Attitude Dynamics and Control of a Nano-Satellite Orbiting Mars

Padraig S. Lysandrou *
The University of Colorado Boulder, Boulder, CO 80301

This project for ASEN5010 Spacecraft Dynamics and Control considers a small satellite orbiting Mars at a low altitude. This spacecraft gathers science data and transfers this data to another satellite orbiting at a higher altitude. Periodically, this spacecraft must transition from nadir-pointing, science gathering mode to sun-pointing mode to recharge the battery system. The three missions goals are nadir-pointing, communicating with the mother spacecraft, and to sun-point. We must develop a simulation architecture to demonstrate closed loop attitude control and verify performance characteristics.

Introduction and Problem Definition

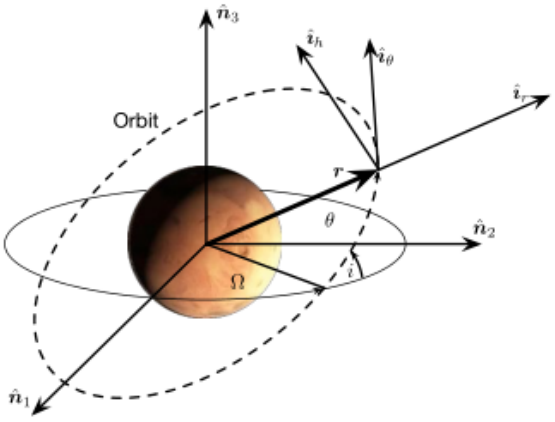
THIS project for ASEN5010, taught by Professor Hanspeter Schaub, considers a data-collection small satellite orbiting mars in a low Martian orbit (LMO) and a larger relay vehicle in a geosynchronous Martian orbit (GMO). The small spacecraft transmits data to the larger vehicle, which relays this information to another target. The small satellite has three main attitude mode: sun-pointing for power collection, nadir-pointing for science collection, and GMO-pointing for data transmission. Therefore, the three main mission goals for this spacecraft attitude control system are: 1) point the sensor platform towards Mars, 2) point the communication platform to the mother spacecraft for relay, 3) point the solar array at the sun for power collection. Both of these spacecraft are on simple circular orbits with known Keplerian orbital elements. We shall write a simulation architecture to demonstrate control of the spacecraft to converge to these three different reference objectives. The reference attitude and angular rate will be generated and tracked depending on the control mode of the spacecraft. Given that we know the inertial motion of both spacecraft, these reference frames are easily determined. Our torque control law shall drive the attitude MRP $\boldsymbol{\sigma}_{B/M}$ and angular rate $\boldsymbol{\omega}_{B/N}$ towards the nominal reference values. The scope of this problem includes reference frame generation, attitude and feedback control characterization, and simulation architecture development.

Problem Statement

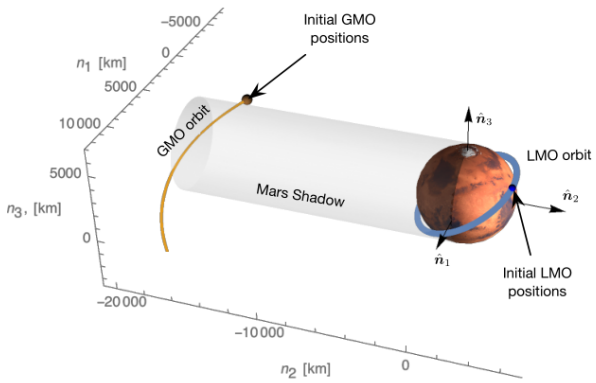
Now that we know we must create a simulation and control law for this spacecraft for each of it's pointing scenarios, let us begin by defining the orbit of the nano-satellite and GMO spacecraft with figure 1(a) and figure 1(b). We know that both spacecraft are in circular two-body orbits with different inclinations, and elements defined in table 1 below. The nano-satellite has an orbital altitude of 400km. Mars has an approximate rotational period of 1 day and 37 minutes, which should be the same period of our GMO spacecraft. This calculation leads to an orbital radius of approximately 20424.2 km. We assume this spacecraft to be in the equatorial, zero inclination plane. Given circular orbits, we can assume their orbital angular rates to be static. Lastly, we must describe the LMO spacecraft body frame construction showed in figure 1.

In the tasks to come, we shall derive each of the pointing frames and their control law. Additionally, we will describe the simulation framework construction. Before we begin with our work on the simulation and controller, we must define our initial conditions and operational parameters. Equations 1-3 are the initial attitude, angular rate, and the fixed intertia matrix that will be used throughout the project.

*PhD Student, Aerospace Engineering Department. Student Member of AIAA.



a) Illustration of the Inertial, Hill, and perifocal geometrical constructions. Taken from ASEN5010 Semester Project sheet.[1]



b) Overlook of each spacecraft orbit.[1]

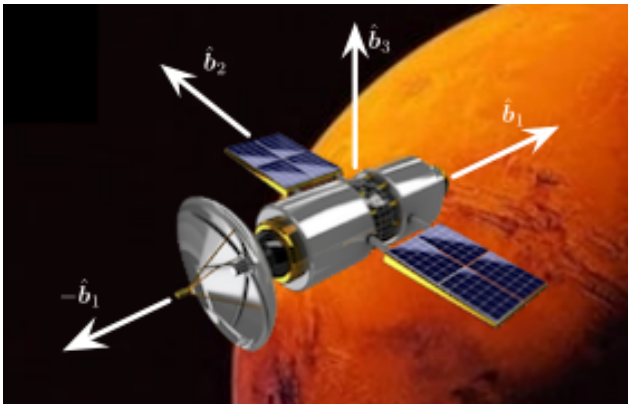


Figure 1: Body frame definition of our LMO spacecraft.[1]

$$\boldsymbol{\sigma}_{B/N}(t_0) = \begin{bmatrix} 0.3 \\ -0.4 \\ 0.5 \end{bmatrix} \tag{1}$$

$$^{\mathcal{B}}\boldsymbol{\omega}_{B/N}(t_0) = \begin{bmatrix} 1 \\ 1.75 \\ -2.2 \end{bmatrix} \text{ deg/s} \tag{2}$$

$$^{\mathcal{B}}[I] = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 7.5 \end{bmatrix} \text{ kg m}^2 \tag{3}$$

We will also use the following Keplerian orbital elements for both spacecraft:

Spacecraft	r	Ω	i	$\theta(t_0)$
LMO	3796.19 km	20°	30°	60°
GMO	20424.2 km	0°	0°	250°

Additionally, we use a gravitational parameter $\mu = 42828.3\text{km}^3/\text{s}^2$. We can also calculate each rotation rate using the circular motion assumption of $\dot{\theta} = \sqrt{\frac{\mu}{r^3}}$. These parameters will be used for all of the tasks and intermediate simulations in this project.

