Attitude Dynamics and Control of a Nano-Satellite Orbiting Mars

Chun-Wei Kong

Introduction

This project considers a small satellite orbiting Mars at a low altitude gathering science data. However, this small satellite needs to transfer this data to a larger mother satellite at a higher altitude. Further, to keep the batteries from draining all the way, periodically the satellite must point its solar panels at the sun to recharge. Thus, three mission goals must be considered by the satellite: 1) point sensor platform straight down at Mars, 2) point communication platform at the mother satellite, and 3) point the solar arrays at the sun. In all scenarios the small spacecraft and mother craft are on simple circular orbits whose motion is completely known. Figure 1 illustrates the mission ConOps.

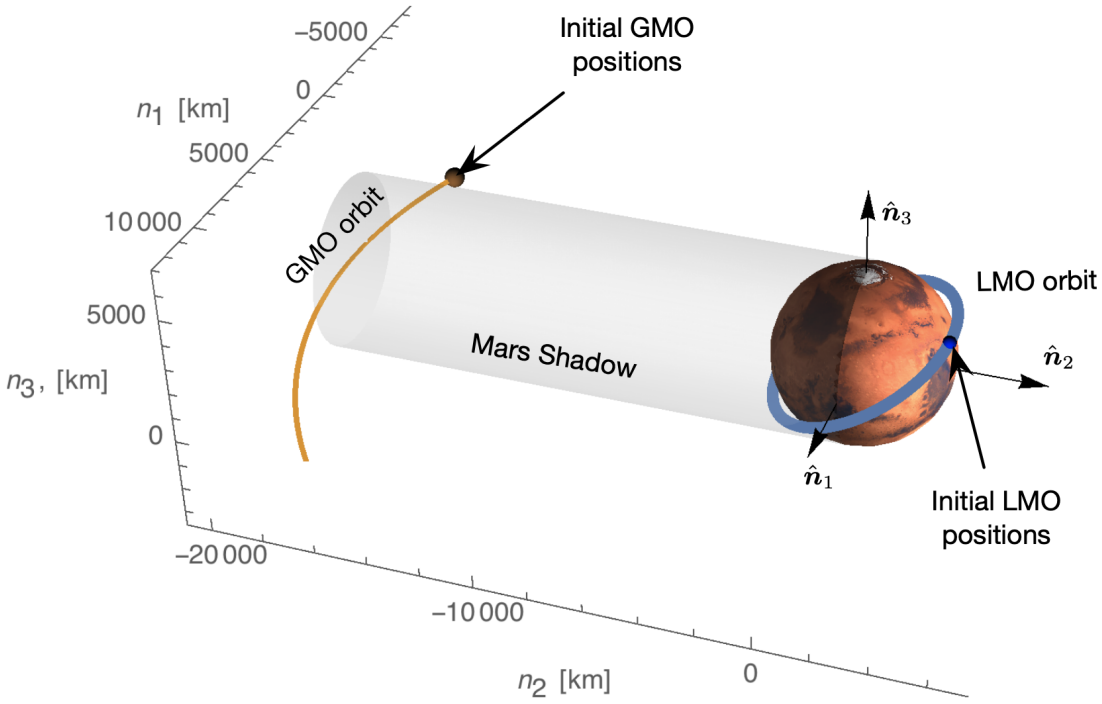

Figure 1: Mission ConOps

Task 1: Orbit Simultion

Spacecraft Inertial Velocity

Consider a spacecraft in a constant circular orbit. Assume the orbit frame $\mathcal{O} : \{\hat{\boldsymbol{i}}_r, \hat{\boldsymbol{i}}_\theta, \hat{\boldsymbol{i}}_h\}$ in Figure 2. The position vector is $\boldsymbol{r} = r\hat{\boldsymbol{i}}_r$. The angular velocity of orbit frame relative to the inertial frame is

$$\boldsymbol{w}_{\mathcal{O}/\mathcal{N}} = \dot{\theta}\hat{\boldsymbol{i}}_h. \tag{1}$$

Then the inertial spacecraft velocity vector $\dot{\boldsymbol{r}}$ is:

$$\dot{\boldsymbol{r}} = r\dot{\theta}\hat{i}_\theta, \tag{2}$$

where r is the constant orbit radius, and $\dot{\theta}$ is the constant orbit rate. The derivation follows transport theorem:[1]

$$\dot{\boldsymbol{r}} = {}^{\mathcal{N}}\frac{d}{dt}\boldsymbol{r} = {}^{\mathcal{O}}\frac{d}{dt}\boldsymbol{r} + \boldsymbol{w}_{\mathcal{O}/\mathcal{N}} \times \boldsymbol{r} = \boldsymbol{0} + \dot{\theta}\hat{\boldsymbol{i}}_h \times r\hat{\boldsymbol{i}}_r \tag{3}$$

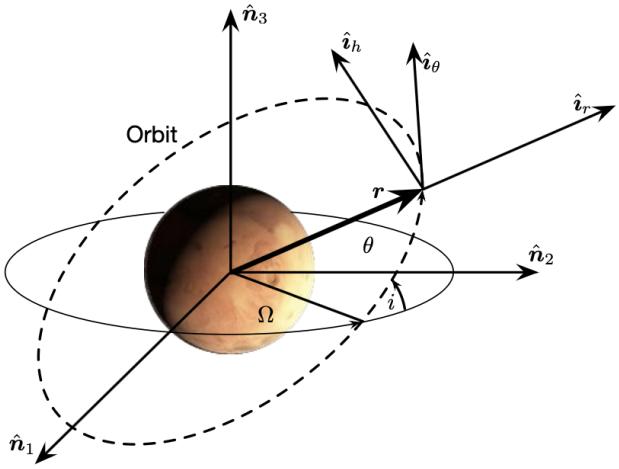

Figure 2: orbit frame

Function Implementation

`[r,v] = get_inertial_pos_vel(t,`$\mathcal{P}$`)` takes time t and orbit parameter $\mathcal{P}$ as inputs and return position and velocity vector of the spacecraft expressed in the inertial frame. $\mathcal{P}$ consists of constant orbit radius, orbit rate, and initial Euler angles.

Validation

The function is validated by calculating the position and velocity of the spacecraft at (1) LMO orbit at $t = 450$ seconds, and (2) GMO orbit at $t = 1150$ seconds. The initial orbit frame orientation is:

$$(\Omega, i, \theta(t_0))_{LMO} = (20, 30, 60) \text{ degree} \tag{4}$$
$$(\Omega, i, \theta(t_0))_{GMO} = (0, 0, 250) \text{ degree} \tag{5}$$

The results are summarized below and validated via the online checker.

