AST 2000 - Part 5
Satellite Launch

Welcome to Part 5 of the AST2000 Spacecraft Project. This is the moment you've been waiting for! You are now going to launch your spacecraft and travel to your destination from Part 3 after first planning your journey by simulating the trajectory.

GOALS

∇ Write a generalized orbital trajectory simulation.

∇ Plan your journey to your destination planet using simulations.

∇ Launch your spacecraft, adjust your planned trajectory on the fly and travel to your destination.

SIMPLIFICATIONS AND ASSUMPTIONS FOR THE JOURNEY

⊗ While in space, the spacecraft is allowed to freely adjust its angular orientation without the use of fuel.

⊗ The duration of a boost is incredibly insignificant when compared to the duration of the entire journey. We may therefore regard them as instantaneous.

⊗ We invoke all assumptions from Part 2 such that we can use our simulated orbits. Specifically, we will also ignore the z-axis.

HELPFUL THEORY

1. Entering a Stable Orbit

In the event that you finally arrive at your destination, you will need enter a stable orbit. [1] In general, the more circular the orbit is, the more stable it will be.

Let $\mathbf{v}_0$ denote the spacecraft's current velocity and $\mathbf{v}_\pm$ denote the velocity corresponding to a circular orbit. The sign of $\mathbf{v}_\pm$ denotes whether the spacecraft traverses the circular orbit counterclockwise (+) or clockwise (-). The orbital injection maneuver $(\Delta\mathbf{v})_{\text{inj}}$ is given by

$$(\Delta\mathbf{v})_{\text{inj}} = \mathbf{v}_\pm - \mathbf{v}_0 \tag{1a}$$

$$\mathbf{v}_\pm = \pm\mathbf{e}_\theta v_{\text{stable}} \tag{1b}$$

where $\mathbf{e}_\theta$ is the tangential unit vector relative to the planet (see Lecture Notes 1B) and v_{stable} is given by

$$G\frac{M_p m}{r^2} = m\frac{{v_{\text{stable}}}^2}{r}$$

$$v_{\text{stable}} = \sqrt{\frac{GM_p}{r}} \tag{2}$$

Here, G is the gravitational constant, M_p is the mass of the planet, m is the mass of the spacecraft and r is the distance from the spacecraft to the center of the planet.

CHALLENGES

All simplifications and assumptions with respect to the solar system and spacecraft from previous parts are also pertient in this part.

A. Simulating the Spacecraft's Trajectory

The goal of this challenge is for you to develop a program that simulates your spacecraft's trajectory through the solar system. You are *not* going to simulate a specific trajectory here! Design the program such that you can simulate any and all trajectories, provided initial time t_0 and initial spacecraft conditions $\mathbf{r}_0$ and $\mathbf{v}_0$. During a trajectory, your spacecraft *is not* able to accelerate using its rocket engine (more details in later challenges). Your program's structure should be reminiscent of this:

`input`	`output`
initial time (yr)	
initial position (AU)	final time (yr)
initial velocity (AU/yr)	final position (AU)
simulation time / no. time steps	final velocity (AU/yr)
time step length	

The trajectory of the spacecraft is given by Newton's Second Law of Motion. The net force on the spacecraft is equal to the gravitational effects from the star and all of the planets:

$$m\ddot{\mathbf{r}} = -G\frac{mM_s}{|\mathbf{r}|^3}\mathbf{r} - \sum_{i=1}^{N}\frac{GmM_i}{|\mathbf{r}-\mathbf{r}_i|^3}(\mathbf{r}-\mathbf{r}_i) \tag{3}$$

where m and $\mathbf{r}$ are the mass and position of the spacecraft respectively, M_s is the mass of the star, and M_i and $\mathbf{r}_i$ are the mass and the position of planet i. Simulate the trajectory using your favorite integration method.

B. Plan your Journey

Do not spend a lot of time perfecting this challenge! When we launch the spacecraft in

the next challenge, you will need to adjust your original plan with correctional boosts.

