its solar panels, 2) collect data from the planet surface, and 3) relay that data and communication information to a mother craft in geostationary Mars orbit (GMO). To do this, the craft must know not only its present location relative to the craft body frame, but its orientation relative to frames that enable the success of these criteria. There will be several frames and angular velocities that require inertial reference frame referencing, which will be necessary to control the LMO craft appropriately.

I. Nomenclature

B	=	body frame
H	=	Hill frame
O	=	general orbit frame
N	=	inertial frame
Rs	=	sun pointing reference frame
Rn	=	nadir pointing reference frame
Rc	=	communication pointing reference frame
$d\theta$	=	orbital rate
$d\boldsymbol{r}$	=	inertial rate
dt	=	time step
n	=	sample index
$\boldsymbol{\omega}$	=	angular velocity in the inertial frame
${}^{X}\mathbf{a}_{Y/Z}$	=	vector quantity a in the Y frame, measured relative to the Z frame, as observed in the X frame

II. Introduction

In order to analyze the problem as specified in the abstract, several subroutines and a object oriented (OO) software package were constructed using the numerical software Matlab. Representing both the low Mars orbit (LMO) and geostationary Mars orbit (GMO) craft as objects allowed for reuse of orientation data, simpler subroutine execution, and allowed for better overall organization. Likewise, creating several Matlab libraries for guidance, navigation, and control (GNC), math utilities, constants, and extraneous support allowed for more in depth debugging.

The breakout of tasks in this report each leverage off of the architecture layout as specified above, and each task can be categorized as either a single subroutine, or group of subroutines. The collection of all tasks allow for the creation of a simulation whose purpose will be to create a dynamic control system for the LMO craft so as to satisfy the problem criteria as described in this report's abstract.

III. Tasks

The following section outlines the steps that were taken in creating the simulation for resolving the requirements in the abstract.

A. Orbit Simulation

1 Senior Software Engineer, Lockheed Martin Space

In order to recreate the orbits of the LMO and GMO craft, it was assumed that both followed a circular trajectory. The LMO craft is positioned 400 kilometers (km) above the surface of Mars, and the GMO craft is 20424.2 km from the center of Mars. Along this circular trajectory, both craft move at a constant rate $d\theta$, such that they maintain a circular orbit. The initial conditions for the orientations are also given in the 3-1-3 symmetric Euler angle attitude sets:

Ω	i	θ(t)	Craft
20º	30º	60º	LMO
0º	0º	240º	GMO

Table 1a: Orbital Initial Conditions

The subroutine for the task of simulating orbits was constructed with inputs for the radius (altitude from center of Mars), as well as the craft's initial 3-1-3 attitude description. The purpose of this subroutine is to give the inertial frame (N) position **r** and velocity d**r** vectors of each craft in orbit. (See Appendix B for code).

Subroutine (pseudo) *See Appendix A for mathematical definitions

inertial(R,O=[Ω,i,θ(t)]**):**

Calculate B(**O**), the orbital rate to angular velocity transform (eulersum, append A)
Use B(**O**) to calculate **w** from $d\boldsymbol{\theta}$ (bmatrix, append A)
Calculate *H* from **O** (gnc.frames.HN, append B)
Use *H* to calculate **r**, $d\boldsymbol{r}$ from **R** and $\boldsymbol{\omega}$

Using the above subroutine gives the following plots for the LMO and GMO crafts, which achieves the desired result of transforming each respective craft's **r** and $d\boldsymbol{r}$ vectors from the orbital *H* frame to the inertial *N* frame.

