Kevin Dennis
U07433850
IDH 4930

Orbital Drift

Description

This project will graph the orbits of the planets in the solar system around the sun. In the non-relativistic Kepler problem, planets follow their orbit eternally. However, if the gravitational forces of the other planets on a given body are taken into account, the bodies orbit begins to slowly precess around the sun. This perihelion precession can be approximated. This project shows the difference in coordinates one can receive based on three different methods of calculation.

Motivation

In my Natural Sciences Honors class, our professor tasked us with planning a “trip” to the surface of Mars and the outer orbit of Saturn. The group I was in was assigned the task of planning the launch, including the day and how the shuttle would get into orbit. I slowly became obsessed with plotting the orbits of the planet and the launch path of the shuttle. Many long nights of attempting to graph these orbits followed, and ended in failure. This project is an attempt to tackle this problem again.

User Interface

The Orbital Drift program is used with the command line. Running the program will open a plotly local html file in the user’s default web browser, which can then be manipulated. The user can provide command line arguments to specify the behavior of the plot. If no options are listed, the program will print an example. Command line options include:

Option	Purpose	Example
-n --nosun	Don’t show the sun	
-g --galileo	Orbits will be plotted using geocentric coordinates	
-c --center	Similar to -g, except it allows the user to define the center planet. Overwrites -g.	-c Venus --center v
-ng --nograph	Does not output the graph	
-fg --forcegraph	Always graphs the planets, regardless of the number of points. (The recommended max is 25000 points)	
-nd --nodiff	Does not output the difference between the different methods. Overwrites -o	
-h --horiz	Includes the Horizon data as coordinates for each planet. This does NOT graph them. See -gh	
-o --output	The differences between methods will printed to the provided file, not stdout.	-o output.txt --output test.txt

-gh --graphhorizon	Graphs Horizon data for each planet. This overwrites the -h option.	
-s --schlyter	Any planets entered after this option will be graphed using the Schlyter Method. Note that this is the default method.	-s Earth -vs Earth -s Mars
-vs --vsop87	Any planets entered after this option will be graphed using VSOP87-C.	-vs Earth Mars -vs Earth Mars -s Venus
-t --mastertimer	Prints to stdout the time the entire program took to run.	
-d --date	Specify the date to start tracking	-d yyyy-mm-dd
-e --dateend	Specify the date to end tracking	-e yyyy-mm-dd

Anything else provided to the program will be interpreted as a planet. To simplify plotting multiple planets, the user can provide abbreviations for the planets. For example, e E Ea and earth would all be interpreted correctly. The program uses a WHERE LIKE search in the database to find the correct planet, and thus any correct letter combination can be provided. Note that Mercury and Mars both begin with M; using m or M as a planet may return either of these options, based on how the database runs it search at that time.

Other Requirements

- Data Source

  For the method devised by Schlyter on his website, approximate elements for each planet are used. These elements were taken from his website, and stored in the elements.csv and schlyter_terms.csv files.

  The more accurate VSOP87-C method uses several thousand terms, comprised of 3 elements, to calculate the points. X, Y, and Z each have their own series. The terms were retrieved from the VSOP87 FTP site. These are contained in the VSOP87_C folder.

  The planetary coordinates for NASA's Horizon service are directly requested through Horizon's Telnet service using the Python telnetlib module, and thus does not need any data stored or input for calculation.
- Data Formatting

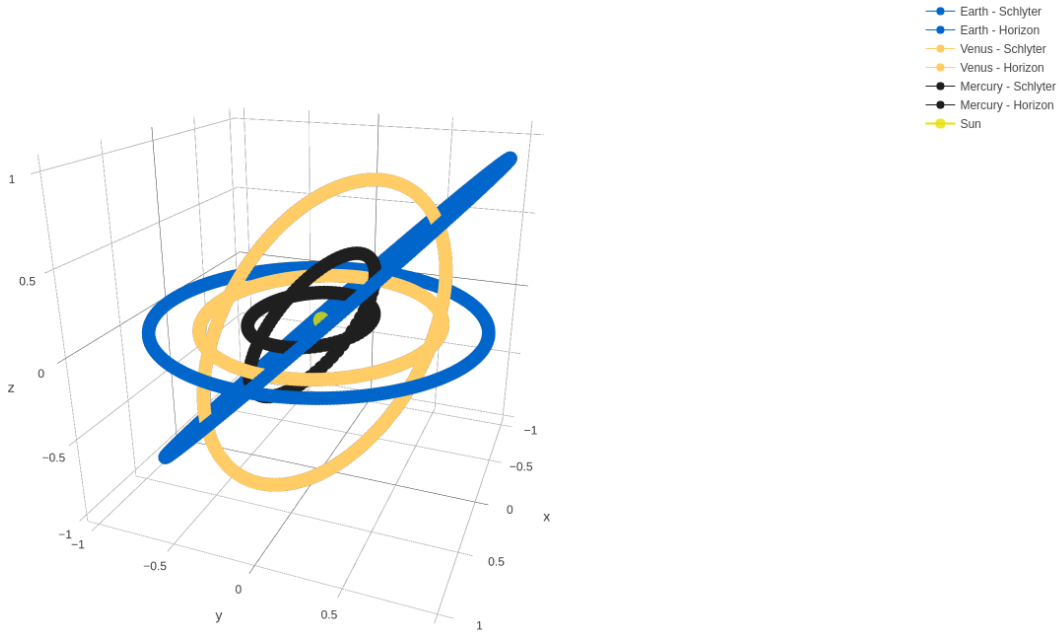
  The data will be stored in a simple, comma-separated (or space delimited for VSOP87) list. These files could be edited to allow the plotting of other celestial bodies, such as asteroids, comets, or even a satellite, with only a few minor tweaks to the database script, and some knowledge of the file layout. A future enhancement I had in mind was allowing users to run the database script, which would prompt them with deleting, inserting, or updating new celestial bodies.
- File I/O

