**Introduction**

1. Function Application Workflow

**Contents**

1. [**kepel\_statvec.m**](#kepel_statvec)
2. [**kepler.m**](#kepler)
3. [**djm.m**](#djm)
4. [**time\_to\_dayf.m**](#time_to_dayf)
5. [**statvec\_kepel.m**](#statvec_kepel)
6. [**delkep.m**](#delkep)
7. [**egm\_read\_data.m**](#egm_read_data)
8. [**orbital\_to\_inertial\_matrix.m**](#orbital_to_inertial_matrix)
9. [**egm\_difeq.m**](#egm_difeq)
10. [**egm\_acc.m**](#egm_acc)
11. [**gst.m**](#gst)
12. [**inertial\_to\_terrestrial.m**](#inertial_to_terrestrial)
13. [**terrestrial\_to\_inertial.m**](#terrestrial_to_inertial)
14. [**rotmaz.m**](#rotmaz)
15. [**Proximus.m**](#proximus)

**Example**

1. [**demo\_egm**](#demo_egm)

**Reference**

1. Vallado, D. A. (2013). Fundamentals of Astrodynamics and Applications (4th ed.). Hawthorne, CA: Microcosm Press.
2. Chandeok Park, 2024. Chapter 2. Kepler’s Equation and Kepler’s Problem, ASTRODYNAMICS I, Yonsei University.

**Introduction**

This document describes the structure and purpose of a set of functions that propagate a satellite orbit by applying the Earth's non-spherical gravity model (EGM) based on given Keplerian elements. The core functions include orbital element conversion, time computation, gravity modeling, definition of numerical equations of motion, and coordinate transformation.

The following section outlines the overall functional workflow in a step-by-step manner, illustrating how each function contributes to the numerical orbit propagation. Additionally, a detailed explanation of each function—including its inputs, outputs, and internal logic—is provided to help readers understand the implementation in a structured way.

At the end of the document, a simulation example demonstrates how these functions are applied in practice, supported by figures and results generated from the full code.

**Function Application Workflow**

**1. Initial Keplerian Element Setup and State Vector Conversion**

* The **kepel\_statvec** function converts the initial Keplerian elements (a,e,i,Ω,ω,M) to an ECI position and velocity vector.
* Internally, the conversion from Mean Anomaly M to Eccentric Anomaly E is computed using the **kepler** function via the Newton-Raphson method.
* The position and velocity in the PQW frame are rotated to the ECI frame using **orbital\_to\_inertial\_matrix** and **rotmaz**.

|  |  |
| --- | --- |
| **Function Name** | **Brief Description** |
| [**kepel\_statvec**](#kepel_statvec) | Converts Keplerian elements to ECI state vector (includes PQW → ECI rotation) |
| [**kepler**](#kepler) | Computes the Eccentric Anomaly (E) from Mean Anomaly (M) using the Newton-Raphson method |
| [**orbital\_to\_inertial\_matrix**](#orbital_to_inertial_matrix) | Generates a rotation matrix to transform from orbital frame (PQW) to inertial frame (ECI) |
| [**rotmaz**](#rotmaz) | Generates a rotation matrix around the z-axis (e.g., Earth rotation) |

**2. Time Conversion for GST Calculation**

* The **djm** function converts the input calendar date to Modified Julian Date (MJD) based on the epoch 1950.0.
* The **time\_to\_dayf** function computes the elapsed seconds from UTC time (hh:mm:ss) within a day.
* These two values are used later in **gst** for calculating the Greenwich Sidereal Time.

|  |  |
| --- | --- |
| **Function Name** | **Brief Description** |
| [**djm**](#djm) | Converts calendar date to Modified Julian Date (MJD), referenced to 1950.0 |
| [**time\_to\_dayf**](#time_to_dayf) | Converts hours, minutes, seconds into elapsed seconds of the day (UTC) |
| [**gst**](#gst) | Calculates Greenwich Sidereal Time (GST) from Modified Julian Date and UTC seconds |

