MAE3145: PROPAGATE

Description

The objective for this project is to write a computer program which first determines the classical orbital elements of an Earth orbiting body given its current position and velocity vectors. Using these orbital elements your program must then compute the future classical orbital elements given a time of flight using a function called `Update`, which uses the restricted two-body assumption. Finally, your program should output these results to a text file for later printing/analysis.

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 separate 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 code. The program you develop must be your own work.

Program Specifications

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

- $\bar{r}_0 \in \mathbb{R}^3$ – the components of the position vector in the Earth Centered Inertial frame given in kilometers (km)
- $\bar{v}_0 \in \mathbb{R}^3$ – the components of the velocity vector in the Earth Centered Inertial frame given in kilometers per second ($\mathrm{km\,s^{-1}}$)
- $\Delta t \in \mathbb{R}^1$ – the time of flight of the spacecraft in minutes (min)

OUTPUTS: The following orbital elements are at the final time $t_0 + \Delta t$.

- 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)

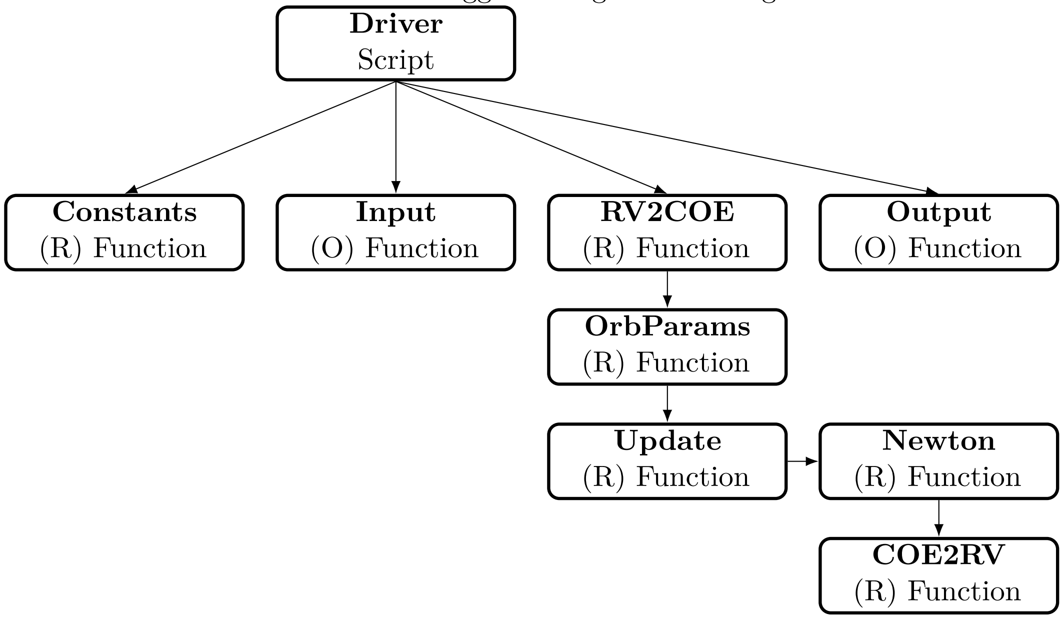
In addition, your program should also compute and output the future position and velocity vectors in the inertial frame at time $t_0 + \Delta t$.

- $\bar{r}_f \in \mathbb{R}^3$ – the components of the position vector in the Earth Centered Inertial frame at the final time in kilometers.
- $\bar{v}_f \in \mathbb{R}^3$ – the components of the velocity vector in the Earth Centered Inertial frame at the final time in kilometers per second.

Plus other outputs which are listed later in the description.

Code Organization

Your program should be modular in constructions. This means you 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:



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 example. Several functions are utilized from the previous project. Those descriptions are not duplicated here.

`Update`

This function "updates" the orbital elements given a time of flight using the restricted two-body assumptions. This function also calls `newton` which solves Kepler's problem using a Newton iteration scheme in order to solve for the future eccentric anomaly E_f.

Input:

- a_i – semimajor axis in kilometers (km)
- e_i – eccentricity (unitless)
- i_i – inclination in radians (rad)
- Ω_i – right ascension of the ascending node in radians (rad)
- ω_i – argument of perigee in radians (rad)
- ν_i – true anomaly in radians (rad)

OUTPUTS:

- a_i – semimajor axis in kilometers (km)
- e_i – eccentricity (unitless)
- i_i – inclination in radians (rad)
- Ω_i – right ascension of the ascending node in radians (rad)
- ω_i – argument of perigee in radians (rad)
- ν_i – true anomaly in radians (rad)

`newton`

This function uses an iterative method to solve Kepler's problem.
Input:

- M – mean anomaly in radians (rad)
- e – eccentricity of orbit

Output:

- E – eccentric anomaly in radians (rad)
- ν – true anomaly in radians (rad)

`COE2RV`

This function is the inverse of our previous `RV2COE`. It will convert the classical orbital elements to the equivalent position and velocity vector in the inertial frame.
Input:

- 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)
- μ – gravitational parameter of attracting body ($\mathrm{km}^3\,\mathrm{s}^{-2}$)

Output:

- $\bar{r} \in \mathbb{R}^3$ – 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 ($\mathrm{km}\,\mathrm{s}^{-1}$)

Test Data

Two test cases are provided with this project. There is a text file named `RV2.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 0.0
-3084.7   30.0  6911.0   5.66  -4.07   3.84 0.0
```

The numbers on each line corresponds to the three elements of the position vector and the three elements of the velocity vector followed by the time of flight (seconds), i.e. $\begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \Delta t \end{bmatrix}$. The solution for Case 1, the first line/spacecraft, is also given in the course software library as `RV2_solution.txt` The cases in `RV2.txt` have a time of flight of 0 s. In addition, since the time of flight is zero, the initial and final orbital elements should be identical.

Using your computer program, you should read the vectors, compute the appropriate values and write them to a separate text file. Do not hard code 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 and time of flights.

After verifying both cases shown in `RV2.txt`, modify the time of flight to one orbital period. This should provide another test with the final orbital elements equal to the initial orbital elements. You are encouraged to create your own test cases to further verify you code. For example, you could test your code using a time of flight of $\frac{1}{4}$ orbital period, which should cause a change of the true anomaly by $90°$.

Finally, your code should be able to handle any arbitrary number of position/velocity vectors; there is an additional input file `PROPAGATE_tle_rvt.txt` that your program should be able to process.

Grading

Finally, there is a file `PROPAGATE_tle_rvt.txt` which contains a large list of position and velocity vectors of several real spacecraft. 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 `PROPAGATE_tle_rvt.txt`