Task 1: Orbit Simulation

Our Hill frame is defined by the basis: $\{\hat{\boldsymbol{i}}_r, \hat{\boldsymbol{i}}_\theta, \hat{\boldsymbol{i}}_h\}$ with the inertial defined as $\{\hat{\boldsymbol{n}}_1, \hat{\boldsymbol{n}}_2, \hat{\boldsymbol{n}}_3\}$. Given the inertial and Hill frame definitions, we know that the position vector of the LMO satellite is $r\hat{\boldsymbol{i}}_r$. Additionally we know that since it is a circular orbit, it has a time invariant angular rate $\boldsymbol{\omega}_{H/N} = \dot{\theta}\hat{\mathbf{i}}_h$. Calculating the vectorial inertial derivative:

$$\dot{\boldsymbol{r}} = \frac{{}^N d}{dt}\boldsymbol{r} = \frac{{}^H d}{dt}\boldsymbol{r} + \boldsymbol{\omega}_{H/N} \times \boldsymbol{r} \tag{4}$$

$$= \dot{\theta}\hat{\boldsymbol{i}}_h \times r\hat{\boldsymbol{i}}_r \tag{5}$$

$$= r\dot{\theta}\hat{\boldsymbol{i}}_\theta \tag{6}$$

Additionally, we can use this information to find the inertial position and velocity vectors by performing transformations using the perifocal frame information. We know that the perifocal frame can be defined by an Euler 3-1-3 rotation defined the set $\{\Omega, i, \theta\}$

$$C_{ECI} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix} \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Which describes a rotation from Earth Centered Inertial frame. Each portion of the DCM is a single-axis rotation. We can then use this to project scalar values in the Hill frame to inertial vectors with the following:

$${}^N\vec{\boldsymbol{r}} = C_{ECI}^T \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

$${}^N\vec{\boldsymbol{v}} = C_{ECI}^T \begin{bmatrix} 0 \\ r\dot{\theta} \\ 0 \end{bmatrix} \tag{9}$$

When the ECI direction cosine matrix is calculated, θ must be propagated over time, as the true anomaly is the only perifocal parameter that is time variant. It is calculated as such: $\theta = \theta_0 + t * \dot{\theta}$.

Task 2: Orbit Frame Orientation

It is simple to generate bases vectors for the Hill frame, under motion, using our new inertial vectors. As stated before, $\mathcal{H} = \{\hat{\boldsymbol{i}}_r, \hat{\boldsymbol{i}}_\theta, \hat{\boldsymbol{i}}_h\}$, which can be constructed with the following:

$$\hat{\boldsymbol{i}_r} = \frac{\boldsymbol{r}_{LM}}{\|\boldsymbol{r}_{LM}\|} \tag{10}$$

$$\hat{\boldsymbol{i}_\theta} = \hat{\boldsymbol{i}_h} \times \hat{\boldsymbol{i}_r} \tag{11}$$

$$\hat{\boldsymbol{i}_h} = \frac{\boldsymbol{r}_{LM} \times \dot{\boldsymbol{r}}_{LM}}{\|\boldsymbol{r}_{LM} \times \dot{\boldsymbol{r}}_{LM}\|} \tag{12}$$

If we stack up these vectors into a matrix $[\hat{\boldsymbol{i}_r}\ \hat{\boldsymbol{i}_\theta}\ \hat{\boldsymbol{i}_h}]$, this defines the direction cosine matrix which takes vectors in the Hill frame to the inertial frame: $[NH]$. We can take the transpose to find the opposite: $[HN] = [\hat{\boldsymbol{i}_r}\ \hat{\boldsymbol{i}_\theta}\ \hat{\boldsymbol{i}_h}]^T$.

Task 3: Sun-Pointing Reference Frame Orientation

The solar panel axis $\hat{\boldsymbol{b}}_3$ must be pointed at the sun, and a reference frame $\mathcal{R}_s$ must be generated such that $\hat{\boldsymbol{r}}_3$ points in the sun direction ($\hat{\boldsymbol{n}}_2$). Given that the solar reference frame is constant with respect to the inertial frame, the ${}^N\boldsymbol{\omega}_{R_sN} = \mathbf{0}$. And our DCM is easily constructed using our assumptions with the following:

$$[R_sN] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{13}$$

Task 4: Nadir-Pointing Reference Frame Orientation

In order to point the payload platform axis $\hat{\boldsymbol{b}}_1$ towards Mars in the nadir direction, the reference frame $\mathcal{R}_n$ must be constructed such that $\hat{\boldsymbol{r}}_1$ points towards the planet. Additionally, we assume that $\hat{\boldsymbol{r}}_2$ is in the direction of the velocity $\hat{\boldsymbol{i}}_\theta$. Therefore we easily can construct a Hill-to-reference DCM which, using our now stated definitions, follows as such:

$$[R_nH] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{14}$$

This is the manifestation of a simple π rotation about the second Hill axis, where the reference flips $\hat{\boldsymbol{i}}_r$ and $\hat{\boldsymbol{i}}_h$. We can then calculate $[HN]$ using our procedure from Task 2. We then generate $[R_nN]$ via the following:

$$[R_nN] = [R_nH][HN] \tag{15}$$

Similarly, given that we are on a circular orbit, and that our reference is an invariant transformation from the Hill frame, we can easily describe ${}^N\boldsymbol{\omega}_{R_nN}$. Given that the reference and Hill angular rates are similar, we know that ${}^H\boldsymbol{\omega}_{R_nN} = [0\ 0\ \dot{\theta}]^T$ and can supply the reference angular rate with the following

$${}^N\boldsymbol{\omega}_{R_nN} = [HN]^T\ {}^H\boldsymbol{\omega}_{R_nN} = [NH][0\ 0\ \dot{\theta}]^T \tag{16}$$

