Final Project : Solar System & Interstellar Mission

Wei-Hsiang Yu 游惟翔

Department of Physics, National Tsing Hua University, Hsinchu, Taiwan

June 24, 2021

Abstract

This project predicts to improve the solar system simulation based on the program we had done in lecture 7 & 8 to make it can simulate **a more real solar system** and use it to find **trajectories of Voyager 2**, which had launched in 1970s in human history. Finally, this project didn't achieve all the goals, but there are still some interesting results during building up this project, I will show you in the following report.

1 Introduction

In lecture 7, professor showed his program about a n-body system, and in the next lecture, the course came to the RK4 method to have higher accuracy way to solve ODE problem. I consider that maybe I can simulate one satellite projected by multiple stars by RK4 method to get the trajectory of the interstellar mission.

So I need a real solar system to meet a real circumstance for satellite launch record, and use a way better than the Euler method to fulfill this simulation, and based on this simulation to compare with the launched record of *Voyager2* [1], which had passed 4 gas giants of our solar system in human history to test this program is exact or not.

Finally, this project didn't finish all the parts I predicted to achieve. But the data acquirement and classification during prepare stage; the plot coding and idea of physics problem in coding stage, and the last problem of call object problems about copy mechanism in Python, they all made me more familiar with computational astrophysics, and I believe those experience will be applied to my near future when I become a graduate student!

2 Ephemeris obtaining

The real solar system ephemeris was imported by following package/database:

- **Time** : *astropy*[2] package in Python
- **Location** : *NAIF*[3] database
  . *jplephem*[4] package in Python

2.1 astropy.time

By using this package, I convert the UTC time into JD time (Julian Day) easily, and thus can modify the launch time and the update dt more convenient.

In this part I create an numpy array to store a time list, whose length is 10 years long and its interval is 1 day.

2.2 NAIF database

NASA's Navigation and Ancillary Information Facility (NAIF) provided a ephemeris of each planet location in our solar system.

I chose **"DE421.bsp"**, a solar system ephemeris between 1900-2050, and use *jplephem* package, which also provided by them to read this file.

So based on the above two steps, I can get a real solar system model(Fig.2), and I will use this data and the RK4 result of our satellite to realize the simulation.

3 Methodology

This part will show the achieved tasks of this project:

- 3D RK4
- Initial condition
- Thruster system of voyager2

3.1 3D RK4

I transform RK4 method we had built in the lecture 8, and applied the 3D simulation by the idea of spherical coordinate (Fig.1).

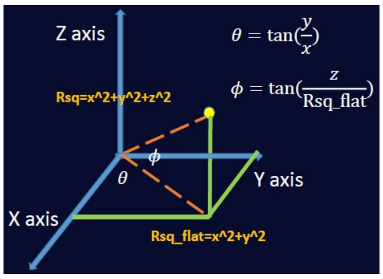
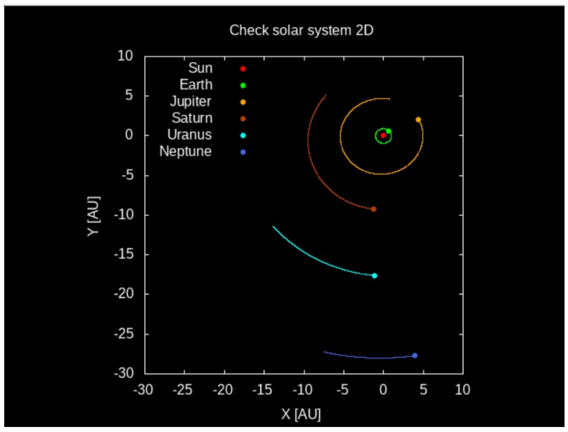


Figure 1: 3D x,y,z components definition

Noted that ϕ angle I define here is different from we usually see in the physics notation, because we already have the Z value from NAIF, so define ϕ as the angle between x-y plane to vector will be more straightforward.

(a) 2D solar system



(b) 3D solar system

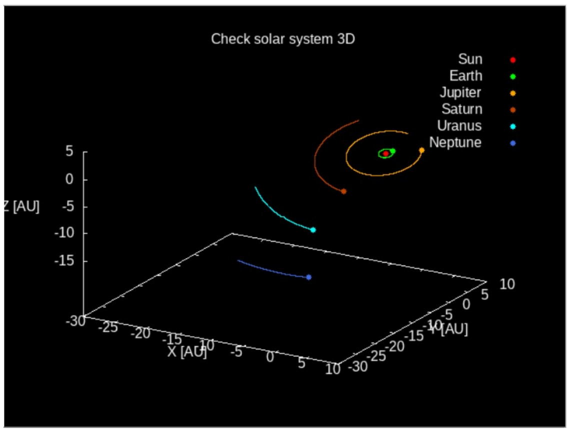


Figure 2: real circumstance of solar system data

3.2 Initial condition

I supposed the initial position of satellite launch will be the height from the Earth ground, so it will perpendicular to Earth angular velocity, and this position can be viewed as the midnight of the Earth, so that's why I chose 00:00 as the start time of Ephemeris. The velocity direction will be parallel to the Earth angular velocity.

