Attitude Dynamics and Control of a Nano-Satellite Orbiting Mars

Jash Bhalavat*
Department of Aerospace Engineering Sciences

This paper considers the dynamics of a nano-satellite in Low Mars Orbit (LMO) represented using Modified Rodrigues Parameters (MRPs) and controlled using thrusters to meet mission objectives such as data transfer to mother spacecraft in Geostationary Mars Orbit (GMO), sun-pointing, and nadir-pointing. All of this is simulated in MATLAB using a custom 4th order Runge-Kutta (RK4) numerical integrator. The attitude controller is a non-linear Proportional-Derivative (PD) controller and the gains for the controller are presented along with its performance during all mission modes. This project successfully this non-linear controller and the spacecraft is able to achieve all of its mission modes throughout the simulation.

I. Nomenclature

${}^N\boldsymbol{r}_{LMO}$	=	Inertial position of LMO spacecraft [km]
${}^N\boldsymbol{r}^T_{LMO}$	=	Transpose of Inertial position of LMO spacecraft [km]
${}^N\hat{\boldsymbol{n}}_1$	=	First axis that defines N frame
${}^N\boldsymbol{\omega}_{B/N}$	=	angular velocity of B frame with respect to (wrt) N frame represented in N frame [rad/s]
$\boldsymbol{\sigma}_{B/N}$	=	MRP that maps vectors in N frame to B frame
${}^B[I]$	=	Moment of Inertia expressed in B frame [kg-m^2]
$\dot{\boldsymbol{r}}$	=	Inertial derivative of $\boldsymbol{r}$ [km/s]
$[BN]$	=	DCM that maps vectors in N frame to B frame

II. Introduction

Consider a small spacecraft in low Mars Orbit (LMO) and a mother spacecraft in geosynchronous Mars orbit (GMO). Both these spacecrafts are in circular orbits around Mars. The LMO spacecraft's payload is a science instrument that needs to be pointed nadir (towards Mars' surface) to take measurements. Additionally, whenever the spacecraft's not eclipsed, it also needs to gather power by pointing its solar arrays to the sun. Lastly, it also needs to transfer the gathered science data to the GMO spacecraft. In order to control its attitude to achieve these attitude scenarios, the spacecraft is equipped with thrusters. The goal of this paper is to control this spacecraft to achieve these following attitude scenarios: 1) nadir-point, 2) GMO-point, and 3) sun-point.

Additionally, here are some key assumptions that are foundational to the simulation:

- The spacecraft orbit does not perturb and is perfectly circular
- The thrusters produce perfect torques, no misalignments or thruster leakage
- The spacecraft does not experience any external torques (gravity gradient, Solar Radiation Pressure (SRP), etc.)
- Sun's position is constantly infinitely far away in $\hat{\boldsymbol{n}}_2$ axis and when the spacecraft has a negative $\hat{\boldsymbol{n}}_2$ component, it is eclipsed
- Spacecraft's moment of inertia does not change as the controller uses the thrusters
- The discrete time step used throughout this simulation (for numerical integration, differentiation, etc.) is 1 second.

Various characteristics of these spacecrafts are presented in Table 1.

*University of Colorado Boulder, 3775 Discovery Dr, Boulder, CO 80303

Table 1 Spacecraft Characteristics

Characteristic	LMO	GMO
Orbit Radius	3796.2 km	20424.2 km
Orbit Rate	0.000884797 rad/s	0.0000709003 rad/s
$\sigma_{B/N}(t_0)$	[0.3, -0.4, 0.5]^T	-
${}^B\omega_{B/N}$	[1.0, 1.75, -2.20]$^T rad/s$	-
${}^B[I]$	diag([10.0, 5.0, 7.5]) kg m^2	-
Right Ascension of Ascending Node (Ω)	20°	0°
Inclination (i)	30°	0°
True Anomaly at initial time ($\theta(t_0)$)	60°	250°

A. LMO Spacecraft Geometry

The spacecraft body frame orientation is shown in Fig. 1.

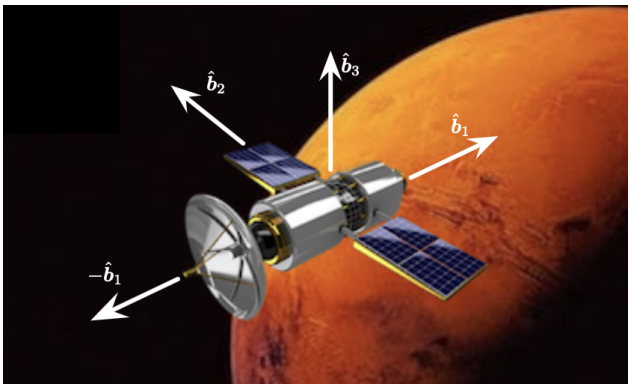

Fig. 1 LMO Spacecraft Body Frame [1]

As seen in Fig. 1, the body frame is given by $\{\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \hat{\mathbf{b}}_3\}$. The antenna that's used to communicate with the GMO spacecraft is aligned with $-\hat{\mathbf{b}}_1$. The science instrument is aligned with $\hat{\mathbf{b}}_1$ and the solar arrays are aligned with $\hat{\mathbf{b}}_3$.

III. Task 1 - Orbit Simulation

The purpose of this task is to derive the inertial spacecraft velocity vector $\dot{\boldsymbol{\nu}}$. The position vector of the LMO spacecraft is constantly ${}^O\mathbf{r} = r\hat{\mathbf{i}}_r$ in the orbit frame (where r is the radius of the circular orbit). To convert that to the inertial frame, the Euler angles need to be calculated. For the orbit frame, 3-1-3 Euler Angle set is used (Ω, i, θ). The initial Euler Angle set for the LMO spacecraft is given - (20°, 30°, 60°) and for the GMO spacecraft - (0°, 0°, 250°). Since, for both these orbits, we're not expecting any perturbations, the right ascension of the ascending node (Ω), and the inclination (i) do not change over time. But, the true anomaly (θ) increases linearly with time. The rate of change of the true anomaly is given as follows for the LMO and GMO spacecrafts:

$$\begin{aligned} \dot{\theta}_{lmo} &= \sqrt{\mu_{mars}/r^3} = 8.8480\text{e}{-4} rad/s \\ \dot{\theta}_{gmo} &= \sqrt{\mu_{mars}/r^3} = 0.0000709003 rad/s \\ \theta(t) &= \dot{\theta}t + \theta(t_0) \end{aligned} \tag{1}$$

where $\mu_{mars} = 42828.3$ km^3/s^2.

Using a simple for loop for the period of the spacecrafts, the 3-1-3 set can then be fully computed for 1 period. The period of an orbit is computed using the radius or the orbit and Mars' garvitational constant - $P = 2\pi\sqrt{r^3/\mu_{mars}}$ sec. Once, the euler angles are computed, the inertial position and velocities can also be computed using Eqn. 2:

$$
\begin{aligned}
{}^{N}\mathbf{r} &= [NO]\,{}^{O}\mathbf{r} \\
\dot{\mathbf{r}} &= \frac{{}^{O}d}{dt}\mathbf{r} + \boldsymbol{\omega}_{O/N} \times \mathbf{r} \\
{}^{N}\dot{\mathbf{r}} &= [NO]\left({}^{O}\boldsymbol{\omega}_{O/N} \times {}^{O}\,\mathbf{r}\right)
\end{aligned}
\tag{2}
$$

where [ON] can be computed by converting the Euler angles and [NO] is the transpose as shown in Eqn. 3:

$$
\begin{aligned}
[ON](t) &= R3(\theta) * R1(i) * R3(\Omega) \\
[NO](t) &= [ON](t)^{T}
\end{aligned}
\tag{3}
$$

$$
R3(x) = \begin{bmatrix} cos(x) & sin(x) & 0 \\ -sin(x) & cos(x) & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
R1(x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(x) & sin(x) \\ 0 & -sin(x) & cos(x) \end{bmatrix}
$$

Additionally, the angular velocity of the orbit frame with respect to the inertial frame is ${}^{O}\boldsymbol{\omega}_{ON} = [0, 0, \dot{\theta}]$ rad/s. The inertial position and velocity for the LMO and GMO spacecrafts at 450 sec and 1150 sec respectively are provided in Table 2:

Table 2 Task 1 Submissions

Spacecraft	Time [sec]	Inertial Position [km]	Inertial Velocity [km/s]
LMO	450	$[-669.29, 3227.5, 1883.2]^{T}$	$[-3.256, -0.79779, 0.21012]^{T}$
GMO	1150	$[-5399.1, -19698, 0]^{T}$	$[1.3966, -0.3828, 0]^{T}$

IV. Task 2 - Orbit Frame Orientation

The purpose of this task is to determine an analytical expression for $[\mathcal{NH}]$. The base vectors for the Hill frame are defined as follows:

$$
{}^{N}\hat{\mathbf{i}}_{r} = \frac{{}^{N}\mathbf{r}_{LMO}}{|{}^{N}\mathbf{r}_{LMO}|}, \quad {}^{N}\hat{\mathbf{i}}_{\theta} = \hat{\mathbf{i}}_{h} \times \hat{\mathbf{i}}_{r}, \quad {}^{N}\hat{\mathbf{i}}_{h} = \frac{{}^{N}\mathbf{r}_{LMO} \times {}^{N}\,\dot{\mathbf{r}}_{LMO}}{|{}^{N}\mathbf{r}_{LMO} \times {}^{N}\,\dot{\mathbf{r}}_{LMO}|}
\tag{4}
$$

The direction cosine matrix (DCM) for this can be found using the following equation:

$$
[\mathcal{NH}] = \{{}^{N}\hat{\mathbf{i}}_{r}, {}^{N}\,\hat{\mathbf{i}}_{\theta}, {}^{N}\,\hat{\mathbf{i}}_{h}\}
\tag{5}
$$

where the base vectors are 3x1 vectors and so the resulting DCM $[\mathcal{NH}]$ is a 3x3 DCM. In order to get the $[\mathcal{HN}]$ DCM, $[\mathcal{NH}]$ is transposed. The following is the DCM at 300 seconds (per task 2 submissions):

$$
[\mathcal{HN}_{300s}] = \begin{bmatrix} -0.046477 & 0.87415 & 0.48343 \\ -0.98417 & -0.12292 & 0.12765 \\ 0.17101 & -0.46985 & 0.86603 \end{bmatrix}
$$