**3. Loading EGM Gravity Model**

* The **egm\_read\_data** function loads the gravitational coefficients (Cnm, Snm) from a file (e.g., **egm\_10.dat**) and limits the degree/order to nmax.
* These coefficients are used later in **egm\_acc** for precise gravity computation.

|  |  |
| --- | --- |
| **Function Name** | **Brief Description** |
| [**egm\_read\_data**](#egm_read_data) | Loads EGM coefficient file (.dat) and applies the user-defined maximum degree (nmax) |
| [**egm\_acc**](#egm_acc) | Computes gravitational acceleration from Earth's aspherical gravity model (in Earth-fixed frame) |

**4. Analytical Orbital Element Drift from J2/J4 Perturbations**

* The **delkep** function calculates the time rates of change of Keplerian elements due to oblateness perturbations (J2 and J4).
* These rates are used to compute the analytically predicted orbital elements:

|  |  |
| --- | --- |
| **Function Name** | **Brief Description** |
| [**delkep**](#delkep) | Calculates the time derivatives of Keplerian elements due to J2 and J4 perturbations |

**5. Numerical Orbit Propagation (ODE45 Integration)**

* The **egm\_difeq** function defines the differential equations used for propagation with the ODE45 solver. It performs:
  + Calculation of Greenwich Sidereal Time using **gst**
  + Transformation from inertial to Earth-fixed frame using **inertial\_to\_terrestrial**
  + Computation of acceleration in the ECEF frame via **egm\_acc** using EGM coefficients
  + Conversion of acceleration back to ECI using **terrestrial\_to\_inertial**
  + Summation with external acceleration (if any) to complete the EOM

|  |  |
| --- | --- |
| **Function Name** | **Brief Description** |
| [**egm\_difeq**](#egm_difeq) | Defines the orbital differential equations with EGM-based gravity (used in ODE45 solver) |
| [**egm\_acc**](#egm_acc) | Computes gravitational acceleration from Earth's aspherical gravity model (in Earth-fixed frame) |
| [**gst**](#gst) | Calculates Greenwich Sidereal Time (GST) from Modified Julian Date and UTC seconds |
| [**inertial\_to\_terrestrial**](#inertial_to_terrestrial) | Converts state vector from ECI frame to Earth-fixed (GTC) frame |
| [**terrestrial\_to\_inertial**](#terrestrial_to_inertial) | Converts state vector from Earth-fixed (GTC) frame to ECI frame |

**6. Post-Processing and Comparison**

* The integrated state vectors are converted back to Keplerian elements using **statvec\_kepel**.
* The **proximus** function is applied to normalize the mean anomaly for consistent angular comparison.
* These values are compared against analytically propagated elements to assess perturbation accuracy.

|  |  |
| --- | --- |
| **Function Name** | **Brief Description** |
| [**statvec\_kepel**](#statvec_kepel) | Converts ECI state vector back to Keplerian elements |
| [**proximus**](#proximus) | Normalizes angular values to be closest to a reference angle (within 2π range) |

**7. Visualization and Result Analysis**

* Time histories of position errors, velocity errors, and Keplerian element deviations are plotted.
* The differences between numerical (EGM-based) and analytical (J2/J4-based) results are visualized in 12 figures, supporting performance evaluation.

**function [statvec] =** **kepel\_statvec (kepel)**

**Input :** semimajor axis, eccentricity, inclination, right ascension of ascending node, argument of perigee, mean anomaly

**Output :** ECI coordinates state vector

**Internal functions** : kepler, rotmax, rotmay, rotmaz

**Detailed Description :**

The following description is limited to general elliptical orbits. The length of the long radius is set to a, the length of the short radius is set to b, and the focal length is set to c. When an object forms an elliptical orbit around a focal point, the closest point to the focal point is defined as periapsis, and the farthest point is defined as apoapsis.

(1) semi-major axis[m]: corresponds to the orbital long radius and is the distance from the center of the elliptical orbit to periapsis.