Task 5: GMO-Pointing Reference Frame Orientation

Now we must construct another reference frame $\mathcal{R}_c$ such that $-\hat{\boldsymbol{r}}_1 =$ points towards the GMO spacecraft. This is simply done by finding the vector which represents the inertial difference in the position of both spacecraft: $\Delta\boldsymbol{r} = \boldsymbol{r}_{LMO} - \boldsymbol{r}_{GMO}$. We can then describe the frame with the following:

$$\hat{\boldsymbol{r}}_1 = \frac{-\Delta\boldsymbol{r}}{\|\Delta\boldsymbol{r}\|} \tag{17}$$

$$\hat{\boldsymbol{r}}_2 = \frac{\Delta\boldsymbol{r} \times \hat{\boldsymbol{n}}_3}{\|\Delta\boldsymbol{r} \times \hat{\boldsymbol{n}}_3\|} \tag{18}$$

$$\hat{\boldsymbol{r}}_3 = \hat{\boldsymbol{r}}_1 \times \hat{\boldsymbol{r}}_2 \tag{19}$$

Stacking these unit vectors as such $[\hat{\boldsymbol{r}}_1\ \hat{\boldsymbol{r}}_2\ \hat{\boldsymbol{r}}_3]$ yields a rotation matrix that, when multiplied by, brings vectors from the tracking reference frame to the inertial frame. Therefore, under a transpose operation we get the following:

$$[R_cN] = [\hat{\boldsymbol{r}}_1\ \hat{\boldsymbol{r}}_2\ \hat{\boldsymbol{r}}_3]^T \tag{20}$$

Finding ${}^N\boldsymbol{\omega}_{R_cN}$ is nontrivial and finding an analytical expression for the time derivative of the DCM can be challenging. Instead, we can use a numerical approach to find a usable solution. We know that the derivative of a DCM is that: $[\dot{C}] = -[\boldsymbol{\omega}^\times][C]$ with $^\times$ denoting the skew symmetric matrix form of a cross product. Therefore we can find the angular rate with the following:

$$\frac{d[R_cN]}{dt} = -[\boldsymbol{\omega}^\times_{\boldsymbol{R_cN}}][R_cN] \tag{21}$$

$$\frac{[R_cN(t+dt)] - [R_cN(t)]}{dt}[NR_c] = -[\boldsymbol{\omega}^\times_{\boldsymbol{R_cN}}] \tag{22}$$

Because we know have a function that determines this reference DCM at any point in time, this numerical derivative is easy to calculate for a small value dt. With knowledge of the skew symmetric form, we can de-skew $[\boldsymbol{\omega}^\times_{\boldsymbol{R_cN}}]$ to find our vector ${}^{R_c}\boldsymbol{\omega}_{\boldsymbol{R_cN}}$. To bring this quantity into the inertial frame we perform ${}^N\boldsymbol{\omega}_{\boldsymbol{R_cN}} = [R_cN(t)]^T\ {}^{R_c}\boldsymbol{\omega}_{\boldsymbol{R_cN}}$.

Task 6: Attitude and Angular Rate Error Evaluation

In this section, we must write a function that, given a current attitude state $\boldsymbol{\sigma}_{BN}$, angular rate ${}^{B}\boldsymbol{\omega}_{BN}$, and desired reference attitude matrix $[RN]$, returns the associate tracking errors $\boldsymbol{\sigma}_{BR}$ and ${}^{B}\boldsymbol{\omega}_{BR}$. First let us start with the simpler angular velocity error:

$$ {}^{B}\boldsymbol{\omega}_{BR} = ({}^{B}\boldsymbol{\omega}_{BN} - [BN]\,{}^{N}\boldsymbol{\omega}_{RN}) \tag{23} $$

We can find the inertial to body DCM transform by performing MRP2C($\boldsymbol{\sigma}_{BN}$) with the following function:

$$ [BN] = \frac{1}{\left(1+\sigma^2\right)^2}\begin{bmatrix} 4\left(\sigma_1^2-\sigma_2^2-\sigma_3^2\right)+\left(1-\sigma^2\right)^2 & 8\sigma_1\sigma_2+4\sigma_3\left(1-\sigma^2\right) & 8\sigma_1\sigma_3-4\sigma_2\left(1-\sigma^2\right) \\ 8\sigma_2\sigma_1-4\sigma_3\left(1-\sigma^2\right) & 4\left(-\sigma_1^2+\sigma_2^2-\sigma_3^2\right)+\left(1-\sigma^2\right)^2 & 8\sigma_2\sigma_3+4\sigma_1\left(1-\sigma^2\right) \\ 8\sigma_3\sigma_1+4\sigma_2\left(1-\sigma^2\right) & 8\sigma_3\sigma_2-4\sigma_1\left(1-\sigma^2\right) & 4\left(-\sigma_1^2-\sigma_2^2+\sigma_3^2\right)+\left(1-\sigma^2\right)^2 \end{bmatrix} \tag{24} $$

Now that we have the tracking error for the angular velocity, we must find the relative error in the modified rodrigues parameter attitude formalism. We could use the relative MRP formula, but it can be understood more easily by converting to DCMs and using fundamental properties of the SO(3) group:

$$ \boldsymbol{\sigma}_{BR} = \text{C2MRP}([BN][RN]^T) \tag{25} $$

The DCM to MRP transform is more complicated and is done by first converting the DCM to a quaternion via Sheppard's method. The first step is to find the maximum of these values, as truth, to constrain that value for the second step:

$$ \begin{array}{ll} \beta_0^2 = \frac{1}{4}(1+\text{tr}([BR])) & \beta_2^2 = \frac{1}{4}\left(1+2[BR]_{22}-\text{tr}([BR])\right) \\ \beta_1^2 = \frac{1}{4}\left(1+2[BR]_{11}-\text{tr}([BR])\right) & \beta_3^2 = \frac{1}{4}\left(1+2[BR]_{33}-\text{tr}([BR])\right) \end{array} \tag{26} $$