V. Task 3 - Sun-Pointing Reference Frame Orientation

The base vectors for the Sun-Pointing Reference frame are defined as follows:

$$^{N}\hat{\mathbf{r}}_{s1} = [-1, 0, 0]^T, \quad ^{N}\hat{\mathbf{r}}_{s3} = [0, 1, 0]^T, \quad ^{N}\hat{\mathbf{r}}_{s2} = \hat{\mathbf{r}}_{s3} \times \hat{\mathbf{r}}_{s1} = [0, 0, 1]^T \tag{6}$$

The DCM for this can be found using the following equation:

$$[NR_s] = \{^{N}\hat{\mathbf{r}}_{s1}, ^{N}\hat{\mathbf{r}}_{s2}, ^{N}\hat{\mathbf{r}}_{s3}\} \tag{7}$$

where the base vectors are 3x1 vectors and so the resulting DCM $[NR_s]$ is a 3x3 DCM. In order to obtain $[R_sN]$ dcm, $[NR_s]$ is transposed. The following is the DCM at 0 seconds (per task 3 submissions) and beyond because the sun is inertially fixed:

$$[R_sN]_{0s} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Since, the sun is inertially fixed, the angular velocity of the sun-pointing reference frame is zero:

$$^{O}\boldsymbol{\omega}_{R_s/N} = ^{N}\boldsymbol{\omega}_{R_s/N} = [0, 0, 0]^T \text{ rad/s} \tag{8}$$

VI. Task 4 - Nadir-Pointing Reference Frame Orientation

The base vectors for the Nadir-Pointing Reference frame are defined as follows:

$$^{N}\hat{\mathbf{r}}_{n1} = \frac{-^{N}\mathbf{r}_{LMO}}{|^{N}\mathbf{r}_{LMO}|}, \quad ^{N}\hat{\mathbf{r}}_{n2} = \frac{^{N}\dot{\mathbf{r}}_{LMO}}{|^{N}\dot{\mathbf{r}}_{LMO}|}, \quad ^{N}\hat{\mathbf{r}}_{n3} = ^{N}\hat{\mathbf{r}}_{n1} \times ^{N}\hat{\mathbf{r}}_{n2} \tag{9}$$

The DCM for this can be found using the following equation:

$$[NR_n] = \{^{N}\hat{\mathbf{r}}_{n1}, ^{N}\hat{\mathbf{r}}_{n2}, ^{N}\hat{\mathbf{r}}_{n3}\} \tag{10}$$

where the base vectors are 3x1 vectors and so the resulting DCM $[NR_n]$ is a 3x3 DCM. In order to obtain $[R_nN]$ DCM, $[NR_n]$ is transposed. The following is the DCM at 330 seconds (per task 3 submissions):

$$[R_nN]_{330s} = \begin{bmatrix} 0.072582 & -0.87058 & -0.48665 \\ -0.98259 & -0.14608 & 0.11478 \\ -0.17101 & 0.46985 & -0.86603 \end{bmatrix}$$

Fig. 2 shows the base vectors (shown in the figure as r1, r2, r3) that make up the nadir-pointing reference frame.

It can be seen that the angular velocity of the Nadir-Pointing Reference frame only rotates along the 3rd basis vector (in the reference frame). That rotation is in the negative direction and exactly matches the true anomaly rate of change of the orbit. Hence, the angular velocity of the reference frame is $^{R_n}\boldsymbol{\omega}_{R_n/N} = [0, 0, -\dot{\theta}]$ rad/s. This can be represented in the inertial frame using the DCM calculated above: $^{N}\boldsymbol{\omega}_{R_n/N} = [NR_n]^{R_n}\boldsymbol{\omega}_{R_n/N}$. The angular velocity at 330 seconds is - $^{N}\boldsymbol{\omega}_{R_n/N} = [0.00015131, -0.00041572, 0.00076626]^T$ rad/s. Since, this is a circular orbit, it has a constant true anomaly rate of change. That is why this angular velocity is constant throughout the simulation as seen in Fig. 3.

VII. Task 5 - GMO-Pointing Reference Frame Orientation

The base vectors for the GMO-Pointing Reference frame are defined as follows:

$$^{N}\hat{\mathbf{r}}_{c1} = \frac{-\Delta\mathbf{r}}{|\Delta\mathbf{r}|}, \quad ^{N}\hat{\mathbf{r}}_{c2} = \frac{\Delta\mathbf{r} \times \hat{\mathbf{n}}_3}{|\Delta\mathbf{r} \times \hat{\mathbf{n}}_3|}, \quad ^{N}\hat{\mathbf{r}}_{c3} = ^{N}\hat{\mathbf{r}}_{c1} \times ^{N}\hat{\mathbf{r}}_{c2} \tag{11}$$

where $\Delta\mathbf{r} = ^{N}\mathbf{r}_{GMO} - ^{N}\mathbf{r}_{LMO}$. The DCM for this can be found using the following equation:

$$[NR_c] = \{^{N}\hat{\mathbf{r}}_{c1}, ^{N}\hat{\mathbf{r}}_{c2}, ^{N}\hat{\mathbf{r}}_{c3}\} \tag{12}$$

where the base vectors are 3x1 vectors and so the resulting matrix $[NR_c]$ is a 3x3 DCM. In order to obtain $[R_cN]$ DCM, $[NR_c]$ is transposed. The following is the DCM at 330 seconds (per task 3 submissions):

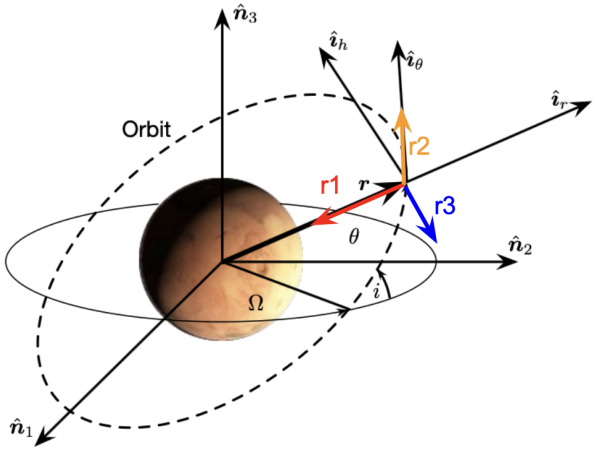

Fig. 2 Nadir-Pointing Reference Frame Base Vectors

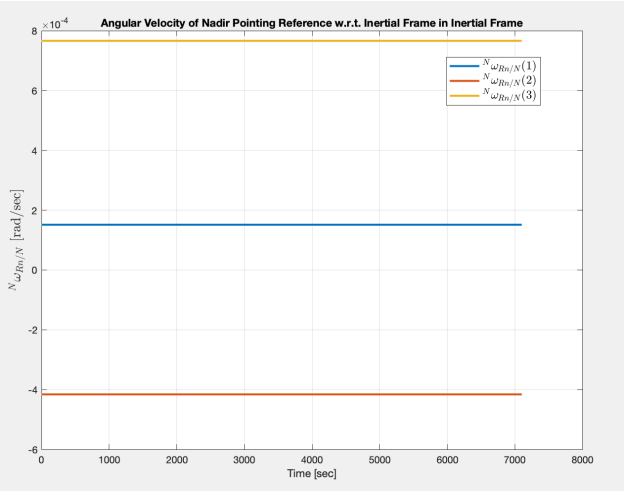

Fig. 3 Nadir-Pointing Reference Frame Angular Velocity

$$[R_cN]_{330s} = \begin{bmatrix} 0.26529113 & 0.96097738 & 0.07837785 \\ -0.96394274 & 0.26610975 & 0 \\ -0.02085711 & -0.07555176 & 0.99692372 \end{bmatrix}$$

The angular velocity of this reference frame can be calculated as such:

$$\begin{aligned} [\dot{R_cN}] &= -[^{R_c}\tilde{\omega}_{R_c/N}][R_cN] \\ [^{R_c}\tilde{\omega}_{R_c/N}] &= -[\dot{R_cN}] * [R_cN]^{-1} \\ [^{R_c}\tilde{\omega}_{R_c/N}] &= -[\dot{R_cN}] * [R_cN]^{T} \\ ^{R_c}\omega_{R_c/N} &= [-^{R_c}\tilde{\omega}_{R_c/N}(2,3), ^{R_c}\tilde{\omega}_{R_c/N}(1,3), ^{R_c}\tilde{\omega}_{R_c/N}(1,2)]^T \\ ^{N}\omega_{R_c/N} &= [R_cN]^T *^{Rc} \omega_{R_c/N} \end{aligned} \tag{13}$$

where the differential equation for the DCM is provided from [2].

This required a numerical differentiation of the DCM. A forward numerical differentiation is implemented which requires knowing the DCM for one discrete time step in the future along with the discrete time step:

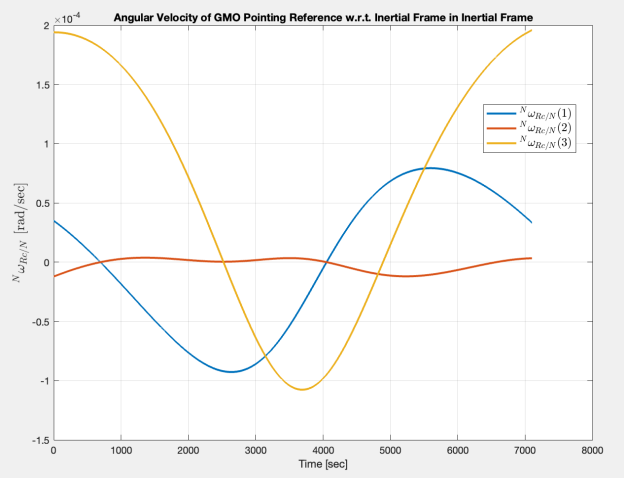

Fig. 4 Angular Velocity of GMO-Pointing Reference Frame with respect to Inerital Frame expressed in Inertial Frame

$$[\dot{R_c N}(t)] = \frac{[R_c N(t+1)] - [R_c N(t)]}{dt} \tag{14}$$

This is feasible for this simulation because the position and velocity vectors are deterministic and can be propagated one step ahead quite easily. The angular velocity at 330 seconds is - ${}^N\boldsymbol{\omega}_{R_c/N} = [0.00001976, -0.00000545, 0.00019129]^T$ rad/s. The full angular velocity vector is shown in Fig. 4.

