THEORY DEVELOPMENT IN SUPPORT OF MATLAB SATELLITE ORBITAL MECHANICS CODE

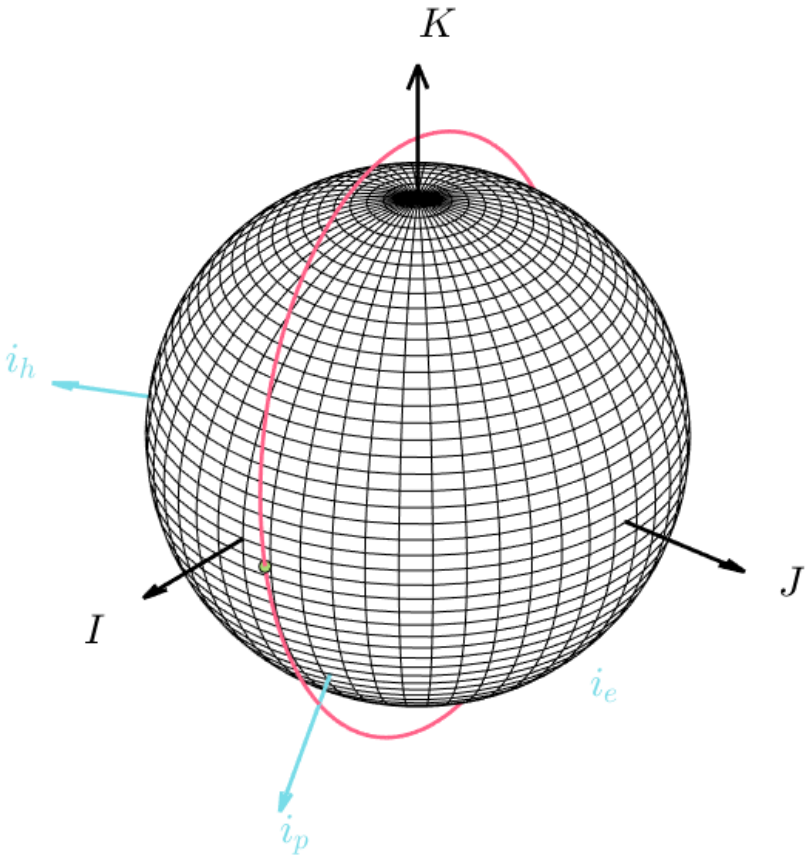

Figure I-2 Spacecraft ECI orbital trajectory

Leif Tinwell
Department of Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey

November 24, 2023

TABLE OF CONTENTS

LIST OF FIGURES

DATA

Earth's gravitational parameter $\mu = 398600.4418\ km^3\ s^{-2}$;
Earth's radius $R_\oplus = 6378.137\ km$;
Earth's angular velocity $\omega_\oplus = 7.2921 \times 10^{-5}\ rad\ s^{-1}$;
$I_{xx} = 2500\ kg\ m^2$;
$I_{yy} = 5000\ kg\ m^2$;
$I_{zz} = 6500\ kg\ m^2$;

INTRODUCTION

This report is written in support of the MATLAB code developed for satellite orbital mechanics. In essence, this report should lead the reader through the implications of the two-body problem, demonstrate how they affect a satellite's motion in space, and provide clear derivation of the underlying physics. It should also discuss the various methods used to determine and predict a satellite's orientation in space, considering a torque-free context, all the while studying their relative merits. Ultimately, the objective of this paper is to explore the fundamentals of space dynamics, specifically, the principles of orbital and attitude mechanics. These hold great significance to spacecraft deployment and navigation in space.

I. TWO BODY-PROBLEM

In order to derive simple analytical results for the motion of a spacecraft about the Earth, it is assumed that any external forces from additional celestial bodies are disregarded - no appreciable force is exerted on the spacecraft from a third body. Likewise, any additional effects of orbit perturbations (*i.e.*, solar wind pressure, atmospheric drag, Earth's oblateness *etc*) are neglected, further simplifying the model. Under these assumptions, it can be stated that the motion of a satellite is purely due to the gravitational interaction with Earth.

The following orbital parameters were considered:

$$\begin{aligned} a &= 7151.6 \text{ km} \\ e &= 0.0008 \\ i &= 98.39 \text{ deg} \\ \Omega &= 10.0 \text{ deg} \\ \omega &= 233.0 \text{ deg} \\ M_0 &= 127.0 \text{ deg} \end{aligned}$$

I.1 Solve Kepler's Equation using Newton's method and finding the initial True Anomaly

Kepler's Equation relates the *Mean Anomaly* M of a satellite to its *Eccentric Anomaly* E and *orbital eccentricity e* as,

$$M = E - e\,sinE. \quad \textbf{Eq. 1}$$

Typically, Eq. 1 is solved numerically using Newton's method by finding the root of

$$f(E) = E - e\,sin(E) - M, \quad \textbf{Eq. 2}$$

in which E is computed iteratively to reduce Eq. 2 below a design tolerance (*i.e.*, *Tol*) using

$$E_{n+1} = E_n - \frac{f(E_n)}{f'(E_n)}, \quad \textbf{Eq. 3}$$

where, $f'(E_n)$ is the first derivative of $f(E_n)$,

$$E_{n+1} = E_n - \frac{E_n - esin(E_n) - M}{1 - ecos(E_n)}. \quad \textbf{Eq. 4}$$

The iterative regression analysis can be initialised by setting $E_n = M_0$, then performed until accuracy is satisfactory (*i.e.*, $|f(E_n)| < Tol$). For a convergence tolerance of 10^{-10}, after two iterations, the algorithm delivers an eccentric anomaly of

$$E = 2.2172\ rads$$

Subsequently, the true anomaly θ of the spacecraft can be determined given the eccentric anomaly E and the orbit eccentricity *e* via

$$\tan\left(\frac{\theta}{2}\right) = \sqrt{\left(\frac{1+e}{1-e}\right)}\tan\left(\frac{E}{2}\right). \quad \textbf{Eq. 5}$$

Solving for θ,

$$\theta = 2\tan^{-1}\left(\sqrt{\left(\frac{1+e}{1-e}\right)}\tan\left(\frac{E}{2}\right)\right), \quad \textbf{Eq. 6}$$

$$\theta = 2.2178\ rads$$

It can then be determined that the satellite is near apoapsis since $\frac{\pi}{2} \leq \theta \leq \frac{3\pi}{2}$. It's important to note that the MATLAB 'atan2' function was used to avoid quadrant issues and the result modulated to [0, 2π].

I.2 Obit propagation via analytical solution of TBP

The *Classical Orbital Elements* (**COE**) uniquely determine an orbit as

$$\boldsymbol{COE} = \begin{bmatrix} a \\ e \\ i \\ \Omega \\ \omega \\ \theta \end{bmatrix}$$

where *a* is the semi-major axis, *e* the orbit eccentricity, *i* the orbit inclination, Ω the right ascension of the ascending node, ω the argument of periapsis and θ the true anomaly. Furthermore, a satellites orbital period P can be related (under TBP assumptions) to the semi-major axis *a* via,

$$P = 2\pi\frac{1}{n}, \quad \textbf{Eq. 7}$$

in which μ, Earth's gravitational parameter, and *n* the mean motion of the satellite

$$n = \sqrt{\frac{\mu}{a^3}}. \quad \textbf{Eq. 8}$$

Applying relevant values into Eq. 7,

$$P = 6018.3262\ s$$

$$\approx 1\ hr\ 40\ min\ 18\ s.$$

The orbital parameters of a satellite, specifically the true, eccentric and mean anomalies, can be projected in time by first expressing the mean anomaly as a function of time,

$$M(t) = M_0 + n(t - t_0). \quad \textbf{Eq. 9}$$

The derivation of E(t) is then analogous to the procedures described in section I.1, to the exception of using Eq. 9 instead. Likewise, θ(t) can be determined using E(t). The resulting plots of eccentric, true and mean anomalies against time are presented in Figure I-1.

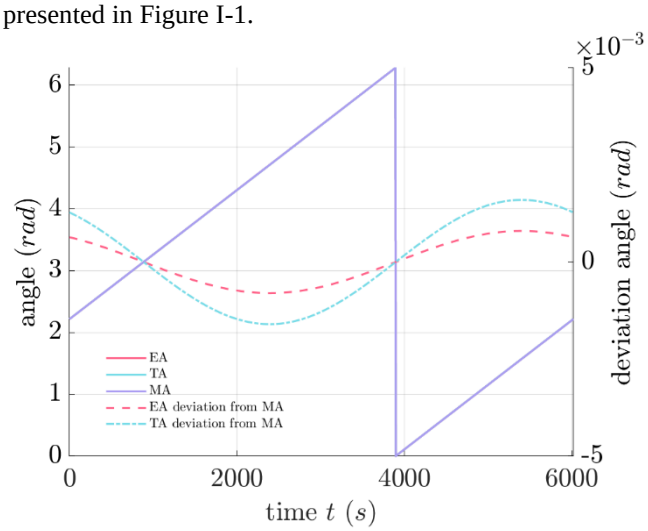

Figure I-1 Mean, Eccentric and True Anomalies vs. time

Here, MA, EA and TA are abbreviations of Mean, Eccentric and True anomalies respectively. Considering the quasi-circular orbit, the anomalies are expected to align. Deviations from the Mean Anomaly are apparent and plotted as dashed lines to aid analysis.

