MATLAB Assignment

Final coursework

Jorge Antonio Chavarín Montoya 6759311

MSc. Space Engineering

Dr. Nicola Baresi & U. Abubacar

EEE3039 – Space Dynamics and Missions

Content

Part I: The Two-Body Problem

Kepler's equation Solution using Newton's Method

Various geometric properties of the orbit of a body can be described by Kepler's equation:

$$M = E - esin(E), \tag{1}$$

where:

- ***M*** is the ***Mean Anomaly***, defined as the angle between periapsis of an orbit and the position of an imaginary body, in this case a spacecraft, that orbits in the same period but at a constant angular velocity.
- ***e*** is the ellipse's ***eccentricity*** that describes how elliptical or non-circular an orbit is. A circular orbit would have an eccentricity of 0, meanwhile elliptical orbits have values between 0 and 1, higher values indicate greater separation between the foci in the ellipse, describing parabolic (*e = 1*) or hyperbolic trajectories (*e > 1*).
- ***E*** is the **Eccentric Anomaly**, the angle measured from the centre of an elliptical orbit to the projection of the spacecraft position on the ellipse's circumference, and the semi-major axis.

To accurate solve Eq. (1) and find solution for *E* given *M* and *e* values, an iterative procedure such as Newton's method should be used. This method finds the root of nonlinear equations of the form $f(x) = 0$ [1], by starting with an initial guess x_i evaluated at $f(x)$ and in its first derivative $f'(x)$, and iteratively, until $|f(x_k)| < tolerance$ previously defined, applying:

$$x_{i+1} = x_i + \delta(x_i); \tag{2}$$

where:

$$\delta(x_i) = -\frac{f(x_i)}{f'(x_i)} \tag{3}$$

In order to apply Newton's method, Kepler's equation (1) has to be defined as:

$$f(E) = E - e \sin E - M \tag{4}$$

Hence, the derivate $f'(E)$ would be:

$$f'(E) = 1 - e \cos E \tag{5}$$

Substituting Eq. (4) and (5) in Eq. (3):

$$\delta(E_i) = -\frac{E_i - e \sin E_i - M}{1 - e \cos E_i} \tag{6}$$

For this specific problem, it will be assumed that the initial guess of the Eccentric Anomaly, would be the same as the Mean Anomaly from

Appendix A. Coursework Data$E_0 = M_o = 127.0\ deg$, with an eccentricity $e = 0.0008$ and a tolerance of $Tol = 1x10^{-10}$. Figure 1 shows the iteration process followed to solve $f(E)$:

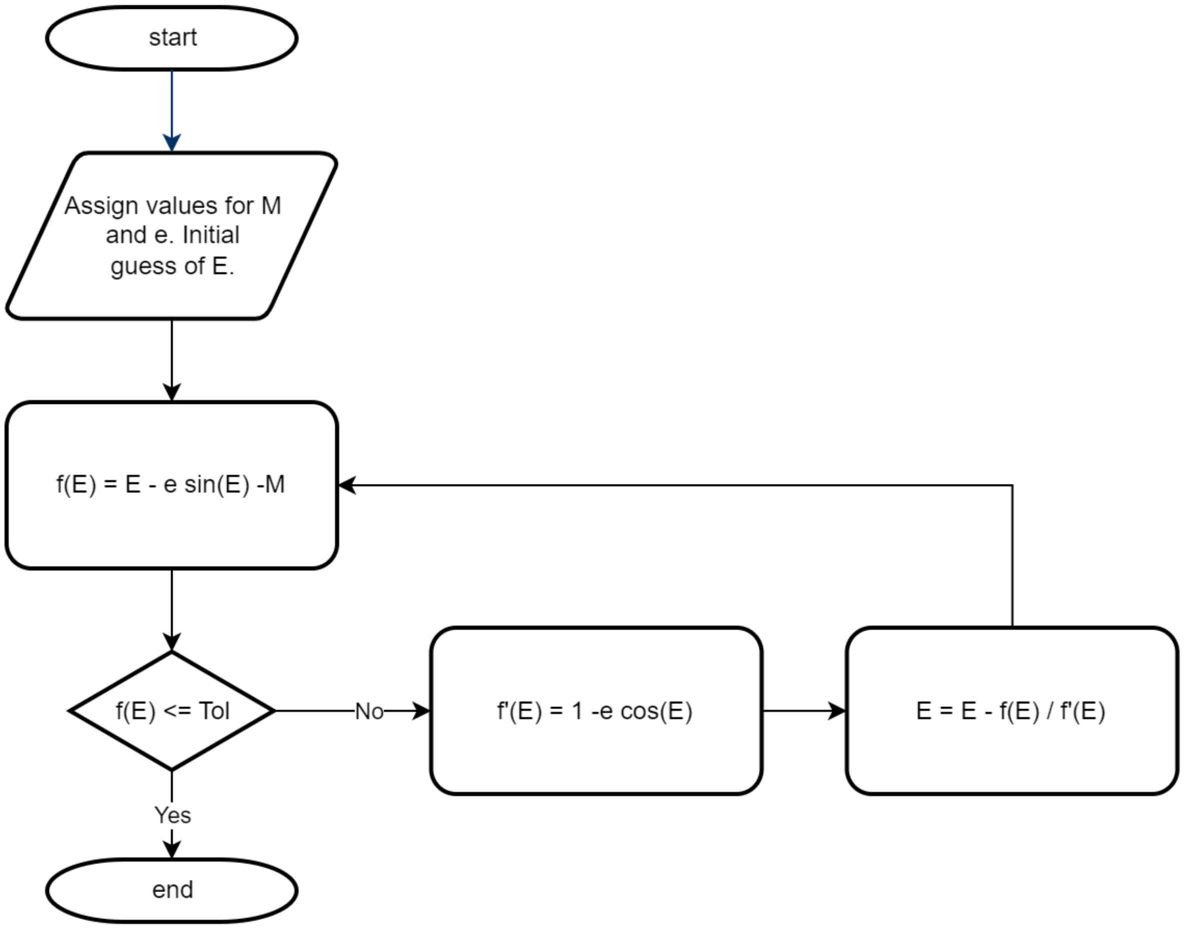

Figure 1. Kepler's Equation solution through Newton's Method.

The MATLAB function **Kepler(e, M, tol)** solves Kepler's equation using Newton's method after **three** iterations. According to the function, the initial Eccentric Anomaly of the satellite is:

$$E = 2.2172\ rad, \tag{7}$$

The True Anomaly can be calculated from the formula:

$$\tan\frac{\theta}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2}; \tag{8}$$

where:

- *E* is the Eccentric Anomaly from (7),
- *e* is the eccentricity of the orbit and,
- θ is the True Anomaly.

By factoring θ, the new equation can be written as:

$$\theta = 2\tan^{-1}\sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2}. \tag{9}$$

Substituting ***E*** and ***e*** in (9) the initial True Anomaly is:

$$\theta = 2.2178\, rad, \tag{10}$$

The satellite is *127.0732 degrees* (*2.2178 rad*) far from periapsis, which means it is closer to the apoapsis, only *52.9268 degrees* (*0.9237 rad)* far from there.

Orbit Propagation via Analytical solution of Two-Body Problem

Eccentric and True Anomalies were obtained at (7) and (10) for a specific Mean Anomaly. The equation that describes the Mean Anomaly over time is given by:

$$M = M_0 + n(t - t_0), \tag{11}$$

where:

- M_0 is the initial Mean Anomaly,
- n is the mean motion of the satellite around the Earth defined as:

$$n = \sqrt{{}^{\mu}/_{a^3}}, \tag{12}$$

  - being a the distance of the semi-major axis of the orbit ($a = 7151.16\ km$) and μ the Gravitational parameter ($\mu = \ 398600.4418\ \frac{km^3}{s^2}$) from

    - Appendix A. Coursework Data
- t_0 is the starting time to calculate *M*,
- *t* is the total time for which *M* will be calculated.

In order to calculate the Mean Anomaly for the entire orbital period ***P*** of the satellite, t should take values from *0* to *P*, $t \in [0, P]$, while $t_0 = 0\ s$. According to Curtis, H. [1] the period P can be obtained by the expression,

$$P = \frac{2\pi}{\sqrt{\mu}} a^{\frac{3}{2}}, \tag{13}$$

Substituting (12) in (13) an alternative expression can be found in different references,

$$P = \frac{2\pi}{n}, \tag{14}$$

As a result, the total period of the satellite is,

$$P = 6018.3262\ s. \tag{15}$$

True an Eccentric Anomalies could be calculated for all the orbital period too. From Eq. (11), let M_t be the Mean Anomaly at a specific time t. Once calculated, this value can be substituted in Eq. (1) to then applied Newton's method to obtain the Eccentricity Anomaly E_t at time *t*. Finally, True Anomaly θ_t can be calculated by substituting E_t in Eq. (9).
These steps must be repeated for each time *t* until $t = P$.
Figure 2 shows the values for all three anomalies (Mean, Eccentric and True). A MATLAB script can be found in Appendix B. MATLAB Script where a time vector of 1000 equal points from t_0 to *P* was created to calculate the anomalies.

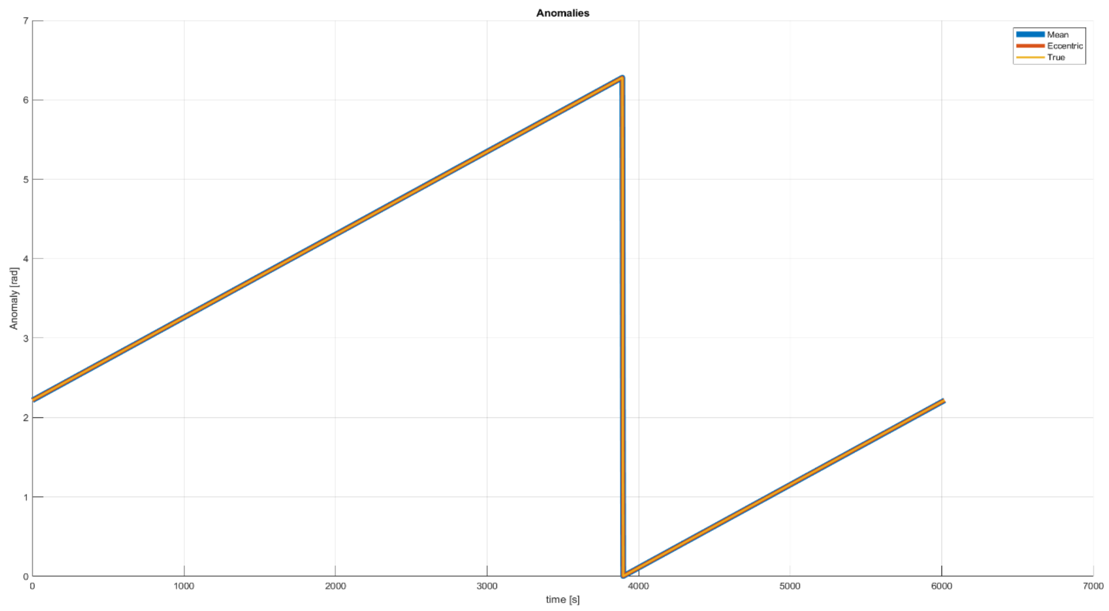

Figure 2. Mean, Eccentric & True Anomalies over satellite's orbital period, expressed in radians.

Since eccentricity of the orbit is approximately zero ($e = 0.0008$), which means that the trajectory of the satellite around the Earth approaches to a circular orbit; the three anomalies have similar values during the orbit.
Figure 2 shows also how the last value of the Mean Anomaly is the same as M_0 (127 deg. ≈ 2.21 rad.), after a complete orbit.

<u>Convert the state of a spacecraft from classic orbit elements to ECI position and velocity components.</u>

The Classic Orbit Elements vector (*c.o.e.*), is represented by:

$$coe = [a \ \ e \ \ i \ \ \Omega \ \ \omega \ \ \theta]^T, \tag{16}$$

where:

- a is the semi-major axis,
- e is the eccentricity of the orbit,
- i is the inclination of the orbit,
- Ω is the Right Ascension of the Ascending Node (RAAN), the angle measured positively at the equatorial plane from the I vector to the Ascending Node (the point in the equatorial plane where the satellite crosses the equator from south to north) [2],
- ω is the argument of perigee, an angle measured from the ascending node to periapsis and,
- θ is the True Anomaly.