VIII. Task 6 - Attitude Error Evaluation

Firstly, let's obtain $\boldsymbol{\sigma}_{B/R}$. This depends on the applicable pointing scenario - sun-point, nadir-point, or GMO-point. For this task, all three are calculated at the first time step. The pointing scenario selection leads to an output of the [RN] DCM. For this task, the MRP $\sigma_{B/N}$ for the initial time step is used - $\boldsymbol{\sigma}_{B/N}(t_0) = [0.3, -0.4, 0.5]^T$. Convert the given $\boldsymbol{\sigma}_{B/N}(t_0)$ to DCM [BN] as such:

$$[C] = [I_{3x3}] + \frac{8[\tilde{\boldsymbol{\sigma}}]^2 - 4(1-\sigma^2)[\tilde{\boldsymbol{\sigma}}]}{(1+\sigma^2)^2}$$
$$\sigma^2 = \boldsymbol{\sigma}^T\boldsymbol{\sigma} \tag{15}$$
$$[\tilde{\boldsymbol{x}}] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

(where [C] is a generic DCM and in this case it can be [BN] and $\boldsymbol{\sigma}$ is a generic MRP and in this case it can be $\boldsymbol{\sigma}_{B/N}$ and $\boldsymbol{x}$ is a generic vector and in this case can be $\boldsymbol{\sigma}_{B/N}$)

and then, $[BR] = [BN][RN]^T$. Then, convert the DCM [BR] to an MRP as such:

$$\boldsymbol{\sigma} = \frac{1}{\zeta(\zeta+2)} \begin{pmatrix} C_{23} - C_{32} \\ C_{31} - C_{13} \\ C_{12} - C_{21} \end{pmatrix} \tag{16}$$

(where [C] is a generic DCM and in this case can be [BR] and $\boldsymbol{\sigma}$ is a generic MRP and in this case can be $\boldsymbol{\sigma}_{B/R}$) Then, let's obtain the ${}^B\boldsymbol{\omega}_{B/R}$. Start with ${}^N\boldsymbol{\omega}_{N/R} = -{}^N\boldsymbol{\omega}_{R/N}$ and convert it to body frame ${}^B\boldsymbol{\omega}_{N/R} = [BN]{}^N\boldsymbol{\omega}_{N/R}$. Then add the angular velocity vectors to obtain the desired angular velocity vector: ${}^B\boldsymbol{\omega}_{B/R} = {}^B\boldsymbol{\omega}_{B/N} + {}^B\boldsymbol{\omega}_{N/R}$.

The attitude errors at the first time step are presented in Table 3 (task 6 submissions).

Table 3 Task 6 Submissions

Reference Frame	$\boldsymbol{\sigma}_{B/R}$	${}^{B}\boldsymbol{\omega}_{B/R}$
Sun-Pointing	[-0.77542077, -0.47386825, 0.04307893]	[0.01745329, 0.03054326, -0.03839724]
Nadir-Pointing	[0.26226523, 0.55470457, 0.0394240]	[0.01684883, 0.03092879, -0.03891576]
GMO-Pointing	[0.01697198, -0.38280275, 0.20761310]	[0.01729708, 0.03065743, -0.03843686]

IX. Task 7 - Numerical Attitude Simulator

Let's integrate the spacecrafts by using the following state vector:

$$X = \begin{bmatrix} \boldsymbol{\sigma}_{B/N} \\ {}^{B}\boldsymbol{\omega}_{B/N} \end{bmatrix}$$

The dynamics are governed by this differential equation:

$$\dot{X} = \begin{bmatrix} \dot{\boldsymbol{\sigma}}_{B/N} \\ {}^{B}\dot{\boldsymbol{\omega}}_{B/N} \end{bmatrix}$$

Assuming that the spacecrafts are rigid, the following dynamics are obeyed:

$$\begin{aligned} {}^{B}[I]\,{}^{B}\dot{\boldsymbol{\omega}}_{B/N} &= -{}^{B}[\tilde{\omega}_{B/N}]\,{}^{B}[I]\,{}^{B}\boldsymbol{\omega}_{B/N} + {}^{B}\,\mathbf{u} \\ \dot{\boldsymbol{\sigma}}_{B/N} &= \frac{1}{4}[(1-\sigma_{B/N}^{2})[I_{3x3}] + 2[\tilde{\sigma}_{B/N}] + 2\boldsymbol{\sigma}_{B/N}\boldsymbol{\sigma}_{B/N}^{T}]\,{}^{B}\boldsymbol{\omega}_{B/N} \end{aligned} \tag{17}$$

where ${}^{B}\mathbf{u}$ is the control vector expressed in the body frame. These equations are referenced from [2] (Eqn. 4.35, 3.160). A custom 4th order Runge-Kutta integrator as provided in [1] is implemented that follows the logic in Fig. 5:

$\boldsymbol{X}_0 = \begin{bmatrix} \boldsymbol{\sigma}_{B/N}(t_0) \\ {}^{B}\boldsymbol{\omega}_{B/N}(t_0) \end{bmatrix}$;
$t_{\text{max}} = \cdots$;
$\Delta t = \cdots$;
t_n = 0.0 ;
while $t_n < t_{max}$ **do**
  if *new control required* **then**
    Evaluate current reference frame states $[RN(t)]$, ${}^{N}\boldsymbol{\omega}_{R/N}(t)$, etc.;
    Determine control tracking errors $\boldsymbol{\sigma}_{B/R}$ and ${}^{B}\boldsymbol{\omega}_{B/R}$;
    Determine control solution $\boldsymbol{u}$;
  end
  $k_1 = \Delta t\boldsymbol{f}(\boldsymbol{X}_n, t_n, \boldsymbol{u})$;
  $k_2 = \Delta t\boldsymbol{f}(\boldsymbol{X}_n + \frac{k_1}{2}, t_n + \frac{\Delta t}{2}, \boldsymbol{u})$;
  $k_3 = \Delta t\boldsymbol{f}(\boldsymbol{X}_n + \frac{k_2}{2}, t_n + \frac{\Delta t}{2}, \boldsymbol{u})$;
  $k_4 = \Delta t\boldsymbol{f}(\boldsymbol{X}_n + k_3, t_n + \Delta t, \boldsymbol{u})$;
  $\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$;
  if $|\boldsymbol{\sigma}_{B/N}| > 1$ **then**
    map $\boldsymbol{\sigma}_{B/N}$ to shadow set;
  end
  $t_{n+1} = t_n + \Delta t$;
  save spacecraft states $\boldsymbol{X}$ and $\boldsymbol{u}$;
end

Algorithm 1: Coding Logic to Simulated Controlled Spacecraft Attitude Motion

Fig. 5 RK4 Logic

A while loop is instantiated and run for discrete time steps until the final time step is reached. This implementation used a constant time step of 1 second. As of this section, none of the controller logic is implemented yet. The slopes (k) are calculated using the spacecraft dynamics. These constants are used to calculate the state for the next time step. After this, the MRP's (first three components of the state) magnitude is checked. If it's greater than 1, the MRP can be switched to its shadow which provides the shorter rotation (<180°). The MRP can be switched to its shadow set using the following equation:

$$\sigma_i^S = \frac{-\sigma_i}{\sigma^2} \tag{18}$$

This is repeated for the next time step until the end of the loop. Using this integrator, angular momentum and rotational energies can also be obtained.

$$\begin{aligned}
{}^B H &= {}^B [I]\, {}^B\omega_{B/N} \text{ Nms} \\
T &= \frac{1}{2}\, {}^B\omega_{B/N}^T *^B [I] *^B \omega_{B/N} \text{ J} \\
{}^N\omega_{B/N} &= [BN]^T\, {}^B\omega_{B/N} \\
{}^N[I] &= [NB]^B[I][NB]^T = [BN]^T\, {}^B[I][BN] \text{ kg-m}^2 \\
{}^N\boldsymbol{H} &= {}^N [I]\, {}^N\omega_{B/N} \text{ Nms}
\end{aligned} \tag{19}$$

The first three components of the state output from the RK4 integrator is ${}^B\boldsymbol{\sigma}_{B/N}$ and the last three components are ${}^B\boldsymbol{\omega}_{B/N}$. The [BN] DCM can be converted from MRP using Eqn. 16.

Table 4 Task 7 Submissions

Time [sec]	Property	Value	Notes
500	${}^B\mathbf{H}$	[0.13789721, 0.13266205, -0.31638781] Nms	$\mathbf{u} = \mathbf{0}$
500	T	0.00938412 J	$\mathbf{u} = \mathbf{0}$
500	$\boldsymbol{\sigma}_{B/N}$	[0.13765932, 0.56027025, -0.03217283]	$\mathbf{u} = \mathbf{0}$
500	${}^N\mathbf{H}$	[-0.26412649, 0.25278185, 0.05526876] Nms	$\mathbf{u} = \mathbf{0}$
100	$\boldsymbol{\sigma}_{B/N}$	[-0.22686076, -0.64138593, 0.24254996]	$\mathbf{u} = [0.01, -0.01, 0.02]^T$ Nm

Firstly, the integration is performed with no control torque and simulated for 500 seconds as seen in Fig. 6. The angular momentum magnitude doesn't change because there is no external torque. This can be more evidently seen for the angular momentum in inertial frame. Similarly, the rotational energy is fairly constant. Small changes can be attributed to numerical errors.

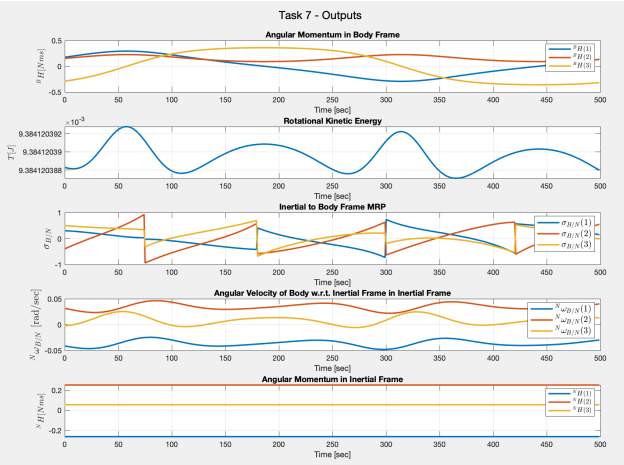

Fig. 6 Task 7 Outputs for 0 Control

Then, the integration is performed for a constant control torque of $[0.01, -0.01, 0.02]^T$ [Nm] and shown in Fig. 7. In this case, because of the constant control torque, the system angular momentum and rotational energy seems to increase over time unlike the zero control torque case. So, the spacecraft spins up and never steadies if a torque is applied constantly.