As an initial step to determining the position and velocity of a spacecraft, mapping the orbital parameters to the ECI frame is achieved with a 3-1-3 set of rotations about the orbital parameters ω, i and Ω, defined as follows,

$$\begin{cases} \boldsymbol{R}_3(\omega) = \begin{bmatrix} \cos(\omega) & \sin(\omega) & 0 \\ -\sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \boldsymbol{R}_1(i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & \sin(i) \\ 0 & -\sin(i) & \cos(i) \end{bmatrix} \\ \boldsymbol{R}_3(\Omega) = \begin{bmatrix} \cos(\Omega) & \sin(\Omega) & 0 \\ -\sin(\Omega) & \cos(\Omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{cases} \quad \textit{Eq. 10}$$

which, when concatenated into the Direction Cosine Matrix (DCM),

$$\boldsymbol{R}_I^P = \boldsymbol{R}_3(\omega)\boldsymbol{R}_1(i)\boldsymbol{R}_3(\Omega), \quad \textit{Eq. 11}$$

transforms the *ECI* frame to the *Perifocal*. Given that $\boldsymbol{R}_P^I = (\boldsymbol{R}_I^P)^T$, the state vector $\boldsymbol{X}_{ECI}$ of the spacecraft can be defined accordingly as,

$$\boldsymbol{X}_{ECI} = \begin{bmatrix} \boldsymbol{r}_I \\ \boldsymbol{v}_I \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \quad \textit{Eq. 12}$$

where $\boldsymbol{r}_I$ and $\boldsymbol{v}_I$ are the spacecraft's position and velocity (*w.r.t ECI* frame) components respectively. The position vector $\boldsymbol{r}$ can be determined as follows,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{R}_P^I \cdot \boldsymbol{r}_P, \quad \textit{Eq. 13}$$

where $\boldsymbol{r}_P$ the position components in the perifocal frame,

$$\boldsymbol{r}_P = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{bmatrix}, \quad \textit{Eq. 14}$$

in which r, the spacecrafts distance from Earth centre, is calculated as,

$$r = \frac{a(1-e^2)}{(1+e\cos\theta)}, \quad \textit{Eq. 15}$$

with a, e, and θ as previously defined. Similarly, the velocity vector $\boldsymbol{v}$ can be calculated via,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{R}_P^I \cdot \boldsymbol{v}_P, \quad \textit{Eq. 16}$$

where $\boldsymbol{v}_P$ is the velocity components in the perifocal frame,

$$\boldsymbol{v}_P = \begin{bmatrix} -\frac{\mu}{h}\sin\theta \\ \frac{\mu}{h}(\cos\theta + e) \\ 0 \end{bmatrix}, \quad \textit{Eq. 17}$$

in which h is the specific angular momentum,

$$h = \sqrt{\mu a(1-e^2)}. \quad \textit{Eq. 18}$$

Finally, repeating this analytical process at each time step allows for the computation of the full *ECI* orbit of the spacecraft. Through this procedure, it then follows that the initial and final values of the spacecraft's state $\boldsymbol{X}_{ECI}$ are

$$\boldsymbol{X}_{ini} = \begin{bmatrix} 7046.1371 \\ 1241.0704 \\ 9.0389 \\ 0.1844 \\ -1.0731 \\ 7.3824 \end{bmatrix} \qquad \boldsymbol{X}_{fin} = \begin{bmatrix} 7046.1371 \\ 1241.0704 \\ 9.0389 \\ 0.1844 \\ -1.0731 \\ 7.3824 \end{bmatrix}$$

Indeed, these states are identical: Since the satellite's state is considered in an unperturbed orbit in an inertial frame, it is

dealt no forces nor acceleration. As such the values are expected to align. The spacecraft's resulting *ECI* trajectory is plotted in Figure I-2. The satellite's trajectory is traced in red around a white sphere, representing Earth. Here, i_e, i_h and i_p shown in blue describe the axes of the perifocal frame.

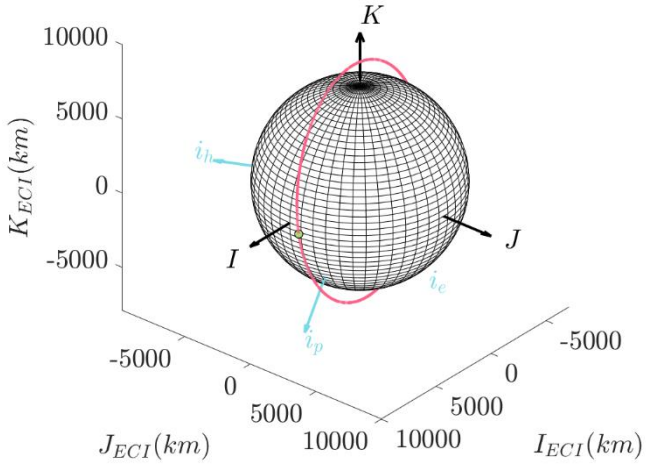

Figure I-2 Spacecraft ECI orbital trajectory

I.3 Numerical integration in the ECI frame

Although the analytical solution excels in precision, it is computationally heavy, since it deals with DCM rotations every iteration. This is often overcome by means of a numerical approach. Under the TBP assumptions, the spacecraft motion can be described as

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r}\,. \qquad \textit{Eq. 19}$$

To propagate the state of a spacecraft in time, a system of first-order ordinary differential equations may be found, then integrated. The satellite's kinematics can be described with the following system derived from Eq. 19,

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v}_x \equiv \ddot{x} = -\dfrac{\mu}{r^3}x \\ \dot{v}_y \equiv \ddot{y} = -\dfrac{\mu}{r^3}y \\ \dot{v}_z \equiv \ddot{z} = -\dfrac{\mu}{r^3}z \end{cases}. \qquad \textit{Eq. 20}$$

Only the six initial conditions $\boldsymbol{r_o}$ and $\boldsymbol{v_o}$ are required to integrate the system. Hence, knowledge of the state of the satellite at a single point in time allows for a complete description of its orbit. The integration was done with `ode113`[1] which excels in orbital dynamics.

[1] **`ode113`** was chosen over **`ode45`** for the integration, as it is known for its accuracy in orbital mechanics. For the same tolerances, **`ode45`'s** error was 2-3 orders of magnitude greater than that of **`ode113`**.

However, unlike an analytical method described in section I.2, which derives an *exact* solution, a numerical approach inherently introduces some degree of inaccuracy. This is shown in Figure I-3, in which the deviation of the numerical integration from its analytical counterpart is plotted over time.

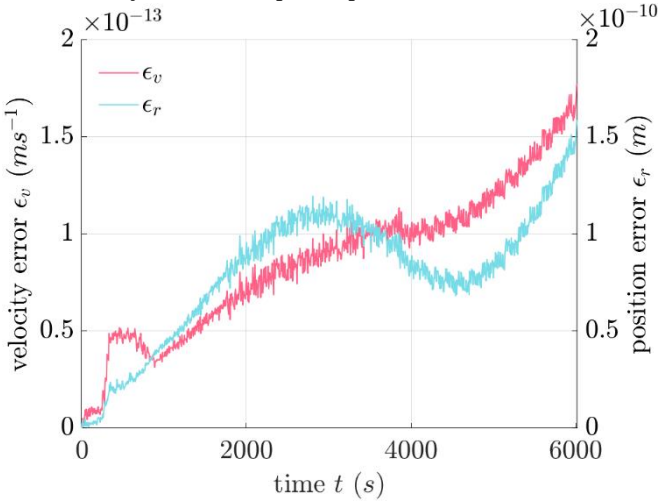

Figure I-3 Position and Velocity Error over time

The integration error in this case is on the order 10^{-10}. When contrasted with the spacecraft's position and velocity values, the numerical integration errors are inconsequential, thereby validating the numerical approach.

Despite this error, the satellite's specific energy ζ should remain relatively constant since it is gravitationally bound to Earth in a quasi-circular orbit. The orbit's specific energy ζ can be defined as the sum of the kinetic and potential energies as

$$\zeta_2 = \frac{v^2}{2} - \frac{\mu}{r}. \qquad \textit{Eq. 21}$$

In a closed Keplerian orbit, the same holds true, regardless of the fact that the satellite's altitude varies significantly. Simply put, there is a cyclic exchange of kinetic and potential energy, implied by conservation of energy. As such, the specific energy of a spacecraft is often approximated as

$$\zeta_1 = -\frac{\mu}{2a}. \qquad \textit{Eq. 22}$$

This relationship holds true regardless of the eccentricity. The values of ζ are plotted over time in Figure I-4, for both Eq. 21 and Eq. 22.

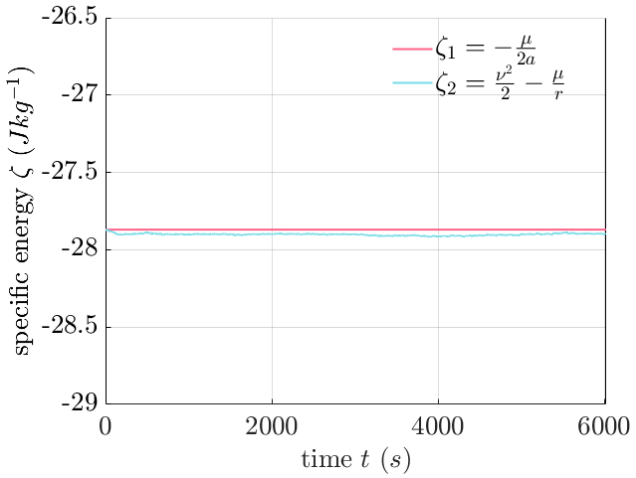

Figure I-4 Specific Energy vs. time

Figure I-4 shows that the specific energy, computed from Eq. 21, remains relatively constant, with minor variation on the order of 10^{-13} J/kg. This supports the notion that it is indeed an integral of motion, thereby validating the Eq. 22 approximation.