(2) eccentricity: orbital eccentricity and defined as c/a. Or, if the distance from the focal point, which is the center of the elliptical orbit, to periapsis, is r\_p and the distance from focal point to apoapsis is defined as r\_a, it is also defined as .

(3) inclination[rad]: defined as the of the orbital plane vector , indicating how inclined the orbital plane is with respect to the kaxis of the ECI coordinate system.

(4) right ascension of ascending node [rad]: The angle formed by ascending node vector in the direction of during the intersection of the orbital plane and the IJ plane of the ECI coordinate system with Iaxis, and the value of is used behind the .

(5) argument of perigee [rad]: , defined as the angle formed by the ascending node vector and the eccentricity vector , uses the value of if it is .

(6) mean anomaly [rad] : defined as , where **E is eccentricity anomaly**.

도표, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 1 Vallado, D. A. (2013). Fundamentals of Astrodynamics and Applications (4th ed.). Hawthorne, CA: Microcosm Press. (Figure 2-2).

**E** is the intersection of a straight line perpendicular to the major axis of the ellipse at the point where the object is currently in orbit, and a circle with the same center as the major radius of the ellipse meets, and is located in the same quadrant as the point on the ellipse. It can be expressed through (: true anomaly), which means the angle of with respect to .

**line 36** : orb2iner : transform the state vector on the PQW coordinate system into the state vector on the ECI coordinate system.

The PQW coordinate system is a coordinate system in which the origin is the focal point of the elliptical motion, the preferred direction , and the fundamental plane corresponding to the orbital plane.

**line 40** : E = kepler(kepel(6), exc); Description below

**line 48** : statvec = [[a\*(cE - exc), a\*c1\*sE, 0]\*orb2iner', [-c3\*sE, c1\*c3\*cE, 0]\*orb2iner'];

The above line converts the state vector and on the PQW coordinates into the state vector on the ECI coordinate system through the rotational transformation of line 36.

Finally, the state vector of the ECI coordinate system are obtained.

**function [eccentric] =** **kepler (mean\_anomaly, eccentricity)**

**Input :** mean anomaly[rad], eccentricity

**Output :** eccentricity anomaly[rad]

**Detailed Description :**

Function to calculate Eccentricity anomaly E through mean\_anomaly M and eccentricity

The initial guess is

which corresponds to the approximate equation of E applied when .

At this time, the search for E is terminated when the error is less than 1e-12 or the number of repetitions is 10 using the Newton-Raphson method to find a value where the function f(E)=E-esin(E)-M representing the error of E becomes 0.

**function [diju] =** **djm (day, month, year)**

**Input :** day(day of month), month(month of the year), year

**Output :** Modified Julian Date in days, referred to 1950.0.

**Detailed Description :**

This function calculates (**Modified Julian Date, MJD**) for a given date (day, month, year). The **Julian date** refers to the number of days between Jan. 1, 4713 B.C. 12:00 and the current time. Modified Julian Date is defined as follows.

The epoch used in this code is B1950.0 with a value of

**line 20** : Days(per year) + current days - epoch correction constant + Days(per month) - leap year correction

**function [dayf] =** **time\_to\_dayf (hours, minutes, seconds)**

**Input :** hours[integer], minutes[integer], seconds

**Output :** seconds elapsed in the day(UTC)

**function [kepel] =** **statvec\_kepel (statv)**

**Input :** ECI coordinates state vector

**Output :** semimajor axis[m], eccentricity, inclination[rad], right ascension of ascending node[rad], argument of perigee[rad], mean anomaly[rad]

**Detailed Description :**

**line 33** : Earth gravity : product of Gravitational Constant of Earth (m^3/kg/s^2) and Mass of Earth (kg).

**line 39** : **ainv** stands for reciprocal of the semimajor axis and is derived from the vis-viva equation

.

**line 41~** : **h** means the orbital plane vector and **hm** means scalar of **h**.

(1) inclination[rad]: defined as the of the orbital plane vector

(2) right ascension of ascending node [rad]:

The angle

formed by ascending node vector

and the value of is used behind the .

