Python Simulation of Solar System

Pablo Morandé (s1976852)

March 2021

Contents

Abstract

The aim of the project was to construct a simple animation of the inner planets of the Solar System using Python. The Beeman Algorithm was used to calculate the position and velocities of the different bodies at each time-step. This algorithm was implemented as it conserves energy (Sum of kinetic and potential) better than other algorithms.

1 Introduction

1.1 Aim

The aim of the project was to build a reasonable simulation of the Solar System, the purpose of building that simulation was to perform two independent experiments.
The first experiment involved showing that the energy of the Solar System is conserved in time.
The second experiment involved launching a satellite in the simulation at $t = 0$ so that eventually it arrived to Mars.

1.2 Theoretical Background

The motion of the planets and the Sun in the Solar System is governed by the Force of Gravity. This force is described by:

$$F_{12} = -\frac{Gm_1m_2}{r^2}\hat{r}_{12}$$

Where F_{12} is the force acting on body one due to body two, and $\hat{r}_{12}$ is the unit vector that has origin in the position of body one and ends in body two. Therefore, it is easy to see that Gravity is always attractive.
Therefore in a system with N bodies, the total force that each body will experience is given by:

$$F_i = \sum_{j=0, j\neq i}^{N} -\frac{Gm_im_j}{r^2}\hat{r}_{ij}$$

In this case F_i is the total force experienced by body i, m_i, m_j are the masses of bodies i and j respectively and $\hat{r}_{ij}$ is the unit vector between the bodies i and j.
The total energy of the system is given by the sum of the potential and kinetic energy of the system. The kinetic energy is calculated by the sum of the individual kinetic energies of each body.

$$K = \sum_{i=0}^{N} \frac{m_iv_i^2}{2}$$

The potential energy of the system is given by:

$$V = -\frac{1}{2}\sum_{j\neq i}^{N} \frac{Gm_im_j}{r_{ij}}$$

1.3 Beeman Algorithm

While there is an analytic solution for the two body problem, there is none for the three body problem or to the N-body problem.Therefore, the only feasible approach is through numerical methods.
The Beeman Algorithm is a method for numerically integrating ordinary differential equations, which can be used to model the N-body problem. The position and velocity at each step is given by:

$$\underline{r}(t+\Delta t) = \underline{r}(t) + \underline{v}(t)\Delta t + \frac{1}{6}[4\underline{a}(t) - \underline{a}(t-\Delta t)]\Delta t^2$$
$$\underline{v}(t+\Delta t) = \underline{v}(t) + \frac{1}{6}[2\underline{a}(t+\Delta t) + 5\underline{a}(t) - \underline{a}(t-\Delta t)]\Delta t.$$

Beeman's Algorithm is a symplectic integrator, which means that energy is conserved (mostly)[1]. The use of numerical approximations makes impossible a full conservation of energy, so it will produce oscillations around the average value.

2 Methods

2.1 Class Structure

The project was structured in 4 classes.

Celestial_Body Class This class was used to represent the celestial objects. Each instance of this class stored its position, velocity and acceleration at time t as a 2D vector (Numpy Array), the acceleration on the previous time-step is also stored by all instanced of this class as it is required for the Beeman Algorithm
The objects of this class were to update its position, velocity and acceleration using the Beeman Algorithm. However, all of these methods required an additional parameter to work, the time-step. This was left as a parameter as all the Bodies of the simulation must have the same time-step, and therefore it was dangerous to leave it as a parameter that could be inputted by the user in the constructor.
The class also included the methods to update the position, velocity and acceleration of the body using Euler-Cromer algorithm, this only used to generate the graphs for the stability of energy. Other methods included in this class were a method to get the sign of the y position of the body and a method to calculate the initial velocity vector from the initial position. The first method is used to check if the body has completed a period[1]. The latter is used to calculate the initial velocity vector if a planet was initially not aligned with the x-axis. The constructor of the class made a difference between three types of Celestial Bodies. Such difference was implemented in order to set up the appropriate initial conditions:

1. Star
   The initial position is set to the centre of the System, and it is assumed to have 0 velocity at the start.

[1]When a Body completes a period it y-sign changes from negative to positive and that can be easily checked when updating the position

2. Planet
   The initial position of the planets was always set at (orbital_radius,0), where the orbital_radius was a parameter of the constructor. The initial velocities of the Planets where determined using the central potential approximation and a circular orbit.

$$v = \sqrt{\frac{GM_{\text{sun}}}{r}}$$

3. Probe
   The initial position of the satellite was set to (orbital_radius_Earth,radius_Earth) as it leaves from the surface of the Earth. There was only one instance of this type. The velocity of the Probe was set to be the velocity of the Earth plus an additional velocity.