I.4 Derive and integrate the equations of motion in the ECEF frame

Up until now, the satellite's motion has been described in both the *Perifocal* and *Earth-Centred-Inertial* reference frames. These are however inadequate for faithfully tracking a satellite's location with respect to a ground reference. In fact, the *Earth-Centred-Earth-Fixed* (*ECEF*) frame is commonly used, which accounts for Earth's rotation, ultimately simplifying ground-based navigation and positioning of a satellite.

To do so, the motion of the satellite must be mapped to the *ECEF* frame. This is achieved by applying transport theorem to Eq. 19, which states

$$\dot{\boldsymbol{r}}_F = \dot{\boldsymbol{r}}_I - \boldsymbol{\omega}_I^F \times \boldsymbol{r}\,, \qquad \textit{Eq. 23}$$

where $\dot{\boldsymbol{r}}_F$ and $\dot{\boldsymbol{r}}_I$ refer to the velocities in the fixed and inertial frames respectively; $\boldsymbol{\omega}_I^F$ is the angular velocity of the *ECEF* frame as seen from the *ECI* frame, which is assumed constant and aligned to the third axis of rotation,

$$\boldsymbol{\omega}_I^F = \begin{bmatrix} 0 \\ 0 \\ \omega_\oplus \end{bmatrix},$$

$\omega_\oplus$ being Earth's angular velocity. To derive the acceleration $\ddot{\boldsymbol{r}}_F$ of the satellite in the *ECEF* frame, the second time derivative of Eq. 23 is taken as,

$$\ddot{\boldsymbol{r}}_F = \ddot{\boldsymbol{r}}_I - \dot{\boldsymbol{\omega}}_I^F \times \boldsymbol{r} - 2\boldsymbol{\omega}_I^F \times \boldsymbol{v}_F - \boldsymbol{\omega}_I^F \times \boldsymbol{\omega}_I^F \times \boldsymbol{r}\,. \qquad \textit{Eq. 24}$$

Here, $\omega_\oplus$ is constant, hence $(\dot{\boldsymbol{\omega}}_I^F = 0)$

$$\ddot{\boldsymbol{r}}_F = \ddot{\boldsymbol{r}}_I - 2\boldsymbol{\omega}_I^F \times \boldsymbol{v}_F - \boldsymbol{\omega}_I^F \times \boldsymbol{\omega}_I^F \times \boldsymbol{r}\,. \qquad \textit{Eq. 25}$$

Expanding the vectors and performing the cross products,

$$\ddot{\boldsymbol{r}}_F = -\frac{\mu}{r^3}\boldsymbol{r} - \begin{bmatrix} -2\omega_\oplus {v_y}^F \\ 2\omega_\oplus {v_x}^F \\ 0 \end{bmatrix} - \begin{bmatrix} -{\omega_\oplus}^2 x \\ -{\omega_\oplus}^2 y \\ 0 \end{bmatrix}. \qquad \textit{Eq. 26}$$

The satellite's motion as seen in the ECEF frame can therefore be fully defined with the following system of equations,

$$\begin{cases} \dot{x} = {v_x}^F \\ \dot{y} = {v_y}^F \\ \dot{z} = {v_z}^F \\ \dot{v}_x = -\frac{\mu}{r^3}x + 2\omega_\oplus {v_y}^F + {\omega_\oplus}^2 x\,. \\ \dot{v}_y = -\frac{\mu}{r^3}y - 2\omega_\oplus {v_x}^F + {\omega_\oplus}^2 y \\ \dot{v}_z = -\frac{\mu}{r^3}z \end{cases} \qquad \textit{Eq. 27}$$

The ECI initial conditions of the satellite are converted to ECEF frame to initialise the integration. Disregarding the effects of polar motion and procession/nutation, the conversion may be done by means of a rotation about the third axis $\widehat{\boldsymbol{Z}}$,

$$\boldsymbol{R}_3(\Theta) = \begin{bmatrix} \cos(\Theta) & \sin(\Theta) & 0 \\ -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textit{Eq. 28}$$

where Θ is the *Greenwich Mean Sidereal Time*, derived as,

$$\Theta = \omega_\oplus (t - t_0) \qquad \textit{Eq. 30}$$

Considering that at $t = t_0 = 0\ s$, both frames are aligned, the rotation matrix $\boldsymbol{R}_3(\Theta)$ reduces to a 3x3 identity matrix $\boldsymbol{I}_3$. Indeed, at t_0, $\boldsymbol{r}_F$ coincides with $\boldsymbol{r}_I$

$$\boldsymbol{r}_F(t_0) = \boldsymbol{I}_3 * \boldsymbol{r}_I \qquad \textit{Eq. 31}$$

The velocity of the spacecraft $\boldsymbol{v}_F$ at t_0 however, must comply with Eq. 23 and account for Earth's rotation,

$$\boldsymbol{v}_F(t_0) = \boldsymbol{I}_3 * (\boldsymbol{v}_I - \boldsymbol{\omega}_I^F \times \boldsymbol{r}) \qquad \textit{Eq. 32}$$

Finally, Eq. 27 is integrated to determine the satellite's orbit as seen in the *ECEF* frame. The resulting 10 orbit trajectory is plotted in Figure I-5.

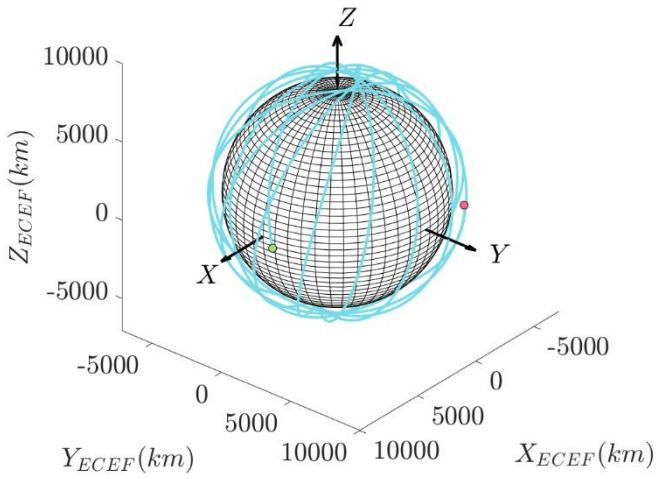

Figure I-5 Ten Orbit Satellite ECEF Trajectory

The initial and final positions of the satellite in Figure I-5 are marked with green and red dots respectively. The initial and final conditions $\boldsymbol{X^F}_{ini}$ and $\boldsymbol{X^F}_{fin}$ as a result of the integration are reported

$$\boldsymbol{X}_{ini} = \begin{bmatrix} 7046.1371 \\ 1241.0704 \\ 9.0389 \\ 0.2749 \\ -1.5869 \\ 7.3824 \end{bmatrix} \qquad \boldsymbol{X}_{fin} = \begin{bmatrix} -3418.1482 \\ 6285.2658 \\ 9.0389 \\ 1.4170 \\ 0.7654 \\ 7.3824 \end{bmatrix}$$

These values align with expectations, since the satellite's trajectory is represented in a rotating frame. If the orbital period were to be a multiple of Earth's rotating period, the satellite's initial conditions should in theory align with the final conditions.

II. ATTITUDE REPRESENTATIONS AND TORQUE-FREE MOTION

The fundamental complement to spacecraft orbital mechanics is attitude dynamics. Where one represents position and velocity in space, the other defines orientation. It shall be shown in this chapter, that much like orbit determination, various methods may be used for attitude modelling, with certain approaches proving more advantageous than others.

One hour before passing through its periapsis, the spacecraft ECI position and velocity coordinates:

$$\boldsymbol{r} = \begin{bmatrix} 6768.27 \\ 870.90 \\ 2153.59 \end{bmatrix} \text{(km)} \quad \boldsymbol{v} = \begin{bmatrix} -2.0519 \\ -1.4150 \\ 7.0323 \end{bmatrix} \text{(km s}^{-1}\text{)}$$

II.1 Direction Cosine Matrices

For a planet orbiting satellite, it is often most convenient to define the orbital reference frame *RSW* in which: $\hat{\boldsymbol{R}}$ points towards the spacecraft away from nadir, $\hat{\boldsymbol{W}}$ orthogonal to the orbital plane and $\hat{\boldsymbol{S}}$ completes the triad. Given the state of the satellite in the *ECI* frame, its state can be determined in the *RSW* frame using a DCM, in a similar fashion as described in section I.2. The DCM that maps the motion of the spacecraft to the orbital frame is unique

$$\boldsymbol{R}_I^O = \begin{bmatrix} \hat{O}_1 \cdot \hat{\boldsymbol{I}} & \hat{O}_1 \cdot \hat{\boldsymbol{J}} & \hat{O}_1 \cdot \hat{\boldsymbol{K}} \\ \hat{O}_2 \cdot \hat{\boldsymbol{I}} & \hat{O}_2 \cdot \hat{\boldsymbol{J}} & \hat{O}_2 \cdot \hat{\boldsymbol{K}} \\ \hat{O}_3 \cdot \hat{\boldsymbol{I}} & \hat{O}_3 \cdot \hat{\boldsymbol{J}} & \hat{O}_3 \cdot \hat{\boldsymbol{K}} \end{bmatrix}. \qquad \textit{Eq. 33}$$