To convert from *c.o.e.* to E.C.I. components the first thing to do is to determinate the position and velocity vectors in the perifocal system, and then rotate them into the geocentric equatorial system [2]. Because the trajectory of the orbit is elliptical, there is no

problem when using the semimajor-axis $\boldsymbol{a}$; that is not the case for example of parabolic orbits where $a = \infty$.
In the perifocal frame $P = \{m_1 | \hat{\imath}_e, \hat{\imath}_p, \hat{\imath}_h\}$ the position vector is defined as:

$$\boldsymbol{r} = r\hat{\boldsymbol{e}}_r = r\ \cos\theta\, \hat{\imath}_e + r\ \sin\theta\, \hat{\boldsymbol{\imath}}_p. \tag{17}$$

The magnitude of **r** is given by:

$$r = \frac{a(1-e^2)}{(1+e\ \cos\theta)}. \tag{18}$$

The velocity vector is the differentiation of Eq. (17) over time:

$$\dot{\boldsymbol{r}} = \left(\dot{r}\ \cos\theta - r\ \sin\theta\, \dot{\theta}\right)\hat{\boldsymbol{\imath}}_e + \left(\dot{r}\ \sin\theta + r\ \cos\theta\, \dot{\theta}\right)\hat{\boldsymbol{\imath}}_p, \tag{19}$$

and from (18):

$$\dot{r} = \frac{\mu}{h} e\ \sin\theta, \tag{20}$$

where μ is gravitational parameter and *h* is defined as:

$$h = \sqrt{\mu a(1-e^2)}. \tag{21}$$

After substituting (18) and (20) in (19) the resultant velocity vector is:

$$\dot{\boldsymbol{r}} = -\left(\frac{\mu}{h}\sin\theta\right)\hat{\boldsymbol{\imath}}_e + (\frac{\mu}{h}(\cos\theta + e))\hat{\boldsymbol{\imath}}_p. \tag{22}$$

Rewriting both position and velocity vectors as 3x1 matrixes:

$$\boldsymbol{r} = [r\ \cos\theta\,, r\ \sin\theta\,, 0]^T, \tag{23}$$

$$\dot{\boldsymbol{r}} = [-\frac{\mu}{h}\sin\theta\,, \frac{\mu}{h}(\cos\theta + e), 0]^T. \tag{24}$$

The next step is to rotate the vectors into the E.C.I. frame. The Direction Cosine Matrix (DCM), from the E.CI. to the Perifocal frame, can be represented as a 3-1-3 sequence of single axis rotations: $[PJ] = R_3(\omega)\, R_1(i)\, R_3(\Omega)$. Therefore, the DCM from the Perifocal frame to the E.C.I. is the transpose of it $[JP] = [PJ]^t$.
Defining [PJ] as:

$$[PJ] = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix} \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{25}$$

The final DCM [JP], after multiplying the rotation matrices and transposing [PJ], the DCM **[JP]** is:

$$\begin{matrix} \cos\Omega\cos\omega - \sin\Omega\sin i\sin\omega & -\cos\Omega\sin\omega - \sin\Omega\cos i\cos\omega & \sin\Omega\sin i \\ \sin\Omega\cos\omega + \cos\Omega\cos i\sin\omega & -\sin\Omega\sin\omega + \cos\Omega\cos i\cos\omega & -\cos\Omega\sin i \\ \sin i\sin\omega & \sin i\cos\omega & \cos i \end{matrix} \tag{26}$$

Multiply the DCM (26) by (23) gives as a result the position vector on ECI frame:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = [JP]\begin{pmatrix} r\ \cos\theta \\ r\ \sin\theta \\ 0 \end{pmatrix}, \tag{27}$$

while, multiplying (26) by (24) gives the velocity vector:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = [JP] \begin{pmatrix} -\frac{\mu}{h}\sin\theta \\ \frac{\mu}{h}(\cos\theta + e) \\ 0 \end{pmatrix}. \tag{28}$$

The new state of the spacecraft in the ECI, could be defined by:

$$\boldsymbol{X} = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} \tag{29}$$

This process must be made for each True Anomaly value over the time vector t. The result is a matrix of True Anomalies vectors where, initial and final values of vector $\boldsymbol{X}$ at t_0 and at the final value of t highlight for being equal:

$$\boldsymbol{X} = \begin{pmatrix} 7046.1370 \\ 1241.0704 \\ 9.0390 \\ 0.1844 \\ -1.0731 \\ 7.3824 \end{pmatrix} \tag{30}$$

This means that the satellite covered a complete orbit and returned to the initial point. Figure 3 shows the trajectory of the spacecraft around the ECI frame, where the red point represents the convergence between the initial and final values of the ECI position vector. In addition, the plot demonstrates that the orbit has an inclination of $i = 98.39\ degrees$, it is parallel to vectors $\hat{\boldsymbol{\imath}}_e$ and $\hat{\boldsymbol{\imath}}_p$.

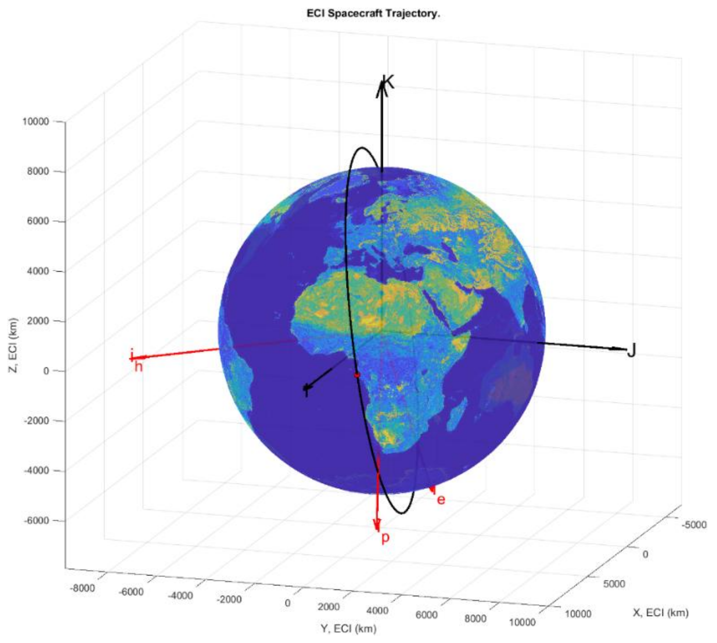

Figure 3. Spacecraft trajectory on ECI frame.

Numerical Integration in the ECI frame

From Two-Body problem, the equation that describes the movement of the satellite taking in consideration only the gravitational influence of the Earth is given by:

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r}, \tag{31}$$

By means of the numerical integration method, second-order differential equations must be converted into a system of first-order equations to know the position and velocity vectors at each time t using the MATLAB function **TBP_ECI()** from Appendix B.

A new vector of the form $\dot{\boldsymbol{X}} = \binom{\dot{\boldsymbol{r}}}{\ddot{\boldsymbol{r}}}$ must be defined where, $\dot{\boldsymbol{r}}$ is the velocity vector, and $\ddot{\boldsymbol{r}}$ is the acceleration vector described in (31). Using the original vector (29), the differentiating vector can be rewritten as:

$$\dot{\boldsymbol{X}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\frac{\mu}{r^3}x \\ -\frac{\mu}{r^3}y \\ -\frac{\mu}{r^3}z \end{pmatrix}, \tag{32}$$

Integration was calculated over the same time vector t from the Two-Body problem while the values from (30) were used as the initial conditions. The value of r is the magnitude of vector $\boldsymbol{r}$.

MATLAB provide two different integration functions *ode45* and *ode113*. Solution may vary depending which integration function is used among with the relative and absolute tolerances. After testing both functions, *ode113* was selected as it seemed to have a better performance in the results. The relative tolerance used was $3\text{x}10^{-14}$ and while the absolute tolerance was $1\text{x}10^{-16}$.

Results from the numerical integration method are equivalent to the position and velocity vectors obtained through the analytical solution of the Two-Body problem with a difference in a scale of 10^{-7} and 10^{-10} respectively.

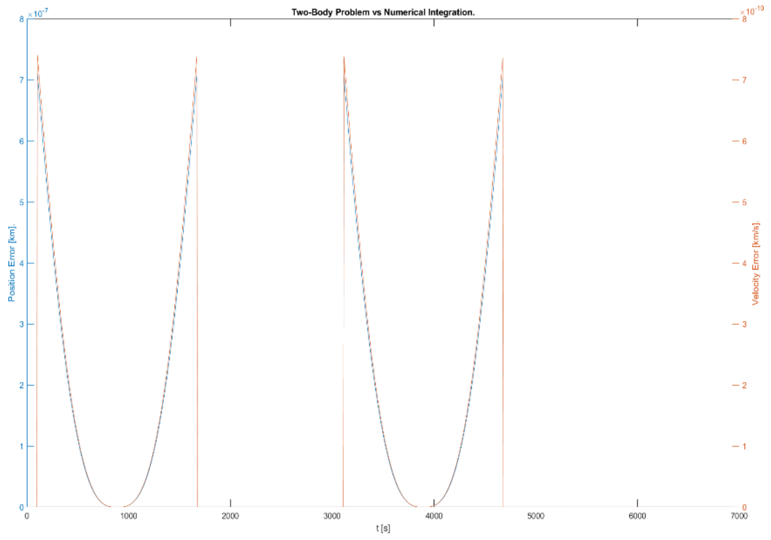

Figure 4.Position and velocity errors over time t between Analytical Solution of Two-Body Problem and Numerical Integration in the ECI frame using ode113.

Once the position and velocity vectors are obtained, the next equation is used to calculate the specific energy $\boldsymbol{\zeta}$:

$$\zeta = \frac{1}{2}v^2 - \frac{\mu}{r}, \tag{33}$$

where r is the magnitud of the position vector and v the magnitude of the velocity at each time t. The graph below shows how the specific energy is approximately $-\frac{\mu}{2a}$, due to the fact that the relation between the velocity and the position magnitudes tends not to vary ove time.

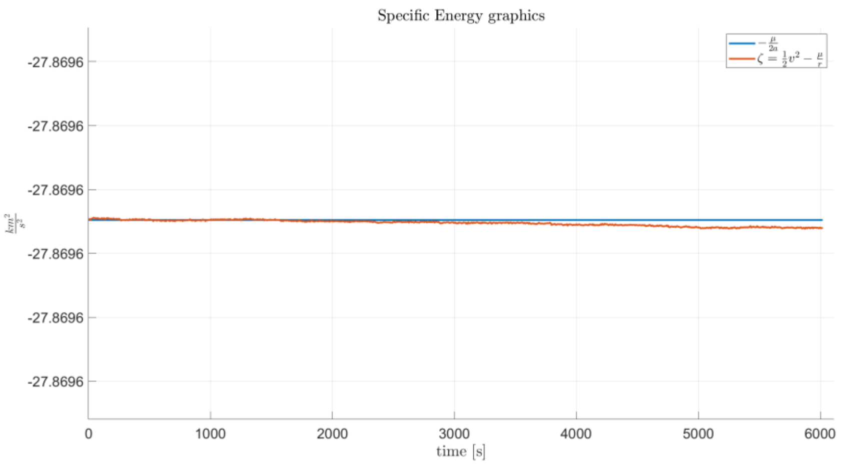

Figure 5. Specific Energy ζ vs $-\frac{\mu}{2a}$.

Derive and Integrate the Equations of Motion in the ECEF frame.

The Earth-Centered Earth-Fixed Frame (**ECEF**) is a geocentric coordinates system fixed to the Earth rotation. The $\hat{\boldsymbol{Z}}$ axis is parallel to the North Pole, the $\hat{\boldsymbol{X}}$ axis intercepts the $\hat{\boldsymbol{Z}}$ axis in the Earth equatorial plane and Greenwich prime meridian. 0° latitude and 0° longitude. The $\hat{\boldsymbol{Y}}$ axis is the orthogonal to $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Z}}$.