The second step is done by computing the rest of the quaternion entries, using our constrained entry, with the following:

$$ \begin{array}{ll} \beta_0\beta_1 = \left([BR]_{23}-[BR]_{32}\right)/4 & \beta_1\beta_2 = \left([BR]_{12}+[BR]_{21}\right)/4 \\ \beta_0\beta_2 = \left([BR]_{31}-[BR]_{13}\right)/4 & \beta_3\beta_1 = \left([BR]_{31}+[BR]_{13}\right)/4 \\ \beta_0\beta_3 = \left([BR]_{12}-[BR]_{21}\right)/4 & \beta_2\beta_3 = \left([BR]_{23}+[BR]_{32}\right)/4 \end{array} \tag{27} $$

The final MRP is calculated with from our quaternion entries using the definition:

$$ \boldsymbol{\sigma}_{BR} = \begin{bmatrix} \frac{\beta_1}{1+\beta_0} \\ \frac{\beta_2}{1+\beta_0} \\ \frac{\beta_3}{1+\beta_0} \end{bmatrix} \tag{28} $$

Task 7: Numerical Attitude Simulator

Now we must numerically integrate our differential equations of motion to simulate the dynamics of our system for both the LMO and GMO spacecraft. Let us define our state vector as the following:

$$ \boldsymbol{X} = \begin{bmatrix} \boldsymbol{\sigma}_{BN} \\ {}^{B}\boldsymbol{\omega}_{BR} \end{bmatrix} \tag{29} $$

Again with $^{\times}$ denoting the skew symmetric matrix form of a cross product, for $\boldsymbol{u}$ control torque vector, we know the the rigid body dynamics obey the following:

$$ [I]\dot{\boldsymbol{\omega}}_{BN} = -[\boldsymbol{\omega}_{BN}^{\times}][I]\boldsymbol{\omega}_{BN} + \boldsymbol{u} - \boldsymbol{L} \tag{30} $$

This can be solved for purely $\dot{\boldsymbol{\omega}}_{BN}$ on the left hand side. We also know the following to be the kinematic differential equation for the MRP attitude formalism:[2]

$$ \dot{\boldsymbol{\sigma}}_{BN} = \frac{1}{4}[(1-\sigma^2)\mathbb{I}_3 + 2[\boldsymbol{\sigma}^{\times}] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T]^{\mathcal{B}}\boldsymbol{\omega}_{BN} \tag{31} $$

We use a fourth order Runge-Kutte algorithm for integration (RK4). Using the nonlinear dynamics function $\dot{\boldsymbol{X}} = f(t, \boldsymbol{X})$, the integration is Algorithm 1. Each point $i \in [1 : N]$ is 1 integration time step, and therefore the full simulation time dtN. Using this simulation framework, we can study our angular momentum $\boldsymbol{H} = [I]\boldsymbol{\omega}_{BN}$ and kinetic energy $T = \frac{1}{2}\boldsymbol{\omega}_{BN}^T[I]\boldsymbol{\omega}_{BN}$ over time.

Algorithm 1 Fourth Order Runge Kutte Integrator

for i = 1:N-1 **do**
  $k_1 = \dot{\boldsymbol{X}}(t(i),\ \boldsymbol{X}(:,i))$
  $k_2 = \dot{\boldsymbol{X}}(t(i) + \frac{dt}{2},\ \boldsymbol{X}(:,i) + \frac{dt}{2}k_1)$
  $k_3 = \dot{\boldsymbol{X}}(t(i) + \frac{dt}{2},\ \boldsymbol{X}(:,i) + \frac{dt}{2}k_2)$
  $k_4 = \dot{\boldsymbol{X}}(t(i) + dt,\ \boldsymbol{X}(:,i) + dtk_3)$
  $\boldsymbol{X}(:,i+1) = \boldsymbol{X}(:,i) + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
end for

Code appended to the end of the document shows the structure of the simulation. We must first describe our initial conditions as part of the state vector, then define our time span, distribute our states, calculate the control, integrate for one step, check the norm of the MRP, populate our state vectors, and continue.

Task 8: Control for Sun, Nadir, and Communication Pointing

Next, the control architecture must be developed. We shall use the linearized closed loop dynamics to determine the proportional and derivative gains.

Let us consider the PD nonlinear feedback control law:

$$\boldsymbol{u} = -K\boldsymbol{\sigma} - [P]\delta\boldsymbol{\omega} + [I]\left(\dot{\boldsymbol{\omega}}_r - [\boldsymbol{\omega}^\times]\boldsymbol{\omega}_r\right) + [\boldsymbol{\omega}^\times][I]\boldsymbol{\omega} - \boldsymbol{L} \tag{32}$$

Let us disregard the external torque modeling error and consider the displacement MRP and deviation angular rate with the following control:

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

Let us look at the closed loop dynamics with this state vector:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\sigma} \\ \delta\boldsymbol{\omega} \end{bmatrix} \tag{34}$$

Using the differential kinematic equation for the MRPs we arrive at the following nonlinear state space formulation, with the closed loop full-state feedback:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{\sigma}} \\ \delta\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{4}B(\boldsymbol{\sigma}) \\ -K[I]^{-1} & -[I]^{-1}[P] \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma} \\ \delta\boldsymbol{\omega} \end{bmatrix} \tag{35}$$

The small angle approximation is employed to linearize this formulation with the following:

$$\begin{bmatrix} \dot{\boldsymbol{\sigma}} \\ \delta\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{4}\mathbb{I}_3 \\ -K[I]^{-1} & -[I]^{-1}[P] \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma} \\ \delta\boldsymbol{\omega} \end{bmatrix} \tag{36}$$

We can find the eigenvalues of this matrix to see how the gains affect the stability of the system. The roots act as the following:

$$\lambda_i = \frac{-1}{2I_i}\left(P_i \pm \sqrt{-KI_i + P_i^2}\right) \quad i = 1, 2, 3 \tag{37}$$