And thus, the DCM that converts the state of the spacecraft from orbital frame to *ECI* frame may be determined, given that $\boldsymbol{R}_O^I = (\boldsymbol{R}_I^O)^T$,

$$\boldsymbol{R}_O^I = \begin{bmatrix} \hat{O}_1 \cdot \hat{\boldsymbol{I}} & \hat{O}_2 \cdot \hat{\boldsymbol{I}} & \hat{O}_3 \cdot \hat{\boldsymbol{I}} \\ \hat{O}_1 \cdot \hat{\boldsymbol{J}} & \hat{O}_2 \cdot \hat{\boldsymbol{J}} & \hat{O}_3 \cdot \hat{\boldsymbol{J}} \\ \hat{O}_1 \cdot \hat{\boldsymbol{K}} & \hat{O}_2 \cdot \hat{\boldsymbol{K}} & \hat{O}_3 \cdot \hat{\boldsymbol{K}} \end{bmatrix}. \qquad \textit{Eq. 34}$$

The inertial components of the spacecraft are determined as

$$\begin{cases} \hat{O}_1 = \hat{\boldsymbol{e}}_r = \dfrac{\boldsymbol{r}}{r} \\ \hat{O}_3 = \hat{\boldsymbol{e}}_h = \dfrac{\boldsymbol{h}}{h} \\ \hat{O}_2 = \hat{\boldsymbol{e}}_\theta = \hat{\boldsymbol{e}}_h \times \hat{\boldsymbol{e}}_r \end{cases}. \qquad \textit{Eq. 35}$$

The inertial components of $\hat{O}_1$ can then be found

$$\hat{O}_1 = \begin{bmatrix} 0.9458 \\ 0.1217 \\ 0.3010 \end{bmatrix}.$$

The inertial components of the specific angular momentum $\hat{O}_3$ can also be calculated, knowing that $\boldsymbol{h} = \boldsymbol{r} \times \boldsymbol{v}$,

$$\hat{O}_3 = \begin{bmatrix} 0.1718 \\ -0.9743 \\ -0.1459 \end{bmatrix}.$$

Finally, the triad is complete by computing the inertial components of $\hat{O}_2$,

$$\hat{O}_2 = \begin{bmatrix} -0.2755 \\ -0.1897 \\ 0.9424 \end{bmatrix}.$$

$\boldsymbol{R}_O^I$ is now fully determined,

$$\boldsymbol{R}_O^I = \begin{bmatrix} 0.9458 & 0.1718 & -0.2755 \\ 0.1217 & -0.9743 & -0.1897 \\ 0.3010 & -0.1459 & 0.9424 \end{bmatrix}.$$

The DCM $\boldsymbol{R}_O^B$ from the orbital frame to the principal axes frame of the satellite can be represented as a 1-2-1 sequence rotation of Euler angles $\boldsymbol{\theta} = [\alpha,\ \beta,\ \gamma]^T$ as

$$\begin{cases} \boldsymbol{R}_1(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \\ \boldsymbol{R}_2(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \\ \boldsymbol{R}_1(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \end{cases} \qquad \textit{Eq. 36}$$

$$\boldsymbol{R}_O^B = \boldsymbol{R}_1(\gamma)\boldsymbol{R}_2(\beta)\boldsymbol{R}_1(\alpha). \quad \textit{Eq. 37}$$

Concatenating Eq. 33 and Eq. 37, the DCM from the ECI frame to the body frame, by definition, is

$$\boldsymbol{R}_I^B = \boldsymbol{R}_O^B * \boldsymbol{R}_I^O \quad \textit{Eq. 38}$$

Substituting in the Euler angles into Eq. 37 and performing the DCM product in Eq. 38,

$$\boldsymbol{R}_I^B = \begin{bmatrix} 0.7908 & 0.3705 & 0.4872 \\ -0.0474 & -0.7565 & 0.6523 \\ 0.6102 & -0.5389 & -0.5807 \end{bmatrix}$$

The orientation of the satellite's principal axes (*i.e.*, attitude of the spacecraft) with respect to the ECI frame has therefore been fully defined.

II.2 Attitude representations

In the previous section II.1, successive rotations about the inertial and orbital frame axes were performed to determine its orientation with respect to the ECI frame. Any motion however, can be represented as a single rotation about a fixed axis, known as *Euler's Principal Axis*. Considering a DCM, one of the eigenvalues must be unity. The implication is that there must exist an eigenaxis $\hat{\boldsymbol{e}}$ such that

$$\boldsymbol{R}\hat{\boldsymbol{e}} = \hat{\boldsymbol{e}}\,. \quad \textit{Eq. 39}$$

A rotation angle about the Euler Axis $\hat{\boldsymbol{e}}$ is denoted as Φ and referred to as *Euler's Angle*. A rotation $\boldsymbol{R}$ can be expressed in terms of the Euler Axis and Euler Angle as

$$\boldsymbol{R} = \cos\Phi\,\boldsymbol{I_3} + (1-\cos\Phi)\hat{\boldsymbol{e}}\hat{\boldsymbol{e}}^T - \sin\Phi\,\hat{\boldsymbol{e}}^{\times}, \quad \textit{Eq. 40}$$

where $\hat{\boldsymbol{e}}^{\times}$ is a matrix cross product

$$\hat{\boldsymbol{e}}^{\times} = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}, \quad \textit{Eq. 41}$$

and

$$\hat{\boldsymbol{e}}\hat{\boldsymbol{e}}^T = \begin{bmatrix} {e_1}^2 & e_1e_2 & e_1e_3 \\ e_1e_2 & {e_2}^2 & e_2e_3 \\ e_1e_3 & e_2e_3 & {e_3}^2 \end{bmatrix}. \quad \textit{Eq. 42}$$

Euler's Angle Φ is calculated as

$$\Phi = \cos^{-1}\left(\frac{\boldsymbol{R^{Tr}} - 1}{2}\right) \quad \textit{Eq. 43}$$

The matrix $\boldsymbol{R}$ is a DCM with elements R_{ij}. The elements of the eigenvector of rotation in terms of R_{ij} are given as

$$\begin{cases} e_1 = \dfrac{R_{23} - R_{32}}{2\sin\Phi} \\ e_2 = \dfrac{R_{31} - R_{13}}{2\sin\Phi} \\ e_3 = \dfrac{R_{12} - R_{21}}{2\sin\Phi} \end{cases} \quad \textit{Eq. 44}$$

Hence, $\boldsymbol{R}_I^B$ derived in section II.1 can in fact be fully described as a single rotation Φ about the axis $\hat{\boldsymbol{e}}$,

$$\Phi = 1.7645 \text{ rads}, \qquad \hat{\boldsymbol{e}} = \begin{pmatrix} 0.6069 \\ 0.0627 \\ 0.2129 \end{pmatrix}.$$

Satellite attitude modelling often employs quaternions for their computational efficiency and absence of singularities. Overall, they present a favourable balance between DCMs and Euler Axis rotations, albeit slightly challenging to conceptualise. A quaternion can be defined in terms of the principal rotation components as

$$\begin{cases} \boldsymbol{q} = \hat{\boldsymbol{e}}\sin\dfrac{\Phi}{2} \\ q_4 = \cos\dfrac{\Phi}{2} \end{cases}. \quad \textit{Eq. 45}$$

Substituting in the previously calculated $(\hat{\boldsymbol{e}}, \Phi)$, the equivalent quaternion $\overline{\boldsymbol{q}}_{\mathrm{I}}^B$ can be derived

$$\overline{\boldsymbol{q}}_{\mathrm{I}}^B = \begin{pmatrix} 0.4687 \\ 0.0484 \\ 0.1644 \\ 0.6354 \end{pmatrix}.$$

If $\hat{\boldsymbol{e}}$ is taken to be a unit vector, then the quaternion $\overline{\boldsymbol{q}}_{\mathrm{I}}^B$ must also be unity (*i.e.*, norm of $\overline{\boldsymbol{q}}_{\mathrm{I}}^B = 1$). In this case though, the norm will be

$$\|\overline{\boldsymbol{q}}_{\mathrm{I}}^B\| \approx 0.8079,$$

it follows that, if $\overline{\boldsymbol{q}}_{\mathrm{I}}^B$ is not unity, therefore, neither is $\hat{\boldsymbol{e}}$. Attitude modelling however, mandates unit quaternions to correctly represent a rotation. Nonetheless, a quaternion can be scaled (normalised) by dividing its components by its norm. As a result, the normalised unit quaternion would be

$$\overline{\boldsymbol{q}}_{\mathrm{I}}^B = \begin{pmatrix} 0.5801 \\ 0.0599 \\ 0.2035 \\ 0.7865 \end{pmatrix},$$

validated by $\|\overline{\boldsymbol{q}}_{\mathrm{I}}^B\| = 1$.