The ECEF is especially useful for spacecraft observation and tracking. Because ECEF is fixed to the Earth rotation, its components differ from the ECI frame:

- *East longitude*, λ. Angle measured from the Greenwich meridian to the meridian where the object to tracked or observed is located.
- *Geocentric latitude,* φ_{gc}. Angle between the desired object and the equatorial plane, measured in the same meridian where the object is located.

In order to map the ECEF frame from the ECI and vice versa, a special rotation matrix of the form:

$$[IF] = [P][N][R][W], \tag{34}$$

Where **[P]** and **[R]** correspond to the precession and nutation of the Earth's rotation Axis. **[R]** describes the rotation of the Earth. **[W]** is the polar motion, describes the orientation of the true North with respect to the instantaneous Rotation Axis of the Earth. Since the polar motion and the precession and nutation are long time effects, for satellite tracking these three terms can be ignored. Thus, the only component used is **[R]**, a single rotation about the third axis $\widehat{\boldsymbol{Z}}$.

$$[IF] = [\underset{I_e}{\cancel{P}}][\underset{I_e}{\cancel{N}}][R][\underset{I_e}{\cancel{W}}] = R_3(-\Theta), \tag{35}$$

From Eq.(31) it is possible to derive the equations of motion of a satellite as seen from the ECEF. Using the transport theorem let define the velocity of the satellite in the ECI:

$$^{I}\dot{\boldsymbol{r}} = (\,^{F}\dot{\boldsymbol{r}} + \omega_{F/I} \times \boldsymbol{r}), \tag{36}$$

Where subscripts and superscripts *F* & *I*, refer to ECEF frame and ECI frame, respectively.

Thus, the acceleration of the satellite as seen from the ECI is given by:

$$^{I}(\,^{I}\ddot{\boldsymbol{r}}) = \,^{F}(\,^{I}\ddot{\boldsymbol{r}}) + \omega_{F/I} \times \,^{I}\dot{\boldsymbol{r}}, \tag{37}$$

Substituting eq. (36) in (37):

$$^{I}(\,^{I}\ddot{\boldsymbol{r}}) = \,^{F}\left(\frac{d}{dt}(\,^{F}\dot{\boldsymbol{r}} + \omega_{F/I} \times \boldsymbol{r})\right) + \omega_{F/I} \times \,^{I}\dot{\boldsymbol{r}}, \tag{38}$$

Solving the equation and assuming the angular velocity of the ECEF frame as seen from the ECI frame is constant $\dot{\omega} = 0$ and parallel to the third axis of the ECEF $\omega_{F/I} = \omega_{\oplus}\widehat{\boldsymbol{Z}} = \omega_{\oplus}\widehat{\boldsymbol{K}}$:

$$^{I}(\,^{I}\ddot{\boldsymbol{r}}) = \,^{F}(\,^{F}\ddot{\boldsymbol{r}}) + \,^{F}(\cancelto{0}{\,^{F}\dot{\omega}_{F/I}}) \times \boldsymbol{r} + \omega_{F/I} \times \,^{F}\dot{\boldsymbol{r}} + \omega_{F/I} \times \,^{F}\dot{\boldsymbol{r}} + \omega_{F/I} \times (\omega_{F/I} \times \boldsymbol{r}) \tag{39}$$

$$^{I}\ddot{\boldsymbol{r}} = [IF]\,^{F}\ddot{\boldsymbol{r}} + 2(\omega_{F/I} \times \,^{F}\dot{\boldsymbol{r}}) + \omega_{F/I} \times (\omega_{F/I} \times \boldsymbol{r}) \tag{40}$$

$$^{F}\ddot{\boldsymbol{r}} = [FI](\,^{I}\ddot{\boldsymbol{r}}) - 2(\omega_{F/I} \times \,^{F}\dot{\boldsymbol{r}}) - \omega_{F/I} \times (\omega_{F/I} \times \boldsymbol{r}) \tag{41}$$

Where $2(\omega_{F/I} \times \,^{F}\dot{\boldsymbol{r}})$ and $\omega_{F/I} \times (\omega_{F/I} \times \boldsymbol{r})$ are the Coriolis and Centrifugal accelerations.

Similar to the process followed in Numerical Integration in the ECI frame, lets define an original vector $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{r}\\ \boldsymbol{v}\end{bmatrix}$. In order to write equations in a first-order system, we need to create a second derivative vector $\dot{\boldsymbol{X}} = \begin{bmatrix}\dot{\boldsymbol{r}}\\ \ddot{\boldsymbol{r}}\end{bmatrix}$ where $\dot{\boldsymbol{r}}$ is the velocity of the satellite $\boldsymbol{v}$. $\ddot{\boldsymbol{r}}$ is described in eq. (41), where $(\,^{I}\ddot{\boldsymbol{r}})$ represents the acceleration of the satellite in the ECI frame defined at Eq. (31), $\omega_{F/I}$ is the angular velocity or the Earth $\omega_{F/I} = 7.2921x10^{-5}\ rad/s$.

Assuming that both frames are aligned at *t=0*, meaning that the angle between ECI and ECEF becomes zero $\Theta = 0$ rad. $[FI]$ becomes an identity matrix $[FI] = [IF]^{-1}$:

$$[IF] = R_3(-\Theta) = \begin{bmatrix}\cos\Theta & \sin\Theta & 0\\ -\sin\Theta & \cos\Theta & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \tag{42}$$

The equation can be rewritten as follows:

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} - 2(\omega_{F/I} \times \boldsymbol{v}) - \omega_{F/I} \times (\omega_{F/I} \times \boldsymbol{r}), \tag{43}$$

and the vector can be written as:

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{v} \\ -\frac{\mu}{r^3}\boldsymbol{r} - 2\left(\omega_{F/I} \times {}^{F}\boldsymbol{v}\right) - \omega_{F/I} \times \left(\omega_{F/I} \times \boldsymbol{r}\right) \end{bmatrix}, \tag{44}$$

To integrate Integration was done using the function *ode113*. The plot below shows the trajectory of the satellite in the ECEF frame for 10 orbit periods. Note that the first and final points are aligned just in the third axis. This is because the orbital period and the period of the Earth are not aligned.

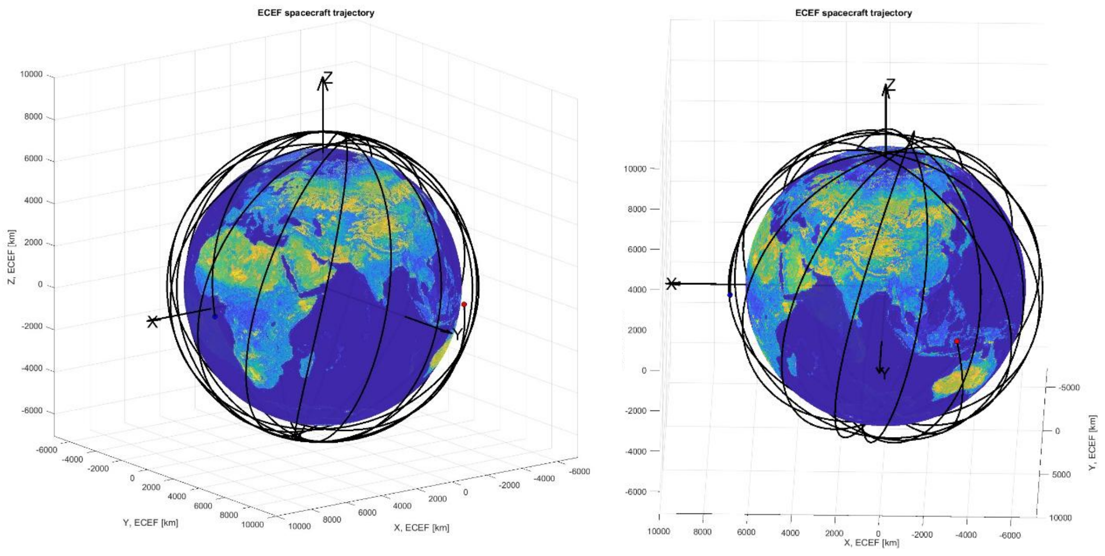

Figure 6. Spacecraft trajectory as seen from different perspectives of the ECEF frame.

Part II: Attitude Representations ad Torque-Free Motion

Spacecraft ECI position and velocity coordinates one hour before passing through its periapsis:

$$r = \begin{bmatrix} 6768.27 \\ 870.90 \\ 21153.59 \end{bmatrix} km, \qquad v = \begin{bmatrix} -2.0519 \\ -1.4150 \\ 7.0323 \end{bmatrix} \frac{km}{s} \tag{45}$$

Direction Cosine Matrices

The orbital frame ***RSW*** is a satellite coordinate system that moves with the satellite [3]. The ***R*** axis ($\hat{\boldsymbol{e}}_r$) points out from the centre of the Earth to the satellite as it moves through the orbit. The ***W*** axis ($\hat{\boldsymbol{e}}_h$) is normal to the orbital plane and shares the direction of the angular momentum vector $\boldsymbol{h}$ which is defined as $\boldsymbol{h} = \boldsymbol{r} \times \boldsymbol{v}$ [3]. The ***S*** ($\hat{\boldsymbol{e}}_\theta$) completes the right-hand rule, completing the orthogonal set, with the same direction of the velocity vector.

The rotation matrix $[IO]$ that goes from RSW frame O to the ECI frame I can be expressed using the definition of the RSW frame, explained in the previous paragraph:

$$[IO] = \begin{bmatrix} \hat{\boldsymbol{e}}_r \\ \hat{\boldsymbol{e}}_\theta \\ \hat{\boldsymbol{e}}_h \end{bmatrix}, \tag{46}$$

where $\hat{\boldsymbol{e}}_r$ and $\hat{\boldsymbol{e}}_h$ are normalized unit vectors: $\hat{\boldsymbol{e}}_r = \frac{\boldsymbol{r}}{r}$ and $\hat{\boldsymbol{e}}_h = \frac{\boldsymbol{h}}{h}$; whereas $\hat{\boldsymbol{e}}_\theta = \hat{\boldsymbol{e}}_h \times \hat{\boldsymbol{e}}_r$.

Coordinates from (45) substitute $\boldsymbol{r}$ and help to calculate $\boldsymbol{h}$ vector. The magnitudes of both vectors follow the next equations: $r^2 = {r_1}^2 + {r_2}^2 + {r_3}^2$ and $h^2 = {h_1}^2 + {h_2}^2 + {h_3}^2$. Thus, the $[IO]$ parameters are:

$$[IO] = \begin{bmatrix} 0.9458 & -0.2755 & 0.1718 \\ 0.1217 & -0.1897 & -0.9743 \\ 0.3010 & 0.9424 & -0.1459 \end{bmatrix}, \tag{47}$$

Given the 1-2-1 sequence of the Euler angles $\theta = [\alpha, \beta, \gamma]^T$, where $\alpha = 30\ deg$, $\beta = 20\ deg$, and $\gamma = 10\ deg$, it is possible to calculate the direction cosine matrix $[BO]$ from the orbital frame O to t he principal axes frame of the satellite B.

Let $[BO] = R_1(\gamma)R_2(\beta)R_1(\alpha)$. Note that the first rotation to be made $R_1(\alpha)$ is written at the last part o f the equation. This is because the order of the rotations matters and will determine the resultant vector orientation. The numbers in the subscripts correspond to the specific sequence of the rotation and mean the axis on which the rotation is made. The next matrixes show the specific rotation for each axis:

$$R_1(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix},$$
$$R_2(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{48}$$
$$R_1(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix},$$

The complete direction cosine matrix $[BO]$ with a 1-2-1 sequence can be defined then as:

$$[BO] = \begin{bmatrix} c\beta & s\beta * c\alpha & -s\beta * c\alpha \\ s\gamma * s\beta & -s\gamma * c\beta * s\alpha + c\gamma * c\alpha & s\gamma * c\beta * c\alpha + c\gamma * s\alpha \\ c\gamma * s\beta & -c\gamma * c\beta * s\alpha - s\gamma * c\alpha & c\gamma * c\beta * c\alpha - s\gamma * s\alpha \end{bmatrix}, \tag{49}$$

where $s = \sin()$ and $c = \cos()$. Substituting the actual α, β and γ angles,

$$[BO] = \begin{bmatrix} 0.9397 & 0.1710 & -0.2962 \\ 0.0594 & 0.7713 & 0.6337 \\ 0.3368 & -0.6131 & 0.7146 \end{bmatrix}, \tag{50}$$

Due to the properties of the RSW cosine matrix, it is possible to rotate from the ECI frame I to the orbital frame O by transposing Eq. (47), ($[OI] = [IO]^T$).