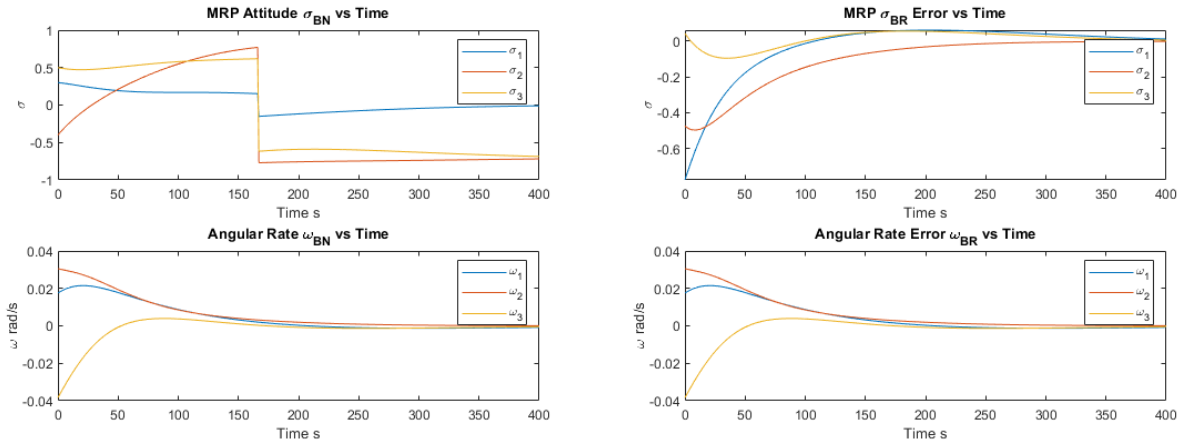
Now, we can choose the gains such that they meet our operation criterion. We want to use single scalar gains for both K and P. We know that we have a decay time of $T = 120$ seconds which constraints out P value with the following:

$$P = \max_i\left(\frac{2I_{ii}}{T}\right) \tag{38}$$

Once we have constrained this gain value, we can take a look at the K gain and damping coefficient for each of the modes. We know the relationship $\xi_i = \frac{P}{\sqrt{KI_i}}$. We can find a solution for the critically damped $\xi = 1$ mode, then pick an inertia that results in the other modes being critically or underdamped, where $\xi \leq 1$. We end with the result that $[P\ K] = [0.1666\ \ 0.0055]$.

Task 8: Sun Pointing Plots

Let us now simulate the LMO spacecraft performing the sun-point maneuver. We calculate the tracking errors with the function developed earlier and use the reference DCM and angular rate vector developed earlier. These tracking errors are used in our control function with the gains determined in the previous step. With our determined gains, we see how the system tracks the reference for a duration of 400 seconds:



a) States vs Time

b) Tracking Error vs Time

Figure 2: These figures show the states and state tracking errors over times for a 400 second simulation of sun-pointing.

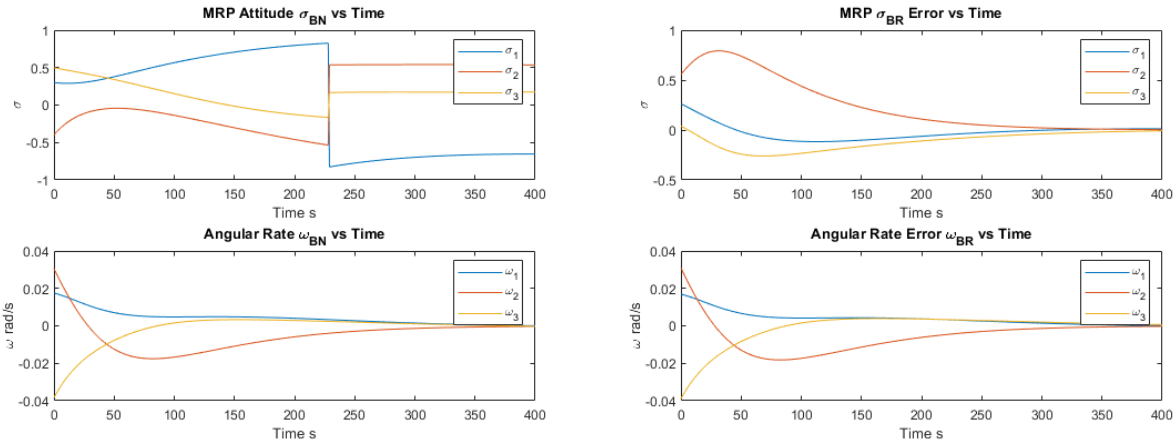
The error figure 2(b) clearly converges to zero in both state and angular rate tracking as time approaches the end of the simulation. This slow time constant of 120 seconds makes sure that the applied torques are small (in the milli-Nm range), such that they are achievable torques for practical actuators.

Task 9: Nadir (Science Mode) Pointing Plots

Now we shall simulate the LMO spacecraft performing purely the Mars-nadir pointing mode, where it gathers data by measuring the surface. The only difference between this mode and the sun-pointing mode are our new DCM and angular rate references. Similarly, we simulate for 400 seconds in figure 3(a) with errors in figure 3(b). Again the error figure 3(b) shows convergence over the simulation time, as we would expect from our control derivation.

Task 10: GMO Spacecraft Communication Pointing Plots

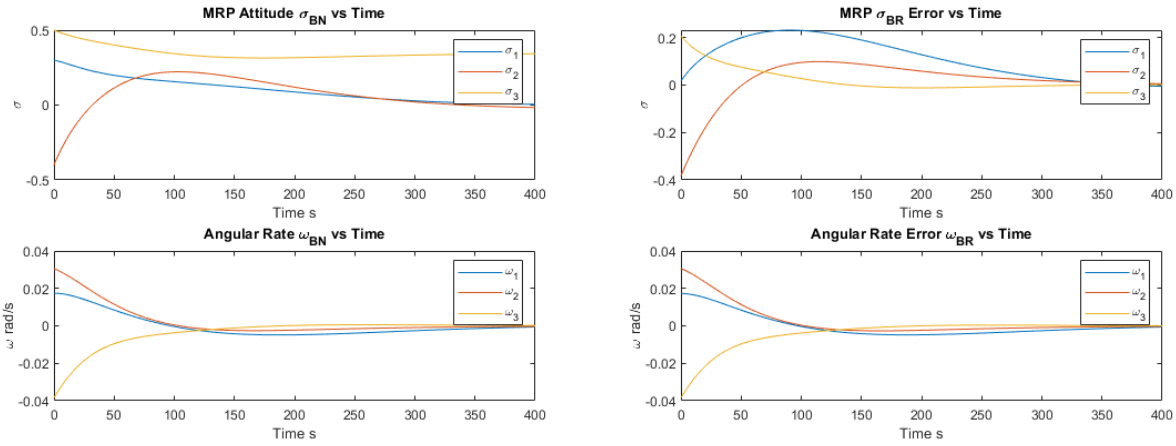
Similarly, we shall apply the control law to the GMO communication-pointing mode. In this mode, the spacecraft tracks the GMO reference attitude and angular rate, in order to uplink information for transmission to Earth. This concludes our individual mode control and we can move onto the full mission simulation with mode mode switching. The 400 second simulation is shown in figure 4(a) with tracking error in 4(b). Again the error figure 4(b) shows convergence over the simulation time, as we would expect from our control derivation.