The rotation matrix $\boldsymbol{R}$ can then be reconstructed from the quaternion $\boldsymbol{q}$ as

$$\boldsymbol{R} = ({q_4}^2 - \boldsymbol{q}^T)\boldsymbol{I_3} + \boldsymbol{2qq}^T - 2q_4\boldsymbol{q}^{\times}, \quad \textit{Eq. 46}$$

which, developed, gives

$$\boldsymbol{R} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_3) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}. \quad \textit{Eq. 47}$$

Furthermore, this same DCM can also be performed as a 3-2-1 sequence about the Euler Angles $\boldsymbol{\Psi} = [\psi,\ \theta,\ \phi]^T$. These angles are calculated from

$$\begin{cases} Yaw, \psi = atan2(R_{12}, R_{11}) \\ Pitch, \theta = \sin^{-1}(-R_{13}) \\ Roll, \phi = atan2(R_{23}, R_{33}) \end{cases}, \quad \textit{Eq. 48}$$

as

$$\begin{bmatrix}\psi\\ \theta\\ \phi\end{bmatrix} = \begin{bmatrix}0.4381\\ -0.5089\\ 2.2982\end{bmatrix} \text{rads.}$$

Finally, the kinematic equations of such a rotating rigid body can be determined from the following relationship[2]

$$\dot{\boldsymbol{\Psi}} = \frac{1}{\cos\theta}\begin{bmatrix}0 & \sin\phi & \cos\phi\\ 0 & \cos\theta\cos\phi & -\cos\theta\sin\phi\\ \cos\theta & \sin\theta\sin\phi & \sin\theta\sin\phi\end{bmatrix}\boldsymbol{\omega}_I^B, \qquad \textit{Eq. 49}$$

where $\dot{\boldsymbol{\Psi}}$ is the time rate change of Euler Angles, $\boldsymbol{\omega}_I^B$ is the angular velocity vector of the spacecraft in body frame coordinates. It is important to note that this equation is singular when $\theta = \frac{\pi}{2}, \frac{3\pi}{2}$. As such, practically speaking, the kinematic equations are typically derived from the quaternions themselves to avoid any singularities.

II.3 Kinematics and Euler's equations

In order to define the full attitude model of the spacecraft, both attitude dynamics and kinematics are required – the latter of which has been derived in section II.2 (Eq. 49). The dynamic equations – often referred to as Euler's equations – are derived as follows. A torque $\boldsymbol{T}$ acting on a body about its centre of mass equates to the time rate change of its angular momentum as

$$\boldsymbol{T} = \dot{\boldsymbol{h}}_B + \boldsymbol{\omega}_I^B \times \boldsymbol{h}\,, \qquad \textit{Eq. 50}$$

where $\boldsymbol{h}$ is the spacecraft's angular momentum and $\dot{\boldsymbol{h}}_{\boldsymbol{B}}$ its time derivative. The total angular momentum of the satellite is given by

$$\boldsymbol{h} = \boldsymbol{I}\boldsymbol{\omega}. \qquad \textit{Eq. 51}$$

The dynamic model of a satellite affected by external torques is then

$$\boldsymbol{T} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}\,, \qquad \textit{Eq. 52}$$

in which $\boldsymbol{I}$ are the principle axes of the spacecraft's inertia matrix. In the definition of the context (*i.e.*, torque-free motion, $\boldsymbol{T} = 0$), Eq. 52 reduces to

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = 0\,, \qquad \textit{Eq. 53}$$

which gives the model for torque-free attitude dynamics. Rearranging Eq. 53 to solve for the satellite's angular acceleration $\dot{\boldsymbol{\omega}}$ **(***i.e.*, Euler's equations**)**,

$$\begin{cases}\dot{\omega}_\psi = \dfrac{(I_2 - I_3)}{I_1}\omega_\theta\omega_\phi\\ \dot{\omega}_\theta = \dfrac{(I_3 - I_1)}{I_2}\omega_\psi\omega_\phi\\ \dot{\omega}_\phi = \dfrac{(I_1 - I_2)}{I_3}\omega_\psi\omega_\theta\end{cases}. \qquad \textit{Eq. 54}$$

The angular velocities of the body frame as seen from the *ECI* frame $\boldsymbol{\omega}_I^B$ are given

$$\boldsymbol{\omega}_I^B = \begin{bmatrix}\omega_\psi\\ \omega_\theta\\ \omega_\phi\end{bmatrix} = \begin{bmatrix}-3.092\cdot10^{-4}\\ 6.6161\cdot10^{-4}\\ 7.4606\cdot10^{-4}\end{bmatrix}, \text{(rad/s)}$$

The attitude model has now been fully defined and can be integrated in time using initial conditions $\boldsymbol{\Psi}$ and $\boldsymbol{\omega}_I^B$**.** Using **`ode113`,** the time evolution of the Euler Angles and angular velocities of the spacecraft may be computed. The results are plotted in Figure II-1 Euler Angles vs. timeFigure II-1 and Figure II-2 respectively. The aforementioned issues with singularities in the kinematics equations are revealed in Figure II-1, as seen by the sudden inflection of the Euler angles at time $t \approx 1100\ s$. As a consequence, the satellite's attitude is indeterminate when using the kinematics equations derived from Euler's angles, further validating the need for quaternions.

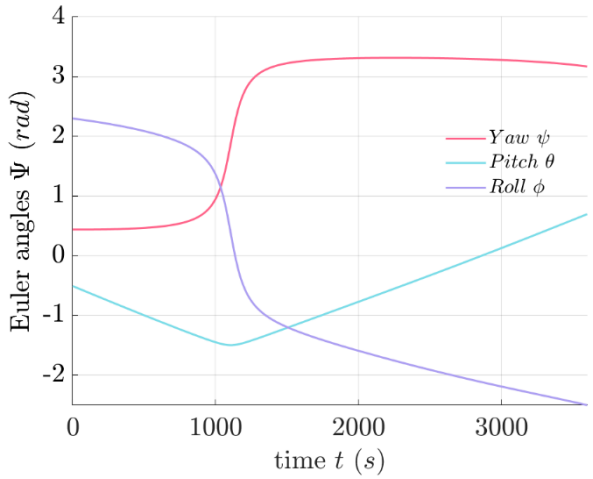

Figure II-1 Euler Angles vs. time

Despite the singularity issues that arise from the propagation of Euler angles, the evolution of the angular velocities is, however, consistent with expectations. Since no torque is applied to the spacecraft, in theory the attitude dynamics should display periodicity.

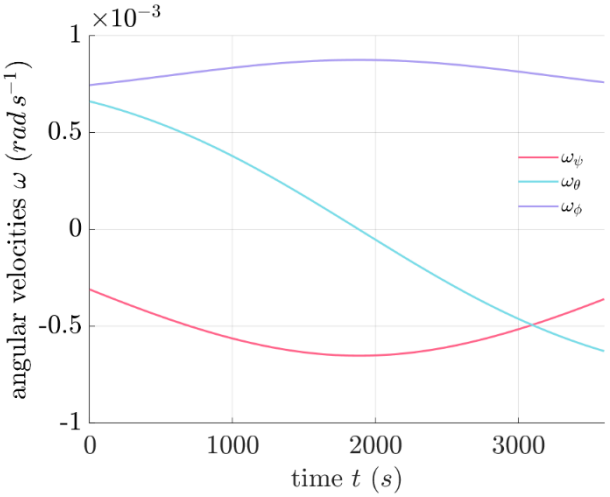

Figure II-2 Angular Velocities vs. time

[2] *Sidi*, 1997

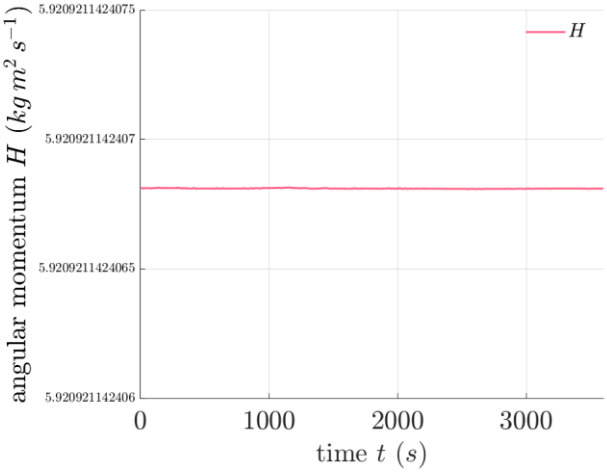

Figure II-5 Angular Momentum vs. time

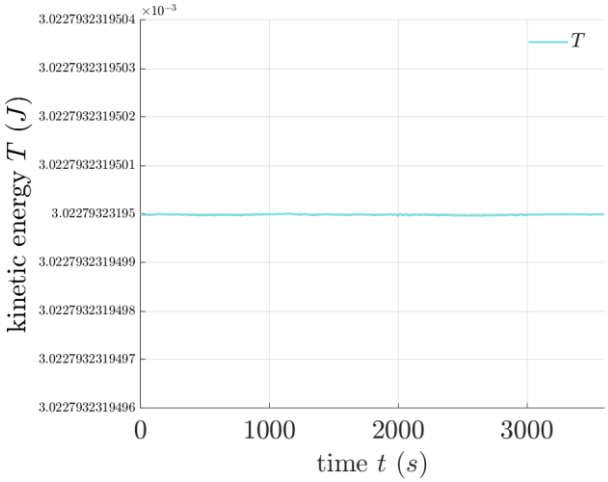

Figure II-3 Rotation Kinetic Energy vs. time

Under the assumption that the spacecraft is a perfectly rigid body in a torque-free environment, both the angular momentum and rotational kinetic energy are expected to be conserved. Figure II-5 and Figure II-3 display the fact that these parameters are indeed integrals of motion.

The Polhode plot shown in Figure II-4 may be used to corroborate the validity of the angular momentum and kinetic energy values. The white sphere, here, represents the momentum sphere of radius $\|\boldsymbol{h}\|$, on which the angular momentum vector traces out a trajectory on its surface, represented in red. Furthermore, the angular momentum vector must also trace a trajectory on the energy ellipsoid, represented in blue. Hence, all the possible values of $\boldsymbol{h}$ must lie on the intersection between both the momentum sphere and the energy ellipsoid. It can also be deduced that the stability of the system is maximal – *i.e.*, the energy required to transfer from one pole of the energy ellipsoid to the other is maximal.

