N-body gravity simulation project

Paresh Dokka

December 2021

Abstract

The aim of the project was to simulate an n-bodied gravitationally interacting system using python. This was done using three numerical methods, Euler, Euler-Cromer, and Verlet. The first step in the simulation of a n-bodied solar system was to start simple, with two bodies. These two bodies were simulated using the Euler approximation. This was built upon, by having the two bodies interact with each other through Euler-Cromer and Verlet as well. Upon being successful the final step was to update the code in a way so it could iterate through all n bodies in the system and produce orbits for each of the bodies. Through qualitative testing, it was seen that the Euler approximations did not conserve energy, had become more inaccurate every iteration, and produced spiralling orbits of the planets. Euler-Cromer and Verlet on the other hand, were very similar in that they had produced stable elliptical orbits, which implied energy systems were conserved to a good approximation and were more accurate over a higher frequency of time steps.

1 Introduction

In recent decades attention has increasingly focused on computer programming. Today, programming is important in almost every area of physics. One of the major uses in the world of physics is to produce theoretical simulations of astronomical phenomena. One of these phenomena is a stable gravitationally interacting system, such as our solar system [4]. There are several numerical approximations that could be used to simulate the properties of the particles within the system, and therefore the system as a whole. The aim of this project is to use three of these approximations, Euler, Euler-Cromer, and the velocity Verlet.

Firstly, consider the case of N massive particles moving under the influence of each other's gravity. The net acceleration of a particle with mass is give by [2]:

$$\overrightarrow{a_i} = \sum_{j \neq i}^{N} \frac{-Gm_j}{|r_{ij}|^2} \widehat{r_{ij}} \tag{1}$$

G is the gravitational constant, m is the mass of the particles, r_{ij} is the distance between masses, and $\widehat{r_{ij}}$ is the unit vector.

When it comes to numerical approximations, the Euler method is the simplest approach, which iterates and allows us to calculate the position and then from that calculate velocity at t+dt, given a t. [4]

$$\overrightarrow{x}_{n+1} \approx \overrightarrow{x_n} + \overrightarrow{v_n}\Delta t \tag{2}$$

$$\overrightarrow{v}_{n+1} \approx \overrightarrow{v_n} + \overrightarrow{a_n}\Delta t \tag{3}$$

We assume acceleration is constant over small dt, and n denotes the current time step, with n+1 the end of the current time step.

An alternative method, the Euler-Cromer method, is very similar to the Euler method, in that its formulae are the same as that of Euler, except the velocity is iterated first, using the end of the step before the beginning of the step.[1]

$$\overrightarrow{v}_{n+1} \approx \overrightarrow{v_n} + \overrightarrow{a_n}\Delta t \tag{4}$$

$$\overrightarrow{x}_{n+1} \approx \overrightarrow{x_n} + \overrightarrow{v_n}\Delta t \tag{5}$$

This simple modification conserves energy for oscillatory problems, unlike Euler, which increases the energy of the oscillator over time.

Finally, there is the Verlet method, derived from the kinematic equation for the motion of any object. It uses the acceleration at the end position to calculate the updated velocity. In fact the equations used here are more commonly known as the velocity Verlet algorithm, as the basic Verlet integration does not incorporate velocities into its equations [3].

$$\overrightarrow{x}_{n+1} \approx \overrightarrow{x_n} + \overrightarrow{v_n}\Delta t + \frac{1}{2}\overrightarrow{a}_n(\Delta t)^2 \tag{6}$$

$$\overrightarrow{v}_{n+1} \approx \overrightarrow{v_n} + \frac{1}{2}(\overrightarrow{a_{n+1}} + \overrightarrow{a_n})\Delta t \tag{7}$$

2 Simulation of 2 bodies

The first step towards making an n-bodied gravitationally interacting system was to begin with two particles. In the exercise followed up by the final project [4], a simple analysis of results between a satellite and the Earth was done. The simulation used Euler as a numerical method to approximate the motion of the particles. An iterative method was used to append the updated positions and velocities of the two bodies into a list. This data was then taken and made a 2D plot of using matplotlib. The bodies were iterated 200,000 times, with a time step of 6 seconds.

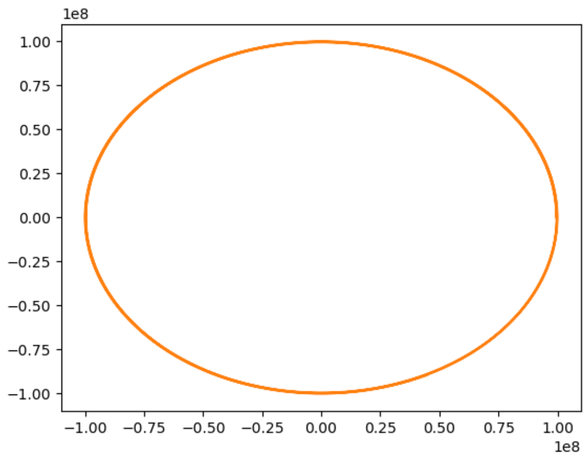


Figure 1: *This the 2D plot of the Earth and a satellite revolving around it. The Earth and satellite have not been modelled, so only their trajectories will be shown. And as such, it is difficult to spot the trajectory of the Earth, as comparatively the Earth barely moves.*

The next step was to repeat the same process; however, this time Euler-Cromer and Verlet were also added as additional methods that could be used. Furthermore, the two bodies used were Mercury and the Sun. The data for the positions and velocities of them were taken from the online JPL Horizons page. A simulation for 2 years, with a time step of 3600s (every hour) was run.

