Simulation of the solar system

Dimitrios Chantzis s2573995

Introduction

The overarching aim of the project is to simulate the sun and the first five planets of the solar system using an n-body Newtonian gravity simulation. Based on this simulation, I performed 3 experiments: measuring the sidereal year of each planet, comparing the evolution of the total energy when using Beeman integration and when using Euler integration, and calculating the velocity and angle necessary for a rocket from earth to reach mars.

Methods General

The core of the code consists of two classes, the Planet class which represents a single planetary body that is affected by gravity and affects others by its gravity, and the PlanetarySystem class, that handles the interactions of each planet, the simulation and the animation. In the actual simulation, the solar system is an instance of the planetary system class, and each planet and the sun are instances of the Planet class. In every iteration of the simulation, the PlanetarySystem class iterates through all planets, and each planet alters its position according to:

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

Note that the acceleration has to be initialized to initialized separately at the start of the simulation.

The PlanetarySystem class then calculates the forces by iterating through all pairs of planets, finding the gravitational force applied to each planet, and each planet updates its position according to:

$$\vec{v}(t+\Delta t) = \vec{\mathrm{v}}(t) + \frac{1}{6}[2\vec{a}(t+\Delta t) + 5\vec{a}(t) - \vec{a}(t-\Delta t)]\Delta t$$

These relations are known as the Beeman algorithm. Δt is the timestep, a fixed parameter of the system. The lower the timestep, the more accurate the simulation will get.

Experiment 1: Planetary periods

Introduction

The purpose of this experiment is to calculate the sidereal periods of the 5 inner planets of the solar system. The sidereal period is the time it takes for a planet to return to its starting position after a revolution relative to distant stars[1]. In the simulation “distant stars” can be

taken to mean the coordinate axes, since they are fixed relative to the motion of the system.

Methods

To calculate the period of each, the simulation compares the y-coordinate of a planet's position with that of its previous position. If the former is positive will the latter is negative, we can conclude that the planet crossed the positive x-axis during the last time step. This means that the planet has completed an orbit. Note that all the planets start directly on the x-axis. This is handled with the check_year method in the Planet class.

To approximate the exact time when the planet crossed the axis, the code assumes that the planet followed an approximately linear path with constant velocity during the last timestep, and thus a more precise value is calculated.

By storing the exact times when a planet completed a period, the average period and standard deviation are calculated at the end.

One possible issue with this method is that the solar system gradually drifts. So, the position of the axes relative to the system changes throughout the simulation, and as time passes, the period calculation becomes progressively less accurate. For this reason, at the start of the simulation I calculate the total momendtum of the system and subtract it from the velocity of the planet a vector equal to the total momentum of the system divided by the total mass. This effectively changes the inertial reference frame of the simulation, making it so the system has a 0 total momentum and does not drift.

Results

The following table compares the planetary period found in literate with those calculated using the simulation for various timesteps, run for 1000 years.

Table 1: the simulated periods of the planets and the actual ones

Planets	Simulated sidereal periods (days) for timestep 0.01 years	Simulated sidereal periods (days) for timestep 0.001 years	Literature value[4] (days)
Mercury	89.89(4)	87.930(16)	87.97
Venus	225.25(6)	224.48(6)	224.70
Earth	365.6(1)	365.15(11)	365.26
Mars	687.2(2)	686.9(2)	686.98
Jupiter	4321	4321(7)	4332

Discussion

The calculations confirm that broadly, the simulation functions appropriately, as the values are reasonably close to the actual ones. The errors for all the values are reasonably small. The standard deviation accounts only for the error in the variance in the periods the planets performed, not the error introduced by the timestep.

As can be seen by comparing the two simulated periods, reducing the timestep increases the precision substantially. The most precise measurements concern the middle three planets, while Mercury's and Jupiter's periods are less precise.

Experiment 2: Beeman integration vs Euler integration

Introduction

Since no analytic solution exists for the n-body problem (n>2), a numerical method must be used to simulate the solar system. Apart from Beeman integration, a simple option is direct Euler integration. Instead of the algorithm already presented, Euler integration updates the positions and velocities with:

