

Satellite project, AST2000

Introduction and useful information

0.1 Project overview

In this project you are going to send a satellite from your home planet, in your own personal star system, to visit another planet in that system. To do this you will use much of the physics you learn in the course and put it into use when solving the various problems you will encounter.

The project will consist of 2 main parts with a total of 10 smaller parts/milestones. The first main part consists of 7 milestones which are all connected and where the main focus is on numerical solutions (although you will also be doing analytical calculations). The last main part will consist of 3 smaller parts which are mostly disconnected to each other but with some connection to the first 7 parts. The solutions here will mostly be found by analytical calculations and the main focus is on understanding difficult concepts. Here is a rough overview of the 10 milestones:

1. In the first part you are going to simulate a simplified rocket engine to help you calculate the amount of fuel you are going to need to boost the rocket to the required location.
2. In the second part you will calculate the orbits of all the planets in your star system and visualize the movement.
3. In the third part you will calculate the trajectory of the satellite from your home planet to your chosen destination. You will also try to find out what instructions to give to the satellite so that it can reach the planet that you want to visit.

Bear in mind that the first 3 parts are only simple simulations to prepare you for part 4-7. It's in these later parts where we actually send out the satellite and check if parts 1-3 work correctly, so don't spend too much time on part 1, 2 and 3!

4. In the fourth part you will write the software to allow the satellite to look around and orient itself to find out where it is and what its velocity is.
5. In the fifth part you will send the *real* satellite on its way. You will use the software you made in part 4 to check and correct the calculations that you did in part 2 and 3 along the way to your destination.

6. Once you have reached the intended planet, part 6 will consist of going into a very low orbit above the planet and taking photographs of the surface. You will then choose a nice spot to land.
7. In the last part of the satellite project, you will be launching the lander module from your satellite and try to land it safely on the surface of the planet.
8. In this part you will explore you planet and other parts of the universe with near-light-speed. Here you should collaborate with another student/group where each of you will see events from different frames of reference.
9. In this part you will be travelling to a black hole and together with another student/group you will make experiments close to the event horizon of the black hole.
10. In this final part you will return to your solar system and explore your star.

0.2 Important information

Read this section carefully before starting your project:

- The first 3 parts, although equally important as the others, are your own simulations of how you think the rockets and the planet trajectories will be. When we actually link your work up to the main class `AST2000SolarSystem` in the last parts of the project, you will probably find that your rocket is not in the place you thought it should be. Don't panic, this is quite normal and actually intended as the last 3 parts are devoted to locating and adjusting your trajectory so that you might safely land on your designated planet.
- Python is the language to use for all of the coding in this project. The module you will receive, and need to use, is a python module, so you need python, or some amount of cleverness, to succeed!
- For this project you will all receive an integer that is your personal seed. This seed determines all the physical properties of the star system that you live in, so everyone gets their own unique star system. If you work in a two-person group, you will have to choose whose seed you want to use for the project.
- Although we try to make this project as realistic as possible, we also make a lot of simplifying assumptions. These assumptions are also used in the python module "`AST2000SolarSystem`" that you will use and interact with, so that these assumptions will be built into the virtual world that your satellite is exploring. This is so that some of the problems become easier to solve, but this also means that if you try to solve the problem in the "correct" way, e.g. by including forces that we specifically say that you should neglect, your results will be wrong. In other words, although it is generally good to be clever, don't try to be too clever!

- We will give you a set of exercises that you will have to do, these exercises will guide you through planning and executing your satellite trip. There is however nothing stopping you from playing around on your own. Using the tools that you will develop and the "AST2000SolarSystem" python module you can explore whatever part of the star system that you would like to see!
- For those of you who will deliver a report at the end (instead of blogging): Your resulting grade will mainly be based on a single scientific report you will hand in at the end of your project, where you describe everything you did in part 1-10. It is advised that you write notes for yourself so that you remember the details of what you did, even after several months.
- You can also choose to write a weekly popular scientific blog about each of the parts instead of handing in a single scientific report. The grade will in this case be based both on your completion of the exercises, and your science communication skills.
- It is very possible that you will encounter bugs or inconsistencies in the module "AST2000SolarSystem". If you think this is the case, then contact the group teacher responsible for this project, and hopefully this can be sorted out.

0.3 What you need

Basically you need just a few things to start this project, you should find a link to all of them on the course webpage. Be sure to check back often, since bugs are constantly being fixed in both the python module and the 3D app.

- A full installation of Python 2.7 or 3.6, with some data science packages. If you are installing on your own computer (any operating system), Anaconda is recommended to get all the needed packages automatically. Specifically the python module needs numpy, scipy and lxml. Plus some plotting package is recommended, like matplotlib.
- The AST2000SolarSystem.pyc module. This is a compiled python-file, which can be included like any other package if you have it in your working directory. You might need to download the correct one for your version of python (2.7 or 3.6).
- A personal seed. To generate this seed, use the myseed.pyc script, with your UiO username, like this:

```
terminal>python myseed.pyc johndoe
76945
```

- To get the visual effects in full 3D, you will also need to download AST2000 Mission Control (MCast).