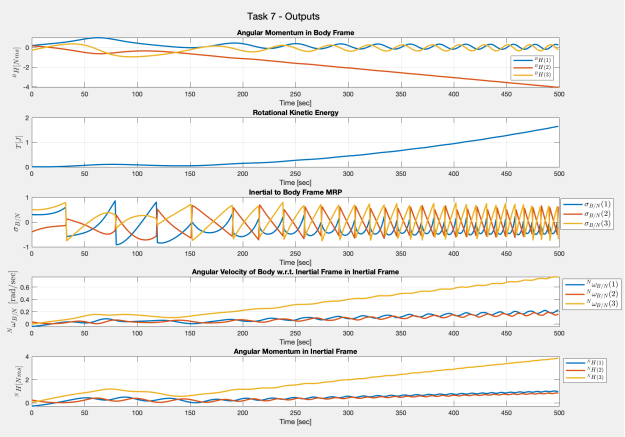

Fig. 7 Task 7 Outputs for constant control - [0.01, -0.01, 0.02]^T Nm

X. Task 8 - Sun Pointing Control

For this task, the integrator mentioned in Task 7 is implemented with feedback control (control in Task 7 was already given). The control is selected based on the following law:

$$^B\boldsymbol{u} = -K\boldsymbol{\sigma}_{B/R} - P^B\boldsymbol{\omega}_{B/R} \tag{20}$$

Here, the MRP of the reference frame and the angular velocities are computed depending on the selected mode. For this task, only the sun-pointing mode is implemented within the integrator. The feedback gains K and P are selected based on some required properties of the controller: The decay time constant of the controller has to be less than or equal to 120 seconds i.e. $T_i \leq 120s$, and the closed loop response for all $\boldsymbol{\sigma}_{B/N}$ should be either critically-damped, or under-damped i.e. $\zeta_i \leq 1$. These requirements aid in selecting feedback gains as provided in [2]:

$$\begin{aligned} T_i &= \frac{2I_i}{P_i} \\ P_i &= \frac{2I_i}{T_i} \end{aligned} \tag{21}$$

The moment of inertia are diag([10, 5, 7.5]) kg-m^2. So, in order to obtain all time constants below 120 s, the maximum moment of inertia must be used to calculate feedback gain P. Then, once P is calculated, the damping ratio equation can be used to calculate the feedback gain K:

$$\begin{aligned} \zeta_i &= \frac{P_i}{\sqrt{K_i I_i}} \\ K_i &= \frac{\left(\frac{P_i}{\zeta_i}\right)^2}{I_i} \end{aligned} \tag{22}$$

Similarly, in order to obtain $\zeta \leq 1$ for all moment of inertias, the minimum moment of inertia must be used to compute the feedback gain K. This leads to the feedback gains shown in Table. 5.

Table 5 Feedback Gains

Feedback Gain	Value
P	0.16667 kg-m^2/sec
K	0.00556 kg-m^2/sec^2

Using the sun-pointing reference frame (as computed in Task 3) and obtaining the attitude error (as computed in Task 6) within the numerical integrator (as shown in Task 7), the control law with the feedback gains is used to steer the

spacecraft such that its solar panel axis $\hat{b}_3$ points at the sun. This task requires $\boldsymbol{\sigma}_{B/N}$ at various times. That is presented below:

Table 6 Task 8 Submissions

Time [sec]	Property	Value
15	$\boldsymbol{\sigma}_{B/N}$	$[0.26559864, -0.15982644, 0.47332788]^T$
100	$\boldsymbol{\sigma}_{B/N}$	$[0.16882911, 0.54823028, 0.57886562]^T$
200	$\boldsymbol{\sigma}_{B/N}$	$[-0.11812708, -0.75786006, -0.59148988]^T$
400	$\boldsymbol{\sigma}_{B/N}$	$[-0.01011126, -0.71884140, -0.68606881]^T$

The state outputs over the spacecraft period are shown in Fig. 8.

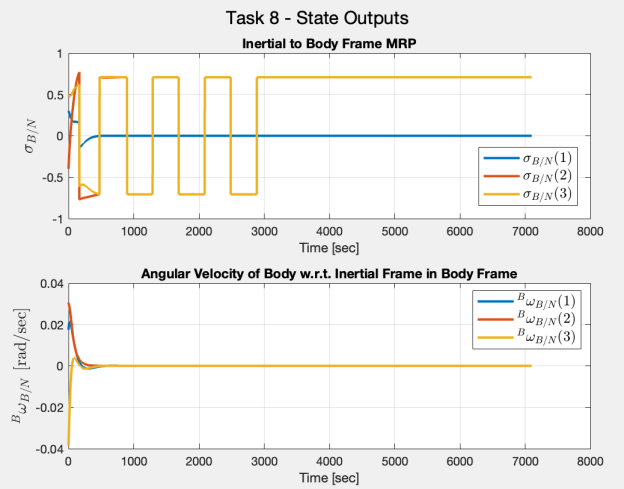

Fig. 8 Task 8 States

Furthermore, the MRP reference frame and the angular velocity along with the control are shown in Fig. 9. The discontinuous behavior can be seen because the MRP is switching to its shadow set. The angular velocity reduces to be 0 rad/sec over time because the sun is inertially fixed. This suggests that the controller is working properly.

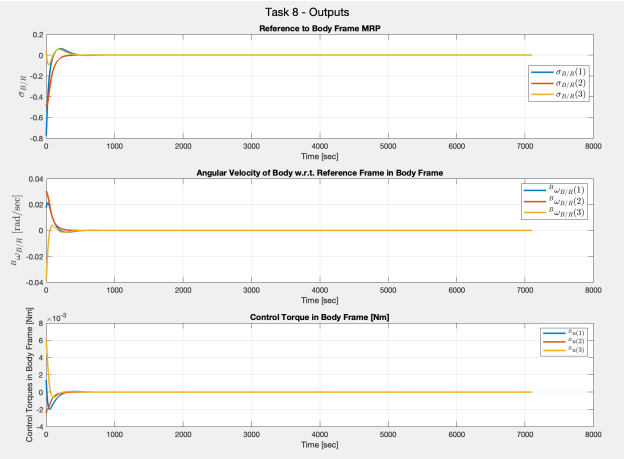

Fig. 9 Sun-Pointing Controller Outputs

As can be seen with the MRP subplot, the body frame is initially offset from the reference frame. Then, as the

controller acts on the spacecraft, the body frame aligns with the reference frame, and the angular velocity of both these frames also end up matching each other, which is why the steady-state response in each axis is close to 0 rad/s.

XI. Task 9 - Nadir Pointing Control

A similar process as Task 8 is implemented for this task. However, the reference frame is nadir-pointing. The same control law and feedback gains are used. Table 7 provides the submissions.

Table 7 Task 9 Submissions

Time [sec]	Property	Value
15	$\boldsymbol{\sigma}_{B/N}$	$[0.29107835, -0.19123835, 0.45350819]^T$
100	$\boldsymbol{\sigma}_{B/N}$	$[0.56612110, -0.13739225, 0.15220670]^T$
200	$\boldsymbol{\sigma}_{B/N}$	$[0.79577465, -0.45980282, -0.12651500]^T$
400	$\boldsymbol{\sigma}_{B/N}$	$[-0.65283837, 0.53489647, 0.17461124]^T$

The state outputs over the spacecraft period are shown in Fig. 10.

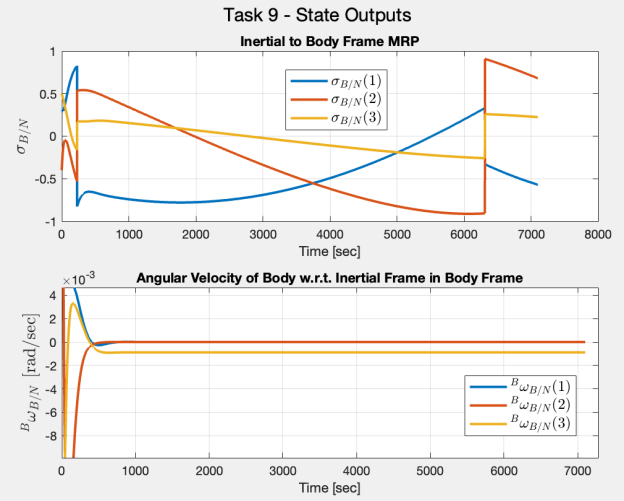

Fig. 10 Task 9 States

Since the controller is the same, a similar behavior as Task 8 can be observed in the outputs from Task 9 in Fig. 11.

The initial offset of the reference and body frames is reduced as the controller acts on the body. Once, the offset is removed, the body and reference frames maintain alignment for the rest of the simulation.

XII. Task 10 - GMO Pointing Control

A similar process as Task 8 and Task 9 is implemented for this task. However, the reference frame is GMO-pointing. The same control law and feedback gains are used. The MRPs are summarized below:

The state outputs over the spacecraft period are shown in Fig. 12.

Since, the controller is the same, a similar behavior as Task 8 and Task 9 can be observed in the outputs from Task 10 in Fig. 13.

The initial offset of the reference and body frames is reduced as the controller acts on the body. Once, the offset is removed, the body and reference frames maintain alignment for the rest of the simulation. Note that in this task, it is not checked if the GMO spacecraft is in the field of view of the LMO spacecraft and the controller is always enforced. This will be updated in the next task.