- LMO:

$$\begin{aligned} {}^{N}\boldsymbol{r}(t = 450\ s) &= [-669.27, 3227.4, 1883.1]^T \text{ km} \\ {}^{N}\dot{\boldsymbol{r}}(t = 450\ s) &= [-3.2559, -0.79777, 0.21011]^T \text{ km/s} \end{aligned} \tag{6}$$

- GMO:

$$\begin{aligned} {}^{N}\boldsymbol{r}(t = 450\ s) &= [-5399.1, -19698, 0]^T \text{ km} \\ {}^{N}\dot{\boldsymbol{r}}(t = 450\ s) &= [1.3966, -0.3828, 0]^T \text{ km/s} \end{aligned} \tag{7}$$

Task 2: Orbit Frame Orientation

Hill Frame Expression

Let Hill frame $\mathcal{H}$ be the orbit frame of the LMO satellite. $\mathcal{H} = \{\hat{\boldsymbol{i}}_r, \hat{\boldsymbol{i}}_\theta, \hat{\boldsymbol{i}}_h\}$ is defined through (3-1-3) Euler angles (Ω, i, θ) with respect to the inertial frame $\mathcal{N} = \{\hat{\boldsymbol{n}}_1, \hat{\boldsymbol{n}}_2, \hat{\boldsymbol{n}}_3\}$. From the definition of Euler angles, the $\mathcal{H}$ frame orientation represented by DCM is:

$$[\mathcal{HN}(t)] = \begin{bmatrix} \cos\theta(t)\cos\Omega - \sin\theta(t)\cos i\sin\Omega, & \cos\theta(t)\sin\Omega + \sin\theta(t)\cos i\cos\Omega, & \sin\theta(t)\sin i \\ -\sin\theta(t)\cos\Omega - \cos\theta(t)\cos i\sin\Omega, & -\sin\theta(t)\sin\Omega + \cos\theta(t)\cos i\cos\Omega, & \cos\theta(t)\sin i, \\ \sin i\sin\Omega, & -\sin i\cos\Omega & \cos i \end{bmatrix} \tag{8}$$

Function Implementation

`[HN]=get_LMO_orbit_frame(t)` takes time t as input and return $[\mathcal{HN}]$ as the DCM representation of the $\mathcal{H}$ frame orientation.

Validation

The function is validated by computing $[\mathcal{HN}(t = 300\ s)]$. The result is shown below and validated via online checker.

$$[\mathcal{HN}(t = 300\ s)] = \begin{bmatrix} -0.046477, & 0.87415, & 0.48343 \\ -0.98417, & -0.12292, & 0.12765 \\ 0.17101, & -0.46985, & 0.86603 \end{bmatrix} \tag{9}$$

Task 3: Sun-Pointing Reference Frame Orientation

Sun-Pointing Reference Frame

Sun Pointing Reference Frame $\mathcal{R}_s = \{\hat{\boldsymbol{r}}_{s,1}, \hat{\boldsymbol{r}}_{s,2}, \hat{\boldsymbol{r}}_{s,3}\}$ such that $\hat{\boldsymbol{r}}_{s,3} = \hat{\boldsymbol{n}}_2,\ \hat{\boldsymbol{r}}_{s,1} = -\hat{\boldsymbol{n}}_1$. Frome the definition of DCM:

$$\{\hat{\boldsymbol{r}}_s\} = \begin{bmatrix} \cos\alpha_{11}, \cos\alpha_{12}, \cos\alpha_{13} \\ \cos\alpha_{21}, \cos\alpha_{22}, \cos\alpha_{23} \\ \cos\alpha_{31}, \cos\alpha_{32}, \cos\alpha_{33} \end{bmatrix} \{\hat{\boldsymbol{n}}\}, \tag{10}$$

we obtain:

$$[\mathcal{R}_s\mathcal{N}] = \begin{bmatrix} -1, & 0, & 0 \\ 0, & 0, & 1 \\ 0, & 1, & 0 \end{bmatrix} \tag{11}$$

Since $[\mathcal{R}_s\mathcal{N}]$ does not change over time, $[\mathcal{R}_s\mathcal{N}(t = 0)] = [\mathcal{R}_s\mathcal{N}]$ and ${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}_s/\mathcal{N}} = \boldsymbol{0}$.

Function Implementation and Validation

`get_Sun_pointing_frame(t)` and `get_w_Sun2N(t)` are implemented to return constant $[\mathcal{R}_s\mathcal{N}]$ and ${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}_s/\mathcal{N}}$. The results are validated via online checker.

Task 4: Nadir-Pointing Reference Frame Orientation

Nadir-Pointing Reference Frame

Nadir Pointing Reference Frame $\mathcal{R}_n = \{\hat{\boldsymbol{r}}_{n,1}, \hat{\boldsymbol{r}}_{n,2}, \hat{\boldsymbol{r}}_{n,3}\}$ such that $\hat{\boldsymbol{r}}_{r,1} = -\hat{\boldsymbol{i}}_r,\ \hat{\boldsymbol{r}}_{n,2} = \hat{\boldsymbol{i}}_\theta$. The orientation of $\mathcal{R}_n$ frame represented by DCM is:

$$\begin{gathered} [\mathcal{R}_n\mathcal{N}(t)] = [\mathcal{R}_n\mathcal{H}][\mathcal{HN}(t)], \text{ where} \\ [\mathcal{R}_n\mathcal{H}] = \begin{bmatrix} -1, & 0, & 0 \\ 0, & 1, & 0 \\ 0, & 0, & -1 \end{bmatrix} \end{gathered} \tag{12}$$

Since $\mathcal{R}_n$ frame does not rotate with respect to $\mathcal{H}$ frame, its rotational vector with respect to $\mathcal{N}$ frame is also $\dot{\theta}\hat{\boldsymbol{i}}_h$. Thus,

$${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}n/\mathcal{N}}(t) = [\mathcal{HN}(t)]^T\dot{\theta}\hat{\boldsymbol{i}}_h \tag{13}$$

Function Implementation and Validation

`get_Nadir_pointing_frame(t)` and `get_w_Nadir2N(t)` are implemented to return $[\mathcal{R}_n\mathcal{N}]$ and ${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}_n/\mathcal{N}}$. The results are validated via online checker.