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t$$

$$\vec{v}(t + \Delta t) = \vec{v} + \vec{a}(t)\Delta t$$

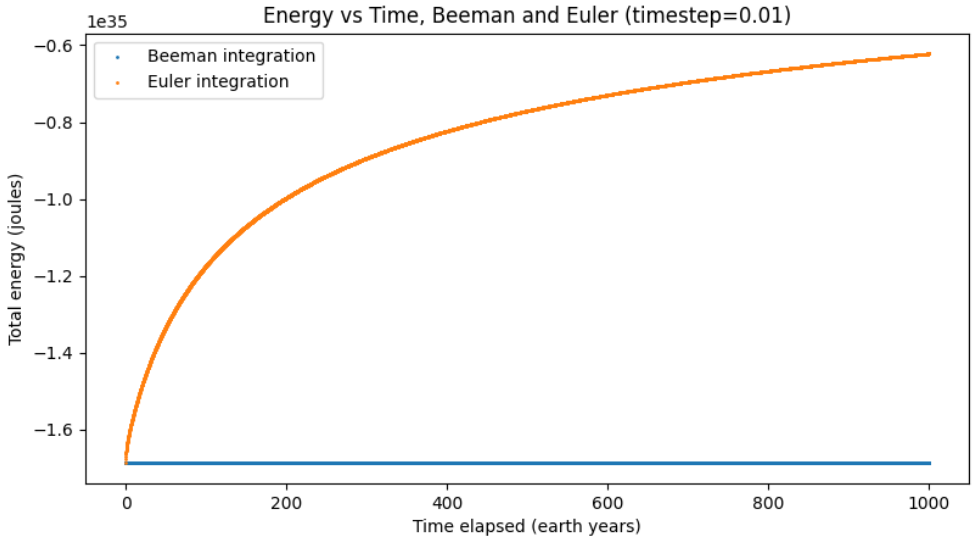
Methods

To compare the two types of integration, I extended the code to add a parameter to the constructor of the Planetary system class that determines the type of numerical integration to be used. The constructor uses getattr to redirect the method perform_step to either perform_step_beeman or perform_step_euler. An advantage of this approach is that most of the code doesn't have to be rewritten, and it is very easy to add more integration methods in the future.

In each timestep, the PlanetSystem calculates and stores the total energy of the system by combining the kinetic energy of each planet with the total gravitational potential energy. At the end, mean energy and its variance are calculated, and a graph of the energy history is displayed.

Results

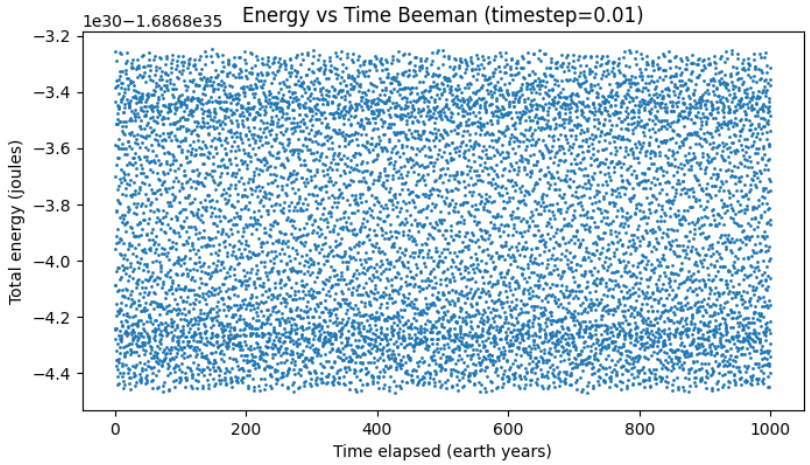
Comparing the evolution of the total energy in both Beeman and Euler integration



Graph 1: Total energy of the system with Beeman and Euler integration

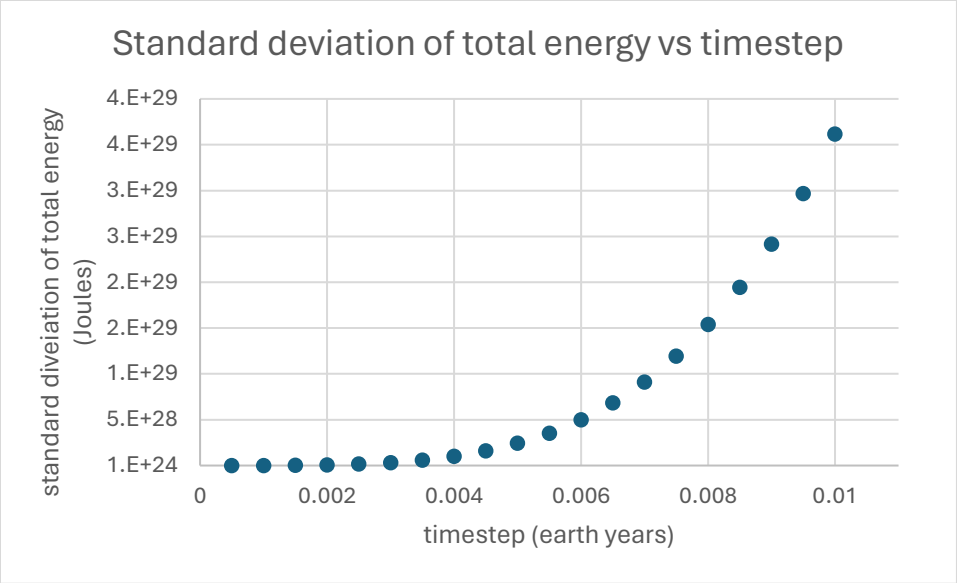
From graph one it looks like Beeman integration produces a completely stable energy evolution, while Euler integration increases the total energy as time passes, indicating that it is much more inaccurate.