With the rotation matrices from the ECI frame I to the orbital frame O, $[OI]$, and from the orbital frame O to the principal axes frame of the satellite B, a new direction cosine matrix $[BI]$ directly from the ECI frame $I$to the principal axes frame B, defined as $[BI] = [BO][OI]$.

Values of this cosine matrix are given below:

$$[BI] = \begin{bmatrix} 0.7908 & 0.3705 & 0.4872 \\ -0.0474 & -0.7565 & 0.6523 \\ 0.6102 & -0.5389 & -0.5807 \end{bmatrix}, \tag{51}$$

Attitude Representations

According to Euler's Principal Rotation Theorem a single rotation through a principal angle $\boldsymbol{\Phi}$ about the principal axis $\hat{\boldsymbol{e}}$ "*can brought a rigid body or a reference frame from an arbitrary initial orientation to an arbitrary final orientation*". [4] The principal axis is fixed in both initial and final orientation. By rotating about $\hat{\boldsymbol{e}}$ by an angle Φ the final reference frame can be obtained from the initial frame, as vector $\hat{\boldsymbol{e}}$ is the same in both frames. Typically, Φ is selected by te shortest rotation about $\hat{\boldsymbol{e}}$, however it is also possible to rotate in the opposite direction by an angle Φ' ($\Phi' = \Phi - 2\pi$). The vector set $(\hat{\boldsymbol{e}}, \Phi)$ is not unique, $(-\hat{\boldsymbol{e}}, -\Phi)$ describe the same orientation as the first set.

The equation that describes the rotation from I to B, same rotation from (51) but using the principal angle and axis given by:

$$[BI] = \cos\Phi\, I_3 - \sin\Phi\, [\tilde{\boldsymbol{e}}] + (1 - \cos\Phi)\hat{\boldsymbol{e}}\hat{\boldsymbol{e}}^T, \tag{52}$$

where:

- $I_3$is the 3x3 identity matrix,
- $[\tilde{\boldsymbol{e}}] = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}$,
- $\hat{\boldsymbol{e}}\hat{\boldsymbol{e}}^T = \begin{bmatrix} e_1^2 & e_1 e_2 & e_1 e_3 \\ e_1 e_2 & e_2^2 & e_2 e_3 \\ e_1 e_3 & e_2 e_3 & e_3^1 \end{bmatrix}$,

Given the resultant matrix, values for e_1, e_2, e_3 and Φ can be obtained using the next set of equations:

$$\Phi = \cos^{-1} \frac{Tr([BI]) - 1}{2}, \tag{53}$$

$$e_1 = \frac{(b_{23} - b_{32})}{2\sin\Phi}, \tag{54}$$

$$e_2 = \frac{(b_{31} - b_{13})}{2\sin\Phi}, \tag{55}$$

$$e_3 = \frac{(b_{12} - b_{21})}{2\sin\Phi}, \tag{56}$$

Where *Tr()* is the trace of $[BI]$ defined by the sum of the elements in the main diagonal. The numerical values of the set $(\hat{\boldsymbol{e}}, \Phi) = ([e_1, e_2, e_3]^T, \Phi)$ are:

$$(\hat{\boldsymbol{e}}, \Phi) = ([0.9392 \quad 0.0970 \quad 0.3295]^T \quad 2.4547\, rad) \tag{57}$$

Quaternions, also known as Euler parameters, are another method to define axis-angle representation using a four-dimensional vector $\boldsymbol{\beta}_{B/_I} = [\beta_0 \quad \beta_1 \quad \beta_2 \quad \beta_3]^T$. They are commonly used in computer applications due primarily to the fact that they allow to make rotations about multiple axis simultaneously, and not sequentially as the normal direction cosine matrix. Each element of the quaternion vector can be defined in terms of the principal rotation elements $(\hat{\boldsymbol{e}}, \Phi)$:

$$\beta_0 = \cos\frac{\Phi}{2}, \tag{58}$$

$$\beta_1 = e_1 \sin\frac{\Phi}{2}, \tag{59}$$

$$\beta_2 = e_2 \sin\frac{\Phi}{2}, \tag{60}$$

$$\beta_3 = e_3 \sin\frac{\Phi}{2}, \tag{61}$$

Using eq. (57) the quaternion elements ca be calculated:

$$\boldsymbol{\beta}_{B/_I} = [0.3368 \quad 0.8843 \quad 0.0913 \quad 0.3102]^T \tag{62}$$

Since $\beta_0 = 0.3368$, is nonnegative, quaternion $\boldsymbol{\beta}$ corresponds to the shortest rotation.

The norm is given by $\boldsymbol{\beta}^2 = {\beta_0}^2 + {\beta_1}^2 + {\beta_2}^2 + {\beta_3}^2 = 1$, because it describes geometrically a four-dimensional unit sphere. To prove this, values from eq. (62) (numbers used in the report are trimmed at 4 decimals) were used to calculate the norm and the result of the operation was equal to ${\boldsymbol{\beta}_{B/_I}}^2 = 1$.

A 3-2-1 sequence of Euler angles (yaw-pitch-roll) is represented by the next rotations in e inverse order: $[BI] = R_1(\phi)R_2(\theta)R_3(\psi)$ and the equations to obtain the angles from the direction cosine matrix for that specific sequence of rotations are:

$$\psi = \text{atan2}(b_{12}, b_{11}), \tag{63}$$

$$\theta = \sin^{-1} -b_{13}, \tag{64}$$

$$\phi = atan2(b_{23}, b_{33}), \tag{65}$$

where b_{nm} is a value at the n-row and m-column of $[BI]$ (51).

The Euler angles $\boldsymbol{\theta} = (\psi \quad \theta \quad \phi)$ are:

$$\boldsymbol{\theta}_{Euler} = (0.438 \quad -0.5089 \quad 2.2982)\, rad \tag{66}$$

$$\boldsymbol{\theta}_{Euler} = (25.1037 \quad -29.1567 \quad 131.6766)\, deg \tag{67}$$

Given the values of the Euler angles, the Kinematic Differential Equation matrix $B(\boldsymbol{\theta})$ can be deducted. Starting from the equation ($\dot{\boldsymbol{\theta}} = [B(\boldsymbol{\theta})]\omega$) that describes the angular velocity vector of the B frame relative to the I frame for the specific 3-2-1 rotation sequence.

Let the vectorial expression for the angular velocity vector of the B frame relative to the I frame expressed in B-frame components:

$$^{B}\left(\boldsymbol{\omega}_{B/I}\right) = R_1(\phi)R_2(\theta)R_3(\psi)\begin{pmatrix}0\\0\\\dot{\psi}\end{pmatrix} + R_1(\phi)R_2(\theta)\begin{pmatrix}0\\\dot{\theta}\\0\end{pmatrix} + R_1(\phi)\begin{pmatrix}\dot{\phi}\\0\\0\end{pmatrix} \tag{68}$$

After the multiplication of the rotational matrixes by the differential angles:

$$^{B}(\boldsymbol{\omega}_{B/I}) = \dot{\psi}\begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} + \dot{\theta}\begin{pmatrix} 0 \\ \cos\phi \\ -\sin\phi \end{pmatrix} + \dot{\phi}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{69}$$

This equation can be rewritten to form a new direction cosine matrix:

$$^{B}(\boldsymbol{\omega}_{B/I}) = \begin{bmatrix} -\sin\theta & 0 & 1 \\ \cos\theta\sin\phi & \cos\phi & 0 \\ \cos\theta\cos\phi & -\sin\phi & 0 \end{bmatrix}\begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} \tag{70}$$

Finally, the matrix must be inverted to find the kinematic differential equation for the 3-2-1 sequence of Euler Angles:

$$\begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \frac{1}{\cos\theta}\begin{bmatrix} 0 & \sin\phi & \cos\phi \\ 0 & \cos\theta\cos\phi & -\cos\theta\sin\phi \\ \cos\theta & \sin\theta\sin\phi & \sin\theta\cos\phi \end{bmatrix}\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \tag{71}$$

Where ψ, θ and ϕ, are the Euler angles from (66), $\boldsymbol{\theta} = (\psi \quad \theta \quad \phi)$. As a result, the matrix $B(\boldsymbol{\theta})$ can be defined numerically as:

$$B(\boldsymbol{\theta}) = 1.1451\begin{bmatrix} 0 & 0.7469 & -0.6649 \\ 0 & -0.5807 & -0.6523 \\ 0.8733 & -0.3639 & 0.3240 \end{bmatrix} \tag{72}$$

Kinematics and Euler's Equations

Let the angular velocity vector of the spacecraft in B-frame coordinates be:

$$\boldsymbol{\omega}_{B/I} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} -3.092 \times 10^{-4} \\ 6.6161 \times 10^{-4} \\ 7.4606 \times 10^{-4} \end{bmatrix}, \qquad rad/s \tag{73}$$

The angular momentum vector about the center of mass of the spacecraft ($\boldsymbol{H} = [I]\omega$) in the B frame $^{B}(\boldsymbol{H})$ can be defined by the next equation:

$$^{B}(\boldsymbol{H}) = {}^{B}[I]\,{}^{B}(\omega) = \begin{pmatrix} I_1\omega_1 \\ I_2\omega_2 \\ I_3\omega_3 \end{pmatrix}, \tag{74}$$

where

- $^{B}[I] = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix}$
- I_1, I_2, I_3 are the principal moments of inertia.
- $^{B}(\omega)$ is the angular velocity in the B-frame.

According to the Euler's equation, the total external torque acting on the satellite $\boldsymbol{L}$ can be defined by

$$\dot{\boldsymbol{H}} = \boldsymbol{L}, \tag{75}$$

By using the transport theorem, the equation above can be expressed as

$$\dot{\boldsymbol{H}} = \frac{{}^{B}d}{dt}(\boldsymbol{H}) + \boldsymbol{\omega} \times \boldsymbol{H} = \boldsymbol{L}. \tag{76}$$

Since the matrix of Inertia $[I]$ is constant over time, in addition to the fact the derivative of the angular velocity vector $\boldsymbol{\omega}$ is the same in the B and the I frames; the derivate of the angular momentum as seen from the B frame can be written as

$$\frac{{}^{B}d}{dt}(\boldsymbol{H}) = [I]\dot{\boldsymbol{\omega}}. \tag{77}$$

Substituting eq. (74) and (77) in (76):

$$\boldsymbol{L} = [I]\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times ([I]\boldsymbol{\omega}). \tag{78}$$

The equation can be rewritten to

$$[I]\dot{\boldsymbol{\omega}} = (\boldsymbol{L} - [\tilde{\boldsymbol{\omega}}][I]\boldsymbol{\omega}). \tag{79}$$

Because the inertia matrix $[I]$ is a diagonal, since the body-fixed coordinate is aligned to the principal body axes, Eq. (79) reduces to

$$\begin{pmatrix} I_1\dot{\boldsymbol{\omega}}_1 \\ I_2\dot{\boldsymbol{\omega}}_2 \\ I_3\dot{\boldsymbol{\omega}}_3 \end{pmatrix} = \begin{pmatrix} L_1 + (I_2 - I_3)\omega_2\omega_3 \\ L_2 + (I_3 - I_1)\omega_1\omega_3 \\ L_3 + (I_1 - I_2)\omega_1\omega_2 \end{pmatrix}. \tag{80}$$

If no external forces act on the spacecraft (torque-free motion), the angular momentum vector as seen from the I frame is fixed in magnitude and direction:

$$\dot{\boldsymbol{H}} = \boldsymbol{L} = 0\,, \tag{81}$$

However, components of the angular velocity vector in the B frame will vary in time. Hence, Eq. (80) can be rewritten to

$$\begin{pmatrix} \dot{\boldsymbol{\omega}}_1 \\ \dot{\boldsymbol{\omega}}_2 \\ \dot{\boldsymbol{\omega}}_3 \end{pmatrix} = \begin{pmatrix} \frac{(I_2 - I_3)}{I_1}\omega_2\omega_3 \\ \frac{(I_3 - I_1)}{I_2}\omega_1\omega_3 \\ \frac{(I_1 - I_2)}{I_3}\omega_1\omega_2 \end{pmatrix}. \tag{82}$$