(3) eccentricity: orbital eccentricity and defined as c/a

The trajectory equation of 2body problem :

By geometry of Kepler’s equation,

(4) eccentricity anomaly E[rad] : E is the intersection of a straight line perpendicular to the major axis of the ellipse at the point where the object is currently in orbit, and a circle with the same center as the major radius of the ellipse meets, and is located in the same quadrant as the point on the ellipse.

(**Related function : kepel\_statvec.m)**

(5) mean anomaly M[rad] : defined as , where E is eccentricity anomaly.

(6) Argument of perigee arpe[rad] : , defined as the angle formed by the ascending node vector and the eccentricity vector , uses the value of if it is .

* ev: Eccentricity vector → unit vector of eccentricity vector
* abev : Magnitude of eccentricity vector
* an : Unit vector of ascending node vector
* fi : condition check
* arpe : Argument of perigee[rad],

**Related function : kepel\_statvec.m**

**function [deltakep] =** **delkep (kep\_el)**

**Input :** semimajor axis, eccentricity, inclination, right ascension of ascending node, argument of perigee, mean anomaly

**Output :** Change rate of kepler elements due to J2, J4 perturbation (change rate of a, e, i are 0)

**Internal functions** :

**Detailed Description :**

The change rate in kepler elements due to J2 and J4 perturbations is obtained.

Since the Earth is not a complete sphere, there is an imbalance in the gravitational field depending on its position. Typically, the radius near the equator becomes larger than the radius of the polar regions due to the rotation of the Earth, and perturbations mean perturbations indicating this imbalance.

: The Earth's Aspherical Gravitational Potential according to the satellite location is described as follows.

where

And is defined as . Where  **is degree, is order**.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

Figure 2 Vallado, D. A. (2013). Fundamentals of Astrodynamics and Applications (4th ed.). Hawthorne, CA: Microcosm Press. (Figure 8-4).

This code shows the changes in Kepler elements due to J2 and J4 perturbations, which are mainly affected near i=0. In the case of a, e, and i, the rate of change is treated as zero because there is a short-term periodic change due to the perturbation, but there is **no secular change**.

**Related function : delkep.m**

**function [] =** **egm\_read\_data(egm\_data\_file, nmax)**

**Input :** egm\_96.dat, egm\_10.dat EGM data file(), nmax(order restriction)

**Output :** egm\_order egm\_length egm\_conv\_f egm\_cc egm\_sc, egm\_ae egm\_gm egm\_pn egm\_qn egm\_ip egm\_nmax (global variable)

**Internal functions** :

**Detailed Description :** This function reads the data from the selected EGM data file to the specified maximum polynomial order **(nmax)** and stores it in the global variable.

Therefore, it reflects the gravitational field from Earth's Aspherical Gravitational Potential to (**nmax = *l***) order.

(**Related function : delkep.m)**

The arrays stored here are as follows.

* egm\_cc: Cosine coefficient array (Cnm)
* egm\_sc: Sine coefficient array (Snm)
* egm\_ae: radius of the earth (meters)
* egm\_gm: Earth's gravitational constant (m^3/s^2).
* egm\_pn, egm\_qn, egm\_ip: additional variables for calculation, which can be used in the modeling process.
* egm\_nmax: User specified maximum order, may be less than file maximum order.
* egm\_order: The maximum number read from the EGM file
* egm\_length: The length of the calculated record
* egm\_conv\_f: the transformation factor of the coefficient

The EGM file lists the following values.

**Related function : delkep.m**

**function [rmx\_i\_o] =** **orbital\_to\_inertial\_matrix(kepel)**

**Input :** semimajor axis[m], eccentricity, inclination[rad], right ascension of ascending node[rad], argument of perigee[rad], mean anomaly[rad]

**Output :** rotation matrix from orbital to inertial frame

**Internal functions** : rotmax, rotmaz, kepler

**Detailed Description :**

This function creates a matrix that transforms the state in the **Orbital frame** to the state in the Inertial frame (**ECI coordinates**) through Orbital elements.