However, focusing only on Beeman reveals fluctuations about the mean.



Graph 2: Total energy

In the simulation for the graph above, he mean energy was $-1.686839 \times 10^{35} J$ with a standard deviation of $3.6 \times 10^{29} J$



Graph 3: The standard deviation of the energy fluctuations with the Beeman method vs time

Discussion

Directly comparing Beeman and Euler integration, it is apparent that Beeman integration conserves energy much more accurately than Euler integration. This is why for all other experiments, Beeman integration is used.

Although Beeman integration has a constant mean energy, there are random fluctuations around the mean. Increasing the timestep decreases the size of the standard deviation of the fluctuations. However, there are diminishing returns in that accuracy improvement (graph 3), suggesting that 0.0002 earth years may be sufficient for most applications.

Experiment 3: Mission to mars

Introduction

The goal of this experiment is to calculate the conditions necessary to launch a rocket from earth to mars. My approach to this was to make an algorithm that minimizes the closest distance reached between the rocket and mars by searching through the space of possible starting conditions.

Methods

To simulate a mission to mars, I extended the core code by adding two classes. First, the Rocket class is a child class of Planet and overrides some of its methods to keep track of its closest approach to mars, and when it was achieved. Second, the MarsMission class, which utilizes the Rocket and PlanetarySystem classes to find the optimal launching angle and speed, and calculate the associated data, like the closest distance.

The rocket can launch at any place a distance of 0.001 AU from the earth's surface. I chose this distance because it is very close to earth, while making sure that the rocket can comfortably escape from the gravitational attraction. Its velocity vector will be perpendicular to the earth's surface and the speed ranges from ±10% the speed of the perseverance mission to mars, 39,600 kph. I chose a mass of 2.2 tons, like the Mars Reconnaissance Orbiter[8].

To calculate the optimal starting conditions (angle and speed), I used a combination of two algorithms. The first separates the range of possible angles and the range of possible velocities into discrete cases and checks every combination of the two to find which on results in the smallest closest approach to mars.

The second algorithm is a first-choice hill climbing algorithm [3]. It is an optimization algorithm with the goal of minimizing or maximizing the value of a function, in this case minimizing the value of the create_simulation method, which accepts an initial angle and speed and returns the distance of closest approach to mars. The algorithm starts with an initial guess for the optimal start conditions and then checks if slightly increasing or decreasing the angle or the speed by given increments improves the closest distance. It then recursively calls itself with the improved conditions. If none of the possible changes are improvements, it calls itself again, this time halving the size of the angle and velocity increments. This process repeats until the desired closest distance is achieved, or a maximum recursion depth is reached.

The first algorithm provides the initial guess for the hill climbing algorithm. To decrease the running time, each simulation is run for only one year. Since the perseverance mission performed the journey in 7 months [5], a solution exists within the constraint. Additionally, this limitation helps with excluding cases where the rocket comes extremely close to mars, but only after multiple revolutions around the sun, which would be a very suboptimal solution.

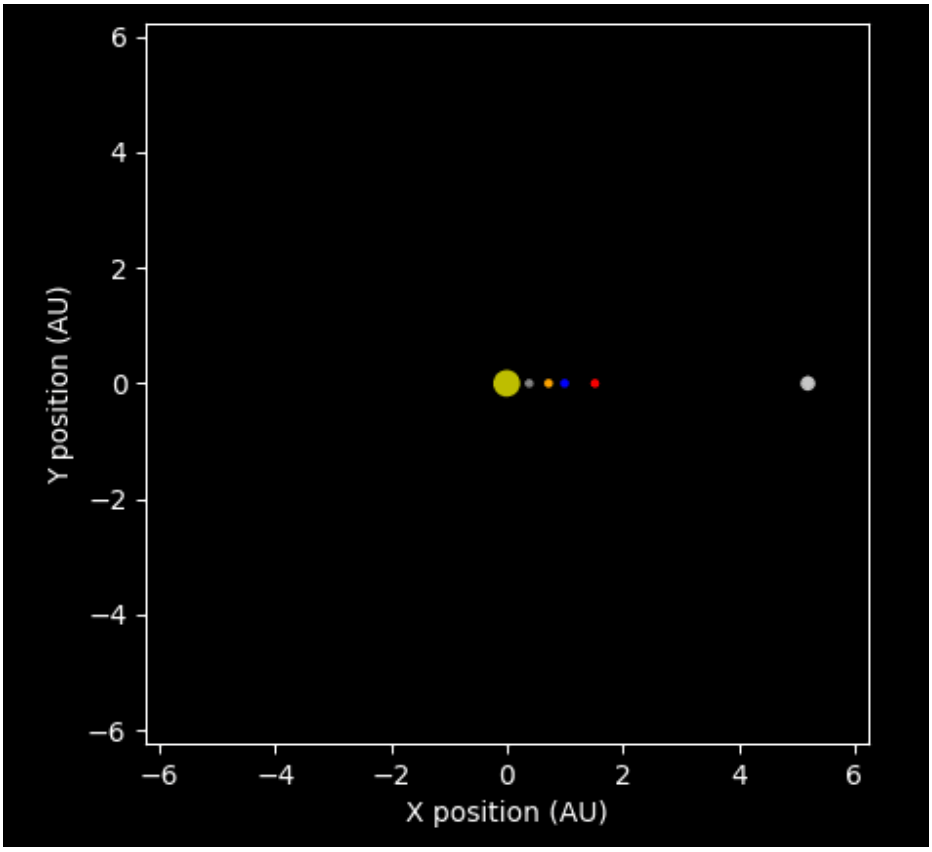
An issue I noticed was that while this process did find good initial conditions, they highly depended on the size of the timestep, and reducing it would vastly increase the closest approach. However, it is not possible to simply perform the process with a smaller timestep, as simulations with low timesteps take a long time.