Task 5: GMO-Pointing Reference Frame Orientation

GMO-Pointing Reference Frame

GMO Pointing Frame $\mathcal{R}_c = \{\hat{\boldsymbol{r}}_{c,1}, \hat{\boldsymbol{r}}_{c,2}, \hat{\boldsymbol{r}}_{c,3}\}$ such that $\hat{\boldsymbol{r}}_{c,1} = -\Delta/|\Delta|$, where $\Delta = \boldsymbol{r}_{GPM}(t) - \boldsymbol{r}_{LGO}(t)$, $\hat{\boldsymbol{r}}_{c,2} = (\Delta \times \hat{\boldsymbol{n}}_3)/(|\Delta \times \hat{\boldsymbol{n}}_3|)$, and $\hat{\boldsymbol{r}}_{c,3} = \hat{\boldsymbol{r}}_{c,1} \times \hat{\boldsymbol{r}}_{c,2}$. Then by definition, the orientation of $\mathcal{R}_c$ frame represented by DCM is:

$$[\mathcal{R}_c\mathcal{N}] = \begin{bmatrix} {}^{\mathcal{N}}\hat{\boldsymbol{r}}_{c,1}^T \\ {}^{\mathcal{N}}\hat{\boldsymbol{r}}_{c,2}^T \\ {}^{\mathcal{N}}\hat{\boldsymbol{r}}_{c,3}^T \end{bmatrix} \tag{14}$$

The rotation vector of $\mathcal{R}_c$ frame (expressed in inertial frame) ${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}c/\mathcal{N}}(t)$ is extracted from the ${}^{\mathcal{N}}[\tilde{\boldsymbol{w}}_{\mathcal{R}c/\mathcal{N}}]$ matrix that is numerically calculated by:

$$ {}^{\mathcal{N}}[\tilde{\boldsymbol{w}}_{\mathcal{R}_c/\mathcal{N}}] = -[\mathcal{R}_c\mathcal{N}]^T[\dot{\mathcal{R}_c\mathcal{N}}][\mathcal{R}_c\mathcal{N}]^T \tag{15} $$

The derivation follows from:

$$ [\dot{\mathcal{R}_c\mathcal{N}}] = -[\tilde{\boldsymbol{w}}_{\mathcal{R}_c/\mathcal{N}}][\mathcal{R}_c\mathcal{N}] \tag{16} $$

Task 6: Attitude Error Evaluation

Attitude Error Evaluation

Given the current body attitude states $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ and ${}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{N}}$, and the current references frame orientation $[\mathcal{R}\mathcal{N}]$ and rates ${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}/\mathcal{N}}$. We want to return the associated attitude tracking errors $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ and ${}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{R}}$. The process to get $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ is:

1. convert $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ to Euler parameters $\boldsymbol{q}(\mathcal{B}\mathcal{N})$
2. convert $\boldsymbol{q}(\mathcal{B}\mathcal{N})$ to DCM $[\mathcal{B}\mathcal{N}]$
3. compute $[\mathcal{B}\mathcal{R}] = [\mathcal{B}\mathcal{N}][\mathcal{R}\mathcal{N}]^T$
4. convert $[\mathcal{B}\mathcal{R}]$ to EP $\boldsymbol{q}(\mathcal{B}\mathcal{R})$ using Sheppard's method:
   (a) Find the largest value of:

$$ \beta_0^2 = \frac{1}{4}(1 + \text{Tr}([\text{DCM}]) \tag{17} $$
$$ \beta_1^2 = \frac{1}{4}(1 + 2[\text{DCM}]_{11} - \text{Tr}([\text{DCM}]) \tag{18} $$
$$ \beta_2^2 = \frac{1}{4}(1 + 2[\text{DCM}]_{22} - \text{Tr}([\text{DCM}]) \tag{19} $$
$$ \beta_3^2 = \frac{1}{4}(1 + 2[\text{DCM}]_{33} - \text{Tr}([\text{DCM}]) \tag{20} $$

   (b) Compute the remaining EPs:

$$ \beta_0\beta_1 = ([\text{DCM}]_{23} - [\text{DCM}]_{32})/4 \tag{21} $$
$$ \beta_0\beta_2 = ([\text{DCM}]_{31} - [\text{DCM}]_{13})/4 \tag{22} $$
$$ \beta_0\beta_3 = ([\text{DCM}]_{12} - [\text{DCM}]_{21})/4 \tag{23} $$
$$ \beta_1\beta_2 = ([\text{DCM}]_{12} + [\text{DCM}]_{21})/4 \tag{24} $$
$$ \beta_3\beta_1 = ([\text{DCM}]_{31} + [\text{DCM}]_{13})/4 \tag{25} $$
$$ \beta_2\beta_3 = ([\text{DCM}]_{23} + [\text{DCM}]_{32})/4 \tag{26} $$

5. convert $\boldsymbol{q}(\mathcal{B}\mathcal{R})$ to $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$ and return

Meanwhile, ${}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{R}}$ is computed by:

$$ {}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{R}} = {}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{N}} - [\mathcal{B}\mathcal{N}]{}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}/\mathcal{N}} \tag{27} $$

Function Implementation and Validation

The above process is implemented in `get_attitude_error(t,` $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$`,` ${}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{N}}$`,` $[\mathcal{R}\mathcal{N}]$`,` ${}^{\mathcal{N}}\boldsymbol{w}_{\mathcal{R}/\mathcal{N}}$`)`. We validate this function at $t = 0$ seconds with three reference $\mathcal{R}_s, \mathcal{R}_n, \mathcal{R}_c$. The results were checked via the online checker.

Task 7: Numerical Attitude Simulator

Attitude State Dynamics

Let the attitude state of the LMO be:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}} \\ {}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{N}} \end{bmatrix} \tag{28}$$

The initial states is

$$\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}(t_0) = [0.3, -0.4, 0.5]^T \tag{29}$$

$${}^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{N}}(t_0) = [1.00, 1.75, -2.20]^t \text{ deg/s} \tag{30}$$

Assume that the LMO spacecraft is rigid and its dynamics is:

$$[I]\dot{\boldsymbol{w}}_{\mathcal{B}/\mathcal{N}} = -[\tilde{\boldsymbol{w}}_{\mathcal{B}/\mathcal{N}}][I]\boldsymbol{w}_{\mathcal{B}/\mathcal{N}} + \boldsymbol{u}, \tag{31}$$

where $\boldsymbol{u}$ is the external control torque vector, and the inertial matrix is

$$[I] = \begin{bmatrix} 10, & 0, & 0 \\ 0, & 5, & 0 \\ 0, & 0, & 7.5 \end{bmatrix} \text{ kg m}^2 \tag{32}$$

An RK4 integrator is written to propagate $\boldsymbol{X}$ with 1 second integration step.