Let define the attitude state of the spacecraft as

$$\boldsymbol{X} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \tag{83}$$

In order to integrate the state in a period of time, a new vector $\dot{\boldsymbol{X}}$ must be defined as in Numerical Integration in the ECI frame: $\dot{\boldsymbol{X}} = \begin{pmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix}$, where $\dot{\boldsymbol{\theta}}$ is defined in Eq. (71) and $\dot{\boldsymbol{\omega}}$ in Eq. (82). The differentiated vector at the n iteration ($\dot{\boldsymbol{X}}_{(n)}$) can be expressed as

$$\dot{\boldsymbol{X}}_{(n)} = \begin{pmatrix} \dot{\boldsymbol{\theta}}_{(n)} \\ \dot{\boldsymbol{\omega}}_{(n)} \end{pmatrix} = \begin{pmatrix} \left[B\left(\boldsymbol{\theta}_{(n-1)}\right)\right]\omega_{(n-1)} \\ [I]^{-1}\left(-\left[\widetilde{\boldsymbol{\omega}}_{(n-1)}\right][I]\boldsymbol{\omega}_{(n-1)}\right) \end{pmatrix} \tag{84}$$

The integration is done in MATLAB for $t \in [0{,}3600]$ s in steps of 10 seconds, that means that the maximum number of iterations is $n = 361$. The initial values at $t = 0\ s$ of the Euler angles $\boldsymbol{\theta} = [\psi \quad \theta \quad \phi]^T$ are the same angle solved at (66), while the initial angular velocity vector $\boldsymbol{\omega} = [\omega_1 \quad \omega_2 \quad \omega_3]^T$ is defined in (73). Tolerances were 3x10^{-12} and 1x10^{-14} for Relative and Absolute tolerance respectively, while ode113 was the integration function used. Figure 7. Euler angles and angular velocity components.Figure 7 shows the time history of the Euler angles along 3600 s.

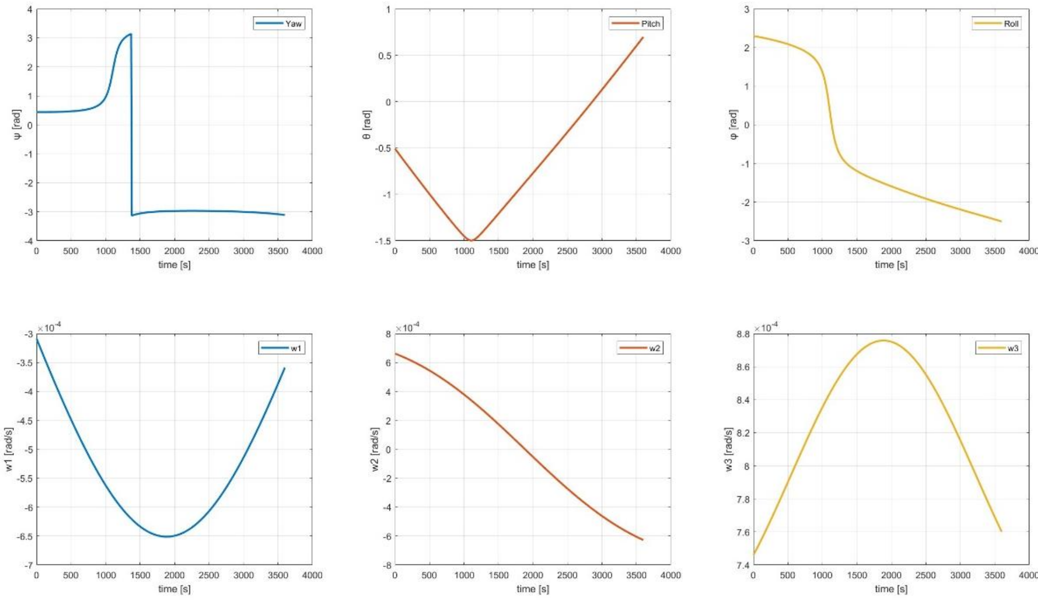

Figure 7. Euler angles and angular velocity components.

Singularity is defined by Pitch (θ) angle, in the $B(\boldsymbol{\theta})$ matrix value: $\frac{1}{\cos\theta}$. If $\theta = \pm\pi\ rad$, multiple combinations of Euler Angles values describe the same attitude matrix. This singularity in real missions would make computers not able to do their calculations.

Although, from Figure 7 it could be appreciated that the pitch angle as if it was reaching the singularity value, Figure 8 zooms- in and validates that the system is never reaching $= \pm\pi\ rad$. Meanwhile, after almost 1500 s the satellite stabilize in its third axis, where the ψ angle tends to stay constant.

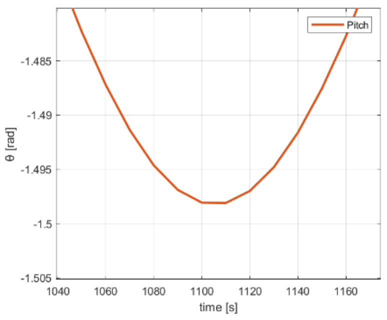

Figure 8. Zoom-in Pitch θ

As it was already mentioned the eq. that describes the angular momentum at each n-state of the spacecraft is.

$$\boldsymbol{H}_n = [I]\omega_n \tag{85}$$

The matrix of Inertia does not change at any iteration as it remains constant during time.

In addition, because the calculations were done in a torque-free motion, see eq. (75), the magnitude of the angular momentum is conserved too, even when observed from the principal axes frame of the satellite. Nevertheless, its components vary according to the rotations from Euler equations. Figure 9 shows the angular momentum magnitude from the equation:

$$H^2 = {H_1}^2 + {H_2}^2 + {H_3}^2 \tag{86}$$

Finally, the rotational kinetic energy of the spacecraft at the n iteration is given by:

$$T_n = \frac{1}{2}\boldsymbol{\omega}_n{}^T\boldsymbol{H}_n, \tag{87}$$

Where $\boldsymbol{\omega}_n$ is the angular velocity calculated from the MATLAB integration function and $\boldsymbol{H}_n$ comes from (85). Figure 9 shows how Torque tends to remain constant.

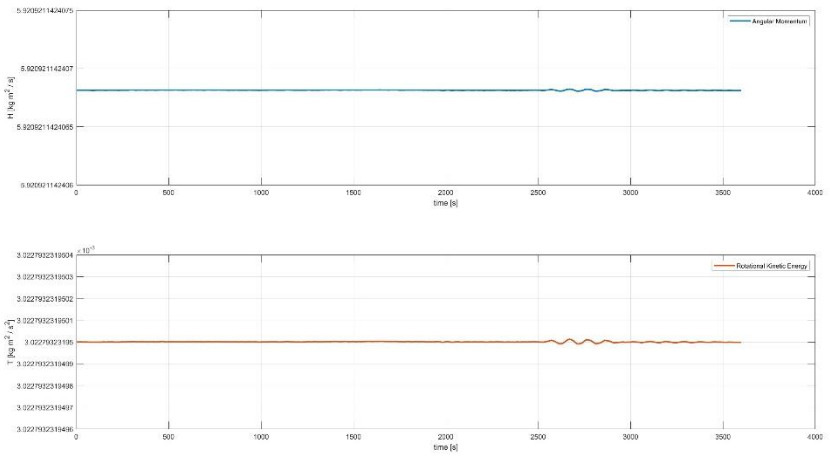

Figure 9. Angular momentum and Torque against time.

References

1. Curtis, H. D. (2014). Orbital Mechanics for Engineering Students. Florida: Butterwort-Heinemann. Retrieved from https://www-sciencedirect-com.surrey.idm.oclc.org/book/9780080977478/orbital-mechanics-for-engineering-students#book-info
2. Vallado, D. A. (2007). Fundamentals of astrodynamics and applications. Hawthorne, California: Microcosm Press. Retrieved from https://contentstore.cla.co.uk/EReader/Index?guid=32d15984-eb28-eb11-80cd-005056af4099&pcid=2668572&t=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJkNjUyNjA0Mi1mOGJjLTRkMWUtYmMxYy0xOTRmZTZmZDZiMDAiLCJuYmYiOjE3MDA1ODkwMzIsImV4cCI6MTcwMDU4OTMzMiwiaWF0IjoxOjox
3. Pesce, V. C. (2023). Modern Spacecraft Guidance, Navigation, and Control. Elsevier. doi:https://doi.org/10.1016/C2020-0-03563-2
4. Schaub, H. J. (2000). Analytical mechanics of space systems. American Institute of Aeronautics and Astronautics. Retrieved from https://ebookcentral.proquest.com/lib/surrey/reader.action?docID=3111684&ppg=118

Appendix A. Coursework Data

- Gravitational parameter:
  - $\mu = 398600.4418\ \frac{km^3}{s^2}$;
- Earth radius:
  - $R_\oplus = 6378.137$ km;
- Earth rotation angular velocity:
  - $\omega_\oplus = 7.2921x10^{-5}\ \frac{rad}{s}$;
- Spacecracft inertia matrix in principal axes frame:
  - $I = \begin{bmatrix} Ixx & 0 & 0 \\ 0 & Iyy & 0 \\ 0 & 0 & Izz \end{bmatrix}$;

  where,
    - $Ixx = 2500\ kg\ m^2$;
    - $Iyy = 5000\ kg\ m^2$;
    - $Izz = 6500\ kg\ m^2$;
- Identity matrix I_3:
  - $I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Initial orbit elements of a satellite in a Keplerian orbit around the Earth:

- Semi-major axis:
  - $a = 7151.16\ km$;
- Eccentricity of the Orbit:
  - $e = 0.0008$;
- Inclination of the Orbit:
  - $i = 98.39\ deg$;
- Right Ascension of the Ascending Node (RAAN):
  - $\Omega = 10.00\ deg$;
- Argument of Periapsis:
  - $\omega = 233.0\ deg$;
- Initial Mean Anomaly:
  - $M_0 = 127\ deg$
- Tolerance factor:
  - $Tol = 1x10^{-10}$.

Appendix B. MATLAB Script

Kepler.m

```
function [E,i] = Kepler(e,M,tol)
% By: Jorge Chavarín
% Kepler's equation solution using Newton's method.
%
% INPUTS:
%   e       Eccentricity
%   M       Mean Anomaly
%   tol     Error tolerance
%
% OUTPUTS:
%   E       Eccentric Anomaly
%   i       Number of iterations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% M = E-e*sin(E)
% f(x,e,M) = E - e*sin(E) - M


% Initial values
E = M;
for i = 1:100                           % Evaluate function 100 times max.

    fE = E - e*sin(E) - M;              % evaluate function in En
    dfE = 1-e*cos(E);                   % evaluate first derivative in En

    % check for convergence using IF statement
    if(abs(fE) <= tol)
        break;
    else
        % Newton's method
        dE = -fE/dfE;                          % delta fE_n
        E = E + dE;                            % update value of fEn
    end
end
```

COERV.m

```
function [X] = COE2RV(coe,u)
% By: Jorge Chavarín
% Convert the state of a spacecraft from classic orbit elements
% to ECI position and velocity components
% INPUTS:
%   coe  Classic Orbit Elements Vector
%               coe         [a, e, i, RAAN, argPer, TA];
%               a           Semi-major axis
%               e           Eccentricity of the Orbit
%               i           Inclination of the Orbit
%               RAAN        Right Ascension of the Ascending Node (RAAN)
%               argPer      Argument of Periapsis
%               TA          True Anomaly
%
%   u    Gravitational Parameter G(m1+m2)
% OUTPUT:
```

```
%   X   ECI position and velocity coordinates

a = coe(1);
e = coe(2);
i = coe(3);
RAAN = coe(4);
argPer = coe(5);
TA = coe(6);


% Rotation Matrix JP
JP = [
    cos(RAAN)*cos(argPer) - sin(RAAN)*cos(i)*sin(argPer),   -
cos(RAAN)*sin(argPer) - sin(RAAN)*cos(i)*cos(argPer),  sin(RAAN)*sin(i);
    sin(RAAN)*cos(argPer) + cos(RAAN)*cos(i)*sin(argPer),   -
sin(RAAN)*sin(argPer) + cos(RAAN)*cos(i)*cos(argPer),  -cos(RAAN)*sin(i);
    sin(i)*sin(argPer),                                    sin(i)*cos(argPer),
cos(i);
    ];
% Map position and inertial velocity vectors from P'frame to ECI.
r = ( a*(1-e^2) ) / (1 + e*cos(TA));
h = sqrt(u*a*(1-e^2));

pos = JP*[r*cos(TA);r*sin(TA);0];                   % Position Coordinates
vel = JP*[-(u/h)*sin(TA); (u/h)*(cos(TA)+e); 0];    % Velocity Coorcinates

% ECI position and velocity coordinates
X = [pos; vel];
end
```