SolarSystem Class This class was used to represent the Solar System. The class was responsible of storing all the Celestial Objects and updating them using the Beeman's Algorithm. The constructor of this class required a time-step as a parameter, this time-step was used in the update of all the bodies (this is a way of forcing all the bodies to have the same time-step in the simulation). The list of bodies that where stored into the instance of this class was determined in the constructor, which loaded the lists of objects (and its information) from a file supplied by the user.
The main methods of this class included the update methods (for both Euler and Beeman's Algorithm). These methods were responsible of updating the position, velocity and acceleration of the bodies in the System.
For the Beeman's update the class would update all the position of the planets, then all the accelerations and finally the velocities. For the Euler-Cromer update, the order of update would be all accelerations, all velocities and finally all positions. Both update methods returned the total energy of the system after the update.
Other methods of this class include methods to calculate the kinetic energy, the potential energy and the total energy at any stage and also methods involved with experiment 2. The latter include a method to get the distance from the probe to the Earth and from the probe to Mars and a method to set Mars in the optimal position for the orbital transfer.

Animation Class This class was used to build the animation, to perform this task each instance contains a SolarSystem instance. The animate method updates the System once and it will add the new position of all the objects to the animation. The plot method of the class sets up the axis for the display and it initializes the animation using FunAnimation of Matplotlib. This class includes other methods to generate the graphs included in the report.

Options Enum class used to represent the options of the simulation. The simulation can be activated with a Probe launching from the Earth at $t = 0$ whit PROBE_RUN or without any Probe using the NORMAL_RUN enum.

2.2 Experiment 1

This experiment consisted in showing whether energy was conserved by using Beeman's algorithm to update the information of the bodies on the Solar System. This section outlines the methods used in this part of the project.
As the Beeman Algorithm is a numerical method it is not expected that energy is perfectly conserved, as all is based on approximations. Also, even if the Algorithm itself allowed for perfect energy conservation there would be float round errors that would make the perfect energy conservation impossible to detect. Therefore, for this experiment it was decided to show the energy of the system against the number of updates but to include also the result that would be obtained by updating the same system with the Euler's method. This makes more sense than just showing the energy using the Beeman's method as perfect conservation is not possible. By including both in the same graph it is possible to compare them and therefore to get an estimation of how good or bad are both of them compared with each other. This was the best method to show the efficiency in conserving energy of Beeman's algorithm as rough data without something to compare would not tell anything.

2.3 Experiment 2

This experiment involved launching a probe from Earth's surface to Mars. As the set up for this experiment is slightly different the constructor of the class Solar System took a parameter (Enum of Options) to decide which type of set up was required.
The set-up for this experiment involved locating Mars in an optimal position for the transfer (44 degrees, based on Hohmann orbit transfer[2]). And adding the probe to the simulation. The usual time-step for the simulation was 3600 seconds, this allowed the planets to move at an appropriate pace. However, this large time-step failed to capture the correct behaviour of the Probe near the Earth (Or other bodies), resulting in nonphysical results like the probe going to the core of the planet. To address this issue a variable time-step was implemented in the system class. If in a probe type run the probe is close (10^8 metres) to the Earth or Mars the time-step would change to 50 seconds until this condition was no longer true.
This set up allowed to perform the experiments with high time-steps and still get viable results. Lowering the time-step of the whole simulation would have been another valid approach. However, undertaking this approach would result in needing a large number of updates to get to Mars with the probe. Hence, it would have increased the computation time.

3 Results

3.1 Orbital Periods

Using the simulation the orbital periods of the first four planets were obtained. The values for the periods of the different planets are shown below in Earth years, the time-step used to get these results was of 1 hour.