Function Implementation

```
I = diag([10, 5, 7.5]);
MRP_0 = [0.3;-0.4;0.5];
w_BN_B_0 = deg2rad([1.0; 1.75; -2.20]);
X0 = [MRP_0; w_BN_B_0];

X_hist = [X0];
U_hist = [];
T_hist = [0.5*w_BN_B_0'*I*w_BN_B_0];
H_hist = [get_BN(MRP_0)'*I*w_BN_B_0];

t_max = 100;
dt = 1;
t_k = 0.0;
while t_k < t_max
    X = X_hist(:,end);

    % get control u
    % u = [0; 0; 0];
    u = [0.01; -0.01; 0.02];

    h1 = dt*get_dXdt(X, t_k, u);
    h2 = dt*get_dXdt(X+0.5*h1, t_k+0.5*dt, u);
    h3 = dt*get_dXdt(X+0.5*h2, t_k+0.5*dt, u);
    h4 = dt*get_dXdt(X+h3, t_k+dt, u);
    X_new = X + (h1 + 2*h2 + 2*h3 + h4)/6;

    % map MRP
    if(norm(X_new(1:3)) > 1)
        MRP = X_new(1:3);
        X_new(1:3) = -MRP/(dot(MRP,MRP));
    end
```

```
    % disp(t_k);
    t_k = t_k + dt;
    X_hist(:,end+1) = X_new;
    U_hist(:,end+1) = u;
    T_hist(:,end+1) = 0.5*X_new(4:6)'*I*X_new(4:6);
    H_hist(:,end+1) = get_BN(X_new(1:3))'*I*X_new(4:6);
end
```

Validation

To validate the RK4 integrator, we first set $\boldsymbol{u} = \mathbf{0}$ and validate the results of angular momentum $\boldsymbol{H} = [I]\boldsymbol{w}_{\mathcal{B}/\mathcal{N}}$, kinetic energy $T = \frac{1}{2}\boldsymbol{w}_{\mathcal{B}/\mathcal{N}}^T[I]\boldsymbol{w}_{\mathcal{B}/\mathcal{N}}$, and MRP attitude $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$. Figures 3 and 4 show that the kinetic energy and the angular momentum components in inertial frame are conserved without external torque. The MRP $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ is validated at $t = 500$ via online checker.

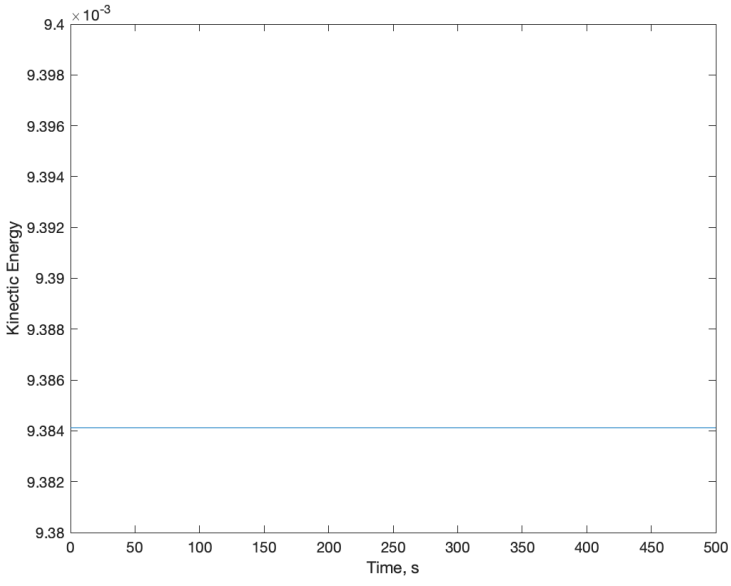

Figure 3: Energy history without external torque

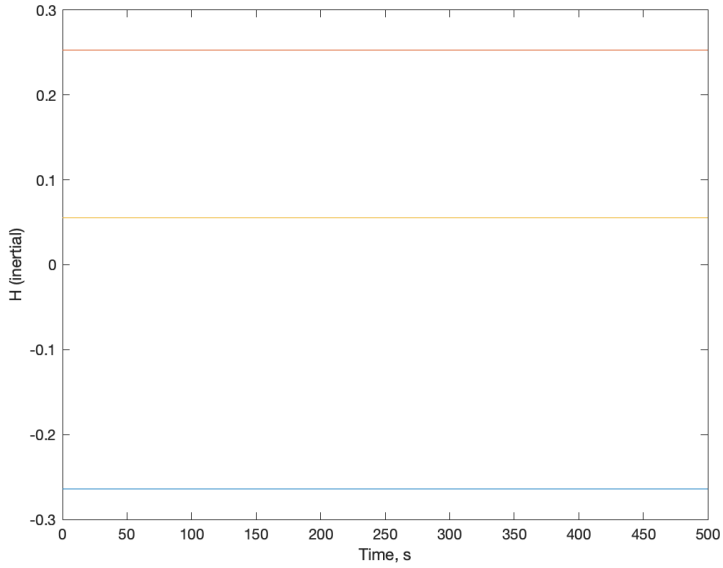

Figure 4: Angular momentum history expressed in inertial frame without external torque

In addition, we set $\boldsymbol{u}(t) = [0.01, -0.01, 0.02]^T$ Nm, and validate the MRP $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ at $t = 100$ seconds via online checker. The $\boldsymbol{\sigma}$ history is provided below.

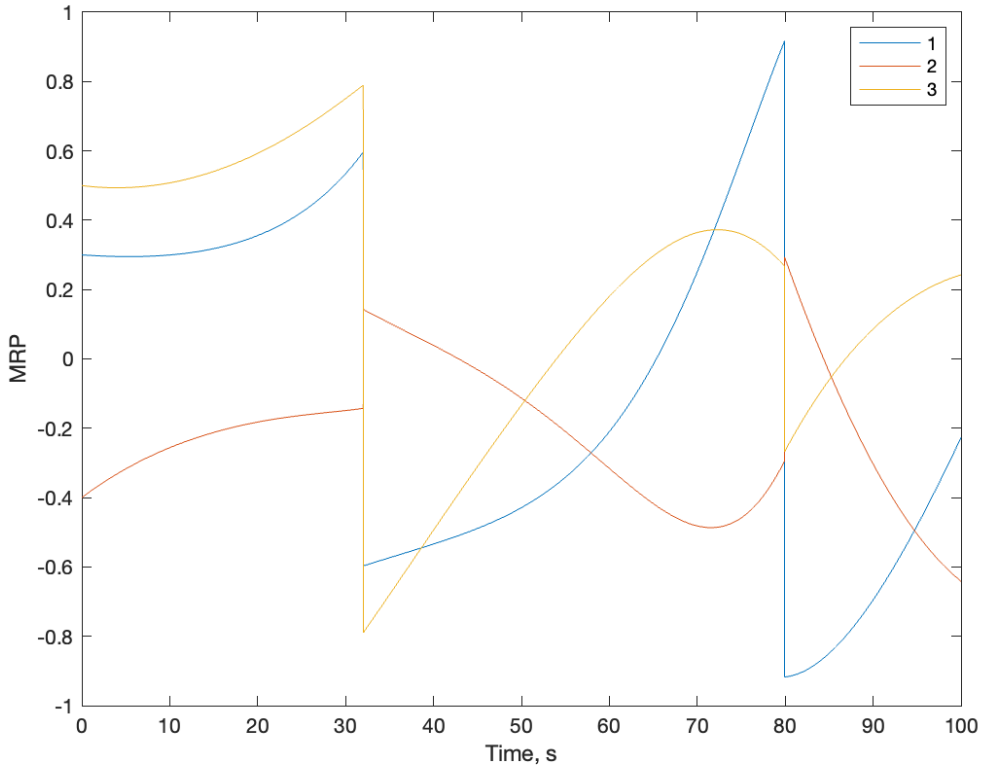

Figure 5: MRP attitude history with external torque

Task 8: Sun Pointing Control

Feedback Control

The attitude feedback control is

$$\boldsymbol{u} = -K\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}} - P^{\mathcal{B}}\boldsymbol{w}_{\mathcal{B}/\mathcal{R}}, \tag{33}$$