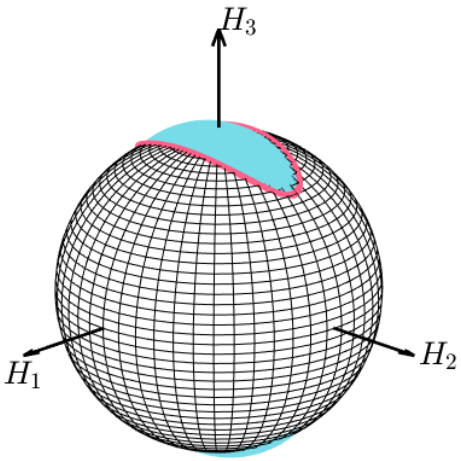

Figure II-4 Polhode Plot

CONCLUSION

This report has in effect demonstrated both the physics and the theory behind space dynamics fundamentals. The two-body problem and the associated orbital mechanics were explored in detail. For this, both an analytical and numerical approach to the problem were suggested for orbit propagation. Additionally, for sake of relevance to satellite navigation, different reference frames and their associated equations of motion were studied, namely that of the ECEF rotating reference frame. Eventually, the topic of attitude mechanics and its associated frameworks/ methods to describing a satellite's orientation in space were investigated. By the end, the report demonstrated the inadequacies of attitude modelling using Euler angles, further validating the necessity for quaternion representation.