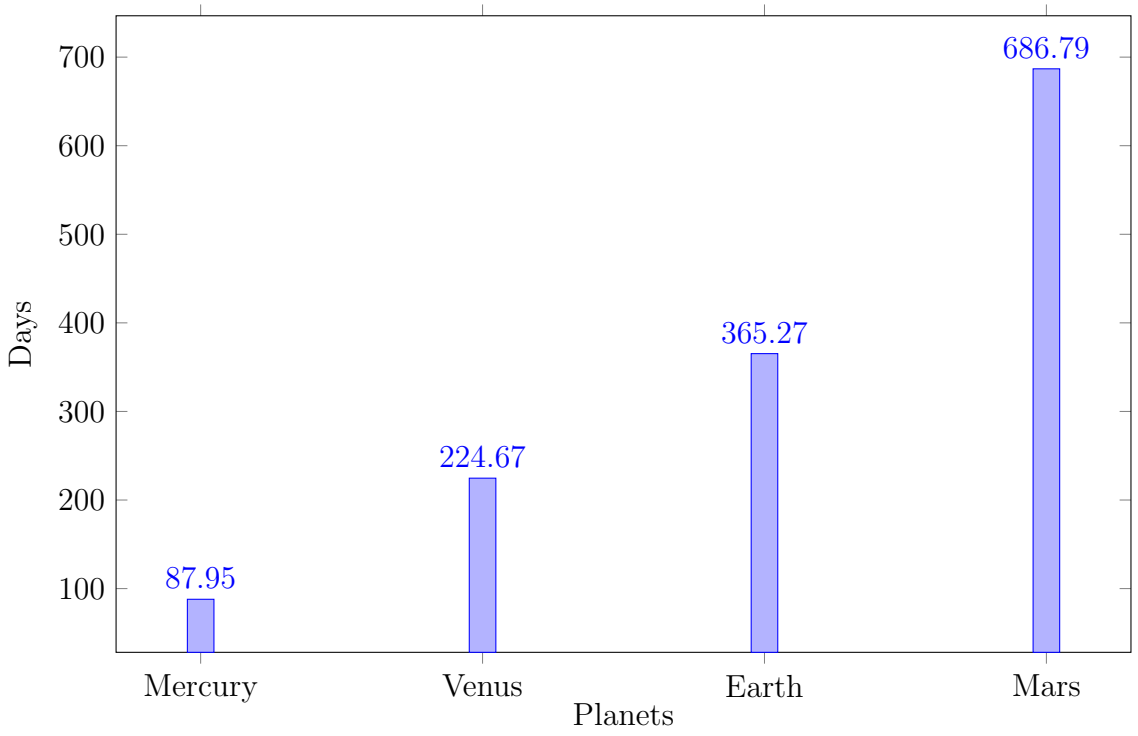

Fig. 1: Displays the periods of all the planets included in the animation. These results represent the average over several periods for each planet

3.2 Experiment 1

This experiment involved showing if the Energy is conserved in the simulation. For that purpose the Euler-Crommer algorithm was also implemented. Two graphs have been included below. The first shows the energy evolution of two identical systems using different methods to update them and the second one displays the evolution of only the first system in a different scale. The data of both graphs were obtained using the same initial system and the same time-step to produce the same results, the only difference is the scale used to present the data.

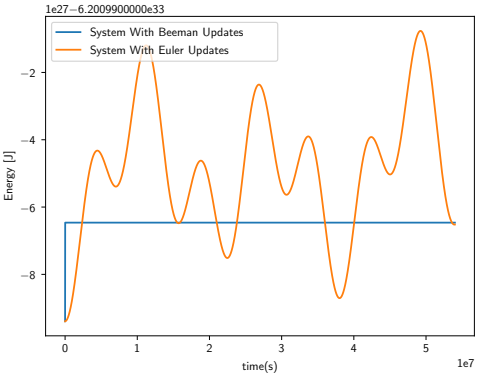

Fig. 2: Energy of two identical systems over time, one of the updated using Beeman's algorithm and the other one using Euler's.

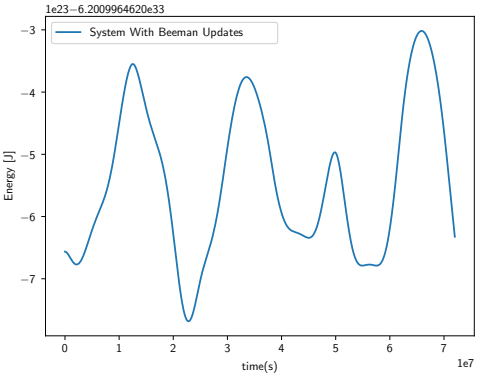

Fig. 3: Energy over time of a system updated with Beeman's algorithm

3.3 Experiment 2

The second experiment involved launching a satellite from Earth at time $t = 0$ and set the initial velocity in a way that it approached Mars. For a time-step of 1 hour and an initial velocity of 11551.7 m/s the closest that the satellite was from Mars was about $10^7 m$. The figure bellow shows the orbits of all the bodies calculated by the simulation under that initial parameters for the satellite.