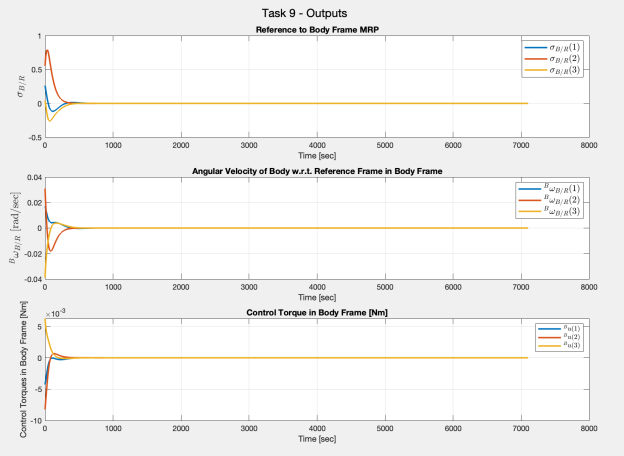

Fig. 11 Nadir-Pointing Controller Outputs

Table 8 Task 10 Submissions

Time [sec]	Property	Value
15	$\boldsymbol{\sigma}_{B/N}$	$[0.26543687, -0.16878831, 0.45949244]^T$
100	$\boldsymbol{\sigma}_{B/N}$	$[0.15614731, 0.22164134, 0.34318895]^T$
200	$\boldsymbol{\sigma}_{B/N}$	$[0.08728425, 0.11935199, 0.31623487]^T$
400	$\boldsymbol{\sigma}_{B/N}$	$[0.00497766, -0.01648733, 0.34243843]^T$

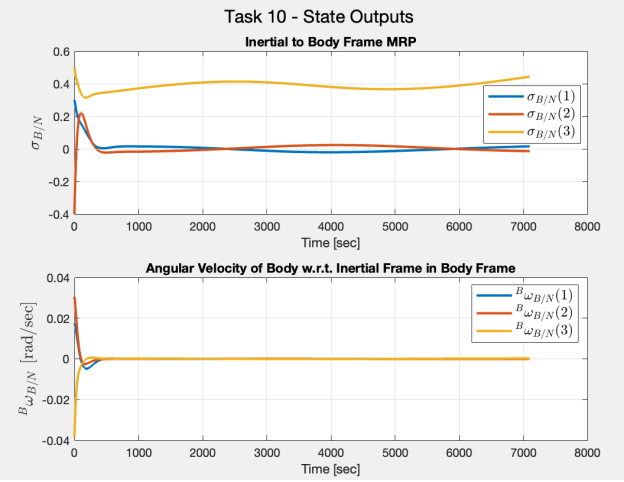

Fig. 12 Task 10 States

XIII. Task 11 - Mission Scenario Simulation

For this task, the full mission scenario is implemented. The pseudo-code shown in Fig. 14 is implemented within the integrator to compute the control:

At each time step, the spacecraft position is used to check whether it's on the sunlit side or the eclipsed side. If it is on the sunlit side, the spacecraft mode is chosen to be sun-pointing. If it is on the eclipsed side, the communication window is checked. If the GMO spacecraft is visible, then the GMO-pointing mode is chosen otherwise the nadir-pointing mode is chosen.

A couple of conditions need to be checked to implement the pseudo-code: is the spacecraft on the sunlit side? is the GMO spacecraft visible?

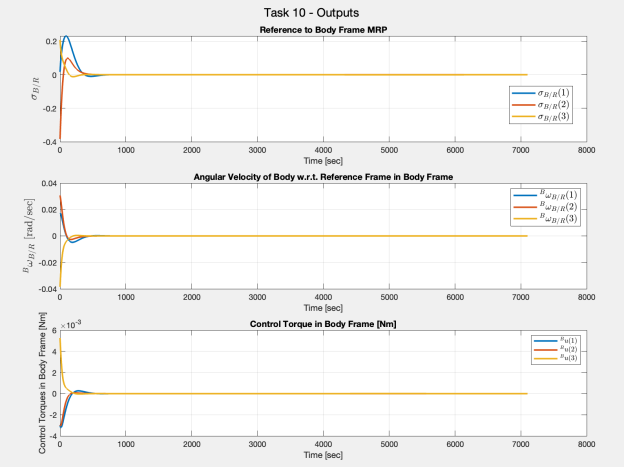

Fig. 13 Nadir-Pointing Controller Outputs

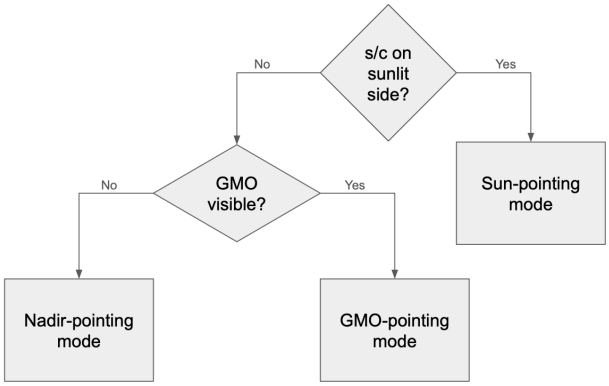

Fig. 14 Pseudocode to Choose Mode

To calculate whether the spacecraft is on the sunlit side or the eclipsed side, only the 2nd component of the inertial position vector of the spacecraft is checked. Since, the sun is on the positive $\hat{\boldsymbol{n}}_2$ axis, if the spacecraft's inertial position's 2nd component is negative, the spacecraft is on the eclipsed side and has to decide whether to be in nadir-pointing or GMO-pointing mode based on GMO visibility.. Otherwise, if its second component is positive, the spacecraft is on the sunlit side and must be in sun-pointing mode.

Secondly, the communication window of the LMO and GMO spacecrafts needs to be computed i.e. is GMO visible?. This is done by simply checking the angular difference between the inertial position vectors of the two spacecrafts. This angle difference can be calculated as follows:

$$\gamma = cos^{-1}\left(\frac{{}^{N}\boldsymbol{r}_{lmo} \cdot {}^{N}\boldsymbol{r}_{gmo}}{|{}^{N}\boldsymbol{r}_{lmo}||{}^{N}\boldsymbol{r}_{gmo}|}\right) \tag{23}$$

If this angle difference is less than or equal to 35°, then the GMO spacecraft is "visible" to the LMO spacecraft and the GMO-pointing reference frame is used to calculate control.

Once the correct reference frame is chosen, this task follows the same process as tasks 8, 9, 10 by calculating the attitude error given the reference frame attitude and angular velocities. Then, the control torque is calculated using the same gains as the previous tasks and that is numerically integrated. This task is propagated for 6500 seconds.

The state outputs for 6500 seconds are shown in Fig. 15. And, the reference MRP and angular velocity along with control are shown in Fig. 16.

(The blue shaded region called communication window is the region where the angle difference between the LMO and GMO spacecrafts is less than or equal to 35° i.e. GMO-spacecraft is visible to the LMO spacecraft. The grey region

Table 9 Task 11 Submissions

Time [sec]	Property	Value
300	$\boldsymbol{\sigma}_{B/N}$	$[-0.04422057, -0.73855063, -0.63065311]^T$
2100	$\boldsymbol{\sigma}_{B/N}$	$[-0.74576509, 0.11392308, 0.15812376]^T$
3400	$\boldsymbol{\sigma}_{B/N}$	$[0.01316091, 0.03981289, 0.39066826]^T$
4400	$\boldsymbol{\sigma}_{B/N}$	$[-0.43315160, -0.73234268, -0.18772582]^T$
5600	$\boldsymbol{\sigma}_{B/N}$	$[-0.00115033, -0.82595563, -0.50443636]^T$

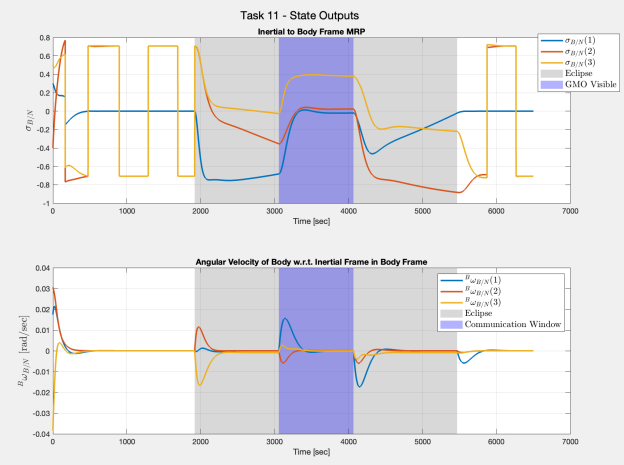

Fig. 15 Task 11 States

is when the spacecraft is eclipsed.)

Compared to the previous 3 tasks, now the spacecraft does not stick to one reference mode during the full duration of the simulation. It switches between sun-pointing, nadir-pointing, and GMO-pointing modes. Initially, since the spacecraft is on the sunlit side, the controller orients the spacecraft in the sun-pointing mode. Then, as the spacecraft is eclipsed (and GMO is not visible), the spacecraft switches to nadir-pointing mode and when the GMO spacecraft becomes visible, the spacecraft switches to GMO-pointing mode. This keeps goings as the conditions change and the spacecraft goes out of the GMO visibility region and out of eclipse. During all of this, the controller is also excited every time a mode change occurs to re-orient the spacecraft. Between these regions, it can be seen that the spacecraft achieves some sort of steady motion right before it has to switch again.

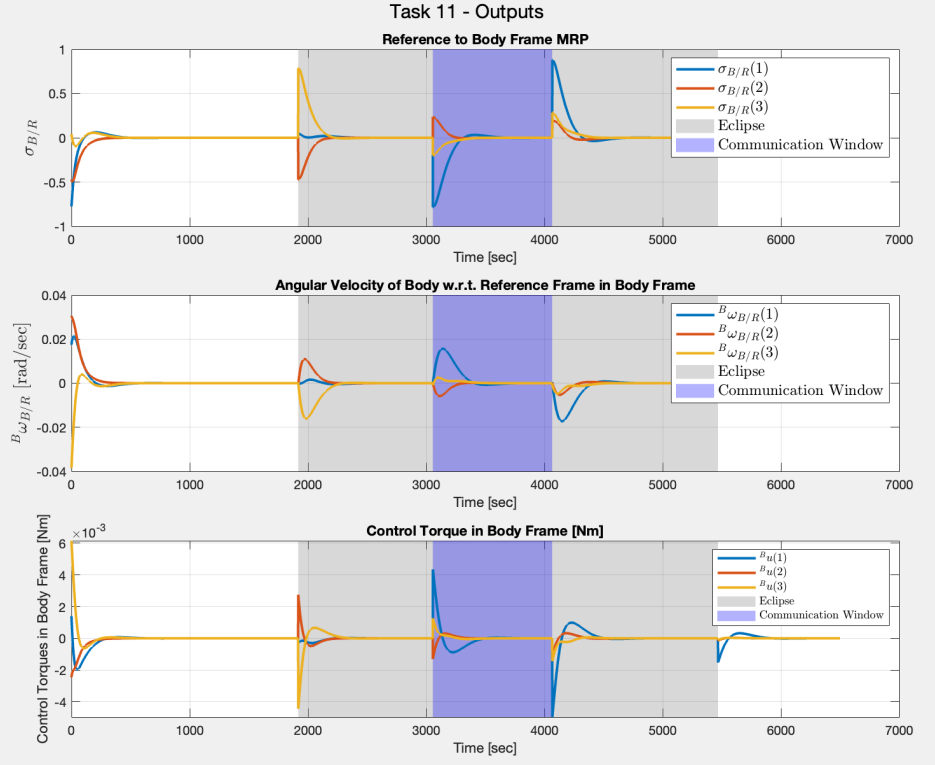

Fig. 16 Full Mission Scenario Outputs

XIV. Conclusion

This paper illustrates a simple spacecraft simulation around Mars with a custom numerical integrator utilizing MRPs and a non-linear PD controller. It can be seen through the various control tasks that such a controller can orient the spacecraft to perform simple mission modes such as sun-pointing, nadir-pointing, and GMO-pointing. However, this simulation does not take into account external environmental torques, variable thrust per thruster, propellant leakage, thruster misalignment, orbit perturbation, etc. However, given the duration of this simulation is less than or equal to 1 orbit, this is a great first-order approximation that is modular and can be built upon to achieve higher fidelity. The high-level goal of designing a thruster-based attitude controller for a nano-satellite is met through this project.