where K and P are real scalars. K and P are determined by the performance requirements: (1) the slowest decay response time is 120 seconds, and (2) at least one component of $\boldsymbol{\sigma}$ must be critically damped $\xi = 1$, while the other shall have $\xi \leq 1$. Using feedback gain selection, the time decay and damping ration are estimated by:

$$T_i = \frac{2I_i}{P} \tag{34}$$

$$\xi_i = \frac{P}{\sqrt{KI_i}}, \tag{35}$$

where $i = \{1, 2, 3\}$, and I_i is the i-th diagonal term of the principal inertia matrix. Thus,

$$K = 0.005556 \tag{36}$$

$$P = 0.166667 \tag{37}$$

Function Implementation and Validation

The implementation is simply modified from task 7, where the section of computing control $\boldsymbol{u}$ is replaced by:

```
RN = get_Sun_pointing_frame(t_k);
w_R2N_N = get_w_Sun2N(t_k);
[MRP_BR, w_BR_Body] = get_attitude_error(t_k, X(1:3), X(4:6), RN, w_R2N_N);
u = -K*MRP_BR - P*w_BR_Body;
```

We validate the MRP attitude response via online checker. Figure 6 plots the state history $\boldsymbol{X}(t)$.

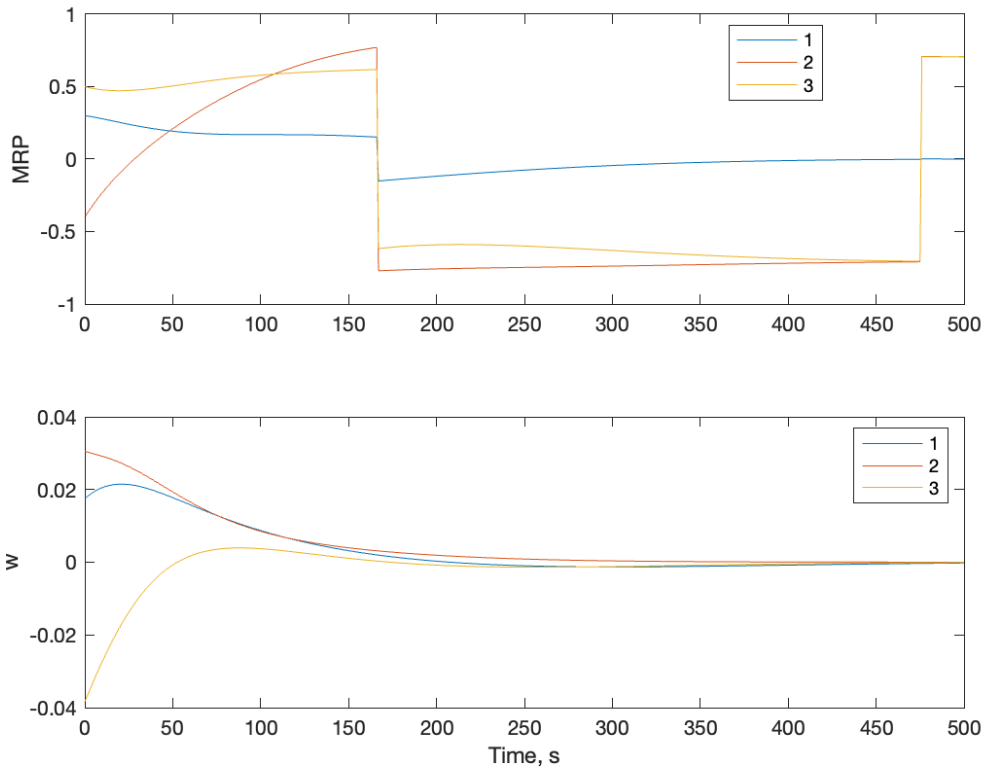

Figure 6: State history in sun pointing control

Task 9: Nadir Pointing Control

The same feedback control law and gains are used. The implementation is done by replacing the code section of getting sun-pointing frame and angular velocity in task 8 by:

```
RN = get_Nadir_pointing_frame(t_k);
w_R2N_N = get_w_Nadir2N(t_k);
```

The results are validated via online checker. Figure 7 shows the state response.