a) States vs Time

b) Tracking Error vs Time

Figure 3: This figures show the states and state tracking errors over times for a 400 second simulation of nadir-pointing.



a) States vs Time

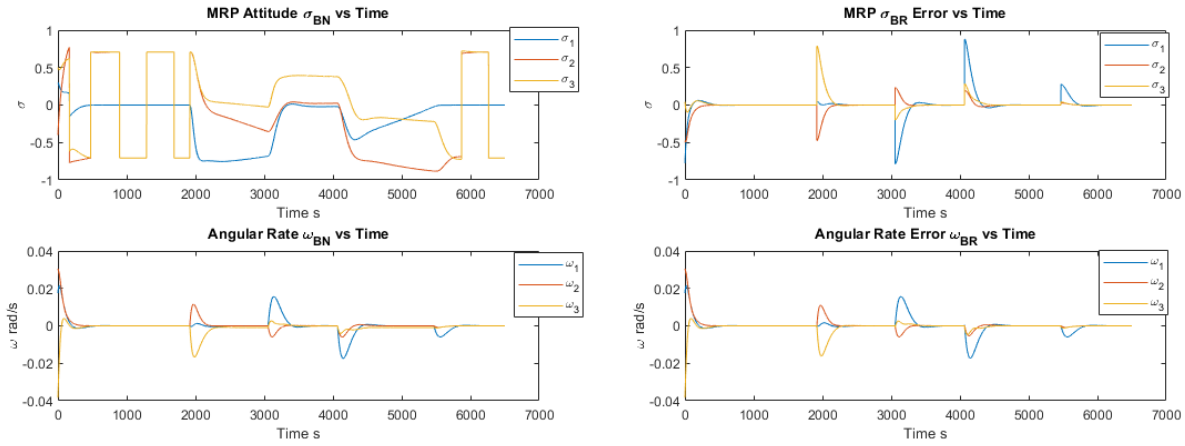
b) Tracking Error vs Time

Figure 4: This figures show the states and state tracking errors over times for a 400 second simulation of GMO-pointing.

Task 11: Mode Switching Mission Scenario

Next, we must simulate the full mission scenario with mode switching. When the inertial position vector of the LMO spacecraft is positive, or it is in the sun-lit region of the planet, it must point towards the sun to gather power; otherwise, we can choose between nadir and GMO pointing regimes. When the spacecraft is in the eclipse regime, we must decide between nadir and GMO pointing. We use the heuristic of there being at most a 35 degree angle displacement between the inertial position vector of the LMO and GMO spacecraft. When the angular difference is within this bound, we point towards the GMO mother ship, otherwise, we nadir-point to collect science data.

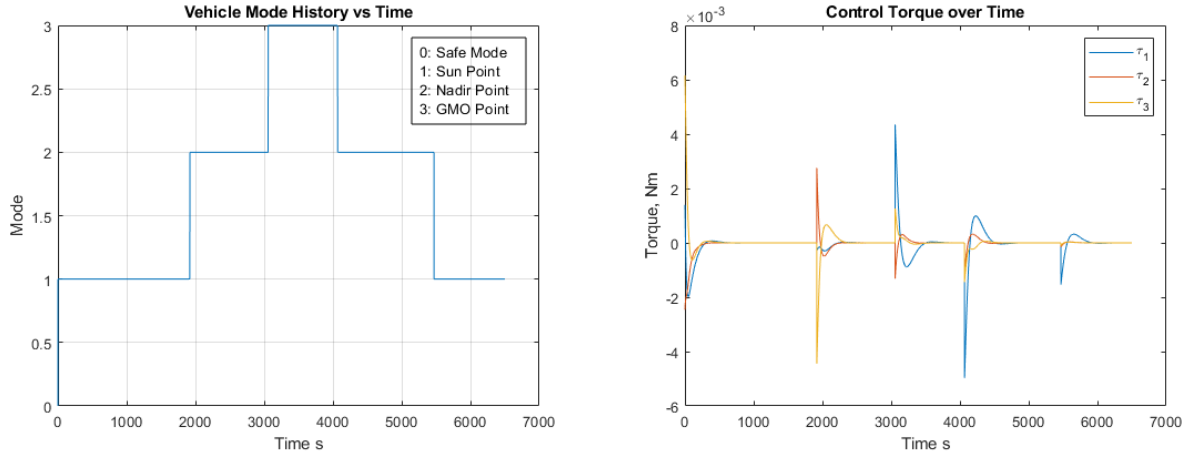
Simulating this mode switching logic for 6500 seconds yields a state plot 5(a) with error figure 5(b). Again, we notice convergence after each context switch, which are denoted in the mode history plot 6(a). The control used over time is displayed in figure 6(b).



a) States vs Time

b) Tracking Error vs Time

Figure 5: These figures show the states and state tracking errors over times for the full 6500 second simulation of operation.



a) This figure shows the mode of the spacecraft over the duration of the mission. This makes sense as the 35 degree angle constraint should only be a small subset of the eclipse duration.

b) This figure shows the on-axis control torques in Nm.

Figure 6: Mode and Control states for the full mission.