  There will be no file input for the project (besides manually editing the text file of planet characteristics or terms), but the program will output a html file that displays the plotted orbits. The html file can be opened and interacted with using any browser that supports javascript, and saved as a png for later use.

Visualization

The Orbital Drift project does not have visual components such as a GUI, but the plotly output does have a simple graphical interface. When the plotly graph is opened, the user is presented with a 3D graph of the planet orbits in the middle of the window. In the top right corner there are several

different options allowing the user to manipulate the image, such as a pan, zoom, and save feature. Below is a snapshot of the plotly graph for the Earth, Venus, and Mars orbiting around the Sun, using the Schlyter method and data from Horizons. (./orbital_drift.py E V M -gh)



Modules

Third-Party modules

Module	URL	Use
Plotly	https://plot.ly/python/	Plotly is used to create the 3D graphs that represent the planet orbits.

Standard Library modules

Module	Use
csv	The csv module is used in the planetDB.py script to read in the different elements and features of the planets from their respective data files. This module makes the reading process simple, as each element in the list can be called by heading, as defined on the file's first line
copy	The copy module is used to create a shallow-copy of the Earth or origin planet when using the -g or -c options. The script creates a copy of the original planet, allowing us to use the non-updated heliocentric coordinates of the origin planet to graph the position of the sun.
datetime	The datetime and timedelta objects are used to store the starting and end dates, and to be able to easily iterate over each day.
math	The calculations for the Schlyter and VSOP87 methods require the use of cos, sin, and tan, as well as the use of square roots. The math module is imported to handle these operations.
os	Used only to test if the planet database already exists when the planetDB.py script is run.

sqlite3	The underlying database used to quickly and easily organize and retrieve planetary terms and elements is a sqlite database. This module provides functions necessary to create and read this database file.
sys	The sys module is mainly used to retrieve command line arguments passed to the program in the main script.
Telnetlib	The telnetlib module is used to connect to the Horizons service that NASA provides.
time	This module is used to receive the system run time of the running program. It is used to measure the time taken for each method and the entire orbital drift program.

Created modules

Module	Use
horizonsConnection	This module handles the Telent connection needed to access NASA's Horizons service. It also handles translating and storing the information read from Horizons into the planet objects.
planet	Defines a planet class, which contains the necessary information to identify and graph the planets, and provides methods for interacting with planets.
planetDBInterface	This module is a wrapper that abstracts interactions with the planet database. The database connections is established when this class is initialized, and then maintained and accessed through this object. Contains methods for routine actions, such as finding a given planet.
plotManager	This module is another wrapper that handles setting up the plotly graphs. Contains methods that can be used to create plotly graph objects from planet objects.
reporting	Contains several methods used for creating the report when -nd is not specified. These methods include timers for each method, and writing the difference file itself.
SchlyterCalc	This module contains the methods required to run the Schlyter calculations on a given planet. Provides a runSchylterCalc method which calls the required methods defined in the module in order.
VSOP87	This module contains the methods required to run the VSOP87 calculations on a given planet. Provides a runVSOP87 method which calls the required methods defined in the module in order.

References

Meeus, Jean. Astronomical Algorithms. Richmond, VA: Willmann-Bell, 1998. Print.

"Planet Positions Using Elliptical Orbits." Planet Positions Using Elliptical Orbits. N.p., n.d. Web. 04 Dec. 2016. <http://www.stargazing.net/kepler/ellipse.html>.

"Planetary Positions with VSOP87." Planetary Positions with VSOP87 - Info :: Caglow. N.p., 25 July 2012. Web. 04 Dec. 2016. <https://www.caglow.com/info/compute/vsop87>.

Schlyter, Paul. "Computing Planetary Positions." Paul Schlyter's Homepage. N.p., n.d. Web. 04 Dec. 2016. <http://www.stjarnhimlen.se/comp/tutorial.html#7>.