**Orbital frame** : The center of the earth is the orgin, and the **x-axis: along the radius vector, from Earth center to satellite**, positive up, y-axis : cross product of the z-axis direction and the x-axis direction, and z-axis : normal to the orbital plane, along the orbital angular momentum.

The orbital frame is a coordinate system **rotated by true anomaly with respect to the z-axis in the PQW coordinate system**. Therefore, a rotation matrix is added in the z-axis direction to orb2iner, which is a transformation matrix from the PQW coordinate system to the ECI coordinate system.

orb2iner : transform the state vector on the PQW coordinate system into the state vector on the ECI coordinate system.

By Kepler’s 1st law and geometry of Kepler’s equation,

**rmx\_i\_o = orb2iner\***, where is rotmaz.

**Related function : kepler.m, statvec\_kepel.m, kepel\_statvec.m**

**function dxdt =** **egm\_difeq (time, x, flag, mjd, dsec, ext\_acc)**

**Input :** Propagation time[sec], ECI coordinates state vector , ODE45 options, Modified Julian Date in days, referred to 1950.0.(mjd), Day time UTC in seconds of initial condition(dsec), external force excluding gravity(disturbance acceleration + control acceleration)

**Output :** Time derivative of the state vector in ECI ,

**Internal functions** : egm\_acc, gst, inertial\_to\_terrestrial, terrestrial\_to\_inertial

**Detailed Description :**

This function creates the derivative of position and velocity to generate the odefun used by the ode45. When this function obtains the acceleration vector, it goes through the geocentric terrestrial coordinates transformation process because the current rotation angle is considered to take into account the Earth's aspherical potential.

1. Extract the velocity part of the inertial coordinate system reference state vector.
2. Calculate the Greenwich Mean Sidereal Time in **gst.m** with Modified Julian date and current time (UTC time + propagation time).
3. Use **inertial\_to\_terrestrial.m** to obtain the state vector in Geocentric Terrestrial Coordinates .
4. Use **egm\_acc.m** to obtain the gravitational acceleration in Geocentric Terrestrial Coordinates
5. Use **terrestrial\_to\_inertial.m** to convert the given gravitational acceleration to gravitational acceleration(ai) in the inertial frame.
6. Add the gravitational acceleration ai to the external acceleration ext\_acc to obtain the acceleration vector.
7. Combine the velocity vector and acceleration vector to obtain .

**Related function : kepel\_statvec.m, time\_to\_dayf.m, djm.m**

**function ac =** **egm\_acc (x)**

**Input :** Position vector in Geocentric Terrestrial Coordinates[m], egm\_order egm\_length egm\_conv\_f egm\_cc egm\_sc, egm\_ae egm\_gm egm\_pn egm\_qn egm\_ip egm\_nmax (global variable)

**Output :** Acceleration vector in Geocentric Terrestrial Coordinates[]

**Detailed Description :**

This function computes the acceleration due to Earth's gravity in geocentric terrestrial coordinates using the Lagrange expansion of Earth's gravity potential.

1. **Auxiliary Variables Initialization**

* r: Magnitude of the position vector.
* q: Ratio of Earth's mean radius to the position vector magnitude.
* t: Sine of the latitude (z-component of the position vector divided by its magnitude).
* u: Cosine of the latitude.
* tf: Tangent of the latitude.
* sc: Magnitude of the x and y components of the position vector.
* sl, cl: Sine and cosine of the longitude, respectively.
* gmr: Earth's gravitational constant divided by the position vector magnitude.

