MAE3145: RV2COE

Description

Your objective is to write a program which determines the classical orbital elements of an Earth-orbiting body given its position and velocity vectors in the Earth-Centered Inertial frame. It is also required to compute other orbital parameters including, radius of perigee, radius of apogee, specific mechanical energy and period.

Project Requirements

After completing the project you must submit the hard copies of your work:

- Complete algorithms for the main driver script as well as a seperate algorithm for each sub-function that you develop. Someone totally unfamiliar with astrodynamics should be able to duplicate your program in any computer language.
- Clear, concise and properly documented and tested code
- Correct outputs from your program which matches the test cases
- Any additional test cases you may have used. Explain why you did or did not use any additional test cases.

Authorized Resources

You may consult with your instructor, the course notes or other reference material, and other students. However, you **MAY NOT** copy another student or any other individuals code. The program you develop must be your own work.

Program Specifications

The following is a description of the inputs and outputs to your program:
INPUTS:

- $\bar{r} \in \mathbb{R}^3$ – the components of the position vector in the Earth Centered Inertial frame given in kilometers (km)
- $\bar{v} \in \mathbb{R}^3$ – the components of the velocity vector in the Earth Centered Inertial frame given in kilometers per second ($\mathrm{km\,s^{-1}}$)

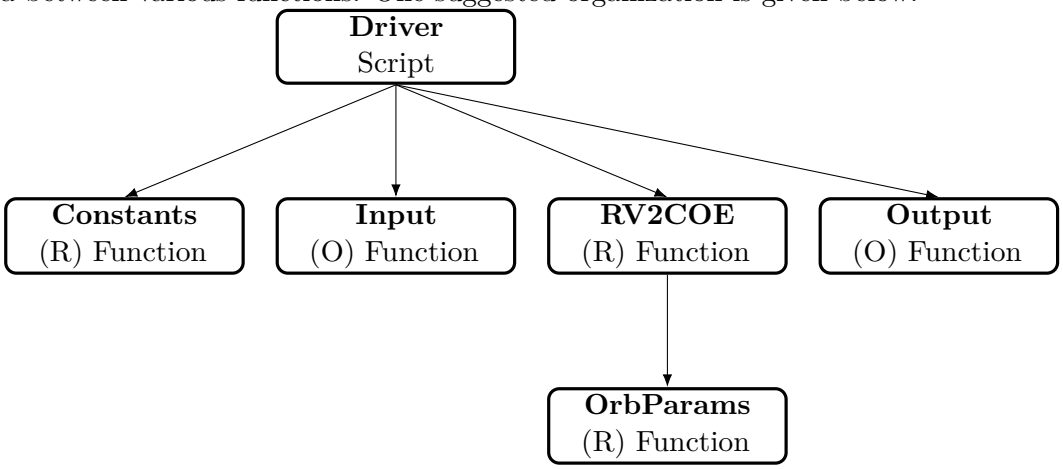
OUTPUTS:

- a – semimajor axis in kilometers (km)
- e – eccentricity (unitless)
- i – inclination in radians (rad)
- Ω – right ascension of the ascending node in radians (rad)
- ω – argument of perigee in radians (rad)
- ν – true anomaly in radians (rad)

Plus other outputs which are listed later in the description.

Code Organization

Your program should be modular in construction. This means you should have separate functions which perform **specific** tasks. Furthermore, each function should be tested and properly documented following the examples shown in class. You **MAY NOT** use global variables but rather must pass data between various functions. One suggested organization is given below:



Required Function Description

The following functions must be written and accept the given inputs and provide the required outputs. Each function should be well tested, with unit tests, and well-documented following the class examples.

`RV2COE`

This function determines an objects six classical orbital elements from its position and velocity vectors.

Input:

- $\bar{r} \in \mathbb{R}^3$ – the components of the position vector in the Earth Centered Inertial frame given in kilometers (km)
- $\bar{v} \in \mathbb{R}^3$ – the components of the velocity vector in the Earth Centered Inertial frame given in kilometers per second ($\mathrm{km\,s^{-1}}$)

OUTPUTS:

- a – semimajor axis in kilometers (km)
- e – eccentricity (unitless)
- i – inclination in radians (rad)
- Ω – right ascension of the ascending node in radians (rad)
- ω – argument of perigee in radians (rad)
- ν – true anomaly in radians (rad)

Optional Functions

The following functions may be useful in the development of your program. It can allow you to further organize the code, but it is not required.