The solution I found was to repeat the hill-climbing algorithm multiple times. Each time, the solution found by the previous run is used as an initial guess for the next one, and the timestep is reduced by half. This process repeats until the same solution works for two consecutive timesteps.

Results

First, I selected the closest distance of 20,428 km, which corresponds to an areostationary orbit [2], the analogue of a geostationary orbit on mars. The optimal angle is 0.00009(7) radians, and the optimal starting velocity 2.30291(7) AU/year. The total travel time was

about 6.74 months. The uncertainties are half the last increment of the angle and velocity. The smallest timestep this was checked to be valid for is 0.00003125 years.



Video 1: The rocket simulation initialized with the optimal parameters.

Discussion

The time taken for the rocket to reach mars was about 6.74 months. The actual perseverance mission took 7 months[5]. The discrepancy can be explained by considering the problems involved in a real mission. The rocket needs time to get into and out of orbit on earth and needs to expend fuel to maneuver itself into an appropriate position for landing or for orbit on mars.

Additionally, these initial conditions determine the initial speed and angle to many significant figures. In a real rocket, it will be impossible to have such accuracy, and adjustments will need to be continuously made along the way.

There are alternative analytic methods for calculating the correct initial positions, like finding the Hohmann transfer orbit, which disregards the gravity of earth and mars, and calculates a perfect elliptical orbit around the sun that would lead a rocket to mars[6]. I decided against them because they often produce suboptimal solutions. Additionally, they are not extendable. The Hohman transfer only works if there is one gravitationally dominant

body. Meanwhile, the algorithm I used can work in any configuration, as it does not make any assumptions about the system.

Conclusion

In summary, the project involved calculating the sidereal periods of the planet, comparing two numerical integration methods, the Beeman algorithm and Euler integration, and finding the optimal launch conditions to send a rocket to mars. Since the simulated orbital periods closely match the actual periods for all the planets, we can conclude that the simulation is accurate. Additionally, Beeman integrations conserves the total energy of the system, with only very minor random fluctuations, while Euler integration quickly deviates from the initial energy. Finally, the launch conditions for the mission to mars were successfully calculated and verified for very small timesteps, producing results similar to those of real missions.

References

[1] Britannica, The Editors of Encyclopaedia. "sidereal period". Encyclopedia Britannica, 30 Nov. 2023, https://www.britannica.com/science/sidereal-period. Accessed 5 April 2024.

[2] "Areostationary Orbit." *Marspedia*, marspedia, 6 Dec. 2019, marspedia.org/Areostationary_orbit.

[3] "Introduction to Hill Climbing: Artificial Intelligence." *GeeksforGeeks*, GeeksforGeeks, 20 Apr. 2023, www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/.

[4] "Solar System Data." *Royal Museums Greenwich*, www.rmg.co.uk/stories/topics/solar-system-data. Accessed 4 Apr. 2024.

[5] "Trip to Mars." *NASA Science*, NASA, mars.nasa.gov/mars2020/timeline/cruise/. Accessed 5 Apr. 2024.

[6] "Chapter 4: Trajectories." *NASA*, NASA, science.nasa.gov/learn/basics-of-space-flight/chapter4-1/. Accessed 5 Apr. 2024.

[7] Computer Simulation Coursebook, 2023, Accessed Apr. 2024

[8] "Launch Vehicle." *Mars Reconnaissance Orbiter*, NASA, mars.nasa.gov/mro/mission/launchvehicle/. Accessed 5 Apr. 2024.