1. **Summation, Sectoral Terms,** **Sin and Cosine Recursions Initialization**
2. **Outer Loop (over m:order)**

* Compute the sectoral terms **pnm(,**legendre polynomial**)** and dpmn(derivative)

1. **Inner Loop (over n:degree)**

* Compute the Compute the recursion coefficients **anm, bnm, fnm**

1. **Update summation variables & Sine and Cosine Recursions**
2. **Compute gradients of summation variables & acceleration vector**

* vl: Component of acceleration in the longitude (east-west) direction.
* vf: Component of acceleration in the latitude (north-south) direction.
* vr: Component of acceleration in the radial (towards/away from the center of the Earth) direction.

It transforms vl, vf, and vr into gradients of the Earth's potential and obtains the acceleration vector in the Geocentric Terrestrial Coordinates through the equation for ac.

**Related function : delkep.m**

**function [gwst] =** **gst (diju, time)**

**Input :** Modified Julian Date in days, referred to 1950.0., UT1 time complement in seconds

**Output :** Greenwich aparent sidereal time in radians

**Detailed Description :**

This function calculates the Greenwich Apparent Mean Sidereal Time (GAST) referenced to the J2000.0 equator and equinox.

**Related function : djm.m**

**function [xterrestrial] =** **inertial\_to\_terrestrial (tesig, xi)**

**Input :** Greenwich sidereal time in radians, referred to 1950.0., ECI coordinates state vector

**Output :** State vector in Geocentric Terrestrial Coordinates

**Internal functions** : rotmaz

**Detailed Description :**

This function transforms a geocentric inertial state vector into geocentric terrestrial coordinates. It turns the position vector and the velocity vector through the rotmaz matrix by the (Greenwich sidereal time) with respect to the z-axis.

**Related function : gst.m, terrestrial\_to\_inertial.m**

**function [xinert] =** **terrestrial\_to\_inertial (tesig, xt)**

**Input :** Greenwich sidereal time in radians, referred to 1950.0., State vector in Geocentric Terrestrial Coordinates

**Output :** ECI coordinates state vector

**Internal functions** : rotmaz

**Detailed Description :**

This function transforms a geocentric terrestrial state vector into geocentric inertial coordinates. It turns the position vector and the velocity vector through the rotmaz matrix by the (-Greenwich sidereal time) with respect to the z-axis.

**Related function : gst.m, inertial\_to\_terrestrial.m**

**function [rot\_mat] =** **rotmaz (angle)**

**Input :** rotation angle[radian]

**Output :** z-axis roation matrix(about given angle)

**Detailed Description :**

This function produces a rotation matrix that rotates by a given angle relative to the z-axis of the currently set coordinate system.

**function [angle] = proxi****mus (angleinp, angleprox)**

**Input :** angleinp(The input angle in radians that needs to be adjusted.), angleprox(The reference angle in radians to which angleinp should be made as close as possible.)

**Output :** The adjusted angle[rad], equivalent to angleinp but close to angleprox.

**Internal functions** :

**Detailed Description :**

In the calculation, it converts kp\_nm(6)(angleinf) to the value closest to kp\_an(6)(angleprox) because kp\_nm or kp\_an can have values above 2pi. For example, if kp\_nm = 2pi - 0.1 [rad], kp\_an = 0.1 [rad], we convert kp\_nm to 0.1 [rad] closest to 0.1.

라인, 도표, 그래프, 폰트이(가) 표시된 사진

자동 생성된 설명

That is, the radian value is converted so that kp\_nm is 30 degrees different from kp\_an in the following situation.

**Example1** **demo\_egm**

Consider the analytic solution of orbital propagation reflecting J2 and J4 perturbations and the numerical solution reflecting the Earth's Aspherical Gravitational Potential up to the nmax order. This problem numerically calculates the value that the following 6 kepler elements change over time by aspherical perturbation in propagation for 6000 sec at a given state. And compare the numerical propagation value with the analytical solution calculated using the current state.

**Initial time** : 2017/07/17, 23hrs:00min:00sec

**Initial condition**: kepler elements(7000000m, 0.01, 98, 0, 35, 0)

The description of each element is as follows.

**Kepler elements**

(1) semi-major axis[m]: corresponds to the orbital long radius and is the distance from the center of the elliptical orbit to periapsis.