Appendix

A. Matlab Code

```
clear; clc; close all;

% ASEN 5010 – Project Code
% Spring 2025
% Jash Bhalavat

%%% Given

r_mars = 3396.19; % km
h = 400; % km
r_lmo = r_mars + h;

mu_mars = 42828.3; % km^3/s^2
```

```
theta_dot_lmo = sqrt(mu_mars/r_lmo^3);

r_gmo = 20424.2; % km
theta_dot_gmo = 0.0000709003; % rad/s

antenna_n_hat_b = [-1, 0, 0]';
sensor_n_hat_b = [1, 0, 0]';
sa_n_hat_b = [0, 0, 1]';

sigma_BN_t0 = [0.3, -0.4, 0.5]';
omega_BN_B_t0 = [1, 1.75, -2.2]' * pi/180; % rad/s
I_B = diag([10, 5, 7.5]); % kg-m^2

omega_0_lmo = 20 * pi/180;
i_0_lmo = 30 * pi/180;
theta_0_lmo = 60 * pi/180;
period_lmo = 2*pi*sqrt(r_lmo^3/mu_mars);

omega_0_gmo = 0 * pi/180;
i_0_gmo = 0 * pi/180;
theta_0_gmo = 250 * pi/180;
period_gmo = 2*pi*sqrt(r_gmo^3/mu_mars);

n_lmo = ceil(period_lmo); % time steps per orbit
n_gmo = ceil(period_gmo);

t_lmo = 0:n_lmo;
t_gmo = 0:n_gmo;

%% Task 1

for i = 1:length(t_lmo)
    theta_lmo(i,1) = wrapTo2Pi(theta_dot_lmo*t_lmo(i) + theta_0_lmo);
    ea_ON_lmo(i,:) = [omega_0_lmo, i_0_lmo, theta_lmo(i,1)];
    [r_N_lmo(i,:), r_dot_N_lmo(i,:)] = inertial_velocity(r_lmo, ea_ON_lmo(i,:), theta_dot_l
end

for i = 1:length(t_gmo)
    theta_gmo(i,1) = wrapTo2Pi(theta_dot_gmo*t_gmo(i) + theta_0_gmo);
    ea_ON_gmo(i,:) = [omega_0_gmo, i_0_gmo, theta_gmo(i,1)];
    [r_N_gmo(i,:), r_dot_N_gmo(i,:)] = inertial_velocity(r_gmo, ea_ON_gmo(i,:), theta_dot_g
end


%% Task 2

% [NT] = [t1_N, t2_N, t3_N]
% Therefore, [NH] = [ir_N, itheta_N, ih_N];

% @300 seconds, t_lmo index = 301
t_300_ind = 301;

DCM_HN_at_300 = orbit_frame(t_300_ind, r_N_lmo, r_dot_N_lmo);
```

```
%% Task 3

[DCM_RsN_at_0, omega_RsN_N_at_0] = sun_reference_frame();

function [DCM_RsN, omega_RsN_N] = sun_reference_frame()
    r1_N = [-1, 0, 0]';
    r3_N = [0, 1, 0]';
    r2_N = cross(r3_N, r1_N);

    % NR = [r1_N, r2_N, r3_N]
    NRs = [r1_N, r2_N, r3_N];
    DCM_RsN = NRs';

    omega_RsN_N = [0, 0, 0]';
end

%% Task 4

for i = 1:length(t_lmo)
    [DCM_RnN(:,:,i), omega_RnN_N(:,i)] = nadir_reference_frame(r_N_lmo(i,:)', r_dot_N_lmo(i
end

DCM_RnN_at_330 = DCM_RnN(:,:,331);
omega_RnN_N_at_330 = omega_RnN_N(:,331);

% DCM_RnN_at_330 = nadir_reference_frame_orientation;

function [DCM_RnN, omega_RnN_N] = nadir_reference_frame(r_N, v_N, theta_dot)
    r1_N = -r_N/norm(r_N);
    r2_N = v_N/norm(v_N);
    r3_N = cross(r1_N, r2_N);

    % NR = [r1_N, r2_N, r3_N]
    NRn = [r1_N, r2_N, r3_N];
    DCM_RnN = NRn';

    % omega_Rn/N_Rn = [0, 0, -theta_dot]
    omega_RnN_Rn = [0, 0, -theta_dot]';
    omega_RnN_N = DCM_RnN' * omega_RnN_Rn;
end


%% Task 5

[DCM_RcN_at_330, omega_RcN_N_at_330] = gmo_reference_frame(r_N_lmo(332,:)', r_N_gmo(332,:)
% [DCM_RcN_at_330, omega_RcN_N_at_330] = gmo_reference_frame(r_N_lmo(16502,:)', r_N_gmo(16

function [DCM_RcN, omega_RcN_N] = gmo_reference_frame(r_LMO_N, r_GMO_N, t, prev_r_LMO_N, pr
    % Time step - 330 seconds
    prev_r_GMO_wrt_LMO_N = prev_r_GMO_N - prev_r_LMO_N;
    prev_r1 = -prev_r_GMO_wrt_LMO_N/norm(prev_r_GMO_wrt_LMO_N);
    n3 = [0 0 1]';
    prev_r2 = cross(prev_r_GMO_wrt_LMO_N, n3) / norm(cross(prev_r_GMO_wrt_LMO_N, n3));
    prev_r3 = cross(prev_r1, prev_r2);
```

```
    prev_NRc = [prev_r1, prev_r2, prev_r3];
    prev_DCM_RcN = prev_NRc';

    % Time step - 331 seconds
    r_GMO_wrt_LMO_N = r_GMO_N - r_LMO_N;
    r1 = -r_GMO_wrt_LMO_N/norm(r_GMO_wrt_LMO_N);
    n3 = [0 0 1]';
    r2 = cross(r_GMO_wrt_LMO_N, n3) / norm(cross(r_GMO_wrt_LMO_N, n3));
    r3 = cross(r1, r2);

    NRc = [r1, r2, r3];
    DCM_RcN = NRc';

    % To calculate omega_RcN_N, use the following property:
    % C_dot = -omega_tilde * C
    % omega_tilde = -C_dot * C'
    % Where C_dot is numerically calculated by DCM_diff/dt
    dt = t - prev_t;
    dcm_diff = DCM_RcN - prev_DCM_RcN;
    dcm_RcN_dot = dcm_diff/dt;
    omega_tilde_Rc = -dcm_RcN_dot * DCM_RcN';
    omega_RcN_Rc = [-omega_tilde_Rc(2,3); omega_tilde_Rc(1,3); -omega_tilde_Rc(1,2)];
    omega_RcN_N = DCM_RcN' * omega_RcN_Rc;
end



%% Task 6

% At t=0
DCM_RsN_at_0 = sun_reference_frame();
omega_RsN_N_at_0 = [0 0 0]';
[DCM_RnN_at_0, omega_RnN_N_at_0] = nadir_reference_frame(r_N_lmo(1,:)', r_dot_N_lmo(1,:)',
[DCM_RcN_at_0, omega_RcN_N_at_0] = gmo_reference_frame(r_N_lmo(1,:)', r_N_gmo(1,:)', t_lmo

[sigma_BRs_at_0, omega_BRs_B_at_0] = attitude_error(t_lmo(1), sigma_BN_t0, omega_BN_B_t0, I
[sigma_BRn_at_0, omega_BRn_B_at_0] = attitude_error(t_lmo(1), sigma_BN_t0, omega_BN_B_t0, I
[sigma_BRc_at_0, omega_BRc_B_at_0] = attitude_error(t_lmo(1), sigma_BN_t0, omega_BN_B_t0, I

function [sigma_BR, omega_BR_B] = attitude_error(t, sigma_BN, omega_BN_B, DCM_RN, omega_RN_
    DCM_BN = mrp_to_dcm(sigma_BN);
    DCM_BR = DCM_BN * DCM_RN';
    sigma_BR = dcm_to_mrp(DCM_BR);

    omega_NR_B = DCM_BN * -omega_RN_N;
    omega_BR_B = omega_BN_B + omega_NR_B;
end


%% Task 7

state_0 = [sigma_BN_t0; omega_BN_B_t0];
t = t_lmo;
```

```
u_in = [0; 0; 0];

% [state, u] = rk4(@(state, t, u_in, I_B)diff_eq(state, t, u_in, I_B), t, state_0, I_B, u_

% t = 500 s @ t ind --> 1001
% omega_BN_B_at_500 = state(1001, 4:6)';
% H_B_at_500 = I_B * omega_BN_B_at_500;
% T_at_500 = 1/2 * omega_BN_B_at_500' * I_B * omega_BN_B_at_500;
%
% sigma_BN_at_500 = state(1001, 1:3)';
% DCM_BN_at_500 = mrp_to_dcm(sigma_BN_at_500);
% omega_BN_N_at_500 = DCM_BN_at_500' * omega_BN_B_at_500;
% I_N = DCM_BN_at_500' * I_B * DCM_BN_at_500;
% H_N_at_500 = I_N * omega_BN_N_at_500;
%
% u_in = [0.01; -0.01; 0.02];
% [state, u] = rk4(@(state, t, u_in, I_B)diff_eq(state, t, u_in, I_B), t, state_0, I_B, u_

% sigma_BN_at_100 = state(201, 1:3)';

function state_dot = diff_eq(state, t, u, I)
    % 6x1 state EOMs
    % Inputs:
    % state - [sigma_BN'; omega_BN_B']
    % I - 3x3 principal MOI [kg-m^2]
    %
    % Output:
    % state_dot - [sigma_BN_dot', omega_BN_B_dot']
    sigma_BN = state(1:3)';
    omega_BN_B = state(4:6)';

    sigma_sq = dot(sigma_BN, sigma_BN);
    sigma_BN_dot = 1/4 * ((1-sigma_sq)*eye(3) + 2*skew_symmetric(sigma_BN) + 2*(sigma_BN*si

    omega_BN_B_dot(1,1) = -(I(3,3) - I(2,2))/I(1,1) * omega_BN_B(2) * omega_BN_B(3) + u(1)/
    omega_BN_B_dot(2,1) = -(I(1,1) - I(3,3))/I(2,2) * omega_BN_B(1) * omega_BN_B(3) + u(2)/
    omega_BN_B_dot(3,1) = -(I(2,2) - I(1,1))/I(3,3) * omega_BN_B(2) * omega_BN_B(1) + u(3)/

    state_dot = [sigma_BN_dot; omega_BN_B_dot];
end


%% Task 8

T = 120;
zeta = 1;

P = 2*I_B(1,1)/T;
K = (P/zeta)^2/I_B(2,2);

[state_task8, u_task8, sigma_BR_task8, omega_BR_B_task8] = rk4_task8(@(state, t, u_B, I_B)

sigma_BN_at_15 = state_task8(16,1:3);
sigma_BN_at_100 = state_task8(101,1:3);
```

```
sigma_BN_at_200 = state_task8(201,1:3);
sigma_BN_at_400 = state_task8(401,1:3);

print_array(sigma_BN_at_15)
print_array(sigma_BN_at_100)
print_array(sigma_BN_at_200)
print_array(sigma_BN_at_400)

%% Figures

figure()
subplot(2,1,1)
plot(t, state_task8(1:end-1,1:3), 'LineWidth',2)