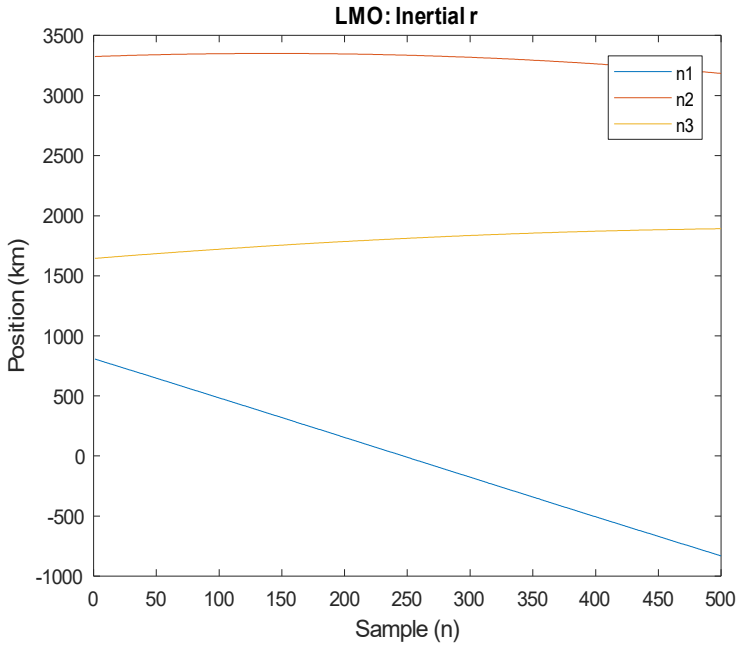

Figure 1a: LMO Inertial Position

1 Senior Software Engineer, Lockheed Martin Space

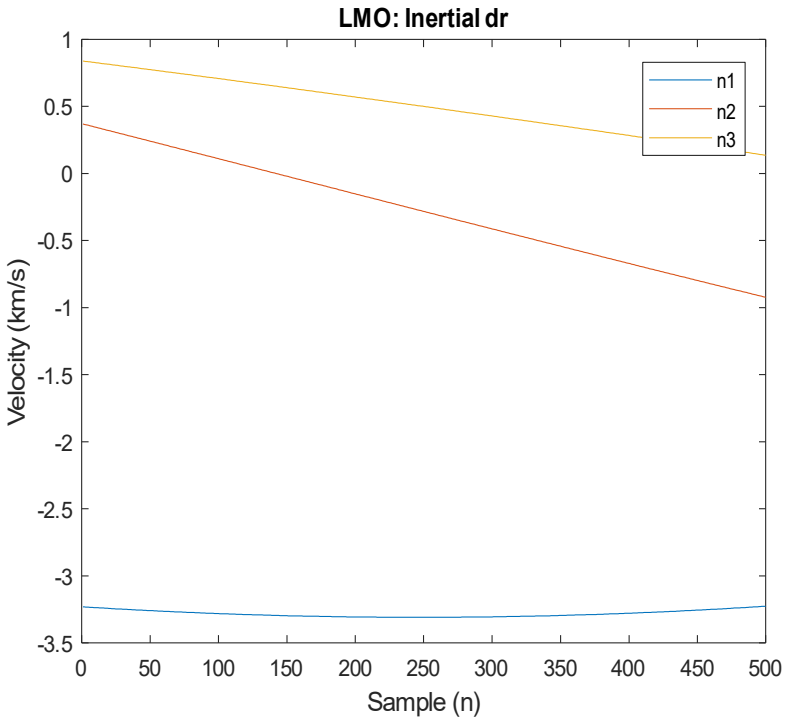

Figure 1b: LMO Inertial Velocity

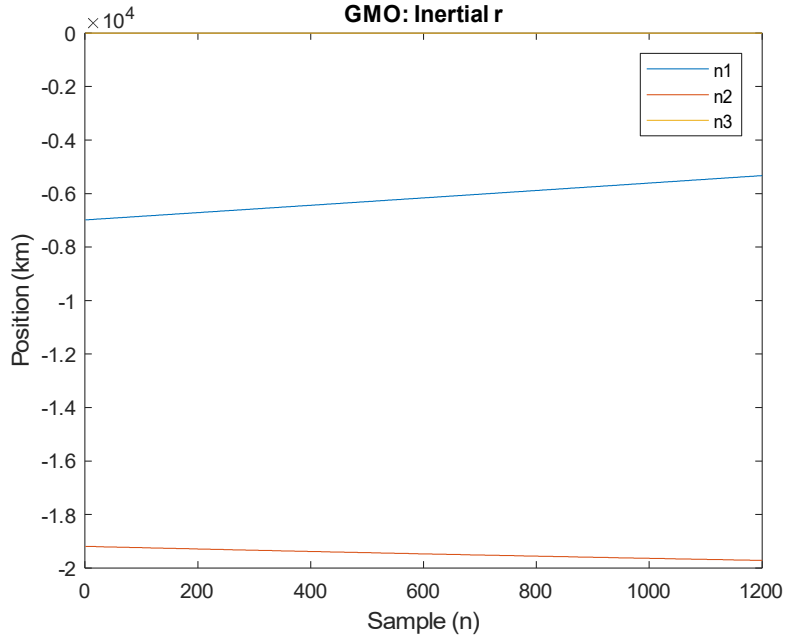

Figure 1c: GMO Inertial Position

1 Senior Software Engineer, Lockheed Martin Space

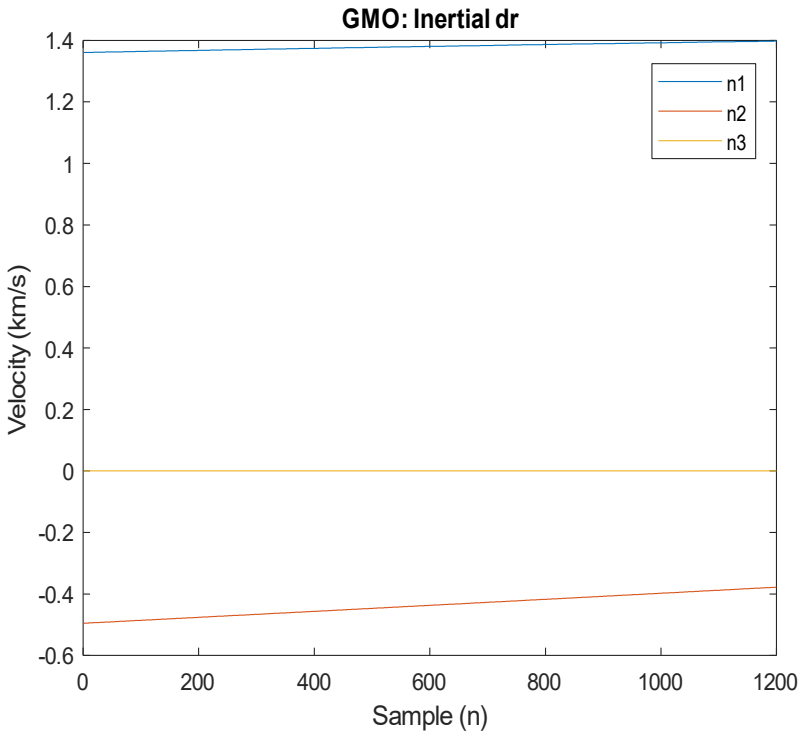

Figure 1d: GMO Inertial Velocity

Further study of these plots, when compared to initial conditions, confirms their accuracy. The LMO orbit does not lie in the elliptic plane of Mars, implying a nonzero n3 coordinate (Figure 1a), as well as a nonzero n3 velocity (Figure 1b). On the contrary, the GMO craft's states depicted by figures 1c and 1d, confirm the expected n3 equal to zero for a satellite lying in the elliptic plane of the body it is orbiting.

The above plots were analyzed with a sampling frequency of 1 Hz, such that the sample n corresponds 1:1 with the time t in seconds. The following values were obtained at the respective sample:

State	Time (s)	Sample	i	j	k
LMO**r**	450	450	-669.3 km	3227.5 km	1883.2 km
LMOd**r**	450	450	-3.2560 km/s	-0.7978 km/s	0.2101 km/s
GMO**r**	1150	1150	-5399 km	-19698 km	0 km
GMOd**r**	1150	1150	1.3966 km/s	-0.3828 km/s	0 km/s

Table 1b: LMO & GMO Query Results