<u>TBP_ECI.m</u>

```
function [dXdt] = TBP_ECI(t, X, u)
%   By: Jorge Chavarín
% Equations of motion rewrited as a system of first order ordinary differential
equation
%
% INPUTS:
%   t       Time
%   X       State vector [position, velocity] or [r,v]
%   u       Gravitational Parameter G(m1+m2)
%
% OUTPUT:
%   dXdt    Time derivate of state vector [vel., acc.]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Two-body problem equations of motion:
% r̈ = d^2r / dt = -(mu/r^3)*r.   % r^3 -> scalar.
%
% Equation rewrited.
% ṙ = dr / dt   = v              % v = vel.
% r̈ = dv / dt   = -(mu/r^3)*r    %r^3 -> scalar
%
% X = [r, v]
% Ẋ = [v, -(mu/r^3)*r]^T  = [rdot, vdot]
% r c = [ (xPos)^2 + (yPos)^2 + (zPos)^2 } ^ (1/2)
r = sqrt((X(1)^2) + (X(2)^2) + (X(3)^2) );

% Constant value used at r̈
cte = -(u/(r^3));

% Vector of zeros
dXdt = zeros(6,1);

dXdt(1) = X(4);
dXdt(2) = X(5);
dXdt(3) = X(6);

dXdt(4) = cte * X(1);
dXdt(5) = cte * X(2);
dXdt(6) = cte * X(3);

end
```

TBP_ECEF.m

```
function [dXdt] = TBP_ECEF(t,X, mu)
%   By: Jorge Chavarín
% Equations of motion rewrited as a system of first order ordinary differential
equation
%
% INPUTS:
%   t       Time
%   X       State vector [position, velocity] or [r,v]
%   mu       Gravitational Parameter G(m1+m2)
%
% OUTPUT:
%   dXdt    Time derivate of state vector [vel., acc.]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transport Theorem problem equations of motion:
%
% Equation rewrited from I frame
% ṙ = dr / dt   = v              % v = vel.  --> Vi
% r̈ = dv / dt   = -(mu/r^3)*r    %r^3 -> scalar in ECI Frame   --> Ai
% W = Wf/i  = Angular velocity of the ECEF frame as seen from the ECI frame
% W = 7.2921x10-5 Z [rad/s]
%
% At ECEF
% i(ṙ) = i(Vf) = Vi - W x r           % Vf is vel. at ECEF seen from the ECI
% ṙ = Vf = [FI] i(Vf)                 % Vf seen from ECEF
% [FI] --> Identity matrix as both frames are parallel at Xo means  θ = 0
% r̈ = Af = Ai - 2(W x Vf) - W x W x r
% r̈ = Af = Ai - 2(W x (Vi - W x r)) - W x W x r
% r̈ = -(mu/r^3)*r - 2(W x (Vi - W x r) - W x W x r
% X = [r, Vf]
% Ẋ = [Vf, Af]^T

% r scalar = [ (xPos)^2 + (yPos)^2 + (zPos)^2 } ^ (1/2)
r = sqrt((X(1)^2) + (X(2)^2) + (X(3)^2) );
```

```
w = 7.2921e-5;                % Angular vel. of the ECEF frame. [rad/s]
W = [0;0;w];
posf = [X(1);X(2); X(3)];
velf = [X(4); X(5); X(6)];

% Constant value used at r̈
Ai = -(mu/(r^3))*X(1:3);
FI = eye(3);
% Vector of zeros
dXdt = zeros(6,1);

% Vf = [EF] (Vi - W x r )
% Vf = Vi - W x r
%dXdt(1:3,1) =  [FI]* [X(4:6) - cross(W, X(1:3))];
dXdt(1:3, 1) = velf;
%  r̈ = Af = [EF] * [Ai - 2(W x (Vi - W x r)) - W x W x r]
%  r̈ = Af = [EF] * [Ai - 2(W x Vf) - W x W x r]
%  r̈ = Af = [EF] * [Ai - Coriolis - Centrifugal]

% Coriolis --> 2(W x (Vi - W x r))
coriolis = 2 * ( cross( W, velf ) );
% Centrifugal --> W x W x r
centrifugal = cross(W, cross(W,posf ));

dXdt(4:6, 1) = [FI]* [Ai - coriolis - centrifugal];

end
```

<u>AttitudeDynamics.m</u>

```
function [dXdt] = AttitudeDynamics(t,X, I)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

alpha = X(1);
beta = X(2);
gamma = X(3);
I1 = I(1,1);
I2 = I(2,2);
I3 = I(3,3);

W = X(4:6);
matrix = [
    0,              sin(gamma),                cos(gamma);
    0,              cos(beta)*cos(gamma),     -cos(beta)*sin(gamma);
    cos(beta),     sin(beta)*sin(gamma),    sin(beta)*cos(gamma)];

B0 = (1/cos(beta)) * matrix  *W;

w_dot = [
    (I2-I3)/I1 * W(2) * W(3);
    (I3-I1)/I2 * W(1) * W(3);
    (I1-I2)/I3 * W(1) * W(2)];

dXdt = [B0; w_dot];
```

```
end
```

