PHYS1321 Computational Methods in Physics
December 8, 2016

Time Evolution of Our Solar System with Earth-Asteroid Collisions

Gerardo Ayes, Nick Hopkins
University of Pittsburgh

ABSTRACT

Aims.
Methods.
Results.

1. Introduction

The problem that we tackled was creating a simulation of the solar system and simulating collisions of astronomical objects with the earth. In order to do this we needed to encode the physics involved. Attractive forces due to gravity are what generate the orbits of celestial bodies. Newton's law of gravitation states that the force of attraction between two objects is directly proportional to the product of their masses and the gravitational constant, while being inversely proportional to the square of their separation distance. This results in the equation: $F_{ji} = -\frac{GM_iM_j}{r^2}$ where M_i and M_j are the masses of the objects and r is the separation distance between the two. This equation is the foundation of what allows us to understand how objects affect one another due to gravity. The collisions and even the orbits of the planets were described using this equation.

It would have arguably been more fun to simulate asteroid collisions with Earth that sent it out to Jupiter's orbit or that ejected it out of our solar system than those presented below. However, in the spirit of being scientific the cases discussed were chosen while keeping in mind more plausible situations given our knowledge of the history of our own solar system.

2. Code Description

The complete repository is available on GitHub[1] with the structure being such that there are several files aptly named to reflect the step in our process for which they were created. Later we unified them to produce a final code that satisfied our initial objectives, described above. Among these is “entire-solar-system-simulation” which begins by defining the relevant constants in our solar system. The format chosen for the organization of this data was, since there are 9 main objects, for 0 to represent the Sun, 1 for Mercury, and continuing on until reaching 8 for Neptune. The constants needed were mass M_i, average orbital distance D_i, and average orbital velocity V_i. The gravitational constant and their respective orbital periods were also established in order to have appropriate time scales when testing the code.

Next was the “genOrbitalMotion” function, this is called by the function in the same cell, “orbitalMotion” with the built in odeint from the scipy module. The duty of the former is to tell odeint what to do during each time step and this was generalized to n objects based off of the initial conditions. When specifying yarr[2] and t[3] the orbital periods mentioned earlier proved most useful and N specified the resolution, how many iterations odeint would go through. From there plotting was through the matplotlib.pyplot module with a legend for ease of tracking each body.

Subsequently there was the animation function. Originally I had not attempted an animation before and thus supposed that either creating png files and saving generated images to them and subsequently displaying them in a gif or creating a figure and updating the data shown using the arrays generated by the odeint function would work. It turns out that the latter with an mp4 file

[1] https://github.com/nah91/Destabilization

[2] initial conditions going as: x, v_x, y, v_y

[3] time array with length N

as the result is the one I got to work first. Thanks to a code found online[4] it was a matter of specifying the os path which solved the issue.
Afterward, the final function I truly needed was not a new one, but rather a modification of the original genOrbitalMotion function to take into account collisions. The simplest method seemed to be to directly alter the function as opposed to try a separate collision function to be called later. The solution was to add an if statement after calculating the separation distance between bodies i and j and to update the position and velocity in both x and y of object j to that of i. Finally, test cases can be implemented varying the mass and radius of a ninth body, namely an asteroid, setting initial conditions such that it collides with Earth and affects its orbit significantly.

3. Analytic Solution

The major calculation throughout our project was that of the acceleration of body i due to the forces $\sum_{j\neq i}^{n} F_{ji}$ where n was the total number of bodies in the system. For example, when considering the solar system's effect on Earth the for loops through i and j in "genOrbitalMotion" calculate the acceleration $\sum_{j\neq 3}^{8} -\frac{GM_j}{r^2}$. Through each iteration of j, this value is added to the sumx and sumy terms to be assigned to the Ax and Ay arrays for assignment into ansarr, which will persist.

4. Test Cases

It may be useful to keep in mind that all the following simulations were run with initial positions at $\theta = \pi$ and thus the planets' positions line up along the customary -x axis and their velocities take them into counterclockwise orbits.
The first test case discussed was that of an asteroid with the mass of Ceres (~9.393e20 kg) with a velocity defined such that its kinetic energy is equal to the difference between the orbits of Mars and Earth: $KE = \frac{1}{2}mv^2 = PE_{Mars} - PE_{Earth}$ The results were slightly surprising as the expected result was an overlap between their orbits after the collision. However, Earth's orbit was affected much less than that. Fig. 1 depicts Earth's orbit with vs. without this asteroid collision. This case is interesting because Ceres is an actual object in our solar system, classified as a dwarf planet, and is an interesting thought experiment.

[4] http://josephrenaud.com/post-drop/2016/data-animation-with-python

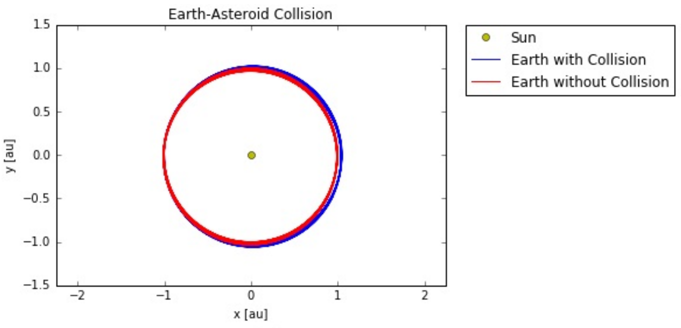

Fig. 1. Orbit of Earth with vs. without a collision, asteroid with mass of Ceres.

The second test case explored was an asteroid with the mass of our Moon and $KE_{asteroid} = \frac{1}{2}mv^2 = PE_{Earth}$. This is similar to what Astronomers theorize resulted in the Earth-Moon system we take for granted today.

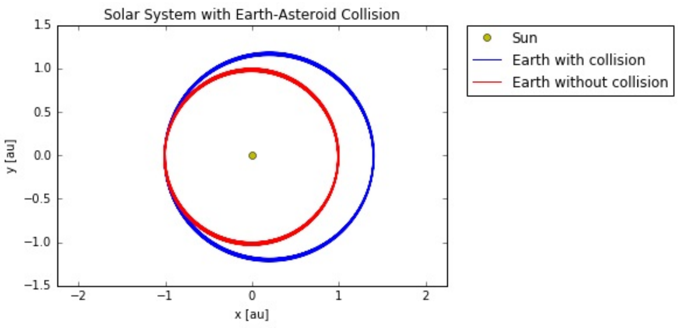

Fig. 2. Orbit of Earth with vs. without a collision, asteroid with mass of Moon.

5. Full Physical System

The entire solar system was considered[5]

6. Results

7. Discussion

Insert phase diagrams here and analysis.

[5] The planets but not their moons, although could be trivially added in since "orbitalMotion" takes in n objects (would need to adjust collision parameter established by 'Sep' variable).

8. Future Directions

9. Acknowledgments

10. Code