Table 1b shows the general vector [i,j,k] for a frame (in this case *N*) at the respective sample n. These values represent the craft's state [**r**,d**r**,] as a function of time, and through simulation, match the expected truth data.

B. The Hill Frame

The Hill frame is formally defined in Appendix A, and this section will be brief due to the thorough analysis in section 3A. The Hill frame was used to transform the time varying 3-1-3 Euler angle set to *N*. The subroutine used to calculated *N* is given below as well as a brief analysis of the frame as compared to truth at time equals 300s:

1 Senior Software Engineer, Lockheed Martin Space

Subroutine (pseudo) *See Appendix A for mathematical definitions

hill(t):

Calculate total samples from t given the sampling frequency
Calculate **O**(n) given table 1a (LMO) (eulersum, append A)
Calculate *H* from **O**(n) (gnc.frames.HN, append B)

Time (s)	Sample	H_1	H_2	H_3
300	300	-0.046477	0.874148	0.483431
300	300	-0.984172	-0.122922	0.127651
300	300	0.171010	-0.469846	0.866025

Table 1b: Hill Frame at 300 Seconds

While table 1B may not be a truly rigorous analysis on its own, when coupled with section 3A, there is a higher degree of confidence added to its accuracy. Additionally, upon further study, it can be shown this frame does indeed belong to SO(3) and is orthogonal, which are required for *H* to be a direction cosine matrix.