I. Conclusion and Future Work

In conclusion, the simulation and control functioned nominally and are extensible to more complicated mission and control architectures. This framework is a useful tool that can be used for future projects requiring attitude determination and control simulation. The immediate next steps are performing close actuator simulation and control. Momentum exchange devices are most likely to be used and have many issues that require deeper understanding and simulation. These traits could be device saturation, singularity avoidance, and control robustness under exogenous torques or Kane damping effects.

References

[1]Schaub, P. D. H., "Attitude Dynamics and Control of a Nano-Satellite Orbiting Mars," 2019.

[2]Schaub, H. and Junkins, J. L., *Analytical mechanics of space systems*, American Institute of Aeronautics and Astronautics, Inc, Reston, Virginia, 4th ed., 2018.

Project Code

Listing 1: Main Script (Simulation Framework)

```
%% Padraig Lysandrou April 4th 2019 -- ASEN5010 Final Project
clc; close all; clear all;

% Problem specified dynamic initial conditions
sigma_BN_0 = [0.3 -0.4 0.5].';
omega_BN_B_0 = deg2rad([1 1.75 -2.2]).';
Ic = diag([10 5 7.5]); p.Ic = Ic;

% Problem specified orbital parameters
mu_M =  42828.3;            % km3/s2
h_LMO = 400;                % km
R_M = 3396.19;              % km
r_LMO = h_LMO + R_M;
r_GMO = 20424.2;
Omega_GMO = 0; inc_GMO = 0; theta_GMO_0 = deg2rad(250);
Omega_LMO = deg2rad(20); inc_LMO = deg2rad(30); theta_LMO_0 = deg2rad(60);
theta_dot_LMO = sqrt(mu_M / (r_LMO^3)); % rad/s, orbital angular rate(circ)
theta_dot_GMO = sqrt(mu_M / (r_GMO^3)); % rad/s, orbital angular rate(circ)

% theta_dot_GMO = 0.0000709003;

% Take care of timing
tend = 6500;
dt = 1;
t = 0:dt:tend;
npoints = length(t);

% Set up the dynamic function as well as state tracking, init conds
f_dot = @(t_in,state_in,param) dynamics(t_in,state_in,param);
vehicle_state = zeros(6,npoints);
vehicle_state(:,1) = [sigma_BN_0; omega_BN_B_0];
p.L = [0.0 0 0].';
p.u = [0 0 0].';
% vehicle_mode_history = zeros(1,npoints);

% Setup the tracking error stuff, and input history stuff
sigma_BR = zeros(3,npoints);
omega_BR = zeros(3,npoints);
control_input = zeros(3, npoints);
sig_int = [0 0 0].';

% Set the initial conditions
H = zeros(3,npoints);
H(:,1) = Ic*omega_BN_B_0;
T = zeros(1,npoints);
T(1) = 0.5*omega_BN_B_0.'*Ic*omega_BN_B_0;