(2) eccentricity: orbital eccentricity and defined as c/a. Or, if the distance from the focal point, which is the center of the elliptical orbit, to periapsis, is r\_p and the distance from focal point to apoapsis is defined as r\_a, it is also defined as .

(3) inclination[rad]: defined as the of the orbital plane vector , indicating how inclined the orbital plane is with respect to the kaxis of the ECI coordinate system.

(4) right ascension of ascending node [rad]: The angle formed by ascending node vector in the direction of during the intersection of the orbital plane and the IJ plane of the ECI coordinate system with Iaxis, and the value of is used behind the .

(5) argument of perigee [rad]: , defined as the angle formed by the ascending node vector and the eccentricity vector , uses the value of if it is .

(6) mean anomaly [rad] : defined as , where E is **eccentricity anomaly**.

도표, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 3 Vallado, D. A. (2013). Fundamentals of Astrodynamics and Applications (4th ed.). Hawthorne, CA: Microcosm Press. (Figure 2-2).

**E** is the intersection of a straight line perpendicular to the major axis of the ellipse at the point where the object is currently in orbit, and a circle with the same center as the major radius of the ellipse meets, and is located in the same quadrant as the point on the ellipse. It can be expressed through (: true anomaly), which means the angle of with respect to .

The initial kepler elements is changed to state vector **stat**: in the inertial coordinate system via [**kepel\_statvec.m**](#kepel_statvec).

The rate of change of kepler elements over time is calculated through the [**delkep.m**](#delkep).

Change the current date and time to **Modified Julian Date** in days and **UTC** using [**djm.m**](#djm) and [**time\_to\_dayf.m**](#time_to_dayf). And set the propagation time at 1-second intervals from 0 to 6000 seconds.

Select the egm data file(Earth's Aspherical Gravitational Potential Coefficient list) through the [**egm\_read\_data.m**](#egm_read_data) function and set the nmax(order restriction) to select up to which order to reflect. Here, the egm\_10.dat file was used and set to nmax=2.

Next, the ODE solver option was set to odeset ('abstol', 1e-4, 'reltol', 1e-4).

And the size of the following variables is set according to the size of the time interval (n = fix(tend/tstep)). r\_time, r\_sma(semimajor aixs), r\_ecc(eccentricity), r\_inc(inclintion), r\_raan(right ascension of ascending node), r\_par(argument of perigee), r\_ma(mean anomaly), r\_dist(distance from the Earth center)

Additionally, r\_rx and r\_ry are x and y coordinates on a plane coordinate system with the origin as the center of the Earth and the x-axis in the perigee direction and the y-axis in the direction perpendicular to the x-axis. In figure 3, it can be seen that the x value at this time is , and the y value is . And r\_rx and r\_ry also match the size of the time interval.

In the case of r\_xo and r\_vo, they correspond to the position vector and the velocity vector, respectively, so they create a matrix consisting of three rows.

The following performs orbit propagation through **ic** indexing. This loop iterates from **tstart** to **tend** with a step size of **tstep**.

**Orbit propagation loop**

1) kp\_an : kepel(initial kepler elements) + delk\*t(Integrate the rate of change of Keplerian elements with respect to time, delk here means the rate of change of kepler elements based on the starting time)

2) sv\_an : Converts the kepler elements at the current time to the state vector in the ECI via [**kepel\_statvec.m**](#kepel_statvec)**.**

3) **xi\_an** : ECI position vector

**vi\_an** = ECI velocity vector

4) **c\_i\_o** : Calculates the rotation matrix to convert from the [**orbital frame**](#Orbital_frame) to the **inertial frame(ECI)** via [**orbital\_to\_inertial\_matrix.m**](#orbital_to_inertial_matrix).