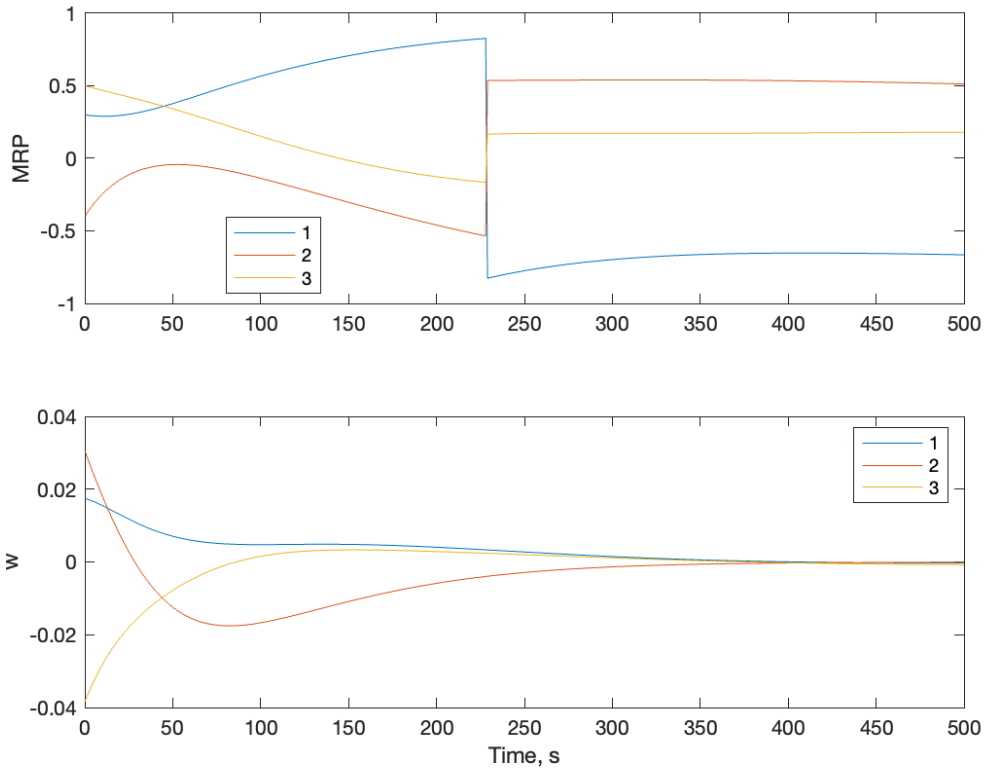

Figure 7: State history in nadir pointing control

Task 10: GMO Pointing Control

The same feedback control law and gains are used. The implementation is done by replacing the code section of getting sun-pointing frame and angular velocity in task 8 by:

```
RN = get_GMO_pointing_frame(t_k);
w_R2N_N = get_w_GMOpointing2N(t_k);
```

The results are validated via online checker. Figure 8 shows the state response.

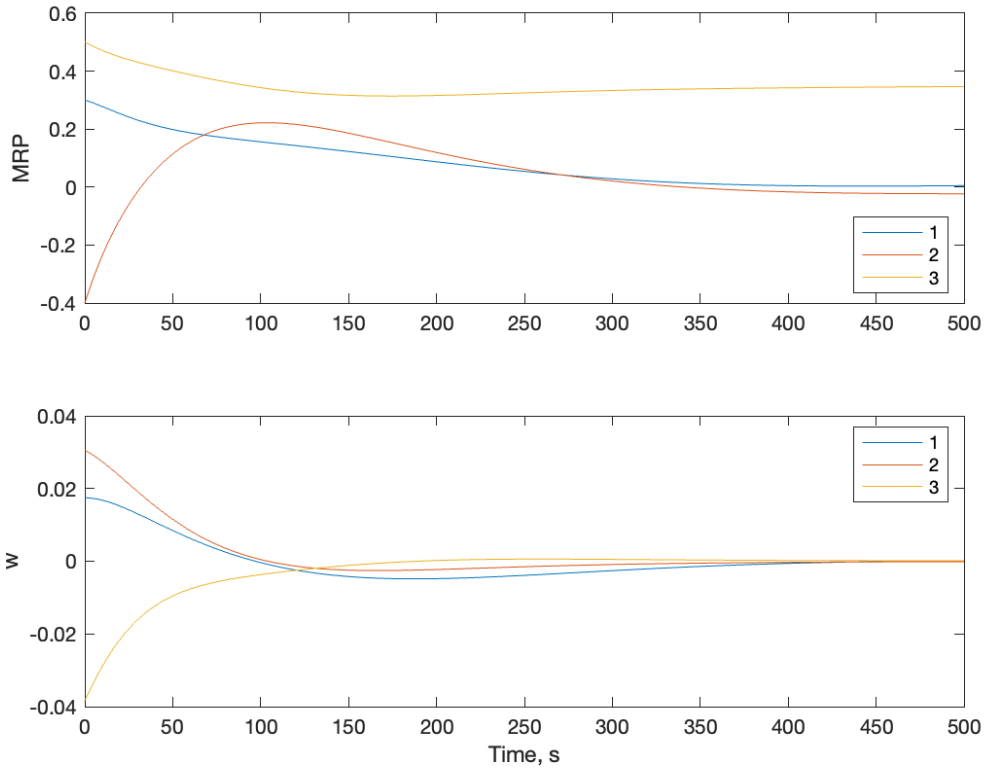

Figure 8: State history in GMO pointing control

Task 11: Mission Scenario Simulation

Misson Control Modes

The full mission is simulated by considering the control mode switching conditions:

1. LMO on sunlit Mars side (the second component of ${}^{N}r_{\text{LMO}} > 0$), do Sun-Pointing Control.
2. LMO not on sunlit Mars side and GMO is visible (the angle between vectors r_{LMO} and r_{GMO} is less than 35 degree), do GMO Pointing Control.
3. LMO not on sunlit Mars side and GMO is not visible, do Nadir Pointing Control.

Control Mode Logic Implementation

```
I = diag([10, 5, 7.5]);
% X = [MRP_B/N; w_B/N_B]
MRP_0 = [0.3;-0.4;0.5];
w_BN_B_0 = deg2rad([1.0; 1.75; -2.20]);
X0 = [MRP_0; w_BN_B_0];
X_hist = [X0];
U_hist = [];
Error_hist = [];

% Controller
K = (1/6)^2/5;
P = 1/6;
mode = "";
```

```
Mode_hist = [];

t_max = 6500.0;
dt = 1;
t_hist = 0:dt:t_max;
for k = 1:length(t_hist)-1
    t_k = t_hist(k);
    X = X_hist(:,end);

    [r_LMO, v_LMO] = get_LMO_pos_vel(t_k);
    [r_GMO, v_GMO] = get_GMO_pos_vel(t_k);
    if(r_LMO(2) > 0)
        mode = "SunPoint";
        Mode_hist(:,end+1) = 0;
    elseif(get_angle_deg(r_LMO, r_GMO) < 35)
        mode = "Communicate";
        Mode_hist(:,end+1) = 1;
    else
        mode = "NadirPoint";
        Mode_hist(:,end+1) = 2;
    end

    % get control u (Sun Pointing)
    if(mode == "SunPoint")
        RN = get_Sun_pointing_frame(t_k);
        w_R2N_N = get_w_Sun2N(t_k);
    elseif(mode == "Communicate")
        RN = get_GMO_pointing_frame(t_k);
        w_R2N_N = get_w_GMOpointing2N(t_k);
    else
        RN = get_Nadir_pointing_frame(t_k);
        w_R2N_N = get_w_Nadir2N(t_k);
    end

    [MRP_BR, w_BR_Body] = get_attitude_error(t_k, X(1:3), X(4:6), RN, w_R2N_N);
    u = -K*MRP_BR - P*w_BR_Body;

    h1 = get_dXdt(X, t_k, u);
    h2 = get_dXdt(X+0.5*h1*dt, t_k+0.5*dt, u);
    h3 = get_dXdt(X+0.5*h2*dt, t_k+0.5*dt, u);
    h4 = get_dXdt(X+h3*dt, t_k+dt, u);
    X_new = X + dt*(h1 + 2*h2 + 2*h3 + h4)/6;

    % ensure norm-1 of MRP
    MRP = X_new(1:3);
    if(norm(MRP) > 1)
        MRP = -MRP/(norm(MRP)^2);
    end
    X_new(1:3) = MRP;

    % store data
    X_hist(:,end+1) = X_new;
    U_hist(:,end+1) = u;
    Error_hist(:,end+1) = MRP_BR;
```

end

Validation

The response of MRP is validated via online checker. Below, we plot the state history $\boldsymbol{X}(t)$, control torque $\boldsymbol{u}(t)$, state error $\boldsymbol{e}(t)$, and control mode $M(t)$, where $M = 0$ is Sun-Pointing mode, $M = 1$ is Nadir-Pointing mode, and $M = 3$ is GMO-Pointing mode.