legend("$\sigma_{B/N}(1)$", "$\sigma_{B/N}(2)$", "$\sigma_{B/N}(3)$", 'FontSize', 12, Inte
ylabel("$\sigma_{B/N}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Inertial to Body Frame MRP")

subplot(2,1,2)
plot(t, state_task8(1:end-1,4:6), 'LineWidth',2)
legend("$^B\omega_{B/N}(1)$", "$^B\omega_{B/N}(2)$", "$^B\omega_{B/N}(3)$", 'FontSize', 12
ylabel("$^B\omega_{B/N}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Inertial Frame in Body Frame")

sgtitle("Task 8 - State Outputs")

figure()
subplot(3,1,1)
plot(t, sigma_BR_task8, 'LineWidth',2)
legend("$\sigma_{B/R}(1)$", "$\sigma_{B/R}(2)$", "$\sigma_{B/R}(3)$", 'FontSize', 12, Inte
ylabel("$\sigma_{B/R}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Reference to Body Frame MRP")

subplot(3,1,2)
plot(t, omega_BR_B_task8, 'LineWidth',2)
legend("$^B\omega_{B/R}(1)$", "$^B\omega_{B/R}(2)$", "$^B\omega_{B/R}(3)$", 'FontSize', 12
ylabel("$^B\omega_{B/R}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Reference Frame in Body Frame")

sgtitle("Task 8 - Outputs")

subplot(3,1,3)
plot(t, u_task8, 'LineWidth', 2)
xlabel("Time [sec]")
ylabel("Control Torques in Body Frame [Nm]")
title("Control Torque in Body Frame [Nm]")
legend("$^Bu(1)$", "$^Bu(2)$", "$^Bu(3)$", Interpreter="latex")
```

```
grid on

%% Task 9

[state_task9, u_task9, sigma_BR_task9, omega_BR_B_task9] = rk4_task9(@(state, t, u_B, I_B)

sigma_BN_at_15 = state_task9(16,1:3);
sigma_BN_at_100 = state_task9(101,1:3);
sigma_BN_at_200 = state_task9(201,1:3);
sigma_BN_at_400 = state_task9(401,1:3);

print_array(sigma_BN_at_15)
print_array(sigma_BN_at_100)
print_array(sigma_BN_at_200)
print_array(sigma_BN_at_400)

%% Figures

figure()
subplot(2,1,1)
plot(t, state_task9(1:end-1,1:3), 'LineWidth',2)
legend("$\sigma_{B/N}(1)$", "$\sigma_{B/N}(2)$", "$\sigma_{B/N}(3)$", 'FontSize', 12, Inte
ylabel("$\sigma_{B/N}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Inertial to Body Frame MRP")

subplot(2,1,2)
plot(t, state_task9(1:end-1,4:6), 'LineWidth',2)
legend("$^B\omega_{B/N}(1)$", "$^B\omega_{B/N}(2)$", "$^B\omega_{B/N}(3)$", 'FontSize', 12
ylabel("$^B\omega_{B/N}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Inertial Frame in Body Frame")

sgtitle("Task 9 - State Outputs")

figure()
subplot(3,1,1)
plot(t, sigma_BR_task9, 'LineWidth',2)
legend("$\sigma_{B/R}(1)$", "$\sigma_{B/R}(2)$", "$\sigma_{B/R}(3)$", 'FontSize', 12, Inte
ylabel("$\sigma_{B/R}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Reference to Body Frame MRP")

subplot(3,1,2)
plot(t, omega_BR_B_task9, 'LineWidth',2)
legend("$^B\omega_{B/R}(1)$", "$^B\omega_{B/R}(2)$", "$^B\omega_{B/R}(3)$", 'FontSize', 12
ylabel("$^B\omega_{B/R}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Reference Frame in Body Frame")
```

```
sgtitle("Task 9 - Outputs")

subplot(3,1,3)
plot(t, u_task9, 'LineWidth', 2)
xlabel("Time [sec]")
ylabel("Control Torques in Body Frame [Nm]")
title("Control Torque in Body Frame [Nm]")
legend("$^Bu(1)$", "$^Bu(2)$", "$^Bu(3)$", Interpreter="latex")
grid on

%% Task 10

[state_task10, u_task10, sigma_BR_task10, omega_BR_B_task10] = rk4_task10(@(state, t, u_B,

sigma_BN_at_15 = state_task10(16,1:3);
sigma_BN_at_100 = state_task10(101,1:3);
sigma_BN_at_200 = state_task10(201,1:3);
sigma_BN_at_400 = state_task10(401,1:3);

print_array(sigma_BN_at_15)
print_array(sigma_BN_at_100)
print_array(sigma_BN_at_200)
print_array(sigma_BN_at_400)

%% Figures

figure()
subplot(2,1,1)
plot(t, state_task10(1:end-1,1:3), 'LineWidth',2)
legend("$\sigma_{B/N}(1)$", "$\sigma_{B/N}(2)$", "$\sigma_{B/N}(3)$", 'FontSize', 12, Inte
ylabel("$\sigma_{B/N}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Inertial to Body Frame MRP")

subplot(2,1,2)
plot(t, state_task10(1:end-1,4:6), 'LineWidth',2)
legend("$^B\omega_{B/N}(1)$", "$^B\omega_{B/N}(2)$", "$^B\omega_{B/N}(3)$", 'FontSize', 12
ylabel("$^B\omega_{B/N}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Inertial Frame in Body Frame")

sgtitle("Task 10 - State Outputs")


figure()
subplot(3,1,1)
plot(t, sigma_BR_task10, 'LineWidth',2)
legend("$\sigma_{B/R}(1)$", "$\sigma_{B/R}(2)$", "$\sigma_{B/R}(3)$", 'FontSize', 12, Inte
ylabel("$\sigma_{B/R}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Reference to Body Frame MRP")
```

```
subplot(3,1,2)
plot(t, omega_BR_B_task10, 'LineWidth',2)
legend("$^B\omega_{B/R}(1)$", "$^B\omega_{B/R}(2)$", "$^B\omega_{B/R}(3)$", 'FontSize', 12
ylabel("$^B\omega_{B/R}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Reference Frame in Body Frame")

sgtitle("Task 10 - Outputs")

subplot(3,1,3)
plot(t, u_task10, 'LineWidth', 2)
xlabel("Time [sec]")
ylabel("Control Torques in Body Frame [Nm]")
title("Control Torque in Body Frame [Nm]")
legend("$^Bu(1)$", "$^Bu(2)$", "$^Bu(3)$", Interpreter="latex")
grid on


%% Task 11

t = 0:6500;

[state_task11, u_task11, sigma_BR_task11, omega_BR_B_task11, sc_eclipsed, angle_difference

eclipsed_time_idx = [find(sc_eclipsed, 1, 'first'), find(sc_eclipsed, 1, 'last')];
comm_window_time_idx = [find(comm_window, 1, 'first'), find(comm_window, 1, 'last')];

sigma_BN_at_300 = state_task11(301,1:3);
sigma_BN_at_2100 = state_task11(2101,1:3);
sigma_BN_at_3400 = state_task11(3401,1:3);
sigma_BN_at_4400 = state_task11(4401,1:3);
sigma_BN_at_5600 = state_task11(5601,1:3);

print_array(sigma_BN_at_300)
print_array(sigma_BN_at_2100)
print_array(sigma_BN_at_3400)
print_array(sigma_BN_at_4400)
print_array(sigma_BN_at_5600)

%% Figures

figure()
subplot(2,1,1)
plot(t, state_task11(1:end-1,1:3), 'LineWidth',2)
xregion(t(eclipsed_time_idx(1)), t(eclipsed_time_idx(2)));
xregion(t(comm_window_time_idx(1)), t(comm_window_time_idx(2)), FaceColor='b');
legend("$\sigma_{B/N}(1)$", "$\sigma_{B/N}(2)$", "$\sigma_{B/N}(3)$", "Eclipse", "Communic
ylabel("$\sigma_{B/N}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Inertial to Body Frame MRP")
```

```
subplot(2,1,2)
plot(t, state_task11(1:end-1,4:6), 'LineWidth',2)
xregion(t(eclipsed_time_idx(1)), t(eclipsed_time_idx(2)));
xregion(t(comm_window_time_idx(1)), t(comm_window_time_idx(2)), FaceColor='b');
legend("$^B\omega_{B/N}(1)$", "$^B\omega_{B/N}(2)$", "$^B\omega_{B/N}(3)$", "Eclipse", "Co
ylabel("$^B\omega_{B/N}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Inertial Frame in Body Frame")

sgtitle("Task 11 - State Outputs")


figure()
subplot(3,1,1)
plot(t, sigma_BR_task11, 'LineWidth', 2)
xregion(t(eclipsed_time_idx(1)), t(eclipsed_time_idx(2)));
xregion(t(comm_window_time_idx(1)), t(comm_window_time_idx(2)), FaceColor='b');
legend("$\sigma_{B/R}(1)$", "$\sigma_{B/R}(2)$", "$\sigma_{B/R}(3)$", "Eclipse", "Communic
ylabel("$\sigma_{B/R}$", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Reference to Body Frame MRP")

subplot(3,1,2)
plot(t, omega_BR_B_task11, 'LineWidth',2)
xregion(t(eclipsed_time_idx(1)), t(eclipsed_time_idx(2)));
xregion(t(comm_window_time_idx(1)), t(comm_window_time_idx(2)), FaceColor='b');