I first try the launch date of Voyager2 1977/08/22 with initial velocity 38.5km/sec [5], but the simulation result was showed in Fig.3, the satellite was slower to arrived the Jupiter orbit so it didn't speed up by Jupiter.

After several times try, I find that increase the initial velocity will make ellipse orbit more flat, early to arrive Jupiter orbit but make the aphelion point far a lot. On the other hand, the launch date can modify the time of satellite arrived Jupiter orbit, but need to be in the launch window based on Voyager1 and Voyager2.

Both of Voyager1 and Voyager2 were sped up by Jupiter (Voyager2 launch on 08/22 and Voyager1 on 09/05), so I let our satellite can be launched during these dates to find a better date to launch.

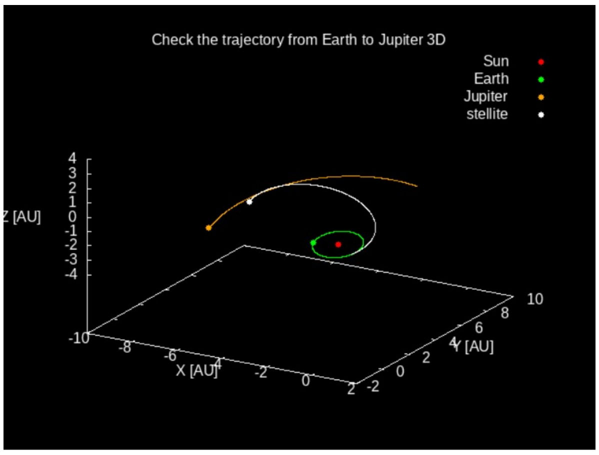


Figure 3: Initial condition on the date Voyager2 launch in the history

3.3 Thruster system

From the introduction on the official website[6] and the information provided by Wiki, they said that there are 16 thrusters(using Hydrazine as fuel[7] [8]) set on Voyager to control the attitude and correct the trajectory.

JPL says that each Hydrazine thruster will have 3 ounces force, so I based on this information to realize our thruster system.

3.3.1 thruster acceleration

I first suppose that there are 8 thrusters (half number of the system) can simultaneously jet in the same direction.

$$F_{thruster} = m_{voyager} \times a_{thruster}$$

$$8 \cdot (\frac{3 \times 0.02835[ounce \rightarrow g]}{1000[g \rightarrow kg]}) \cdot 9.8 = 721.9 \times a_{thruster} \quad (1)$$

3.3.2 thruster dt

Because the time of turning on the thruster will suppose in minute unit, but the simulation of our silar system and Voyager 2 update dt will in hour or day unit.(i.e. dt will be different.)

So there will be a ratio times into the thruster's acceleration to make the unit of our thruster system's acceleration as same as the gravity provided by planets.

3.3.3 thruster timing

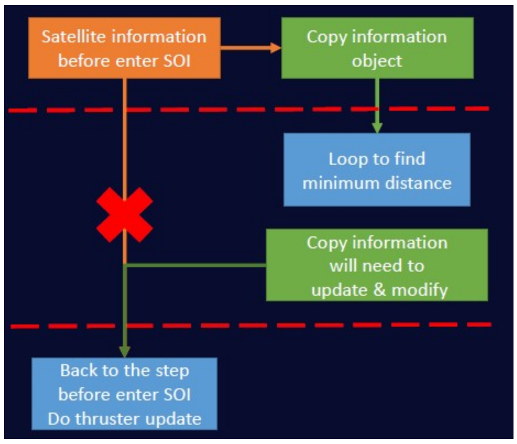
In this simulation, I determine the sphere of influence(SOI) [9] as the definition of whether our voyager 2 is closed to the target planet or not.Since SOI describes the dominated area of the planet gravity impact rather than the farther but massive object(like Sun) does.

I consider that the thruster system can provide a more speed up or down gain with the help of target planet gravity.

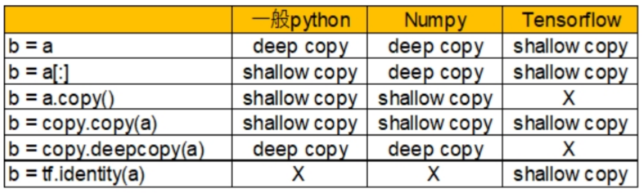
The equation of getting SOI is:

$$r_{SOI} \approx a(\frac{m}{M})^{2/5} \quad (2)$$

(a) Schematic diagram of finding direction subroutine


Satellite information before enter SOI
Copy information object
Loop to find minimum distance
Copy information will need to update & modify
Back to the step before enter SOI Do thruster update

(b) copy mechanism



	一般python	Numpy	Tensorflow
b = a	deep copy	deep copy	shallow copy
b = a[:]	shallow copy	deep copy	shallow copy
b = a.copy()	shallow copy	shallow copy	X
b = copy.copy(a)	shallow copy	shallow copy	shallow copy
b = copy.deepcopy(a)	deep copy	deep copy	X
b = tf.identity(a)	X	X	shallow copy

Figure 4: Copy object problem in Python

3.3.4 thruster direction determination

The thruster direction is decided to the vector from the planet to the satellite when the satellite come to the closest distance. This direction can modify the trajectory in the most economical case.