C. The Sun Pointing Frame

The sun pointing frame is defined as a permutation of *N*, and as such, is the easiest of the three relative reference frames to simulate. It is defined in such a way so as to point the body axis b3, also referred to as the solar panel axis, toward the sun so as to charge the LMO batteries. In addition to the sun pointing frame, the angular velocity of the LMO, as seen in *Rs*, and relative to *N*, is also of interest. This angular velocity interpretation will be required for later sections where pointing control and error is discussed (sections F and H).

The following subroutine calculates *Rs* as well as angular velocity ${}^{N}\boldsymbol{\omega}_{R/N}$ for this frame:

Subroutine (pseudo) *See Appendix A for mathematical definitions

sun(t):

Call **hill(t)** and transpose the resulting matrix (Section B)
Multiply matrix by P to obtain *RsH* (gnc.frames.SN, append B)
Multiply *RsH* and *HN* and return resulting *Rs* (gnc.frames.SN, append B)

Call **inertial(R,O)** (Section A)
Calculate angular velocity in *H* (gnc.rates.orbit, append B)
Transform angular velocity from *H* to *Rs* using *Nrs*RsN*NH* (gnc.rates.sun, append B)

The final step in the subroutine may seem pointless due to the fact that the first two multipliers equate to identity, but in order to ensure the frame of the angular velocity is in fact correct, and the description is as we require it (sun relative to inertial) is as well, this must be performed. The sun pointing case is unique in that the matrix P is a permutation of identity, so this multiplier can easily be overlooked, but it is important to realize this is simply a special case. Ignoring this multiplier in any other circumstance will not yield the desired result. The permutation matrix P is defined below:

$$P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Equation 3a: Permutation Matrix P

1 Senior Software Engineer, Lockheed Martin Space

Due to the fact the matrix P is a permutation of the inertial frame, this implies the angular velocity in *Rs* relative to *N* will be a permutation of the velocity in *N*. Due to this, the motion in *Rs* as seen by *N* will appear stationary. Another way of phrasing this is to say that since all positions are simply reflections from their original vantage points, the relative motion between the reflection and the original state is the same. Mathematically, this means that ${}^{N}\boldsymbol{\omega}_{R/N}$ is the zero vector:

$$ {}^{N}\omega_{R/N} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} $$

Equation 3b: Inertial Frame Angular Velocity (sun pointing relative to inertial) (rad/s)

D. The Nadir Pointing Frame

Nadir pointing is calculated very similarly to sun pointing, but is a permutation of the Hill frame. The purpose of *Rn* is to align the +b1 axis so that it is normal to the surface of Mars. This simulates data collection from the planet's surface and is to be performed when we are not in solar charge mode or GMO communication mode. This will be discussed in later sections where control is the focus of discussion (refer to table 8a).

The following subroutine calculates *Rn* as well as angular velocity ${}^{N}\boldsymbol{\omega}_{R/N}$ for this frame:

Subroutine (pseudo) *See Appendix A for mathematical definitions

nadir**(t):**

Call **hill(t)** and transpose the resulting matrix	(Section B)
Multiply matrix by P to obtain *RnN*	(gnc.frames.PN, append B)
Call **inertial(R,O)**	(Section A)
Calculate angular velocity in *H*	(gnc.rates.orbit, append B)
Transform angular velocity from *H* to *Rn* using *Nrn*RsN*NH*	(gnc.rates.nadir, append B)

Similar to 3D, this frame is a permutation of a previously defined frame. However, unlike the previous section, there is a time dependence associated with the frame, seeing as how it is a permutation of *H*. This permutation is defined as follows:

$$ P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} $$

Equation 4a: Permutation Matrix P

The time dependence of the frame, along with the permutation P*H* leads to the following values at time 330s for the nadir pointing direction cosine matrix:

Time (s)	Sample	Rn_1	Rn_2	Rn_3
330	330	0.072582	-0.870578	-0.486648
330	330	-0.982592	-0.146079	0.114775
330	330	-0.171010	0.469846	-0.866025

Table 4a: Nadir Pointing at 330s

Additionally, since there is a time dependence, the angular velocity ${}^{N}\boldsymbol{\omega}_{R/N}$ will not be the zero vector. This is due to the fact that similarly to *H*, the motion in nadir pointing relative to *N* is varying. The angular velocity is therefore too a permutation of the angular velocity ${}^{N}\boldsymbol{\omega}_{H/N}$:

1 Senior Software Engineer, Lockheed Martin Space

$$ {}^{N}\omega_{R/N} = \begin{vmatrix} 0.000151 \\ -0.000416 \\ 0.000766 \end{vmatrix} $$

Equation 4b: Inertial Frame Angular Velocity (nadir pointing relative to inertial) (rad/s)

E. The GMO pointing frame

The communications frame *Rc*, or GMO pointing frame, is oriented such that the -b1 axis of the LMO points toward the GMO for the purpose of relaying data. The constraint on this frame is such that the GMO must be in a thirty five degree conic section in order for the communication relay to function properly. This will be discussed further in one of the later sections (table 8a).

The GMO pointing frame is defined as such:

$$ \hat{r}_1 = -({}^{N}\vec{r}_{GMO} - {}^{N}\vec{r}_{LMO}) $$

$$ \hat{r}_2 = -\frac{\hat{r}_1 \times \hat{n}_3}{||\hat{r}_1 \times \hat{n}_3||} $$

$$ \hat{r}_3 = \hat{r}_1 \times \hat{r}_2 $$

Equations 5a-5c: GMO pointing frame

As shown in equations 5a-5c, this frame is not a permutation of a previously defined frame and must be calculated for each sample. Upon construction of *Rc*, ${}^{N}\boldsymbol{\omega}_{R/N}$ is calculable only through finite differences. This is due to the complexity of the frame and the inability to simply permute the angular velocity. The following subroutine for the frame and angular velocity calculation shows this in more detail:

Subroutine (pseudo) *See Appendix A for mathematical definitions

communication(t):

Call **inertial(R,O)**	(Section A)
Call **hill(t)** and transpose the resulting matrix	(Section B)
Transform *H* state description to *N*	(Section B)
Calculate difference between inertial GMO and LMO positions	(gnc.frames.CN, append B)
Populate remaining frame basis vectors via eqs 5a-5c	(gnc.frames.CN, append B)
Repeat the above to obtain time t+1	(gnc.rates.comm, append B)
Perform finite difference using *Rc(t)* and *Rc(t+1)*	(gnc.rates.comm, append B)
Unpack the resulting cross product to obtain ${}^{N}\boldsymbol{\omega}_{R/N}$	(gnc.rates.comm, append B)

The resulting frame and angular velocity was analyzed via testing at time 330s to obtain the following data:

Time (s)	Sample	$\boldsymbol{Rc}_1$	$\boldsymbol{Rc}_2$	$\boldsymbol{Rc}_3$	${}^{N}\boldsymbol{\omega}_{R/N}$
330	330	0.265476	0.960928	0.078357	0.00001976 rad/s
330	330	-0.963892	0.266294	0.00	-0.00000546 rad/s
330	330	-0.020866	-0.075528	0.996925	0.00019129 rad/s

Table 5a: Communication (GMO) Pointing Frame and Resulting Angular Velocity at Time 330 Seconds

This data was taken using a 1Hz sampling frequency.

1 Senior Software Engineer, Lockheed Martin Space

F. Attitude Error Determination

In order to ensure controller accuracy of the frames described in sections 3C through 3E, an error tracking subroutine is introduced. This subroutine is examined for each frame at the initial time of t=0s:

Subroutine (pseudo) *See Appendix A for mathematical definitions

error(t):

Call <reference frame>**(t=0)**	(Section <C,D,or E>)
Calculate *B* from $\boldsymbol{\sigma}_{B/N}(t=0)$	(gnc.controls.afc, append B)
Construct *RB* frame DCM	(gnc.controls.afc, append B)
Calculate $\boldsymbol{\sigma}_{B/R}(t=0)$ and ${}^{B}\boldsymbol{\omega}_{B/R}(t=0)$	(gnc.controls.afc, append B)

The resulting error at the initial time zero is:

$\boldsymbol{\sigma}_{B/N}$	$\boldsymbol{\sigma}_{B/Rs}$	${}^{B}\boldsymbol{\omega}_{B/Rs}$	$\boldsymbol{\sigma}_{B/Rn}$	${}^{B}\boldsymbol{\omega}_{B/Rn}$	$\boldsymbol{\sigma}_{B/Rc}$	${}^{B}\boldsymbol{\omega}_{B/Rc}$
0.3 rad	-0.775421 rad	0.017453 rad/s	0.262265 rad	0.016849 rad/s	0.016972 rad	0.017297 rad/s
-0.4 rad	-0.473868 rad	0.030543 rad/s	0.554705 rad	0.030929 rad/s	-0.382803 rad	0.030657 rad/s
0.5 rad	0.043079 rad	-0.038397 rad/s	0.039424 rad	-0.038916 rad/s	0.207613 rad	-0.038437 rad/s

Table 6a: Frame Tracking Error for MRP Set and Associated Angular Velocity (*B* relative to *R*)

This subroutine is critical for feedback control as the goal of the controller should be to improve upon the accuracy over time by minimizing the error in the frames relative to the control torque in *B*. The next section covers this in greater detail.