Main Script> main.m

```
% By: Jorge Chavarin
clear all; close all; clc;
format longG

%% DATA
u = 398600.4418;        % [km3/s2]  Gravitational Parameter G(m1+m2)
Re = 6378.137;          % [km]      Earth Ratio
w = 7.2921e-5;          % [rad/s]   Angular vel. Earth rotation
Ixx = 2500;             % [kg m^2]
Iyy = 5000;             % [kg m^2]
Izz = 6500;             % [kg m^2]
% m2km = 1/ 1000;
% Ixx = Ixx*(m2km^2);
% Iyy = Iyy*(m2km^2);
% Izz = Izz*(m2km^2);
I = [                   % Spacecraft inertia matrix
    Ixx, 0,     0;          % in principal axes frame.
    0,   Iyy,   0;
    0,   0,     Izz];

%% Part 1: Two-Body Problem %%
% Keplerian orbit elements:
a = 7151.16 ;           % [km]      Semi-major axis
e = 0.0008;             % []        Eccentricity of the Orbit

% Inclination of the Orbit
i_deg = 98.39;          % [deg]
i = deg2rad(i_deg);     % [rad]

% Right Ascension of the Ascending Node (RAAN)
RAAN_deg = 10.0;        % [deg]
RAAN = deg2rad(10.0);   % [rad]

% Argument of Periapsis
argPer_deg = 233;       % [deg]
argPer = deg2rad(233);  % [rad]

Mo_deg = 127;           % [deg]     Initial Mean Anomaly
Mo = deg2rad(127);      % [rad]

tol = 1e-10;            % []        Tolerance factor

%% Week 2
% Solving Keplers equation using Newton's Method and Find Initial True
% Anomaly
fprintf('<strong>Mean Anomaly (Mo)</strong>\n Mo = <strong>%.4f
[deg]</strong>.\n\n', Mo_deg);
[E,ii] = Kepler(e,Mo,tol);      % [rad] E = Eccentric Anomaly
Edeg = rad2deg(E);              % [deg] Eccentric Anomaly
fprintf('<strong>Ecentric Anomaly (E)</strong> after %d iterations:\n', ii);
fprintf('E = <strong>%.4f [rad]</strong> = <strong>%.4f [deg]</strong>.\n\n', E,
Edeg);

% Calculating True Anomaly

%%%% True Anomaly Equation %%%%
theta = 2*atan2(sqrt(1+e) * tan(E/2), sqrt(1-e));
thetadeg = rad2deg(theta);                            % [deg] True Anomaly
fprintf('<strong>True Anomaly (θ)</strong>:\nθ = <strong>%.4f [rad]</strong> =
<strong>%.4f [deg]</strong>.\n', theta, thetadeg);
fprintf('\nThe satellite is near <strong>apoapsis</strong>.\n\n');

%% Week 3
%%% Orbit Propagation via Analytical %%
%%% Solution of Two-Body Problem %%%%

% Orbit Period
n = sqrt(u/a^3);                        % [rad/s]   Mean  Motion (rad/s)
P = 2*pi/n;                             % [s]       Period of the Orbit(s)
fprintf('Orbit Period: %.4f (s)\n', P);


% Create time vector of 1000 equally spaced points t∈[0,P]
% assuming t0 = 0 s
t0 = 0;
t = linspace(t0, P+t0, 1000);

% Calculate Mean Anomally
MA = Mo + n*(t-t0);                     % [rad] Mean Anomaly

% Empty vectors of size P, to store Anomalies
TA = t; EA = t;                         % [rad] vectors
EAdeg = t; TAdeg = t; MAdeg = t;        % [deg] vectors

for j = 1:length(t)
    [EA(j),ii] = Kepler(e,MA(j),tol);    % [rad] EA = Excentricity Anomaly
    TA(j) = 2*atan2(sqrt(1+e) * tan(EA(j)/2), sqrt(1-e)); % [rad] True Anomaly

    % Classic Orbits Elements
    % coe = [a; e; i; Ω; ω; θ]
    coe(:,j) = [a, e, i, RAAN, argPer, TA(j)];

    % Calculate ECI position and velocity
    X(:,j) = COE2RV(coe(:,j), u);      % [[km], [km/s]] [pos, vel]
end

for jj = 1:6
    XX(jj, :) = [X(jj, 1), X(jj, end)];
end

% X = [x; y; z; ẋ; ẏ; ż]
fprintf('X      Initial values       Final Values \n')
fprintf('x:  %15f%18f\n', XX(1,1), XX(1,2));
fprintf('y:  %15f%18f\n', XX(2,1), XX(2,2));
fprintf('z:  %15f%18f\n', XX(3,1), XX(3,2));
fprintf('ẋ: %15f%18f\n', XX(4,1), XX(4,2));
fprintf('ẏ: %15f%18f\n', XX(5,1), XX(5,2));
fprintf('ż: %15f%18f\n', XX(6,1), XX(6,2));

% Plot Anomalies
% Radians
figure('Name','Anomalies','units','normalized','outerposition',[0 0 1 1]); hold
on; grid on;
plot(t, wrapTo2Pi(MA), 'LineWidth', 6); plot(t, wrapTo2Pi(EA), 'LineWidth', 4);
plot(t, wrapTo2Pi(TA), 'LineWidth', 2);
legend('Mean','Eccentric', 'True');
title('Anomalies');
xlabel('time [s]')
ylabel('Anomaly [rad]')

hold off;

%{
% Degrees
% Convert Anomalies form radians to degrees
EAdeg = rad2deg(EA);          % [deg] Eccentric Anomaly
TAdeg = rad2deg(TA);          % [deg] True Anomaly
MAdeg = rad2deg(MA);          % [deg] Mean Anomaly
figure('Name','Anomalies','units','normalized','outerposition',[0 0 1 1]); hold
on; grid on;
plot(t, MAdeg, 'LineWidth', 4); plot(t, EAdeg, 'LineWidth', 2); plot(t, TAdeg,
'LineWidth', 2);
legend('Mean','Eccentric', 'True');
title('Anomalies');
xlabel('time [s]')
ylabel('Anomaly [deg]')
hold off;
%}

% Degrees
%{
MA_pi_deg = rad2deg(MA_pi);
EA_pi_deg = rad2deg(EA_pi);
TA_pi_deg = rad2deg(TA_pi);
figure('Name','Anomalies','units','normalized','outerposition',[0 0 1 1]); hold
on; grid on;
plot(t, MA_pi_deg, 'LineWidth', 4); plot(t, EA_pi_deg, 'LineWidth', 2); plot(t,
TA_pi_deg, 'LineWidth', 2);
legend('Mean','Eccentric', 'True');
title('Anomalies');
xlabel('time [s]')
ylabel('Anomaly [deg]')
hold off;
%}

%3D Plot of the ECI spacecraft trajectory
% Create figure
figure('Name', 'ECI Frame','units','normalized','outerposition',[0 0 1 1])
[Xe,Ye,Ze] = sphere(50);
```

```
% Plot Earth
img = imrotate(imread('BlueMarble_square.png'), 0);
surf(Re*Xe, Re*Ye, Re*Ze, 'CData', img, 'EdgeColor', 'none', 'FaceColor',
'texturemap', 'FaceAlpha','.8'); hold on; axis equal;
% surf(Re*Xe, Re*Ye, Re*Ze, 'EdgeColor', '#A2142F', 'FaceColor', 'c',
'FaceAlpha','1'); hold on; axis equal;
title('ECI Spacecraft Trajectory.');
xlabel('I, ECI (km)');
ylabel('J, ECI (km)');
zlabel('K, ECI (km)');

% Plot ECI axes
quiver3(0, 0, 0, 1, 0, 0, 1e4, 'k', 'LineWidth', 2);    % I-axis
quiver3(0, 0, 0, 0, 1, 0, 1e4, 'k', 'LineWidth', 2);    % J-axis
quiver3(0, 0, 0, 0, 0, 1, 1e4, 'k', 'LineWidth', 2);    % K-axis
text(1e4, 0, 0, 'I', 'FontSize', 20, 'Interpreter', 'tex', 'Color', 'k')
text(0, 1e4, 0, 'J', 'FontSize', 20, 'Interpreter', 'tex', 'Color', 'k')
text(0, 0, 1e4, 'K', 'FontSize', 20, 'Interpreter', 'tex', 'Color', 'k')


% Plot Trajectory
plot3(X(1,:), X(2,:), X(3,:), 'k', 'LineWidth', 2);
plot3(X(1,1), X(2,1), X(3,1), 'ok', 'MarkerFaceColor', 'b');
plot3(X(1,end), X(2,end), X(3,end), 'ok', 'MarkerFaceColor', 'r');


%%%{
% Calculate eccentricity vector, specific angular momentum vector, and
% complete the triad
r = X(1:3, 1);
v = X(4:6, 1);
h = cross(r, v);
ee = cross(v, h)/u - r/norm(r);

ie = ee/norm(ee);
ih = h/norm(h);
ip = cross(ih, ie)/norm(cross(ih, ie));

quiver3(0, 0, 0, ie(1), ie(2), ie(3), 1e4, 'r', 'LineWidth', 2);
quiver3(0, 0, 0, ip(1), ip(2), ip(3), 1e4, 'r', 'LineWidth', 2);
quiver3(0, 0, 0, ih(1), ih(2), ih(3), 1e4, 'r', 'LineWidth', 2);
text(1e4*ie(1), 1e4*ie(2), 1e4*ie(3), 'i_e', 'FontSize', 20, 'Interpreter',
'tex', 'Color', 'r')
text(1e4*ip(1), 1e4*ip(2), 1e4*ip(3), 'i_p', 'FontSize', 20, 'Interpreter',
'tex', 'Color', 'r')
text(1e4*ih(1), 1e4*ih(2), 1e4*ih(3), 'i_h', 'FontSize', 20, 'Interpreter',
'tex', 'Color', 'r')

hold off
%%%}
%% Week 4 %%%%%
ode_opt = odeset('RelTol', 3e-14, 'AbsTol', 1e-16);

[tout, Xout] = ode113(@TBP_ECI, t, X(:, end), ode_opt, u);
```

```
% Function to extract ode from main script
%[tout, Xout] = TBP_ECI_ode(t, X(:, end), u);   % Ẋ = [ṙ; v̇] =[v; -(mu/r^3)*r]

% Position and Velocity error Vector over time t
dX = Xout.' - X;

% Calculate Positionand Velocity Scalar error.
eV = zeros(1,length(t));
eX = zeros(1,length(t));
for tt = 1:length(t)
    eX(tt) = sqrt( dX(1,tt)^2 + dX(2,tt)^2 + dX(3,tt)^2);
    eV(tt) = sqrt( dX(4,tt)^2 + dX(5,tt)^2 + dX(6,tt)^2);
end

%Plot Position and Velocity Errors. Two-Body Problem vs Numerical Integration.
figure('Name', 'Pos & Vel Errors');
title("Two-Body Problem vs Numerical Integration.");
yyaxis left
plot(t,eX)
ylabel("Position Error [km].")
yyaxis right
plot(t,eV)
ylabel("Velocity Error [km/s].")
xlabel('t [s]');

% Week 4.3 Specific Energy
% ζ = 1/2(v^2)- mu/r = cte
specific_energy = zeros(1, length(t)) -u/(2*a);                 % [km/s]
SE = zeros(1,length(t));                          % Specific Energy ζ zeros vector
err = zeros(1,length(t));                         % Error zeros vector
for tt = 1:length(t)
    r = sqrt( Xout(tt, 1)^2 + Xout(tt, 2)^2 + Xout(tt, 3)^2);   % [km]
    v = sqrt( Xout(tt, 4)^2 + Xout(tt, 5)^2 + Xout(tt, 6)^2 );  % [km/s]
    SE(tt) =  ( (v^2) /2) - ( u/r );              % Specific Energy ζ [km^2/s^2] using
actual pos and vel
    err(tt) = SE(tt) - -u/(2*a);      % Calculating error
    %specific_energy(tt) = -u/(2*a);           % [km/s]
end

% Plot Specific Energy
figure('Name', 'Specific Energy graphics'); hold on; grid on;
plot(t, specific_energy, 'LineWidth', 2);

plot(t, SE, 'LineWidth', 2);
% legend('−µ/2a', 'ζ');
lg =legend('$-\frac{\mu}{2a}$','$\zeta = \frac{1}{2}v^2 - \frac{\mu}{r}$');
set(lg, 'Interpreter', 'latex');
title('Specific Energy graphics');
xlabel('time [s]');
ylabel('$\frac{km^2}{s^2}$', 'Interpreter', 'latex');
% $x^2+e^{\pi i}$
ytickformat('%.4f');
hold off;

%% Week 5
```

```
period_earth = 86164;
% period_diff = period_earth/P;
t2 = linspace(t0, 10*P, 1000);      % Time vector of 10 Orbit Periods
FI = eye(3);                        % I3 Identity matrix.
W = [0;0;w];                        % Vector Angular vel. of Earth rotation.
[rad/s]
Xin = Xout(end, :).';               % [pos; vel]

% Calculationg Velocity from ECEF
% ṙ = Vf = [FI] i(Vf)                 % Vf seen from ECEF
Xin(4:6, 1) = [FI] * [Xin(4:6) - cross(W, Xin(1:3))];   % [vx; vy; vz] [km/s]

ode_opt = odeset('RelTol', 3e-14, 'AbsTol', 1e-16);
[tout, Xecef] = ode113(@TBP_ECEF, t2, Xin, ode_opt, u);

%[tout, Xecef] = TBP_ECEF_ode(t2, Xin, u);
Xecef = Xecef.';

% Create figure
figure('Name', 'ECEF spacecraft trajectory',
'units','normalized','outerposition',[0 0 1 1]);

% Plot Earth
%img = imrotate(imread('BlueMarble_square.png'), 0);
surf(Re*Xe, Re*Ye, Re*Ze, 'CData', img, 'EdgeColor', 'none', 'FaceColor',
'texturemap', 'FaceAlpha','.9'); hold on; axis equal;
% surf(Re*Xe, Re*Ye, Re*Ze, 'EdgeColor', 'none', 'FaceColor', 'c'); hold on;
axis equal;
xlabel('X, ECEF [km]');
ylabel('Y, ECEF [km]');
zlabel('Z, ECEF [km]');
title('ECEF spacecraft trajectory');
% Plot ECEF axes
quiver3(0, 0, 0, 1, 0, 0, 1e4, 'k', 'LineWidth', 2);    % I-axis
quiver3(0, 0, 0, 0, 1, 0, 1e4, 'k', 'LineWidth', 2);    % J-axis
quiver3(0, 0, 0, 0, 0, 1, 1e4, 'k', 'LineWidth', 2);    % K-axis
text(1e4, 0, 0, 'X', 'FontSize', 20, 'Interpreter', 'tex', 'Color', 'k')
text(0, 1e4, 0, 'Y', 'FontSize', 20, 'Interpreter', 'tex', 'Color', 'k')
text(0, 0, 1e4, 'Z', 'FontSize', 20, 'Interpreter', 'tex', 'Color', 'k')


% Plot Trajectory
plot3(Xecef(1,:), Xecef(2,:), Xecef(3,:), 'k', 'LineWidth', 2);