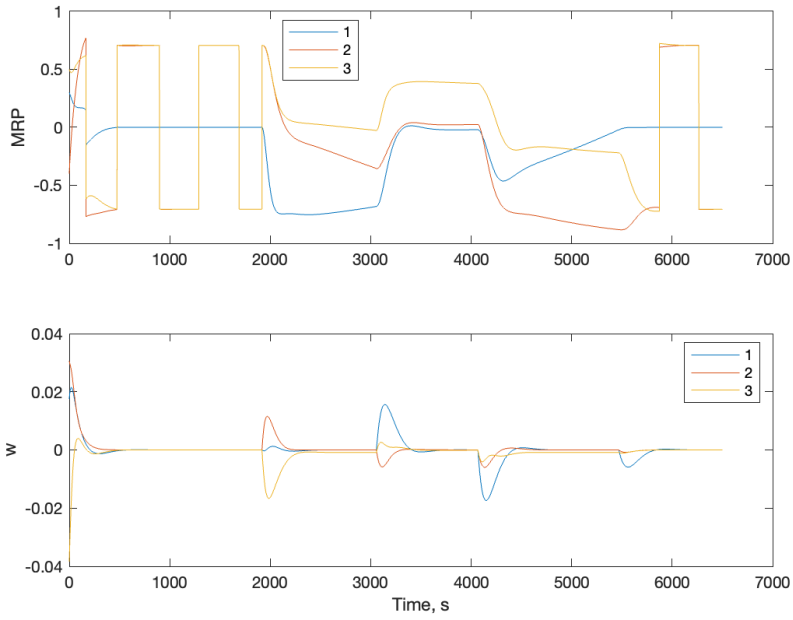

Figure 9: State history

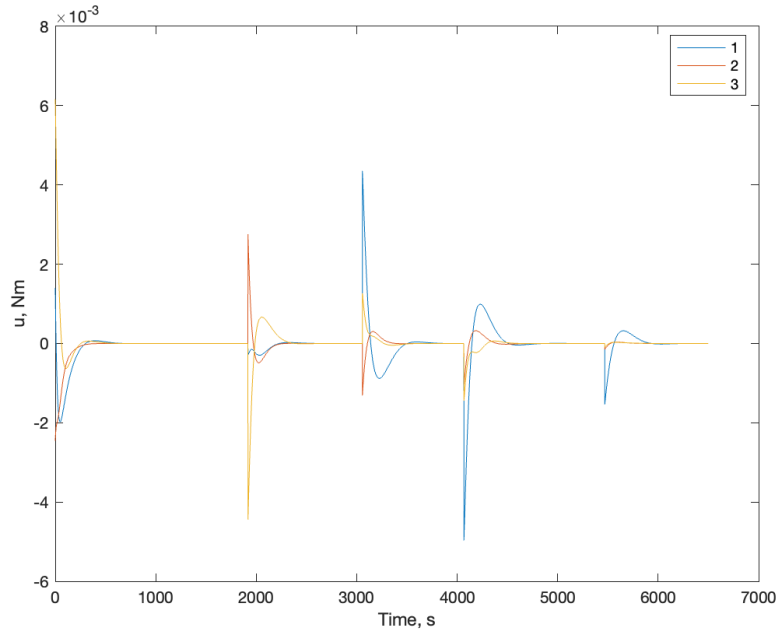

Figure 10: Control history

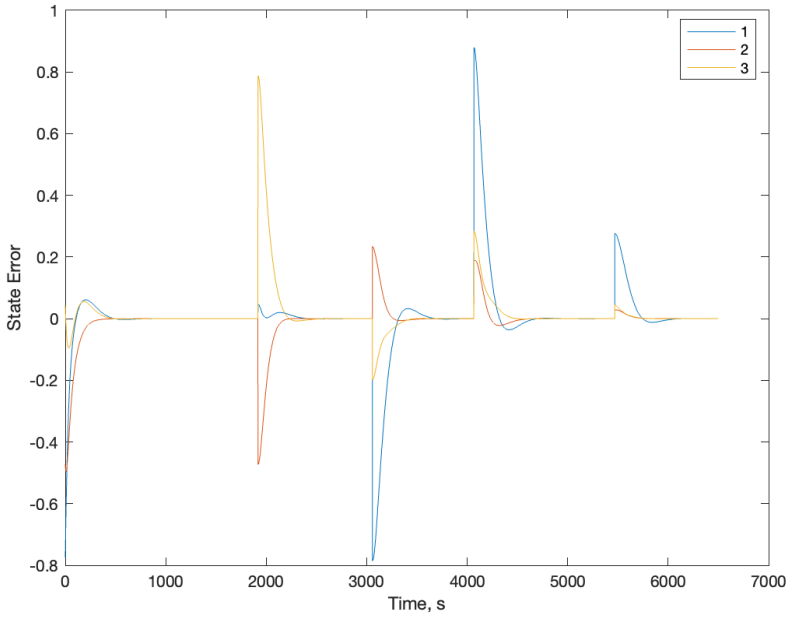

Figure 11: State error history

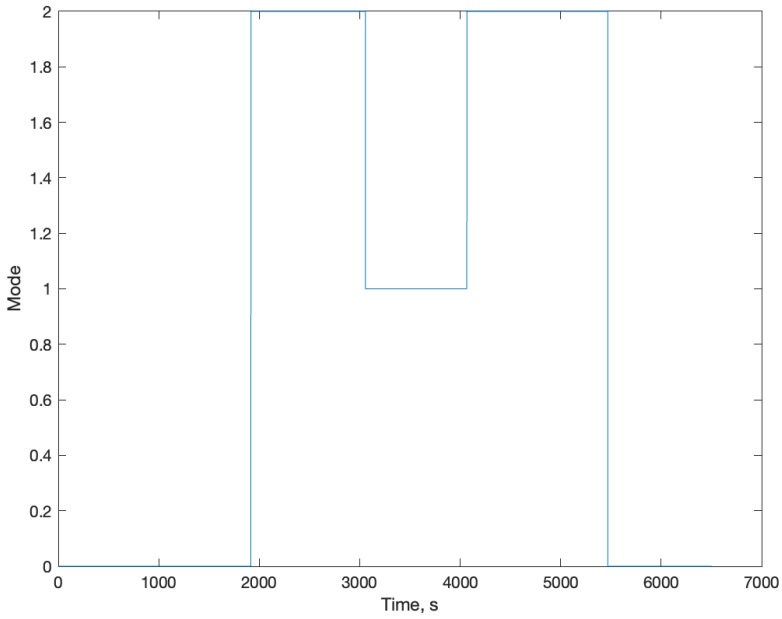

Figure 12: Mode history

3D Animation

A 3D animation of the full mission is created. Below, we show four snapshots at $t = 502, 2102, 3502, 4502$ seconds.