APPENDIX

```
clearvars; close all; clc; format longG;

%Physical Constants
GM = 398600.4418;                                                    %Earth's gravitational parameter
[km^3/s^2]
Re = 6378.137;                                                       %Earth's equatorial radius [km]
Te = 86164;                                                          %Earth's rotational period [s]
we = 2*pi / Te;                                                      %Angular velocity of spacecraft
[rad/s]
Im = [2500 0       0;
      0    5000    0;
      0    0       6500];                                          %Inertia matrix of spacecraft

AngVel = [-3.092e-4; 6.6161e-4; 7.4606e-4];

tol = 10e-10;
ode_opt = odeset('RelTol', 1e-13, 'AbsTol', 1e-14);

%Classical Orbital Elements
a = 7151.16;                                                          %Semi-major axis [km]
ECC = 0.0008;                                                         %Eccentricity [~]
I = deg2rad(98.39);                                                   %Inclination [rad]
RAAN = deg2rad(10);                                                   %RAAN [rad]
argP = deg2rad(233);                                                  %Argument of periapsis [rad]
MA0 = deg2rad(127);

%Solve Kepler's equation
[E0, ~] = Kepler(ECC, MA0, tol);                                      %Eccentric Anomaly in [rad]

%Compute the intial True Anomaly of the spacecraft
TA0 = 2*atan2(sqrt(1+ECC)*tan(E0/2), sqrt(1-ECC));                    %True Anomaly in [rad]
fprintf('Initial True Anomaly TA0: %.4f rads\n', TA0);

%Check location along orbit
if pi/2 < TA0 && TA0 < 1.5*pi
    fprintf('The spacecraft is near its apoapsis\n');
else
    fprintf('The spacecraft is near its periapsis\n');
end

%Compute orbital period of spacecraft
n = sqrt(GM/a^3);                                                     %Mean  motion [rad/s]
P = 2*pi/n;                                                           %Orbit period [s]

fprintf('Orbit Period P: %.4f (s)\n', P);

%Create time vector
t0 = 0;
t = linspace(t0, P + t0, 1000);                                        %Time vector [s]
MAt = mod(MA0 + n * (t - t0), 2*pi);                                   %Mean anomaly as a funciton of
time [rad]

E_matrix = zeros(1, length(t));                                        %Initialising E_matrix

%Propgate state of satellite : MA(t), E(t), TA(t) using COE
for tt = 1:length(t)

    %Solve Kepler's equation
    [E,~] = Kepler(ECC, MAt(tt), tol);                                 %dEdt eccentric anomaly as a
function of time
    E_matrix(tt) = E;                                                  %store value in matrix for
plotting

    TA(tt) = mod(2*atan2(sqrt(1+ECC)*tan(E/2), sqrt(1-ECC)), 2*pi);    %Compute True Anomaly as a
function of time

    %Store COE & convert to Cartesian coords / COE2RV
    COE(:, tt) = [a, ECC, I, RAAN, argP, TA(tt)];                      %Classical Orbital Elements ma-
trix
    X(:,tt) = COE2RV(COE(:,tt), GM);                                   %State vector from classical
orbital elements to RV propagation
end
```

```
%Integrate equation of motion
[~, Xout] = ode113(@TBP_ECI, t, X(1:6, 1), ode_opt, GM);                    %Integrate the equation of mo-
tion of the ECI TBP using ODE45
Xout = Xout';                                                               %Transpose of the output
v_Xout = sqrt(Xout(4, :).^2 + Xout(5, :).^2 + Xout(6, :).^2);               %Velocity magnitude from Xout
integration
r_Xout = sqrt(Xout(1, :).^2 + Xout(2, :).^2 + Xout(3, :).^2);               %Position magnitude from Xout
integration

diff = abs((X(1:6, :) - Xout(1:6, :)));                                     %Position and velocity dis-
crepency from COE2RV propagation
rdiff = sqrt(diff(1, :).^2 + diff(2, :).^2 + diff(3, :).^2);                %Position error magnitude
vdiff = sqrt(diff(4, :).^2 + diff(5, :).^2 + diff(6, :).^2);                %Velocity error magnitude

%Compute specific energy of the spacecraft at every time stamp
for i = 1:length(t)
    sp_e(i) = 0.5 * v_Xout(i)^2 - GM / r_Xout(i);                           %Specific energy of the space-
craft, should be approx. constant and can be approximated to -GM / 2*a
    sp_e2(i) = -GM / (2*a);
end

w = [0 0 we]';
FI = eye(3);

X_ini_ECEF = [FI * X(1:3, 1); FI * X(4:6, 1)-cross(w, X(1:3, 1))];          %Spacecraft initial conditions
in ECEF frame

%Integrate the equations of motion of the satellite in ECEF frame
[~, X_ECEF] = ode113(@TBP_ECEF, 10*t, X_ini_ECEF, ode_opt, GM);
X_ECEF = X_ECEF';

%Initial position and velocity in the ECI frame 1h into orbit
r6 = [6768.27 870.90 2153.59]';
v6 = [-2.0519 -1.4150 7.0323]';

%Initial Euler Angles
alpha = deg2rad(30);
beta = deg2rad(20);
gamma = deg2rad(10);

%Rotation matrices 1-2-1 sequence
R1_alpha = [  1             0               0;
              0             cos(alpha)      sin(alpha);
              0             -sin(alpha)     cos(alpha)];

R2_beta =  [  cos(beta)     0               -sin(beta);
              0             1               0;
              sin(beta)    0               cos(beta)];

R1_gamma = [  1             0               0;
              0             cos(gamma )     sin(gamma);
              0             -sin(gamma )    cos(gamma)];;

%DCM from Orbital to Body frame
R_BO = R1_gamma * R2_beta * R1_alpha;

%RSW Orbital frame axes
O1 = r6 / norm(r6);
h6 = cross(r6, v6);
O3 = h6 / norm(h6);
O2 = cross(O3, O1);

%DCM from Inertial to Orbital frame
R_OI = [O1(1) O1(2) O1(3);
        O2(1) O2(2) O2(3);
        O3(1) O3(2) O3(3)];

%DCM from Inertial to Body frame
R_BI = R_BO *  R_OI;

EulAng BO = acos((trace(R_BI) - 1) / 2);                                    %Determine Euler angles to rotate from
Orbital to Body frame

%Euler's principle axes
EulAX_1 = (R_BI(2,3) - R_BI(3,2)) / (2 * sin(EulAng_BO));
EulAX_2 = (R_BI(3,1) - R_BI(1,3)) / (2 * sin(EulAng_BO));
EulAX_3 = (R_BI(1,2) - R_BI(2,1)) / (2 * sin(EulAng_BO));
```

```
EulAx = [EulAX_1 EulAX_2 EulAX_3]';

%Quaternion representation
q123 = EulAx * sin(EulAng_BO / 2);                                  %Determinne Euler parameters / quater-
nions
q4 = cos(EulAng_BO/2);
q = [q123; q4];
scale = norm(q);                                                    %nomalised value of quaternion
q = q/scale;                                                        %normalise quaternion values

%DCM from Inertial to Body frame from quaternions
q_BI = [q(1)^2 - q(2)^2 - q(3)^2 + q(4)^2 2*(q(1)*q(2) + q(4)*q(3)) 2*(q(1)*q(3) - q(4)*q(2));
       2*(q(1)*q(2) - q(4)*q(3)) -q(1)^2 + q(2)^2 - q(3)^2 + q(4)^2 2*(q(2)*q(3) + q(4)*q(1));
       2*(q(1)*q(3) + q(4)*q(2)) 2*(q(2)*q(3) - q(4)*q(1)) -q(1)^2 - q(2)^2 + q(3)^2 + q(4)^2];

%Euler Angles from Inertial to body frame
psi = atan2(q_BI(1,2), q_BI(1,1));
theta = asin(-q_BI(1,3));
phi = atan2(q_BI(2,3), q_BI(3,3));
EulAng_BI = [psi; theta; phi];

%Initial attitude state
X_ini_Att = [EulAng_BI; AngVel];

t2 = linspace(0, 3600, 360);
[~, X_Att] = ode113(@AttitudeDynamics, t2, X_ini_Att, ode_opt, Im);  %Integrate attitude of spacecraft /
propagate in time

X_Att = X_Att';

%Compute Angular momentum and rotational kinetic energy of spacecraft
for ii = 1:length(t2)
    H(:,ii) = Im*X_Att(4:6, ii);                                    %Angular momentum
    Hmag(:,ii) = norm(H(:,ii));                                     %Magnitude of angular momentum
    T(:,ii) = 0.5 * (X_Att(4:6, ii)' * H(:,ii));                    %Rotational kinentic energy
    Tmag(:,ii) = norm(T(:,ii));                                     %magnitude of kinetic energy
end

%% Plots
%########## Specific Energy Plot ##########################################################
hfig = figure;
ax = gca;
hold on;
sp_diff = abs(sp_e - sp_e2);
yyaxis right
plot(t, sp_e, 'LineWidth', 1.5, 'DisplayName', 'Specific energy', 'Color', "#78DCE8");
set(gca, 'YTick', [])
ax.YColor = 'black';
ax.YColor = 'black';

yyaxis left
plot(t, sp_e2, 'LineWidth', 1.5, 'LineStyle', '-','DisplayName', 'Specific energy', 'Color', "#FF6188");
xlabel('time $t$ $(s)$');
ylabel('specific energy $\zeta$ $(Jkg^{-1})$', 'Color', "#3ce0d5");
ax.YColor = 'black';
ax.YColor = 'black';
xlim([min(t), max(t)]);
lgd = legend('Location', 'best');
lgd.EdgeColor = 'black';
xlim([min(t), max(t)]);

legend({'$\zeta_{1}=-\frac{\mu }{2a}$', '${\zeta_{2}}=\frac{\nu ^{2}}{2}-\frac{\mu }{r}$'});
grid on;

picturewidth = 17.6;
hw_ratio = .75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])

% print(hfig, 'specific energy plot', '-dpng', '-painters')
```

```
%########## VELOCITY AND POSITION ERROR PLOT ############################################################
hfig = figure;
ax = gca;

yyaxis left
plot(t, vdiff, 'Color', "#FF6188", 'LineWidth', 1, 'DisplayName', '$\epsilon_v$');
xlabel('time $t$ $(s)$')
ylabel('velocity error $\epsilon_v$ $(ms^{-1})$')
ax.YColor = 'black';

yyaxis right;
plot(t, rdiff, 'Color', "#78DCE8", 'LineWidth', 1, 'DisplayName', '$\epsilon_r$');
ylabel('position error $\epsilon_r$ $(m)$', 'Color', "#e62740");
ax.YColor = 'black';

lgd = legend('Location', 'best');
lgd.EdgeColor = 'black';
xlim([min(t), max(t)]);
grid on;

picturewidth = 17.6;
hw_ratio = 0.75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'velocity and error plot', '-dpng', '-painters')

%########## ANOMALY VS TIME PLOT ##########################################################
hfig = figure;
yyaxis left;
hold on;
plot(t, E_matrix, 'LineWidth', 1.5, 'LineStyle', '-', 'DisplayName', 'EA', 'Color', "#FF6188");
plot(t, TA, 'LineWidth', 1.5, 'LineStyle', '-',  'DisplayName', 'TA', 'Color', "#78DCE8");
plot(t, MAt, 'LineWidth', 1.5, 'LineStyle', '-', 'DisplayName', 'MA', 'Color', "#AB9DF2");
xlabel('time $t$ $(s)$');
ylabel('angle $(rad)$', 'Color', 'k');
lgd = legend('Location', 'best');
xlim([min(t), max(t)]);
ax = gca;
ax.YColor = 'black';
grid on;
yyaxis right;
Ediff = E_matrix - MAt;
TAdiff = TA - MAt;
plot(t, Ediff, 'LineWidth', 1.5, 'LineStyle', '--', 'DisplayName', 'EA deviation from MA', 'Color',
"#FF6188");
plot(t, TAdiff, 'LineWidth', 1.5, 'LineStyle', '-.', 'DisplayName', 'TA deviation from MA', 'Color',
"#78DCE8");
ylim([-5e-3 5e-3]);
ylabel('deviation angle $(rad)$', 'Color', 'k')
ax.YColor = 'black';

picturewidth = 17.6;
hw_ratio = 0.75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
lgd.FontSize = 10;
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'Anomaly plot', '-dpng', '-painters')

%########## ORBIT PLOTS ###################################################################

% create orbit plot
[Xe,Ye,Ze] = sphere(50);

% Create figure
hfig = figure;

% Plot Earth
```

```
mesh(Re*Xe, Re*Ye, Re*Ze, 'EdgeColor', 'k'); hold on; axis equal;
xlabel('$I_{ECI} (km)$');
ylabel('$J_{ECI} (km)$');
zlabel('$K_{ECI} (km)$');