plot3(Xecef(1,1), Xecef(2,1), Xecef(3,1), 'ok', 'MarkerFaceColor', 'b');
plot3(Xecef(1,end), Xecef(2,end), Xecef(3,end), 'ok', 'MarkerFaceColor', 'r');
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Week 6
% Spacecraft ECI position and velocity coordinates 1 hour after periapsis.
r_w6 = [6768.27;  870.9;2153.59];     % [km] Position vector
v_w6 = [-2.0519; -1.4150; 7.0323];  % [km s^-1] Velocity vector

% RSW Orbital Frame
h_w6 = cross(r_w6, v_w6);           % h = r x ṙ
```

```
er = r_w6 / norm(r_w6);
eh = h_w6 / norm(h_w6);
e0 = cross(eh,er);

IO = [er, e0, eh];
fprintf('\n\nThe [IO] matrix is:\n<strong>[IO]</strong>=\n');
for j = 1:3
    fprintf('%.4f  %.4f  %.4f\n', IO(j,1), IO(j,2), IO(j,3));
end
OI = transpose(IO);
fprintf('\n\nThe [OI] matrix is:\n<strong>[OI]</strong>=\n');
for j = 1:3
    fprintf('%.4f  %.4f  %.4f\n', OI(j,1), OI(j,2), OI(j,3));
end

% θ =[α; β; γ]
alpha = 30;                 % [deg] First axis - roll
beta = 20;                  % [deg] Second axis - pitch
gamma = 10;                 % [deg] First axis rot- roll

% Angles in radians
% betaRad = deg2rad(beta);
% alphaRad = deg2rad(alpha);
% gammRad = deg2rad(gamma);

% Direction Cosine Matrix [BO]
%BO2 = Euler_matrix("roll", gamma)*Euler_matrix("pitch", beta) *
Euler_matrix("roll", alpha);
BO = [  cosd(beta), sind(beta)*sind(alpha), -sind(beta)*cosd(alpha);
        sind(gamma)*sind(beta), -
sind(gamma)*cosd(beta)*sind(alpha)+cosd(gamma)*cosd(alpha),
sind(gamma)*cosd(beta)*cosd(alpha)+cosd(gamma)*sind(alpha);
        cosd(gamma)*sind(beta), -cosd(gamma)*cosd(beta)*sind(alpha)-
sind(gamma)*cosd(alpha), cosd(gamma)*cosd(beta)*cosd(alpha)-
sind(gamma)*sind(alpha)
];
fprintf(['\nDirection cosine matrix <strong>[BO}</strong> from the orbital frame
<strong>O</strong> ' ...
    'to the principal axes frame of the satellite
<strong>B</strong>:\n\n<strong>[BO]</strong>=\n']);
for j = 1:3
    fprintf('      %.4f  %.4f  %.4f\n', BO(j,1), BO(j,2), BO(j,3));
end

% B = [BO][OI]I
% B = [BI]I
% [BI] = [BOI][OI]
BI = BO*OI;
fprintf(['\nDirection cosine matrix <strong>[BI]</strong> from the ECI Frame '
...
    '<strong>I</strong> to the principal axes frame of the spacecraft
<strong>B</strong>:\n']);
for j = 1:3
    fprintf('%.4f  %.4f  %.4f\n', BI(j,1), BI(j,2), BI(j,3));
```

```
end
% Comparing [BO][OI] with BI given from the assignment.
% Result should be 0 o very near cero due to floating point in matlab
BI_assignment = [0.7908, 0.3705, 0.4872; -0.0474, -0.7565, 0.6523; 0.6102, -
0.5389, -0.5807];
% BI_compare = BI - BI_assignment;


%% Week 7

% Calculating values of Euler's Principal Axis and Angle

% [BJ] = cos(Φ) I3 - sin(Φ)[ẽ] + ( 1-cos(Φ) )êê^T
% [BJ] = [b11, b12, b13; b21, b22, b23; b31, 32, b33]
% Φ = cos^-1( [BJ]^T -1 / 2 )
% e1 = (b23-b32) / (2sinΦ)
% e2 = (b31-b13) / (2sinΦ)
% e3 = (b12-b21) / (2sinΦ)
% Φ = phi
% E = ẽ
% EE = êê^T

phi_rad = acos( (trace(BI)-1) / 2);     % [rad] Principal Angle
phi = rad2deg(phi_rad);                 % [deg] Principal Angle
den = 2*sin(phi_rad);
e1 = (BI(2,3)-BI(3,2))/den;             % e1 Axis
e2 = (BI(3,1)-BI(1,3))/den;             % e2 Axis
e3 = (BI(1,2)-BI(2,1))/den;             % e3 Axis
rot_axis = [e1; e2; e3];                % Priincipal Axis vector

fprintf("Euler's Principal Angle & Axis <strong>(ê,Φ)</strong>:\n");
fprintf(" <strong>Φ</strong> = <strong>%.4f rad</strong>\n", phi_rad);
fprintf("<strong>e1</strong> = <strong>%.4f</strong>\n" + ...
    "<strong>e2</strong> = <strong>%.4f</strong>\n" + ...
    "<strong>e3</strong> = <strong>%.4f</strong>\n", e1, e2, e3);
% Quaternions

B0 = cosd(phi/2);
Bcte = sind(phi/2);
B1 = e1*Bcte;
B2 = e2*Bcte;
B3 = e3*Bcte;
% B^2 B/J = B0^2 = B1^2 = B2^2 = B3^2;
BB = B0^2 + B1^2 + B2^2 + B3^2;
B = cosd(phi/2)^2 + sind(phi/2)^2*(e1^2+e2^2+e3^2);

fprintf("\nEuler's Parameters (Quaternions):\n");
fprintf("<strong>β</strong> = %.4f + %.4f<strong>i</strong> +" + ...
    " %.4f<strong>j</strong> + %.4f<strong>k</strong>\n", B0, B1, B2, B3);

fprintf("\nSince β0 = %.4f is <strong>nonnegative</strong> it means β
corresponds to the shortest rotation.\n", B0);

% Since B0 is nonnegative it means B corresponds to the shortest rotation.
% as phi < 180 (phi =140)
% Calculate 3-2-1 Euler Angles sequence
% Wb/j = alpha_dot K + beta_dot a + gamma_dot F
% A = {a, b, c}
% D = {d, e, f}
% [BD] = R1(roll)
% [BA] = [BD][DA] = R1(roll)R2(pitch)
% [BJ] = [BD][DA][AJ] = R1(roll)R2(pitch)R3(yaw)
% [BJ] = [BD][DA][AJ] = R1(alpha)R2(beta)R3(gamma)
%^B(Wb/j) = [BJ](0;0;gamma) + [BA](0;beta;0) + [BD](alpha;0;0)
%^B(Wb/j) = R3(gamma)R2(beta)R1(alpha)(alpha;0;0) + R3(gamma)R2(beta)(0;beta;0)
+ R3(gamma)(0;0;gamma)
% r_B = BI*r_ECI;

% Eulers angles in radians
yaw_rad = atan2(BI_assignment(1,2),BI_assignment(1,1));
pitch_rad = asin(-BI_assignment(1,3));
roll_rad = atan2(BI_assignment(2,3),BI_assignment(3,3));
% Eulers angles in degrees
yaw = rad2deg(yaw_rad);
pitch = rad2deg(pitch_rad);
roll = rad2deg(roll_rad);
eulers_angles = [yaw_rad; pitch_rad; roll_rad];

% Week 7.3 pt1: Reporting values of 3-2-1 Eule Angles sequence values
fprintf('Euler Angles = [ψ, θ, ϕ]\n' );
fprintf('<strong>ψ</strong> = <strong>%.4f rad</strong>    ', yaw_rad);
fprintf('<strong>θ</strong> = <strong>%.4f rad</strong>    ', pitch_rad);
fprintf('<strong>ϕ</strong> = <strong>%.4f rad</strong>\n\n', roll_rad);

fprintf('<strong>ψ</strong> = <strong>%.4f deg</strong>   ', yaw);
fprintf('<strong>θ</strong> = <strong>%.4f deg</strong>   ', pitch);
fprintf('<strong>ϕ</strong> = <strong>%.4f deg</strong>\n\n', roll);
%{
% 3-2-1 Euler Angles sequence values matrix [BJ] compare with original [BI]
BJ = [
    cos(pitch_rad)*cos(yaw_rad), cos(pitch_rad)*sin(yaw_rad), -sin(pitch_rad);
    sin(roll_rad)*sin(pitch_rad)*cos(yaw_rad)-cos(roll_rad)*sin(yaw_rad),
sin(roll_rad)*sin(pitch_rad)*sin(yaw_rad)+cos(roll_rad)*cos(yaw_rad),
sin(roll_rad)*cos(pitch_rad);
    cos(roll_rad)*sin(pitch_rad)*cos(yaw_rad)+sin(roll_rad)*sin(yaw_rad),
cos(roll_rad)*sin(pitch_rad)*sin(yaw_rad)-sin(roll_rad)*cos(yaw_rad),
cos(roll_rad)*cos(pitch_rad)];

fprintf('Direction cosine matrix <strong>[BJ]</strong> of a 3-2-1 Euler Angles
sequence(yaw, pitch, roll):\n\n');
fprintf('<strong>[BJ]</strong> =\n');
for j = 1:3
    fprintf('%3.4f  %3.4f  %2.4f\n', BJ(j,1), BJ(j,2), BJ(j,3));
end
%}
% Week 7.3 pt2 Reporting values of B(θ)
B_theta = [
    0,              sin(roll_rad),                  cos(roll_rad);
    0,               cos(pitch_rad)*cos(roll_rad),    -
cos(pitch_rad)*sin(roll_rad);
    cos(pitch_rad), sin(pitch_rad)*sin(roll_rad),
sin(pitch_rad)*cos(roll_rad)];
B_cte = 1/cos(pitch_rad);


fprintf('Kinematic Differential Equation <strong>θ˙= B(θ)ω</strong> of a 3-2-1
Euler Angles sequence(yaw, pitch, roll):\n\n');
fprintf('<strong>B(θ)</strong> =\n');
for j = 1:3
    if j == 2
        fprintf('(%.4f) * ', B_cte);
    else
        fprintf('           ');
    end
    fprintf('[%3.4f  %3.4f  %2.4f] (w%1d)\n', B_theta(j,1), B_theta(j,2),
B_theta(j,3),j);
end


%% Week 8

w1 = -3.092e-4;                         % [rad/s]
w2 = 6.6161e-4;                         % [rad/s]
w3 = 7.4606e-4;                         % [rad/s]

wB = [w1; w2; w3];                      % [rad/s]

X_B = [eulers_angles; wB];              %[rad, rad/s]

t_attitude = linspace(0, 3600, 361);    %t0=0[s], tend=3600[s]

ode_opt = odeset('RelTol', 3e-12, 'AbsTol', 1e-14);
% Different solutions using ode45() and ode113()
% [tout, X_Att_Dyn] = ode45(@AttitudeDynamics, t_attitude, X_B, ode_opt, I);
[tout, X_Att_Dyn] = ode113(@AttitudeDynamics, t_attitude, X_B, ode_opt, I);
X_Att_Dyn=X_Att_Dyn.';                  % [rad, rad/s]
X_Att_Dyn(1:3,:) = wrapToPi(X_Att_Dyn(1:3,:));
X_Att_Dyn_deg = X_Att_Dyn*180/pi;       % [deg, deg/s]


% Plot euler angle and angular velocities in one 2x1 window.
figure('Name', 'Attitude State of the spacecraft.',
'units','normalized','outerposition',[0 0 1 1]);
title('Euler Properties');
tiledlayout(2,1);

nexttile; hold on; grid on;
title('Angles')
xlabel('time [s]');
ylabel('θ [rad]');
plot(tout, X_Att_Dyn(1,:), 'LineWidth', 2); plot(tout, X_Att_Dyn(2,:),
'LineWidth', 2); plot(tout, X_Att_Dyn(3,:), 'LineWidth', 2);
legend('Yaw','Pitch', 'Roll');
```

```
hold off;

nexttile; hold on; grid on;
title('Angular velocities');
xlabel('time [s]');
ylabel('w [rad/s]');
plot(tout, X_Att_Dyn(4,:), 'LineWidth', 2); plot(tout, X_Att_Dyn(5,:),
'LineWidth', 2); plot(tout, X_Att_Dyn(6,:), 'LineWidth', 2);
legend('w1','w2', 'w3');
hold off;

% Plot euler angle and angular velocities in one 2x3 window.
figure('name', 'Attitude State of the Spacecraft.',
'units','normalized','outerposition',[0 0 1 1]);
tiledlayout(2,3);

%ψ, θ, ϕ
nexttile
plot(tout, X_Att_Dyn(1,:), 'LineWidth', 2, 'Color', "#0072BD");
legend('Yaw');
ylabel('ψ [rad]');
xlabel('time [s]');
grid on;

nexttile
plot(tout, X_Att_Dyn(2,:), 'LineWidth', 2, 'Color', "#D95319");
legend('Pitch');
ylabel('θ [rad]');
xlabel('time [s]');
grid on;

nexttile
plot(tout, X_Att_Dyn(3,:), 'LineWidth', 2, 'Color', "#EDB120");
legend('Roll');
ylabel('ϕ [rad]');
xlabel('time [s]');
grid on;

nexttile
plot(tout, X_Att_Dyn(4,:), 'LineWidth', 2, 'Color', "#0072BD");
legend('w1');
ylabel('w1 [rad/s]');
xlabel('time [s]');
grid on;

nexttile
plot(tout, X_Att_Dyn(5,:), 'LineWidth', 2, 'Color', "#D95319");
legend('w2');
ylabel('w2 [rad/s]');
xlabel('time [s]');
grid on;

nexttile
plot(tout, X_Att_Dyn(6,:), 'LineWidth', 2, 'Color', "#EDB120");
legend('w3');
```

```
ylabel('w3 [rad/s]');
xlabel('time [s]');
grid on;

fprintf(['\nBecause of the 3-2-1 sequence naturality of the rotation, ' ...
    'Pitch angle (θ) almost reaches singularity (+-π).\n' ...
    '<strong>θ ≈ -85.8 deg | θ ≈ 1.49 rad</strong> at <strong>t = 1100
s</strong>\n']);

% Angular Momentum
H = [Ixx*X_Att_Dyn(4,:); Iyy*X_Att_Dyn(5,:); Izz*X_Att_Dyn(6,:)];
H_radius = sqrt(H(1,:).^2 + H(2,:).^2 + H(3,:).^2);

% Rotational Kinetic Energy

for n=1:361
    T(n) = (1/2)*(X_Att_Dyn(4,n).'*H(1,n) + X_Att_Dyn(5,n).'*H(2,n) +
X_Att_Dyn(6,n).'*H(3,n));
end


% Plot euler angle and angular velocities in one 2x3 window.
figure('name', 'Angular momentum & Rotational Kinetic Energy.',
'units','normalized','outerposition',[0 0 1 1]);
tiledlayout(2,1);

nexttile
plot(tout, H_radius, 'LineWidth', 2);
legend('Angular Momentum');
ylabel('H [kg m^2 / s]');
xlabel('time [s]');
grid on;

nexttile
plot(tout, T, 'LineWidth', 2,'Color', "#D95319");
legend('Rotational Kinetic Energy');
ylabel('T [kg m^2 / s^2]');
xlabel('time [s]');
grid on;
```