`hnevec`

This function determines the angular momentum, ascending node, and eccentricity vectors given the position and velocity vectors.
Input:

- $\bar{r} \in \mathbb{R}^3$ – the components of the position vector in the Earth Centered Inertial frame given in kilometers (km)
- $\bar{v} \in \mathbb{R}^3$ – the components of the velocity vector in the Earth Centered Inertial frame given in kilometers per second ($\mathrm{km\,s^{-1}}$)

OUTPUTS:

- $\bar{h} \in \mathbb{R}^3$ – the components of specific angular momentum vector in the Earth Centered Inertial frame
- $\bar{n} \in \mathbb{R}^3$ – the components of the ascending node vector in the Earth Centered Inertial frame
- $\bar{e} \in \mathbb{R}^3$ – the components of the eccentricity vector in the Earth Centered Inertial frame

`angles`

This function determines all of the angular orbital elements. Ensure that you perform quadrant checks on all of the angles.
Input:

- $\bar{r} \in \mathbb{R}^3$ – the components of the position vector in the Earth Centered Inertial frame given in kilometers (km)
- $\bar{v} \in \mathbb{R}^3$ – the components of the velocity vector in the Earth Centered Inertial frame
- $\hat{h} \in \mathbb{R}^3$ – the components of specific angular momentum vector in the Earth Centered Inertial frame
- $\hat{n} \in \mathbb{R}^3$ – the components of the ascending node vector in the Earth Centered Inertial frame
- $\hat{e} \in \mathbb{R}^3$ – the components of the eccentricity vector in the Earth Centered Inertial frame

OUTPUTS:

- i – inclination in radians (rad)
- Ω – right ascension of the ascending node in radians (rad)
- ω – argument of perigee in radians (rad)
- ν – true anomaly in radians (rad)

`orbitalparameters`

This function determines other useful orbital parameters associated with the orbit.
Input:

- a – semimajor axis in kilometers (km)
- e – eccentricity (unitless)

Outputs:

- r_p – radius of perigee in kilometer (km)
- r_a – radius of apogee in kilometer (km)
- sme – specific mechanical energy in kilometer (km)
- P – period of the orbit in seconds (s)

Input and Output

You may consider creating separate functions which can read and write to various text files. These are encouraged but are optional.

Test Data

Two test cases are provided with this project. There is a text file named `RV1.txt` which contains the position and velocity vectors of two spacecraft. The data is copied below.

```
8840.0  646.0 5455.0  -0.695  5.25  -1.65
-3084.7  30.0  6911.0   5.66  -4.07   3.84
```

The numbers on each line corresponds to the three elements of the position vector and the three elements of the velocity vector, i.e. $\begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}$. The solution for Case 1, the first line/spacecraft, is given below.

```
R (km)    =  8840.0000 I    646.0000 J   5455.0000 K    Mag= 10407.6866
V (km/s)  =    -0.6950 I      5.2500 J     -1.6500 K    Mag=     5.5469

Radius of Perigee           (km) =  6260.5311
Radius of Apogee            (km) = 11134.4744
Energy              (km^2/sec^2) =   -22.9147
Period                   (hours) =     2.2423
Semimajor Axis              (km) =  8697.5027
Eccentricity                     =     0.2802

Inclination                           (deg) =    33.9987
Right Ascension of the Ascending Node (deg) =   250.0287
Argument of Perigee                   (deg) =   255.5372
True Anomaly                          (deg) =   214.8548
```

Using your computer program, you should read the vectors, compute the appropriate values and write them to a seperate text file. Do not hardcode the data into your program, but rather read them from the file. While only two cases are given here, your program should be able to convert an arbitrary amount of position and velocity vectors. Within the `MAE3145_library` there is a file which contains the expected solutions associated with `RV1.txt`. Additional information is provided in this file which can aid in your debugging and testing process.

Grading

Finally, there is a file `RV2COE_tle_rv.txt` which contains a large list of position and velocity vectors of several real spacecraft. In addition, you'll notice that the values are provided with additional precision as compared to `RV1.txt`. Furthermore, beyond simply arriving at the correct solution, a well developed algorithm and properly documented/tested code is the main goal of this project. Your project grade will consist of the following:

- 50% – complete algorithm demonstrating the process used to compute the various orbital elements. Your grade will be determined on the clarity of your writing and the algorithm presented. Remember, a person without any astrodynamics background should be able to duplicate your program in any language of their choice.
- 25% – properly documented and tested code. Your functions should be properly documented and each should have sufficient unit-testing to ensure the validity of the outputs. You will be graded on the coverage, i.e. does every function have a test, and your documentation, is it easy to use your function based on the documentation.
- 10% – provide output from your program which matches the provided test cases.
- 15% – able to match a random output from `RV2COE_tle_rv.txt`