% Plot ECI / ECEF axes
quiver3(0, 0, 0, 1, 0, 0, 1e4, 'k', 'LineWidth', 2);    % I/X-axis
quiver3(0, 0, 0, 0, 1, 0, 1e4, 'k', 'LineWidth', 2);    % J/Y-axis
quiver3(0, 0, 0, 0, 0, 1, 1e4, 'k', 'LineWidth', 2);    % K/Z-axis
text(1.22e4, 0, 0, '$I$', 'Color', 'k')
text(0, 1.1e4, 0, '$J$', 'Color', 'k')
text(0, 0, 1.1e4, '$K$', 'Color', 'k')

% Plot Trajectory in ECI frame
plot3(X(1,:), X(2,:), X(3,:),'LineWidth', 2, 'Color', "#FF6188");
plot3(X(1,1), X(2,1), X(3,1), 'ok', 'MarkerFaceColor', "#A9DC76"); %start indicator
plot3(X(1,end), X(2,end), X(3,end), 'ok', 'MarkerFaceColor', "#A9DC76");%end indicator
view(130, 30);

% 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);
e = cross(v, h)/GM - r/norm(r);

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

quiver3(0, 0, 0, ie(1), ie(2), ie(3), 1e4, 'Color', "#78DCE8", 'LineWidth', 2);
quiver3(0, 0, 0, ip(1), ip(2), ip(3), 1e4, 'Color', "#78DCE8", 'LineWidth', 2);
quiver3(0, 0, 0, ih(1), ih(2), ih(3), 1e4, 'Color', "#78DCE8", 'LineWidth', 2);
text(1e4*ie(1), 1e4*ie(2), 1.2e4*ie(3), '$i_{e}$', 'FontSize', 20, 'Interpreter', 'tex', 'Color',
"#78DCE8")
text(1e4*ip(1), 1e4*ip(2), 1.1e4*ip(3), '$i_{p}$', 'FontSize', 20, 'Interpreter', 'tex', 'Color',
"#78DCE8")
text(1e4*ih(1), 1.15e4*ih(2), 1e4*ih(3), '$i_{h}$', 'FontSize', 20, 'Interpreter', 'tex', 'Color',
"#78DCE8")

% Plot Trajectory in ECEF frame
plot3(X_ECEF(1,:), X_ECEF(2,:), X_ECEF(3,:), 'LineWidth', 2, 'Color', "#78DCE8");
plot3(X_ECEF(1,1), X_ECEF(2,1), X_ECEF(3,1), 'ok', 'MarkerFaceColor', "#A9DC76");
plot3(X_ECEF(1,end), X_ECEF(2,end), X_ECEF(3,end), 'ok', 'MarkerFaceColor', "#FF6188");
grid off

picturewidth = 17.6;
hw_ratio = .9;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'ECI plot', '-dpng', '-vector')

%########################## POLHODE PLOT ############################
%create orbit plot
[H1,H2,H3] = sphere(50);

% Create figure
hfig = figure;

% Plot Earth
mesh(Hmag(1)*H1, Hmag(1)*H2, Hmag(1)*H3, 'EdgeColor', 'k'); hold on; axis equal;
axis off

%plot momentum vector
plot3(H(1,:), H(2,:), H(3,:),'LineWidth', 5, 'Color', "#FF6188");

% Plot H axes
quiver3(0, 0, 0, 1, 0, 0, 10, 'k', 'LineWidth', 2);
quiver3(0, 0, 0, 0, 1, 0, 10, 'k', 'LineWidth', 2);
quiver3(0, 0, 0, 0, 0, 1, 10, 'k', 'LineWidth', 2);
text(12, 1, 0, '$H_{1}$', 'Color', 'k')
text(0, 10.25, 0, '$H_{2}$', 'Color', 'k')
text(0, 0, 10.25, '$H_{3}$', 'Color', 'k')
```

```
grid off
view(135, 20);

picturewidth = 17.6;
hw_ratio = 1;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])

[h1, h2, h3] = ellipsoid(0, 0, 0, sqrt(2*Im(1,1)*T(1)), sqrt(2*Im(2,2)*T(1)), sqrt(2*Im(3,3)*T(1)), 50);
surf(h1, h2, h3, 'LineStyle', 'none', 'FaceColor', "#78DCE8"); hold on; axis equal;

% print(hfig, 'PoleHhode plot', '-dpng', '-painters')

%################ EULER ANGLES PLOT ##########
hfig = figure;
ax = gca;
hold on

plot(t2, X_Att(1,:), 'Color', "#FF6188", 'LineWidth', 1.5, 'DisplayName', '$Yaw$ $\psi$');
plot(t2, X_Att(2,:), 'Color', "#78DCE8", 'LineWidth', 1.5, 'DisplayName', '$Pitch$ $\theta$');
plot(t2, X_Att(3,:), 'Color', "#AB9DF2", 'LineWidth', 1.5, 'DisplayName', '$Roll$ $\phi$');
xlabel('time $t$ $(s)$')
ylabel('Euler angles $\Psi$ $(rad)$')
ax.YColor = 'black';

lgd = legend('Location', 'best');
lgd.EdgeColor = 'black';
xlim([min(t2), max(t2)]);
grid on;

picturewidth = 17.6;
hw_ratio = 0.75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
lgd.FontSize = 15;
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'Euler Ang plot', '-dpng', '-painters')

%################ ANGULAR VELOCITY PLOT ##########
hfig = figure;
ax = gca;
hold on

plot(t2, X_Att(4,:), 'Color', "#FF6188", 'LineWidth', 1.5, 'DisplayName', '$\omega_{\psi}$');
plot(t2, X_Att(5,:), 'Color', "#78DCE8", 'LineWidth', 1.5, 'DisplayName', '$\omega_{\theta}$');
plot(t2, X_Att(6,:), 'Color', "#AB9DF2", 'LineWidth', 1.5, 'DisplayName', '$\omega_{\phi}$');
xlabel('time $t$ $(s)$')
ylabel('angular velocities $\omega$ $(rad\,s^{-1})$')
ax.YColor = 'black';

lgd = legend('Location', 'best');
lgd.EdgeColor = 'black';
xlim([min(t2), max(t2)]);
grid on;

picturewidth = 17.6;
hw_ratio = 0.75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
lgd.FontSize = 15;
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'Ang velocities plot', '-dpng', '-painters')

%################ ANGULAR MOMENTUM AND ROTATIONAL KINETIC ENERGY PLOT ##########
```

```
hfig = figure;
ax = gca;
hold on

plot(t2, Hmag, 'Color', "#FF6188", 'LineWidth', 1.5, 'DisplayName', '$H$');
xlabel('time $t$ $(s)$')
ylabel('angular momentum $H$ $(kg\,m^{2}\,s^{-1})$')
ax.YColor = 'black';

lgd = legend('Location', 'best');
lgd.EdgeColor = 'black';
xlim([min(t2), max(t2)]);
grid on;

picturewidth = 17.6;
hw_ratio = .75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
lgd.FontSize = 15;
ax.YAxis.FontSize = 10;
ax.YLabel.FontSize = 20;
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'Ang momentum plot', '-dpng', '-painters')

hfig = figure;
ax = gca;
hold on

plot(t2, Tmag, 'Color', "#78DCE8", 'LineWidth', 1.5, 'DisplayName', '$T$');
xlabel('time $t$ $(s)$')
ylabel('kinetic energy $T$ $(J)$')
ax.YColor = 'black';

lgd = legend('Location', 'best');
lgd.EdgeColor = 'black';
xlim([min(t2), max(t2)]);
grid on;

picturewidth = 17.6;
hw_ratio = .75;
set(findall(hfig, '-property', 'FontSize'), 'FontSize', 20)
lgd.FontSize = 15;
ax.YAxis.FontSize = 10;
ax.YLabel.FontSize = 20;
set(findall(hfig, '-property', 'Box'), 'Box', 'off')
set(findall(hfig, '-property', 'Interpreter'), 'Interpreter', 'latex')
set(findall(hfig, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
set(hfig, 'Units', 'centimeters', 'Position', [3 3 picturewidth hw_ratio*picturewidth])
pos = get(hfig, 'Position');
set(hfig, 'PaperPositionMode', 'Auto', 'PaperUnits', 'centimeters', 'PaperSize', [pos(3), pos(4)])
% print(hfig, 'Rot kin En plot', '-dpng', '-painters')


function [E, i] = Kepler(ECC, MA, tol)
% Function to solve Kepler's equation M = E - e * sinE
%
% Inputs:
%   ECC     : Eccentricity
%   MA      : Mean anomaly [rad] must be 0 <= M <= pi
%   tol     : Threshold tolerance
%
% Outputs:
%   E       : eccentric anomaly
%   i       : Iteration count

if ECC >= 0.99 | ECC < 0                    %Error exit if e is negative or greater than 0.99

    error('Input Error: the eccentricity must be contained within 0 < e < 1');
    return

else

    En = MA;                                %initialise algorithm with 'educated guess'
```

```
    i = 0;                                  %initialise iteration count
    fEn = En - ECC * sin(En) - MA;          %initialise F(En)

    while abs(fEn) > tol                   %iterate while F(En) is higher than threshold value

        i= i+1;                              %add to iteration counter
        fEn = En - ECC * sin(En) - MA;
        fpEn = 1 - ECC * cos(En);
        Ens = En - fEn/fpEn;                 %Newton's method
        En = Ens;                            %update En with iterated value

    end
end

E = mod(En, 2*pi);                                      %update E with final computed value



function [X] = COE2RV(COE, GM)
%{
convert state of spacecraft from COE to ECI position and veloctiy
components

Inputs:
    COE : Matrix of classical orbital elements
    GM : Earth's gravitational parameter [km3/s2]

Outputs:
    X : State vector of spacecraft in ECI

Method:

    Calculate r = a*(1-e^2) / (1 + e cosTA)
    Calculate h = sqrt(mu*a*(1-e^2))
    Rotation matrix from perifocal frame to ECI  :
        [x y z] = ROTA * (r*cosTA r*sinTA 0)
        [xdot ydot zdot] = ROTA * (-(GM/h)*sinTA (GM/h)*(cosTA+e) 0)
%}

%Classical Orbital paramters
a = COE(1);
ECC = COE(2);
I = COE(3);
RAAN = COE(4);
argP = COE(5);
TA = COE(6);

%Compute position vector in perifocal frame
pos = a * (1-ECC^2) / (1 + ECC * cos(TA));
r_P = [pos * cos(TA); pos * sin(TA); 0];

%Compute specific angular momentum
h = sqrt(GM * a * (1 - ECC^2));

%Comput velocity vector (rdot) in perifocal frame
v_P = [-(GM/h)*sin(TA); (GM/h)*(cos(TA) + ECC); 0];

%Rotation matrices
R3_RAAN = [  cos(RAAN)      sin(RAAN)      0;
            -sin(RAAN)     cos(RAAN)      0;
            0              0              1];
R1_I =    [  1        0           0;
             0        cos(I)      sin(I);
             0        -sin(I)     cos(I)];

R3_argP = [  cos(argP)      sin(argP)    0;
            -sin(argP)     cos(argP)      0;
            0              0              1];

ECItoP = R3 argP*R1_I*R3_RAAN; %3-1-3 rotation to go from Perifocal frame to ECI
PtoECI = ECItoP';

r_ECI = PtoECI * r_P;
v_ECI = PtoECI * v_P;
X = [r_ECI; v_ECI];
```

```
function [Xdot_ECI] = TBP_ECI(t, X, GM)

%{

Function to output system of first order ODEs to solve equations of motion
in ECI frame

Inputs:
    t : time [s]
    X : State Vector of spacecraft in ECI
    GM : Earth's gravitational parameter [km3/s2]

Output:
    Xdot_ECI = dXdt time derivative of cartesian coordinates

Method:
    rewrite TBP as system of 6 first-order ODEs Xdot = [...]
    Use ODE45 to integrate the equations of motion over time --> X(t)

%}

%Define state vector / extract components
x1 = X(1);
x2 = X(2);
x3 = X(3);
x4 = X(4);
x5 = X(5);
x6 = X(6);

r = sqrt(x1^2 + x2^2 + x3^2);

%TBP as system of first-order ODEs
x1dot = x4;
x2dot = x5;
x3dot = x6;
x4dot = -(GM/r^3) * x1;
x5dot = -(GM/r^3) * x2;
x6dot = -(GM/r^3) * x3;

Xdot_ECI = [x1dot, x2dot, x3dot, x4dot, x5dot, x6dot]';


function [Xdot_ECEF] = TBP_ECEF(t, X, GM)

%{

Function to output system of first order ODEs to solve equations of motion
in ECEF frame

Inputs:
    t : time [s]
    X : State Vector of spacecraft in ECEF frame
    GM : Earth's gravitational parameter [km3/s2]

Output:
    Xdot_ECEF = dXdt time derivative of cartesian coordinates

Method:
    rewrite TBP as system of 6 first-order ODEs Xdot = [...]
    Use ODE45 to integrate the equations of motion over time --> X(t)

%}

r = norm(X(1:3));

%Angular velocity of the ECEF frame as seen from ECI frame
we = 7.29212351699038e-5;
w = [0 0 we]';

%TBP as system of first-order ODEs
xdot = X(4:6);
vdot = -(GM/r^3) * X(1:3) - cross(2*w, (X(4:6))) - cross(w, cross(w, X(1:3)));

Xdot_ECEF = [xdot; vdot];
```

```
function [Xdot_Att] = AttitudeDynamics(t, X, Im)

%{

Function to integrate attitude dynamics of satellite

Inputs:
    t : time [s]
    X : Attitude state vector of spacecraft
    I : Inertia matrix of spacecraft body

Output:
    Xdot_Att = attitude dynamics as function of time

%}

B = [0          sin(X(3))               cos(X(3));
     0          cos(X(2))*cos(X(3))     -cos(X(2))*sin(X(3));
     cos(X(2))  sin(X(2))*sin(X(3))     sin(X(2))*cos(X(3))];

AngVel = X(4:6);

EulAng_dot = (1 / cos(X(2))) * B * AngVel;
EulEq1 = -(cross(AngVel, Im*AngVel));
EulEq = EulEq1' / Im;

Xdot_Att = [EulAng_dot; EulEq'];
```