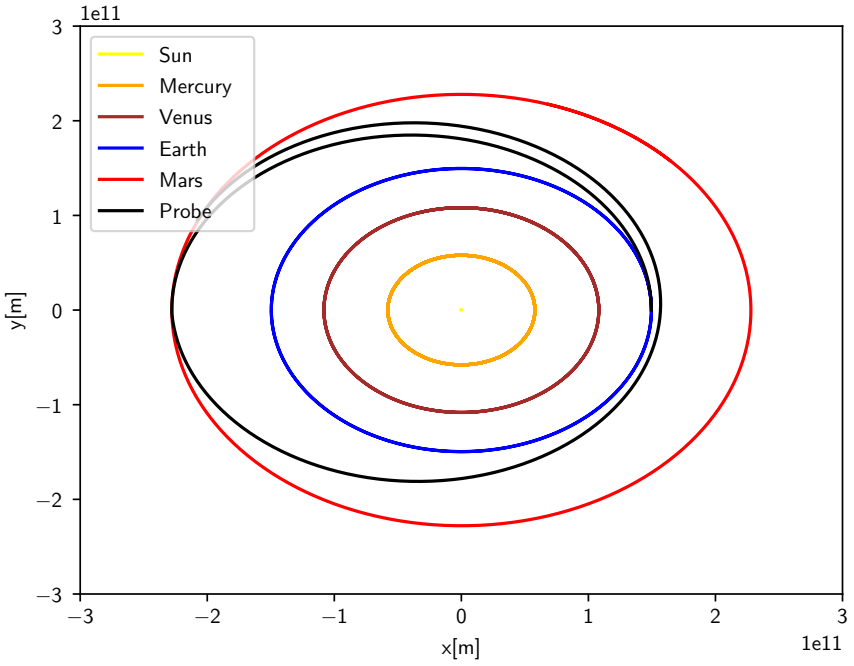

Fig. 4: This Figure displays the orbits for all the bodies on the simulation given that the initial velocity of the satellite (black line) is the optimum for it to approach Mars.

4 Discussion

4.1 Orbital Periods

It can be seen from **Fig.** 1 that the obtained values for the periods using the simulations are close to the actual ones[3]. Discrepancies between the actual values and the ones found in the simulation can be accounted by the fact that the initial velocities of all the planets were calculated using the central-body potential approximation and assuming circular orbits which is not correct. However, these discrepancies were small (all of them were within 0.001[3] of the actual value) and some discrepancy would always be expected as the method used represents an approximation of the behaviour of the whole solar system.

4.2 Experiment 1

The data shown in **Fig.** 2 suggests that the Beeman Algorithm is much better than Euler-Crommer algorithm to conserve energy. The only defect that can be noticed from the graph is the initial jump of energy after the first update of the system. This can be justified by the fact that the initial velocities were calculated under the assumption of circular orbits and a central body potential. These assumptions are corrected after the first update when the Beeman Algorithm starts to calculate the appropriate positions and velocities for the bodies. After that initial jump the energy of the system updated by the Beeman Algorithm appears to be constant. However, a closer look to the data in **Fig.** 3 suggests that the energy is not totally constant and that it oscillates over time. This was expected as full conservation cannot not be achieved by any numerical method and even if the model captured the complete behaviour of the Solar System, there would be random errors due to rounding float numbers.

4.3 Experiment 2

The results obtained were sensible, the velocity is about 11000 m/s (39600km/h)which is approximately the escape velocity from Earth and about the same velocity as Perseverance's launch velocity[4]. This velocity is also the same order of magnitude as the velocity required to get to Mars using two Hohmann orbits transfer, one to get the Low Earth's orbit and then another one to get to Mars[5]. The distance obtained for this initial velocity was also sensible as it's the same order of magnitude as the distance from Mars to Deimos.
These results were possible with a big time-step like 3600 seconds or 10000 seconds (in this case the optimal velocity was 11560 m/s) thanks to the variable time-step that was implemented in the simulation. Without this feature those initial conditions and time-step would result in nonphysical orbits for the satellite.
The orbit shown in **Fig.** 4 shows that when the satellite approaches Mars its trajectory is affected by Mars gravity and as a result its orbit does no longer intersect Earth's. By changing the initial velocity it was possible to make the orbit of the satellite intersect the Earths in expense of not getting that close to Mars.

5 Conclusion

Overall, the given model was found to capture really accurate the behaviour of the bodies of the Solar Systems. The small discrepancies found between the real data and the calculated by the simulation might arise from the fact that the model is just an approximation and that the computer might make some errors in the calculations when rounding float numbers.
The two experiments were implemented in a way that the obtained results were sensible. This was specially important in the second experiment, as conducting this experiment without a variable time-step resulted in non-physical results for high time-steps (as 3600 seconds).

References

[1] Denis Donnelly and Edwin Rogers. Symplectic integrators: An introduction. 73(10):938–945.

[2] NASA. Let's go to mars! calculating launch windows. `https://www.jpl.nasa.gov/edu/teach/activity/lets-go-to-mars-calculating-launch-windows/`. Accessed: 2020-04-09.

[3] NASA. Period data. `https://nssdc.gsfc.nasa.gov/planetary/factsheet/`. Accessed: 2020-04-09.

[4] NASA. Trip to mars. `https://mars.nasa.gov/mars2020/timeline/cruise/`. Accessed: 2020-04-09.

[5] Arthur Stinner and John Begoray. Journey to mars: The physics of travelling to the red planet. *Physics Education*, 40, 01 2005.