Satellite project, AST 1100

Part 2: Calculating planetary orbits

In this part the goal is to calculate and make a list of the positions of all the planets at each timestep for a long period of time. You can then use this list as a reference when you later need to know the position of the planets at some particuar time.

This part is similar to exercise 1B.7 in part1B (except the part on the 3-body problem which you do not need here). Please read through that exercise before you read on.

In order to simplify this problem we make a few assumptions:

- We neglect the gravitational force between the planets.
- We neglect the force on the star from the planets and keep the star fixed at the origin. (in an exercise further down you will relax this assumption, but only for that exercise, for all other calculations during this project work, the assumption of the star fixed to the origin is necessary to make the communication with the SolarSystem class correct)
- All planetary orbits are in the same plane (defined to be the x-y plane).
- All planets orbit AND self-rotate counterclockwise.

The star is assumed stationary at the origin of our coordinate system, and all the planets orbit independently of each other.

For each of the planets you need to solve the following differential equation, given by Newton's second law

$$m_P \frac{d^2\mathbf{r}}{dt^2} = \mathbf{F_G} = -\frac{Gm_P m_S}{r^3}\mathbf{r}, \tag{1}$$

where m_P is the mass of the planet, m_S is the mass of the star, $\mathbf{r}$ is the position vector of the planet and $r = |\mathbf{r}| = \sqrt{x^2 + y^2}$ is the distance between the star and the planet. Use the Euler-Cromer method to solve this ordinary differential equation set numerically for each planet.

Hints :

- Write the second order differential equation (eqn 1) into a set of first order differential equations for the variables $\mathbf{r}$ and $\mathbf{v}$.
- Use smart units! We recommend AU's (astronomical units) for length, years for time, and solar masses, $M_\odot$, for mass. In these units the gravitational constant is just given by

$$G = 4\pi^2[\mathrm{AU}^3\mathrm{yr}^{-2}\mathrm{M}_\odot^{-1}].$$

Optional exercise: Can you show this?

- Use at least 20000 timesteps per year, and run for 10-20 years (depending on how far you want to travel in with the satellite!) This sounds like alot, but we will need the positions of the planets with very good accuracy later. This should also be a motivation to optimize your code: try to find ways to make it faster.
- Solve the problem for one planet first and make a plot of the orbit to make sure your code works.
- Use numpy arrays! They have many useful properties (like elementwise operations, splitting etc.). Another useful property for this part is that you can easily save a numpy array directly to file. Much of the data that you will recieve later will be given in terms of saved numpy arrays, so learn how to handle these. This makes it very easy to use the results from this part later. Here is an example:

```
1   import numpy as np
2   # File for saving positions.
3   outFile = open('positionFile.npy','wb')
4   ...
5   #Calculate results
6   #Generate numpy array with positions and
7   #a corresponding array of times.
8   ...
9   #Save combined array to file
10  np.save(outFile, [positions,timeArray])
11  outFile.close()
```

And if we want to use the arrays in another code

```
1   import numpy as np
2   # File with saved positions.
3   inFile = open('positionFile.npy','rb')
4   # Put positions and times into new arrays
5   [planetPositions, times] = np.load(inFile)
6   inFile.close()
```

Part 2.1: Visualizing and checking the orbit of the planets

In order to visualize and check your orbits you can dump your positions and velocities to an xml-file and use the SolarSystem Viewer exactly as described in exercise 1B.7.

Finally, when your positions are calculated and appear correct in SolarSystem Viewer, you can check whether your orbital calculations are sufficiently exact using the built-in function `checkPlanetPositions` in the AST1100SolarSystem class. The class takes 3 arguments, planet positions, years of simulation and number of time steps pr. year in that order. You may enconter numerical problems or other kinds of problems later in the project if your orbital calculations are not sufficienly exact. If you are within a 3% margin the module will allow

you to continue to the next part. If not, try smaller time steps and check your calculations again. This test has to be passed to be able to move on.

When you pass this test, a npy file called `positionsHomePlanet` will be created. This npy file contains the exact positions of your home planet and you should use these positions further on in your project. If you do not use this you risk launching your satellite from the wrong coordinates (this will not be the case if you launch at $t = 0$, but if you wait a while before you launch your launch position could well be at the moon if you're really unlucky. If you launch at $t = 0$ you can therefor disregard this npy file.).The reason for this is that even if you make very tiny time steps, there will always be some numerical errors present. These might be small in a relative sense, but in particular if you are waiting a few years after $t = 0$ to launch, the absolute error in position may be several hundred km.

Part 2.2: Can an extraterrestrial discover your solar system?

In this exercise you will check whether extraterrestrials in a solar system far away will be able to discover that there are planets orbiting your star by looking at its velocity curve. You need to do exercise 1C.5 where it is described in detail how you can achieve this. In that exercise you will now calculate also the orbit of the star around the center of mass. You may also find the light curve of your star if you wish (optional part of exercise 1C.5).

It is important to note that the planetary orbits that you calculate in this exercise, where the motion of the star is taken into account, should **not** be used further in this project. In the rest of this project you must use the assumption of the star fixed at the origin, as you did above, in order to communicate correctly with the SolarSystem class.