With your destination decided, your final task is to design a plan that will bring the spacecraft to your destination. Your "plan" will describe a launch sequence and a set of subsequent boosts:

→ For the launch sequence you must decide the initial time and position of the launch.

→ For the boosts you must decide the point in time and velocity difference of each boost.

The duration of a boost (i.e. an acceleration with the rocket engine) is miniscule compared to the time-scale of the journey. For this reason we are going to ignore the time spent accelerating and assume that all boosts occur instantaneously in the solar system's frame of reference. After performing a boost, the spacecraft will be given a new velocity, but remain at the same position.

Figure 1 below shows an example of how a journey *might* look like, the commands for such a trip would look something like:

$$\begin{aligned} t = 1.2 \text{ yr} \quad &: \quad \text{launch from } \mathbf{r}_0 = \mathbf{r}_{\text{planet}} + (R, 0) \\ t = 1.3 \text{ yr} \quad &: \quad \text{boost } \Delta\mathbf{v} = (0.1, 0) \text{ AU/yr} \end{aligned}$$

where R is the radius of the planet. **Note that the numbers are not to scale, this is just for illustrative purposes.**

Inbetween the commands, the spacecraft will do nothing except coasting freely: the only thing acting on the satellite is the net gravitational force from the star and all the planets. Recall your program from challenge A: use this program to simulate the motion of your satellite inbetween boosts. (This is why you wrote the program.)

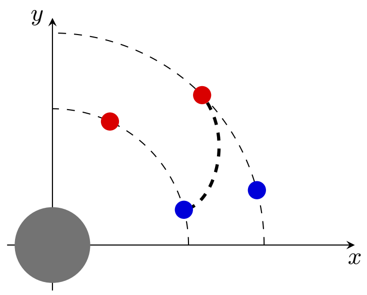

FIG. 1. An illustration of how the journey *might* look like: The inner orbit belongs to your home planet, the blue positions represent $t = 1.3$ yr (when the spacecraft performs the boost) and the red positions represent $t = 2.1$ yr (when the spacecraft arrives at the destination). The thick dashed line connecting the two orbits is the spacecraft's trajectory.

Furthermore, remember to take the accuracy of your simulations into account when planning your journey! For each trajectory, note down the number of time steps/the time step length you used in your simulation.

In addition to developing the commands, you also need to know how much fuel each boost will require. In Part 1 you designed a program that calculates the amount of fuel burnt during a boost - use this program here. In case you run out of fuel, return to your programs from Part 1 and readjust the amount of fuel you bring with you. Note that you may need to adjust other parameters as well: the amount fuel you bring with you may have an effect on whether or not your launch is successful.

Finally, the spacecraft needs to get sufficiently close to the destination so that you are able to perform an orbital injection maneuver. This distance l is given by:

$$l = |\mathbf{r}| \sqrt{\frac{M_p}{10 M_s}} \tag{4}$$

where $|\mathbf{r}|$ is the distance from the spacecraft to the star, M_p is your destination planet's mass and M_s is the mass of your star. Those who work in groups deduced this formula in part 3.

C. Sending the Spacecraft

You are now (finally) going to launch your spacecraft and begin your journey towards your destination. Use your plan from the previous challenge as a starting point. Use your orientation software from Part 4 to correct your spacecraft's actual trajectory in favor of your simulated trajectory.

As you probably experienced in the previous challenge, a simulated orbital trajectory is very dependent on the numerical accuracy of your simulation. As for real space probes, there may be several unknown systematic errors such as small astronomical bodies (moons, etc.) not accounted for, uncertainties in planetary orbits, etc. For this reason, your spacecraft's actual trajectory will likely deviate from the simulated one. This deviation is in many cases significant enough for your spacecraft to miss its destination if you follow your plan from the previous challenge.

In order to reach your destination you will need to perform trajectory corrections. The idea is for you to follow your plan from the previous challenge, but perform correctional trajectory corrections as illustrated in figure 2: using your orientation software from Part 4, you will need to compare your spacecraft's simulated position with your spacecraft's actual position. Then correct the trajectory with a small boost or go back to the previous challenge and adjust your planned trajectory and boosts slightly using the updated information about your correct position.