G. Numerical Attitude Simulation

This section is broken into two parts; one for explanation of the Runge-Kutta $O(h^4)$ integrator (RK4), and another for the implementation/analysis of the integrator as it applies to the simulation.

GI. RK4 Integrator

The RK4 integrator is the integrator of choice for the main body of the simulation controller. It is used to integrate the MRP set $\boldsymbol{\sigma}_{B/N}(t)$, which in turn is used to calculate frame tracking errors for feedback control. The goal of the RK4 is to find a stable numerical approximation based on the step size h, that minimizes error over time in the integration. A visual summary of RK4 is below:

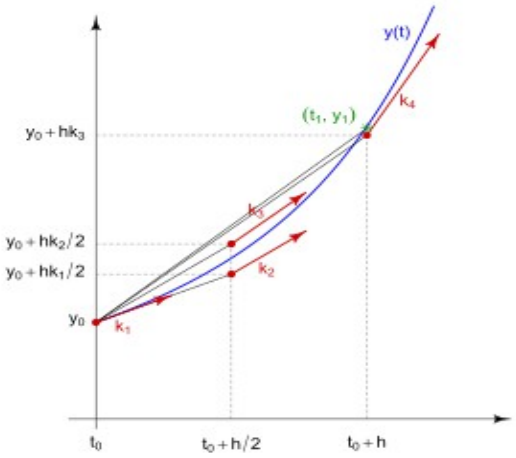

Figure 7a: RK4 Approximation (image courtesy of Wikipedia)

1 Senior Software Engineer, Lockheed Martin Space

The following subroutine in the simulation is the process for the RK4:

Subroutine (actual implementation)

integrate(f,g,u)

```
% Integrate the vector g at time i using the appropriate
% integrand function f

% Using a modified Runge-Kutta Method (RK4) where only discrete times
%  are input and a normalization factor is used to correct the offset.
h = tools.get.constant.value.h;

k1 = h*f(g,u);
k2 = h*f(g+k1/2,u);
k3 = h*f(g+k2/2,u);
k4 = h*f(g+k3,u);

k = (1/6)*(k1+2*k2+2*k3+k4);
```

This subroutine is the crux of the state determination for the whole of the sim and is the core code that drives the PD controller discussed in section H.

GII. Implementation Analysis

RK4 is referred to as a "work horse" as far as numerical integrators go, due to its robustness and ability to obtain high fidelity for appropriate choices of the time step. The integrator described in the previous section was verified as functional by calculating the angular velocity and MRP attitude description at select times, with a final check of its ability to handle piece-wise constant control signals (as this is what is required for the controller implementation).

For verification, the tables below summarizes the results of the test cases against the RK4 integrator implementation described in section GI:

H (*B*, t=500s)	H (*N*, t=500s)	T (t=500s)	$\sigma_{B/N}$ (u=0, t=500s)	$\sigma_{B/N}$ (u!=0, t=100s)
0.137897 kg*m^2/s	-0.264126 kg*m^2/s	0.009384 J	0.137659 rad	-0.22686074 rad
0.132662 kg*m^2/s	0.252782 kg*m^2/s	-	0.560270 rad	-0.64138592 rad
-0.316388 kg*m^2/s	0.055269 kg*m^2/s	-	-0.032173 rad	0.24254998 rad

Table 7a: Verification of RK4 for Varying State Data

In the above table, the **u** used for the MRP attitude descriptions corresponds to the following:

For **u**=0

$$\vec{u} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} Nm$$

Equation 7a: Null Control Torque

For **u**!=0

$$\vec{u} = \begin{vmatrix} 0.01 \\ -0.01 \\ 0.02 \end{vmatrix} Nm$$

Equation 7b: Constant Control Torque

The supporting subroutines used for angular momentum and rotational kinetic energy for this verification can be found in Appendix B, under the tools.physics library.