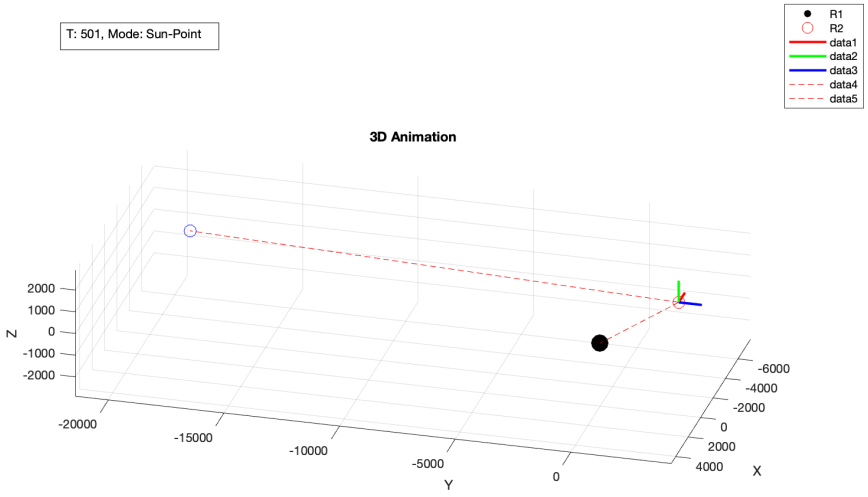

Figure 13: $t = 502$ **seconds**

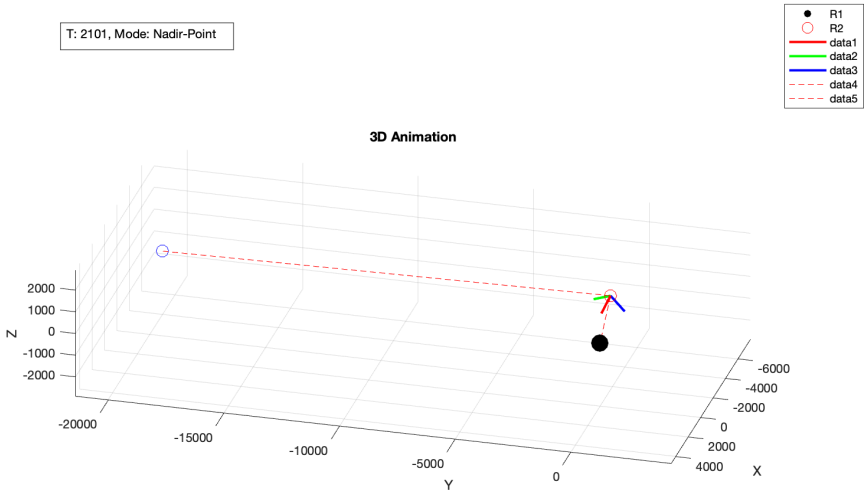

Figure 14: $t = 2102$ **seconds**

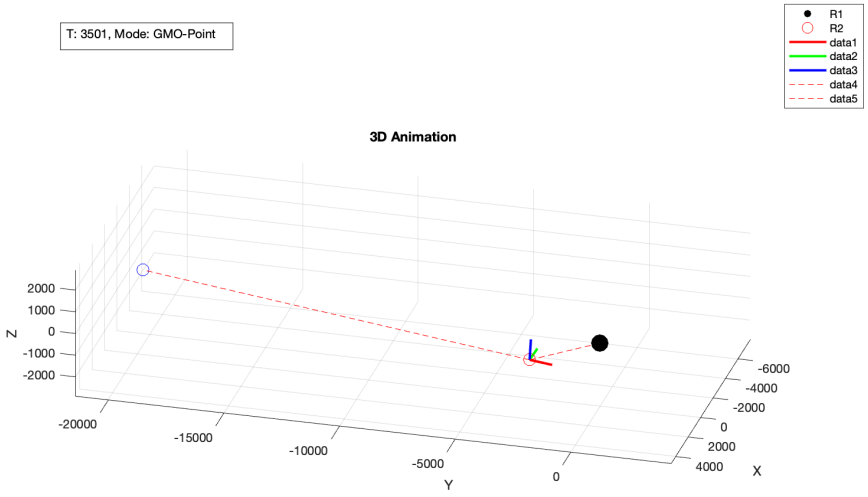

Figure 15: $t = 3502$ **seconds**

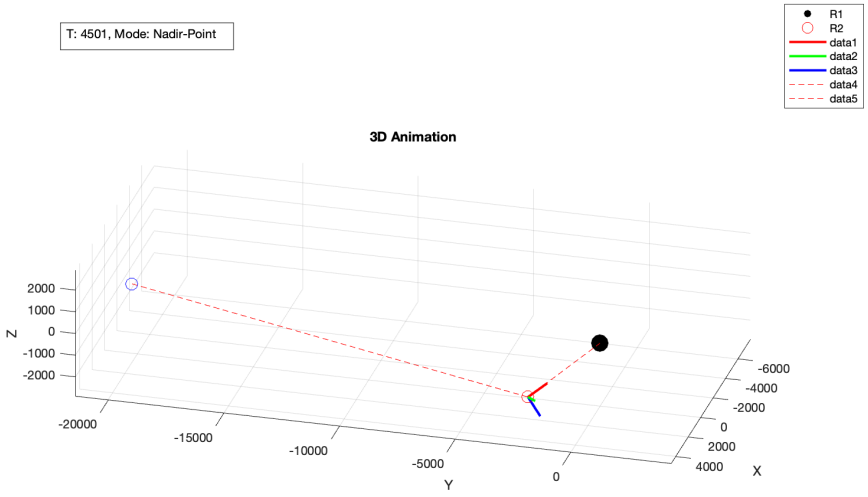

Figure 16: $t = 4502$ **seconds**

References

[1] Schaub, H. and Junkins, J. L., *Analytical mechanics of space systems*, Aiaa, 2003.