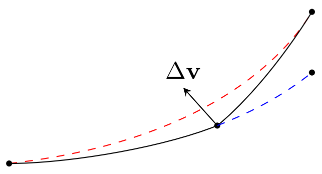

FIG. 2. Trajectory corrections. The actual trajectory (black) deviates from the planned trajectory (red), then a trajectory correction ($\Delta\mathbf{v}$) is used to prevent the trajectory from drifting further away (blue).

To send your spacecraft on its way, you need to create an instance of the `SpaceMission` class, set the launch parameters, launch the rocket and verify the launch results and your manual orientation. You can then begin the interplanetary travel by calling the `begin_interplanetary_travel` method. You command the spacecraft by calling methods like `boost`, `coast` and `orient` on the resulting `InterplanetaryTravel` instance, as described in the documentation. Your task is to successfully place your spacecraft in a stable orbit around your destination planet.

In case you wish to use the spacecraft's camera during the journey you can assume that $P \approx 1000$ and $F = 70°$. Use this as a guide to decide whether or not your planet is visible on camera. This is not to say pictures of your star are worthless, everything in space is generally quite pretty. Create as many pictures and videos as you like!

D. Orbit stability

This is optional for those working alone.

This challenge assumes that your spacecraft is orbiting your destination planet.

1. A very short time after performing the orbital injection maneuver, orient the spacecraft and determine:

   r : The distance from the spacecraft to the center of the planet.

   v_r : The radial component of the spacecraft's velocity with respect to the planet.

   v_θ : The angular component of the spacecraft's velocity with respect to the planet.

2. Assume a two-body system and use your values for r, v_r and v_θ in order to determine:

   a : The semi-major axis.

   b : The semi-minor axis.

   ϵ : The eccentricity.

   P : The orbital time period.

   $\rightarrow$: The apoapsis.

   $\rightarrow$: The periapsis.

   Note that apoapsis and periapsis is the same as aphelion and perihelion, except the latter technically only applies for orbits around the Sun.

3. Let the spacecraft do a couple of orbits and check if the values are consistent.

VERIFICATION/CONFIRMATION

In order to verify that your spacecraft has successfully entered a stable orbit, try to perform an orientation a couple of years later. Is the spacecraft still in orbit? You can also try to record a video of your destination a couple of years later, you will immediately notice whether your spacecraft is still in orbit or not.

WRITING

Once you've finished this part, it will be time to write about your progress thus far. At the very least you should complete the method sections for all the parts before you move on to the next challenge. It is *very* difficult to write up the entire project during the final weeks.

FIG. 3. Trajectory-planning, motivationally oriented, orbital duck. Launched.

AST 2000 - Part 5
Tips, Hints & Guiding Questions

I. IN GENERAL

This part is by far the part whose length varies the most from student/group to student/group depending on their solar system, their planned trajectory for the space probe, etc. Your best bet is to follow your planned simulated trajectory and then making small corrections during your actual space flight using the orientation software. If done correctly, Part 5 should not take too much time. Be thorough from the start!

II. HINTS FOR SOME OF THE CHALLENGES

A. The Trajectory Simulation

In order to access the positions of the planets at an arbitrary time t, you need to interpolate your planetary orbits. Figure 4 shows a linear interpolation of four points. You can easily accomplish this type of linear interpolation via the Python module `scipy.interpolate` (it's fast and safe).

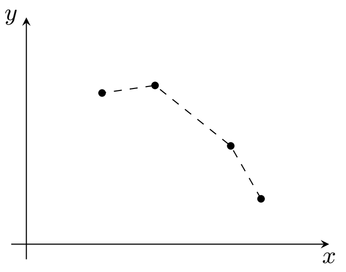

FIG. 4. An example of linear interpolation inbetween four points.

B. Describing your Final Orbit

Remember that Lecture Notes 1B introduces analytical expressions for r, v_θ, v_r and various other quantities. No advanced calculations are needed here, keep it simple. [2]

[1] Lithobraking does not count as valid arrival, by the way.
[2] In the KISS principle we trust.