0.4 The AST2000SolarSystem class

In order to find information about your own virtual star system you need to use the AST2000SolarSystem class. To use this class just put the "AST2000SolarSystem.pyc" file in the folder of your source code and include it. To get your information you need to make an instance of this class. If your seed is e.g. 42:

```
from AST2000SolarSystem import AST2000SolarSystem
seed = 42
myStarSystem = AST2000SolarSystem(seed)
```

All the properties of your star system are given as variables in this class, so you can access them directly once you have made an instance:

```
massOfMyStar = myStarSystem.star_mass
radiusOfMyStar = myStarSystem.star_radius
N = myStarSystem.number_of_planets
```

The planet properties are numpy arrays, so you need to index them with a number. The number of your home planet is 0, otherwise the numbers are not in any particular order:

```
massOfHomePlanet = myStarSystem.mass[0]
radiusOfHomePlanet = myStarSystem.radius[0]
```

Here is a complete list of the different properties:

```
# Data about the star system.
number_of_planets # Number of planets in the system
temperature      # Surface temperature of the star [K]
star_radius      # Radius of star [km]
star_mass        # Mass of the star [solar masses]

# Data about planets, arrays of length numberOfPlanets.
a                # Semi-major axes of planets [AU]
e                # Eccentricity of planets
radius          # Radiuses of planets [km]
omega           # Initial angle of planets' orbits.
psi             # Angle of semi-major axis for the planets
mass            # Mass of the planets [solar masses]
period          # Rotational periods (day lengths) [earth days]
x0              # Initial x-position of planets [AU]
y0              # Initial y-position of planets [AU]
vx0             # Initial x-velocity of planets [AU/Yr]
vy0             # Initial y-velocity of planets [AU/Yr]
rho0            # Atmosphere density at surface [kg/m^3]
```

You will receive more information about how to use different parts of this class as you will need it.

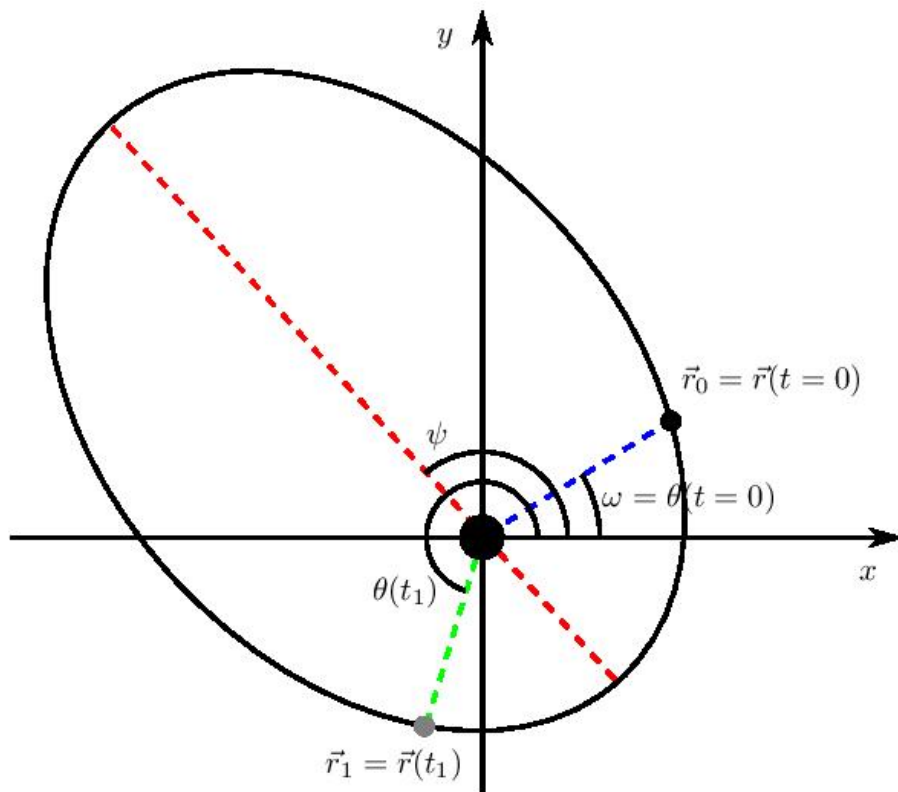


Figure 1: Figure to explain the geometry of the orbit. The figure shows the whole orbit of the planet, as well as the position at two different times, $t = 0$ and $t = t_1$.

0.5 Specifications of the satellite and lander

Here is some technical information about the satellite and the lander that you will need later:

	Satellite	Lander
Mass [kg]	1100	90
Cross-sectional area [m ²]	15	6.2

Note that the lander has a parachute (with area of your choice) that it can deploy during landing.

0.6 The first exercise

This part only concerns those of you planning on using your own computer to solve parts of or the entire project (using a strong-ish personal computer is recommended)

The university computers run Python version 2.7.3, and the data about your star system relies heavily on drawing random numbers using the Python `random` module. You should check that your own Python version outputs the same sequence of random numbers as one of the university computers, given an identical integer seed. This can be done with the `test_random_number_generator` method. So the first thing you should do is to run the following simple script from a directory in which the `AST2000SolarSystem` module is saved:

```
from AST2000SolarSystem import *  
  
system = AST2000SolarSystem(1)  
system.test_random_number_generator()
```

This will print to terminal whether the test passed or failed. If it passes, you may simply move on. If the test fails, it most likely means you are running a version of Python which is not 2.7 or 3.6. In order to fix the discrepancy between the generators, be sure that you have python version 2.7 or 3.6 installed and active. If the test still does not pass, you should contact the group teacher responsible for this project immediately.