H. PD Controller

This section is dedicated to the controller at the core of the LMO simulation. The selected control algorithm is a proportional derivative controller, meaning there is a single feedback loop for driving error to smaller values. This is critical for the information outlined in section F, where error is introduced in the frame tracking estimation. Additionally, since this control law will require only the estimated parameters outlined in F, it will work across all 3 major reference frames; sun,

1 Senior Software Engineer, Lockheed Martin Space

nadir, and GMO. All three applications of the PD controller will be analyzed in this section.

The following control law describes the PD controller to be used:

$$^{B}\vec{u} = -K\vec{\sigma}_{B/R} - P^{B}\vec{\omega}_{B/R}$$

Equation 8a: PD Control Law

The gains K and P are the proportional and derivative gains respectably. The restriction on the system is that the slowest response decay time must be at most 120 seconds. Additionally, the system must be, at worst, critically stable, meaning an eigenvalue of 1. To calculate the appropriate values for each gain given these restrictions, the following equations are examined:

$$P = \frac{2I}{T}$$
$$K = 2\frac{P^2}{I}$$

Equation 8b: PD Control Gains

Using the given inertia tensor (See Appendix A), K and P can be reduced to scalars by taking the infinity norm of each matrix, respectively. The infinity norm will return the largest eigenvalue for a diagonal matrix (which is the form for I), which is representative of the spectral radius for the inertia tensor. This will ensure the controller will have a response that is at most, critically dampened.

The objective of this controller is to minimize frame tracking error and to orient the LMO to the required mode, given the attitude description at time t. The three modes and their attitude criteria are summarized in table 8a below.

Mode	Controller Frame	Description
Solar Array Charging	Sun Pointing	LMO inertial axis j is non-negative
Data Collection	Nadir Pointing	Not in either Sun or GMO Pointing
Communication	Communication (GMO) Pointing	LMO inertial axis j < 0 and angular separation from GMO is < 35 degrees

Table 8a: Control Mode Descriptions

From the above descriptions, it is easy to determine the inertial j axis value given the subroutine in section A. This is the method that was used in order to determine whether or not the LMO would utilize the Sun Pointing Control Mode (SPCM). Communication, or GMO, Pointing required more analysis, given the criteria being a conic section of 35 degrees. In order to accomplish this, and given SPCM is evaluated in *N*, it is sensible to evaluate the angle between GMO and LMO in *N* as well. To do this, the following projection was utilized to calculate the angle between the two at time t:

$$\vec{r}_{GMO} \cdot \vec{r}_{LMO} = \left|\vec{r}_{GMO}\right| \left|\vec{r}_{LMO}\right| cos(\theta(t))$$

Equation 8c: Projection of GMOr on LMOr

Given this information, establishing Nadir Pointing is as simple as failing to meet the criteria of the other two pointing modes. From here, the full simulation can be constructed. Below, in table 8b, are the MRP attitude results for each frame pointing mode given the above control description, and in table 8c, the attitude description at queried times throughout the 6500 second simulation run.

Time (s)	SPCM (rad)	NPCM (rad)	CPCM (rad)
15	0.265599 -0.159826 0.473328	0.291084 -0.191235 0.453508	0.265437 -0.168788 0.459491
100	0.168829 0.548230 0.578866	0.566220 -0.137224 0.152201	0.156142 0.221641 0.343165
200	-0.118127 -0.757860 -0.591490	0.796080 -0.459456 -0.126529	0.087276 0.119351 0.316191
400	-0.010111 -0.718841 -0.686069	-0.652998 0.534536 0.174547	0.004972 -0.016482 0.342390

Table 8b: Attitude Description for Different PD Control Modes

1 Senior Software Engineer, Lockheed Martin Space

Time (s)	i (rad)	j (rad)	k (rad)
300	-0.044221	-0.738551	-0.630653
2100	-0.745748	0.113589	0.158043
3400	0.013172	0.039819	0.390682
4400	-0.432869	-0.732540	-0.187818
5600	-0.000990	-0.825990	-0.504474

Table 8c: Full 6500 Second Simulation Run MRP Attitude Description Queries

The successful run of this simulation and the engagement of the various control modes can be further visualized in the figure below (figure 8a). It is easy to spot the transition points on this figure, as the various modes have very different behavior (due to the various orientations in which the controller is active).