legend("$^B\omega_{B/R}(1)$", "$^B\omega_{B/R}(2)$", "$^B\omega_{B/R}(3)$", "Eclipse", "Co
ylabel("$^B\omega_{B/R}$ [rad/sec]", 'FontSize', 14, Interpreter="latex")
xlabel("Time [sec]")
grid on
title("Angular Velocity of Body w.r.t. Reference Frame in Body Frame")

sgtitle("Task 11 - Outputs")

subplot(3,1,3)
plot(t, u_task11, 'LineWidth', 2)
xregion(t(eclipsed_time_idx(1)), t(eclipsed_time_idx(2)));
xregion(t(comm_window_time_idx(1)), t(comm_window_time_idx(2)), FaceColor='b');
xlabel("Time [sec]")
ylabel("Control Torques in Body Frame [Nm]")
title("Control Torque in Body Frame [Nm]")
legend("$^Bu(1)$", "$^Bu(2)$", "$^Bu(3)$", "Eclipse", "Communication Window", Interpreter=
grid on


%% Functions

function ang = ang_diff(r1, r2)
    ang = acos(dot(r1, r2)/(norm(r1)*norm(r2)));
end

function [x, u_B, sigma_BR, omega_BR_B, sc_eclipsed, angle_difference, comm_window] = rk4_
```

```
    dt = t(2) - t(1);

    % x is nx6
    x = x0';

    for i = 1:length(t)
        sigma_BN = x(i,1:3)';
        omega_BN_B = x(i,4:6)';

        r_N = r_N_lmo(i,:);
        % n1_n3_pos = sqrt(r_N(1)^2 + r_N(3)^2);
        sc_eclipsed(i) = false;
        % if r_N(2) < 0 && n1_n3_pos <= r_mars
        if r_N(2) < 0
            sc_eclipsed(i) = true;
        end

        angle_difference(i) = ang_diff(r_N, r_N_gmo(i,:));
        comm_window(i) = angle_difference(i) <= 35*pi/180;
        if sc_eclipsed(i) == false
            [DCM_RN, omega_RN_N] = sun_reference_frame();
        else
            if comm_window(i)
                [DCM_RN, omega_RN_N] = gmo_reference_frame(r_N_lmo(i+1,:)', r_N_gmo(i+1,:)
            else
                [DCM_RN, omega_RN_N] = nadir_reference_frame(r_N_lmo(i,:)', r_dot_N_lmo(i,:
            end
        end
        [sigma_BR(i,:), omega_BR_B(i,:)] = attitude_error(t(i), sigma_BN, omega_BN_B, DCM_R
        u_B(i,:) = (-K*sigma_BR(i,:) -P*omega_BR_B(i,:))';

        k1 = dt * fn(x(i,:), t(i), u_B(i,:), I_B)';
        k2 = dt * fn(x(i,:) + k1/2, t(i) + dt/2, u_B(i,:), I_B)';
        k3 = dt * fn(x(i,:) + k2/2, t(i) + dt/2, u_B(i,:), I_B)';
        k4 = dt * fn(x(i,:) + k3, t(i) + dt, u_B(i,:), I_B)';

        x(i+1,:) = x(i,:) + 1/6 * (k1 + 2*k2 + 2*k3 + k4);

        sigma_BN_ip1 = x(i+1,1:3);
        if norm(sigma_BN_ip1) > 1
            x(i+1,1:3) = mrp_shadow(sigma_BN_ip1);
        end
    end
end

function [x, u_B, sigma_BR, omega_BR_B] = rk4_task10(fn, t, x0, I_B, K, P, r_N_lmo, r_dot_N

    dt = t(2) - t(1);

    % x is nx6
    x = x0';

    for i = 1:length(t)
```

```
        sigma_BN = x(i,1:3)';
        omega_BN_B = x(i,4:6)';
        if i == length(t)
            sigma_BRc(i,:) = zeros([3,1]);
            omega_BRc_B(i,:) = zeros([3,1]);
            u_B(i,:) = zeros([3,1]);
        else
            [DCM_RcN, omega_RcN_N] = gmo_reference_frame(r_N_lmo(i+1,:)', r_N_gmo(i+1,:)',
            [sigma_BRc(i,:), omega_BRc_B(i,:)] = attitude_error(t(i), sigma_BN, omega_BN_B

            u_B(i,:) = (-K*sigma_BRc(i,:) -P*omega_BRc_B(i,:))';
        end

        k1 = dt * fn(x(i,:), t(i), u_B(i,:), I_B)';
        k2 = dt * fn(x(i,:) + k1/2, t(i) + dt/2, u_B(i,:), I_B)';
        k3 = dt * fn(x(i,:) + k2/2, t(i) + dt/2, u_B(i,:), I_B)';
        k4 = dt * fn(x(i,:) + k3, t(i) + dt, u_B(i,:), I_B)';

        x(i+1,:) = x(i,:) + 1/6 * (k1 + 2*k2 + 2*k3 + k4);

        sigma_BN_ip1 = x(i+1,1:3);
        if norm(sigma_BN_ip1) > 1
            x(i+1,1:3) = mrp_shadow(sigma_BN_ip1);
        end
    end
    sigma_BR = sigma_BRc;
    omega_BR_B = omega_BRc_B;
end

function [x, u_B, sigma_BR, omega_BR_B] = rk4_task9(fn, t, x0, I_B, K, P, r_N_lmo, r_dot_N_

    dt = t(2) - t(1);

    % x is nx6
    x = x0';

    for i = 1:length(t)
        sigma_BN = x(i,1:3)';
        omega_BN_B = x(i,4:6)';
        [DCM_RnN, omega_RnN_N] = nadir_reference_frame(r_N_lmo(i,:)', r_dot_N_lmo(i,:)', th
        [sigma_BRn(:,i), omega_BRn_B(:,i)] = attitude_error(t(i), sigma_BN, omega_BN_B, DCM

        u_B(i,:) = (-K*sigma_BRn(:,i) -P*omega_BRn_B(:,i))';

        k1 = dt * fn(x(i,:), t(i), u_B(i,:), I_B)';
        k2 = dt * fn(x(i,:) + k1/2, t(i) + dt/2, u_B(i,:), I_B)';
        k3 = dt * fn(x(i,:) + k2/2, t(i) + dt/2, u_B(i,:), I_B)';
        k4 = dt * fn(x(i,:) + k3, t(i) + dt, u_B(i,:), I_B)';

        x(i+1,:) = x(i,:) + 1/6 * (k1 + 2*k2 + 2*k3 + k4);

        sigma_BN_ip1 = x(i+1,1:3);
        if norm(sigma_BN_ip1) > 1
            x(i+1,1:3) = mrp_shadow(sigma_BN_ip1);
```

```
        end
    end
    sigma_BR = sigma_BRn;
    omega_BR_B = omega_BRn_B;
end

function [x, u_B, sigma_BR, omega_BR_B] = rk4_task8(fn, t, x0, I_B, K, P)

    dt = t(2) - t(1);

    % x is nx6
    x = x0';

    for i = 1:length(t)
        sigma_BN = x(i,1:3)';
        omega_BN_B = x(i,4:6)';
        [DCM_RsN, omega_RsN_N] = sun_reference_frame();
        [sigma_BRs(:,i), omega_BRs_B(:,i)] = attitude_error(t(i), sigma_BN, omega_BN_B, DCM

        u_B(i,:) = (-K*sigma_BRs(:,i) -P*omega_BRs_B(:,i))';

        k1 = dt * fn(x(i,:), t(i), u_B(i,:), I_B)';
        k2 = dt * fn(x(i,:) + k1/2, t(i) + dt/2, u_B(i,:), I_B)';
        k3 = dt * fn(x(i,:) + k2/2, t(i) + dt/2, u_B(i,:), I_B)';
        k4 = dt * fn(x(i,:) + k3, t(i) + dt, u_B(i,:), I_B)';

        x(i+1,:) = x(i,:) + 1/6 * (k1 + 2*k2 + 2*k3 + k4);

        sigma_BN_ip1 = x(i+1,1:3);
        if norm(sigma_BN_ip1) > 1
            x(i+1,1:3) = mrp_shadow(sigma_BN_ip1);
        end
    end
    sigma_BR = sigma_BRs;
    omega_BR_B = omega_BRs_B;
end

function dcm = mrp_to_dcm(mrp)
    mrp_sq = norm(mrp)^2;
    mrp_tilde = skew_symmetric(mrp);
    dcm = eye(3) + (8 * mrp_tilde * mrp_tilde - 4 * (1-mrp_sq) * mrp_tilde)/(1+mrp_sq)^2;
end

function mrp = dcm_to_mrp(dcm)
    zeta = sqrt(trace(dcm) + 1);
    mrp = 1/(zeta*(zeta+2)) * [dcm(2,3)-dcm(3,2); dcm(3,1)-dcm(1,3); dcm(1,2)-dcm(2,1)];
end

function DCM_HN = orbit_frame(t, r_vec, r_dot_vec)
    r = r_vec(t,:)';
    r_dot = r_dot_vec(t,:)';
    i_r_N = r / norm(r);
    i_h_N = cross(r, r_dot) / norm(cross(r, r_dot));
    i_theta_N = cross(i_h_N, i_r_N);
```

```
    NH = [i_r_N, i_theta_N, i_h_N];
    DCM_HN = NH';
end

function [r_N, r_dot_N] = inertial_velocity(r, ea, theta_dot)
    % Inertial s/c velocity vector for a circular orbit
    % Inputs
    % r - radius
    % ea - 3-1-3 euler angles
    %
    % Outputs
    % r_N - inertial position
    % r_dot_N - inertial velocity

    r_O = [r, 0, 0]';
    omega_ON_O = [0, 0, theta_dot]';

    ON = R3(ea(3)) * R1(ea(2)) * R3(ea(1));


    r_N = ON' * r_O;
    r_dot_N = ON' * cross(omega_ON_O, r_O);

end
```


Acknowledgments

The author would like to thank Prof. Dr. Vishala Arya and TA Karina Rivera for their feedback and guidance.

References

[1] Arya, V., "Attitude Dynamics and Control of a Nano-Satellite Orbiting Mars," April 2025.

[2] Hanspeter Schaub, J. L. J., *Analytical Mechanics of Space Systems*, American Institute of Aeronautics and Astonautics, Inc., 2018.