MAE3145: COMFIX

Description

Your first mission as an orbital analyst has reached your capable hands. Your task is to develop a program to convert radar observation data into Classical Orbital Elements (COEs).

Each radar observation includes the tracking site geodetic latitude, longitude, and its altitude above sea level. The sites are equipped to measure vehicle range, pointing angles, Doppler shift, angular rates and record the observations in Universal Time (UTC). The tracking data is reported as:

- **LINE 1**
  - Site latitude (North is positive and South is Negative) in degrees
  - Site longitude (East is positive and West is Negative) in degrees
  - Site altitude in meters
  - Observation time in Julian Date
- **LINE 2**
  - Satellite ID number - 4 digit integer
  - Vehicle range in kilometers
  - Vehicle azimuth in degrees
  - Vehicle elevation in degrees
  - Range rate in kilometers per second
  - Azimuth rate in degrees per second
  - Elevation rate in degrees per second

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 given in `comfix.dat` and `comfix_solution.txt`
- Correct outputs which match the input data given in `COMFIX_tle_measurement.txt`

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.

Algorithm

Write a structured algorithm that shows your approach to writing a computer program to perform all of the tasks described above. This should be a complete **sequential** list of the **equations and logic (including loop)** that you will use to write your program. The details of sub-algorithms are only required for procedures that are new for this project, not those provided to you or from a previous project. Instead, just mention what procedure will be used and the inputs and outputs of the procedure, e.g. "Calculate orbital elements given position and velocity vectors using `rv2coe`".

When completed, anyone should be able to write your code **solely** using the algorithm in **any** computer language of their choice. Thus, define all symbols before you use them, and do not write the equations and logic using any language specific terminology, i.e. Something like " Find the length of the vector using `norm(x)` " is unacceptable.

Your algorithm must be **typed**, which will serve you well when you document your final code. This is also a good opportunity to practice your technical writing skills in LaTeX.

Final COMFIX Deliverables

Your program must process the `COMFIX.DAT` data file and generate output that matches the orbital elements in `COMFIX.OUT` to at least four decimal places. The input file, `COMFIX.DAT`, contains five observations, but your program should be able to process an arbitrary number of observations.

Submit the following on the due date:

- Fully documented driver script
- Each procedure which you wrote or modified, fully documented (no library routines)
- Computer generated results which match `COMFIX.OUT`

Software Library

A library of support functions are provided for your use. This library gives you a model for the structure of your project as well as several functions you can use in the development of your own programs. Furthermore, the library shows the standards used in this class for:

- Proper documentation
- Proper testing

Program Specifications

The following is a description of the required functions that your software program must include. You will be making use of many of the functions that you have been developing over the course of the semester.

`lla2ecef`

This function transforms a ground station location to a vector in the Earth Centered Earth Fixed reference frame.

Inputs:

- ϕ - site geodetic latitude in radians

- λ - site longitude in radians
- h - site altitude above mean sea level in kilometers

Output:

- $\vec{r}_{ecef} \in \mathbb{R}^{3\times1}$ - site position vector in the Earth centered Earth Fixed reference frame and given in kilometers

`ecef2eci`

This function will return the rotation matrix to transform a vector from the Earth Centered Earth fixed frame to the Earth Centered Inertial frame.
Inputs:

- JD - Julian Date for transformation

Outputs:

- R – rotation matrix from ECEF to ECI

This function will make use of `gstlst`.

`rvtopos`

This function converts radar measurements to a range and range rate vector in the topocentric reference frame.
Input :

- ρ - range from the site to spacecraft in kilometers
- α - azimuth in topocentric reference frame in radians
- β - elevation in topocentric reference frame in radians
- $\dot{\rho}$ - range rate in kilometers per second
- $\dot{\alpha}$ - azimuth rate in radians per second
- $\dot{\beta}$ - elevation rate in radians per second

Output:

- $\vec{r}_{sez}$ - position of spacecraft in topocentric reference frame in kilometers
- $\vec{v}_{sez}$ - velocity of spacecraft in the topocentric reference frame in kilometers per second

`sez2ecef`

This function will transform a vector in the topocentric reference frame to a vector in the Earth Centered Earth Fixed reference frame.
Inputs:

- ϕ - site geodetic latitude in radians
- λ - site longitude in radians
- h - site altitude above mean sea level in kilometers

Outputs:

- R - rotation matrix from the topocentric to ECEF reference frame