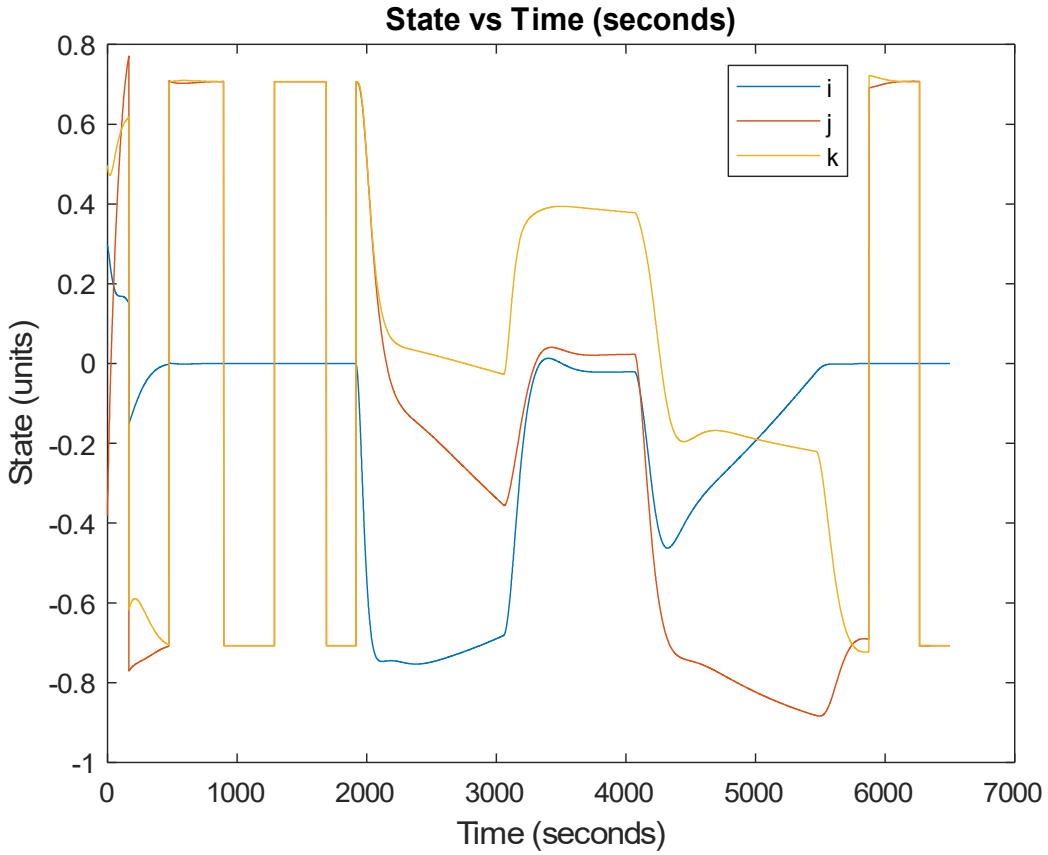

Figure 8a: Full Simulation Run (6500s)

IV. Conclusion

From the above simulation and analysis, the software and digital controller were successful in meeting the criteria laid out in the abstract of this report. A simple PD controller with feedback for minimizing frame tracking error proved to be a sufficient tool, given the relationships between frames were correct. While the mathematics presented in this report are not advanced, it did prove difficult to keep track throughout the subroutines as to which frame the data was present in. To rectify this, an object oriented approach was taken, which can be viewed in depth in Appendix B.

1 Senior Software Engineer, Lockheed Martin Space

Appendix A

This section is dedicated to the mathematical equations utilized throughout subroutines for the analysis component of this report.

Useful Equation Sets:

Bmatrix:

$$\begin{bmatrix} sin(x_3)sin(x_2) & cos(x_3) & 0 \\ cos(x_3)sin(x_2) & -sin(x_3) & 0 \\ cos(x_2) & 0 & 1 \end{bmatrix}$$

EulerSum:

$$k = f + (nh)df$$

Inertia Tensor I:

$$^B \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 7.5 \end{bmatrix}$$

H (angular momentum):

$$^B I \, ^B\vec{\omega}$$

T (rotational kinetic energy):

$$\frac{^B\vec{\omega} \, ^B I \, ^B\vec{\omega}}{2}$$

Appendix B

Github:

References

"Runge–Kutta Methods." *Wikipedia*, Wikimedia Foundation, 23 Apr. 2019, en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods.

Schaub, Hanspeter, and John L. Junkins. *Analytical Mechanics of Space Systems*. American Institute of Aeronautics and Astronautics, Inc., 2018.

1 Senior Software Engineer, Lockheed Martin Space