5) tspan : Defines the time range for the ODE solver.

ext\_acc : Total external acceleration by adding disturbance(such as air drag) and control accelerations

6) In ode45, [**egm\_difeq.m**](#egm_difeq) is applied as an odefun (dynamic equation) and the stat is updated. In **egm\_difeq.m**, the list of earth's aspherical potential read from [**egm\_read\_data.m**](#egm_read_data) is reflected. Additional information to be entered into ode45 is **Modified Julian Date (mjd), UTC time (dfra), and external force (ext\_acc)** calculated previously.

7) sv\_nm : Propagated state vector, corresponding to the point at t+tstep of the ode 45.

**xi\_nm** : propagated inertial posititon vector

**vi\_nm** : propagated inertial velocity vector

stat : current state vector update

8)kp\_nm : Converts sv\_nm, a numerical propagated state vector, to kepler elements.

ea\_nm : Calculates the eccentricity anomaly from mean anomaly & eccentricity via [**kepler.m**](#kepler).( Refer to above description )

9) dist : Calculate distance[m] from earth

(In [figure 3](#Figure_3), , so . By Trajectory equation

If rearrange it for cos : .

Substitute into the Trajectory equation, )

10) Initialize cont\_acc (orbit control acceleration) and dist\_acc (disturbance specific forces) to zero. If you want to give a time-varying external acceleration, you need to change those terms.

11) **Store data**

**r\_time**[sec]: The current time value.

**r\_xo**[km]:

Convert the ECI coordinates position error obtained by subtracting **xi\_an** (*analytic solution obtained by expressing the Kepler element change rate by reflecting J2 and J4 permeations and multiplying that value by time*) from **xi\_nm** *(the position where the Aspherical Gravitational Potential coefficients were read and numerically propagated*) to the value in the [**orbital frame**](#Orbital_frame) through the c\_i\_o' matrix.

At this time, c\_i\_o is a rotating matrix, so is established.

**r\_vo**[m/s]: In the same way as r\_xo, after subtracting the analytically obtained vi\_an from the velocity obtained numerically from the ECI coordinate system, **vi\_nm**, it is converted into a velocity value in the [**orbital frame**](#Orbital_frame).

**r\_dist**[km]: The current distance from the center of the earth

**r\_rx**[km]: The x-coordinate on the orbital plane, as shown in [Figure 3](#Figure_3).

**r\_ry**[km]: The y-coordinate on the orbital plane, as shown in [Figure 3](#Figure_3).

Both r\_rx and r\_ry are calculated from numerical solution values.

**r\_sma**[km]: Indicates the difference between semi-major axis values of numerical and analytic solutions.

**r\_ecc**: Indicates the difference between eccentricity values of numerical and analytic solutions.

**r\_inc**[deg]: Indicates the difference between inclination values of numerical and analytic solutions.

**r\_raan**[deg]: Indicates the difference between right ascension of ascending node values of numerical and analytic solutions.

**r\_par**[deg]: Indicates the difference between argument of perigee values of numerical and analytic solutions.

**r\_ma**: Using the [**proximus.m**](#proximus) function, it converts mean anomaly of numerical solution to the closest value to mean anomaly of analytic solution. And Indicates the difference between argument of perigee values of numerical and analytic solutions.

ic: Increments the data index.

**Output visualization**

figure 1 : Plot the x, y, and z components of **r\_xo** with respect to time.

figure 2 : Plot the x, y, and z components of **r\_vo** with respect to time.

figure 3 : Plot the **r\_dist** with respect to time.

figure 4 : Plot the **r\_rx, r\_ry** 2 dimension trajectory.

figure 5 : Plot the **x components of r\_xo** with respect to **y components of r\_xo**.

figure 6 : Plot the **x components of r\_xo** with respect to **z components of r\_xo**.

figure 7 : Plot the **semi-major axis**[km] with respect to time.

figure 8 : Plot the **eccentricity** with respect to time.

figure 9 : Plot the **inclination**[rad] with respect to time.

figure 10 : Plot the **right ascension of ascending node**[rad] with respect to time.

figure 11 : Plot the **argument of perigee**[rad] with respect to time.

figure 12 : Plot the **mean anomaly**[rad] with respect to time.