The direction determination mechanism is:
1. Determine whether enter SOI or not.
2. Stop recording trajectory and copy Voyager2 orbital information as a new object into another update subroutine.
3. Find the closest condition and get thrust direction.
4. Back to the position of entering SOI, starting to do thrust.

But in the step 2 and 4, I faced the problem of copying the orbital information.(Fig.4(a)) This problem will be discussed in the next section.

4 Dilemma

In the previous section I said there was a false data manipulation when the subroutine copied the object in the different prediction, I will show the bug detail and the solution.

The error was happened when the update subroutine in step4(blue object in Fig.4(a) left down side),which was mainly executed the thruster effect, wants to use the originally orbital information in step 2(orange object in Fig.4(a)) to continue doing update by adding thrust.

But this following update subroutine got a wrong orbital information from the copy object(green object in Fig.4(a) right side),which was modified by the previous update subroutine to get thruster direction by doing minimum distance find.

The reason is that the copy mechanism **Python will use "deep copy" when user only type: a=b** 4(b), which means that the object/variable/array will be copied the whole itself including the store address to create a new copy one.[10][11]. That means when we call this object,either the original or copy thing, this action will all point into the same address in the computer memory.

The above explanation shows that why I got the modified orbital information rather than the original one, because the value at this address was changed by its address cascade property. This concept is like the pointer and reference when we study call by value or call by reference in the programming language course.

The solution is to import another package called : copy to achieve the "shallow copy" [12] 4(b) (will create a copy object in different address)

I originally misunderstood the abrupter displacement is the result of thruster speed up effect, so I didn't stop to deal with this problem and start to develop the next step, finding the best thrust mechanism.

Until in the next step start to execute to test the feasibility, the print value of thruster parameter changing made me found something mistakes due to the unreasonable large displacement even only turn on thruster in 1 sec.

5 Improvement

This part will talk about the original idea I design in the step after finding the thruster direction, but the delay caused by copy problem made this step unfinished.

5.1 thrust mechanism (unfinished)

Now the subroutine go back to the first step when satellite enter SOI, so after this step will start to turn on the thruster.

Iteration will start to turn on the thruster 1.2.3... update dt (in my test update dt=1 day, and the dt ratio of thruster was initially set as 1 day for developing this part code.), and the other time for arriving the closest distance will still without thrusting, just use the gravity by Sun and target planet to do update.

After running every dt step from the SOI to the closest distance, each thruster result (thrust from just dt=1 or until the time closest to target planet) will be compare, the compare can base on the dt value since dt can represent how much time we turn on the thruster to find minimum turn on time.

But if the result of turn on thruster for whole time(from enter SOI to the closest of target planet) can't arrive the gravity assistance, the simulation need to stop and go back to modify the initial condition.

If the result of turn on thruster have the minimum during whole time, maybe the program can do another thrust mechanism, which starts to turn on the thruster in the second moment when entering the SOI, but this method will take lots of time, I guess may this part can achieve by bisection method premise this problem has only the best case.

Experience

I think this project make me practice to reorganize a program when I need to start it in the beginning without advance code, and know more Python skills such as the data convert between Numpy and Pandas and the plot instructions.

And I also studied a lot of numerical algorithms, astronomy knowledge, the skills of typing a decent report and the English presentation. Thank you for teaching me all of this!

References

[1] Voyager 2 - Planetary Voyage
https://web.archive.org/web/20131127192310/http://voyager.jpl.nasa.gov/science/planetary.html

[2] **astropy.time** package documentation
https://docs.astropy.org/en/stable/time/index.html

[3] **NAIF** : NASA's Navigation and Ancillary Information Facility
https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/a_old_versions/

[4] **jplephem** package documentation
https://pypi.org/project/jplephem/

[5] Voyager2 simulation
http://spiff.rit.edu/classes/phys216/workshops/w8c/voyager/voyager2.html

[6] Voyager thruster force on Jet Propulsion Laboratory
https://voyager.jpl.nasa.gov/mission/did-you-know/

[7] Voyager thruster introduction news
https://spaceflightnow.com/2017/12/03/voyager1probefireslongdormantthrustersininterstellarspace/

[8] Voyager thruster system on wiki
https://en.wikipedia.org/wiki/Voyager_1

[9] Sphere_of_Influence (SOI)
https://en.wikipedia.org/wiki/Sphere_of_influence_(astrodynamics)

[10] Deep copy explanation - 1
https://medium.com/@johnnyliao/python%E9%81%BF%E9%96%8Bdeep-copy%E7%9A%84%E9%99%B7
%E9%98%B1-%E5%AF%A6%E9%9A%9B%E5%9C%A8numpy-tensorflow%E7%9A%84%E6%87%89%
E7%94%A8-bebbdd247535

[11] Deep copy explanation - 2
https://www.facebook.com/519265398171665/posts/1212531952178336/

[12] Deep copy solution : Package copy
https://ithelp.ithome.com.tw/articles/10221255