```

```
% PD controller gain initialization
P = max(diag(Ic).*(2/120));
K = (P^2)./Ic(2,2); % still not sure why this is the gain he wants...

% xi_1 = sqrt((P^2)./(K.*Ic(1,1))) % keeps ≤1
% xi_2 = sqrt((P^2)./(K.*Ic(2,2))) % does not keep ≤ 1
% xi_3 = sqrt((P^2)./(K.*Ic(3,3))) % does not keep ≤ 1


%% Start the simulation
vehicle_mode = 0;
vehicle_mode_history = vehicle_mode;

tic
for i = 1:npoints-1
    % Pull out state values to be used below
    sigma_BN = vehicle_state(1:3,i);
    omega_BN = vehicle_state(4:6,i);

    % Determine the inertial small sat and GMO vectors
    theta_LMO = theta_LMO_0 + t(i)*theta_dot_LMO;
    [rN_LMO,¬] = oe2rv_schaub(r_LMO,mu_M,Omega_LMO,inc_LMO,theta_LMO);
    theta_GMO = theta_GMO_0 + t(i)*theta_dot_GMO;
    [rN_GMO,¬] = oe2rv_schaub(r_GMO,mu_M,Omega_GMO,inc_GMO,theta_GMO);

    % Determine the mode of the spacecraft
    if rN_LMO(2)≥0
        % put the system into sun pointing mode
        vehicle_mode = 1;
    elseif acosd((rN_LMO.'*rN_GMO)/(norm(rN_LMO)*norm(rN_GMO))) < 35
        % put the system into GMO data transfer mode
        vehicle_mode = 3;
    else
        % put the system into nadir point science mode
        vehicle_mode = 2;
    end

    vehicle_mode_history = [vehicle_mode_history vehicle_mode];


    % Determine the trackgin DCM/Omega based on vehicle mode
    switch vehicle_mode
        case 1      % Sun pointing energy gather mode
            RN = dcm_RsN(t(i));
            omega_RN = omega_RsN(t(i));
            [sigma_BR(:,i), omega_BR(:,i)] = ...
                track_error(sigma_BN,omega_BN,RN,omega_RN);
        case 2      % Nadir pointing science mode
            RN = dcm_RnN(t(i));
            omega_RN = omega_RnN(t(i),theta_dot_LMO);
            [sigma_BR(:,i), omega_BR(:,i)] = ...
                track_error(sigma_BN,omega_BN,RN,omega_RN);
        case 3      % GMO comm pointing mode
            RN = dcm_RcN(t(i));
            omega_RN = omega_RcN(t(i));
            [sigma_BR(:,i), omega_BR(:,i)] = ...
                track_error(sigma_BN,omega_BN,RN,omega_RN);
        case 4      % Safemode hold
            sigma_BR(:,i) = [0 0 0].';
            omega_BR(:,i) = [0 0 0].';
        otherwise   % Safemode hold
            sigma_BR(:,i) = [0 0 0].';
            omega_BR(:,i) = [0 0 0].';
    end

    % Calculate the control input from mode error
    sig_int = sig_int + dt.*(sigma_BR(:,i));
    p.u = (-K.*sigma_BR(:,i)) + (-P.*omega_BR(:,i)); % + -0.01.*sig_int ;
```

```
    control_input(:,i) = p.u;


    % Pull out the angular momentum and the energy for debugging and
    % verification
    H(:,i) = Ic*omega_BN;
    T(i) = 0.5*omega_BN.'*Ic*omega_BN;

    % RK4 step for the spacecraft dynamics
    k_1 = f_dot(t(i),vehicle_state(:,i),p);
    k_2 = f_dot(t(i)+0.5*dt, vehicle_state(:,i)+0.5*dt*k_1,p);
    k_3 = f_dot((t(i)+0.5*dt),(vehicle_state(:,i)+0.5*dt*k_2), p);
    k_4 = f_dot((t(i)+dt),(vehicle_state(:,i)+k_3*dt), p);
    vehicle_state(:,i+1) = vehicle_state(:,i) + (1/6)*(k_1+(2*k_2)+(2*k_3)+k_4)*dt;

    % Perform the nonsingular MRP propagation attitude check
    s = norm(vehicle_state(1:3,i+1));
    if s > 1
        vehicle_state(1:3,i+1) = -(vehicle_state(1:3,i+1) ./(s^2));
    end
end
toc


%% Plot data here.
close all;

% figure; plot(t, H); title('angular momentum');
figure; subplot(2,1,1);
plot(t, vehicle_state(1:3,:)); title('MRP Attitude \sigma_{BN} vs Time');
xlabel('Time s'); ylabel('\sigma'); legend('\sigma_1','\sigma_2','\sigma_3');
subplot(2,1,2);
plot(t, vehicle_state(4:6,:)); title('Angular Rate \omega_{BN} vs Time')
xlabel('Time s'); ylabel('\omega rad/s'); legend('\omega_1','\omega_2','\omega_3');
% figure; plot(t, vecnorm(H)); title('Angular Momentum over time')

% figure; semilogy(t, T); title('system angular energy over time')
figure; subplot(2,1,1);
plot(t, sigma_BR); title('MRP \sigma_{BR} Error vs Time');
xlabel('Time s'); ylabel('\sigma'); legend('\sigma_1','\sigma_2','\sigma_3');
subplot(2,1,2);
plot(t, omega_BR); title('Angular Rate Error \omega_{BR} vs Time')
xlabel('Time s'); ylabel('\omega rad/s'); legend('\omega_1','\omega_2','\omega_3');

figure; plot(t, vehicle_mode_history); title('Vehicle Mode History vs Time')
xlabel('Time s'); ylabel('Mode');
grid on; hold on;
dim = [0.7 0.6 0.3 0.3];
str = {'0: Safe Mode','1: Sun Point','2: Nadir Point', '3: GMO Point'};
annotation('textbox',dim,'String',str,'FitBoxToText','on');
hold off;

figure; plot(t, control_input); title('Control Torque over Time');
xlabel('Time s'); ylabel('Torque, Nm');
legend('\tau_1','\tau_2','\tau_3');

% Saving things for Checkoff 7
% save_to_txt('H500B.txt', H(:,end));
% save_to_txt('T500.txt',T(end));
% save_to_txt('MRP500.txt',vehicle_state(1:3,end));
% save_to_txt('H500N.txt', MRP2C(vehicle_state(1:3,end)).'*H(:,end));

% close all;
% Saving things for Checkoff 8-10:
% save_to_txt('gains.txt', [P K]);
% save_to_txt('MRP15.txt', vehicle_state(1:3,16));
% save_to_txt('MRP100.txt', vehicle_state(1:3,101));
% save_to_txt('MRP200.txt', vehicle_state(1:3,201));
```

```
% save_to_txt('MRP400.txt', vehicle_state(1:3,401));

% close all;
% Saving things for Checkoff 11:
% save_to_txt('MRP300.txt', vehicle_state(1:3,301));
% save_to_txt('MRP2100.txt', vehicle_state(1:3,2101));
% save_to_txt('MRP3400.txt', vehicle_state(1:3,3401));
% save_to_txt('MRP4400.txt', vehicle_state(1:3,4401));
% save_to_txt('MRP5600.txt', vehicle_state(1:3,5601));
```

Listing 2: System Dynamics RHS

```
function f_x = dynamics(¬,state_in,p)
%%This function defines all of the system dynamics
    % pull out struct constants
    Ic = p.Ic;

    % distribute the states for use! these are row vectors.
    sigma_BN =  state_in(1:3);
    omega_BN =  state_in(4:6);

    % MRP integration, remember to perform norm check on this badboi: 3X1
    sigma_dot = (0.25.*((1 -(sigma_BN.'*sigma_BN))*eye(3) + 2*skew(sigma_BN.') ...
        + (2*sigma_BN*(sigma_BN.'))))*omega_BN;

    % Angular Rate Integration
    omega_dot = Ic\((-skew(omega_BN)*(Ic*omega_BN)) + p.L + p.u);

    % Send out the derivative
    f_x = [sigma_dot.' omega_dot.'].';
end
```

Listing 3: Tracking Error Determination

```
function [sigma_BR, omega_BR] = track_error(sigma_BN,omega_BN,RN,omega_RN)
%track_error return attitude and angular rate tracking error

% first tackle the error in attitude...
% sigma_RN = C2MRP(RN);
% sigma_BR = relMRP(sigma_BN.', sigma_RN.');
BN = MRP2C(sigma_BN);
sigma_BR = C2MRP(BN*(RN.'));
omega_BR = (omega_BN - BN*omega_RN);
end
```

Listing 4: Tracking Error Determination

```
function [rN,vN] = oe2rv_schaub(r,mu,Omega,incl,theta)
%oe2rv JUST FOR THIS PROJECT: NOT ROBUST FOR ANYTHING
    R3_omega = [ cos(theta)  sin(theta)  0
                -sin(theta)  cos(theta)  0
                 0       0     1];
    R3_Omega = [ cos(Omega)  sin(Omega)  0
                -sin(Omega)  cos(Omega)  0
                  0        0     1];
    R1_inc = [1       0          0
              0   cos(incl)  sin(incl)
              0  -sin(incl)  cos(incl)];
    % Take perifocal coordinates and transform to ECI coords
    perif_to_ECI = (R3_omega*R1_inc*R3_Omega).';
    rN =  perif_to_ECI*([r 0 0].');
    % This is an angular rate that only works for circular orbits:
    % basically just mean motion
    n = sqrt(mu/r^3);
    vN = perif_to_ECI* ([0 r*n 0].');
end
```