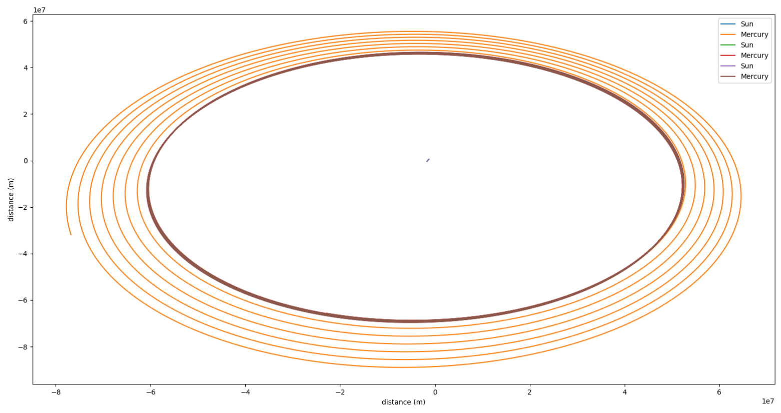


Figure 2: *Represents the trajectory of Mercury through Euler, Euler-Cromer, and Verlet. As you can see Euler's (orange) trajectory starts to spiral off as time goes, whereas Cromer and Verlet on the other hand keep a stable orbit.*

The difference in the numerical approximations can be seen here in the way that they've simulated the orbit of Mercury. Euler gets more inaccurate over longer iterations, whereas Cromer and Verlet are quite opposite. Although the difference between Cromer and Verlet is indistinguishable here, it can be seen over larger time steps.

3 Simulation of an n-bodied Solar system

Next, the 2-body simulation was evolved to take in the entire solar system. To do so, a further file named Systembodies was created to hold the information on each of the planets. [1](JPL Horizons). The Simulation file was modified so that all the updates work under a for loop to update all the planets' positions, velocities and accelerations. The planets were updated for 2 years, with a time step on 3600s. Here are the simulations of the three numerical approximations.[5]

3.1 Euler method

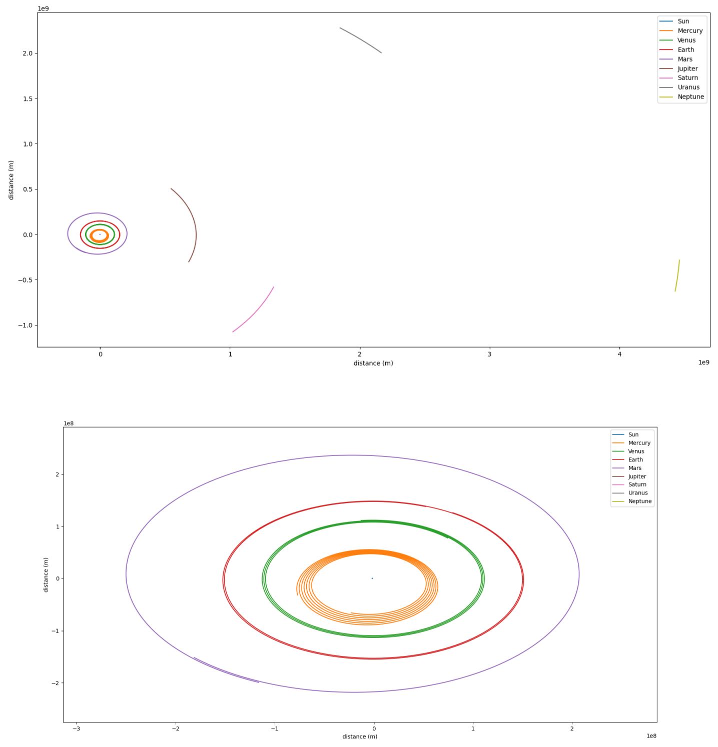


Figure 3: *Here are the trajectories of all the planets in our solar system, calculated through the Euler method. The top figure shows the entire system, whereas the bottom figure shows the first four planets' trajectory in a higher resolution.*

Euler is a first order differential equation method, so the local error is proportional to the square of the step size, and the global error is proportional to the step size itself. The Euler method; however, is an energy increasing method. So as the method is iterated over, the sequence of solutions produced will have an increasing numerical oscillations. Since each updated solution has more and more oscillations, the energy will increase and the system generates a much less accurate solution.

3.2 Euler-Cromer method

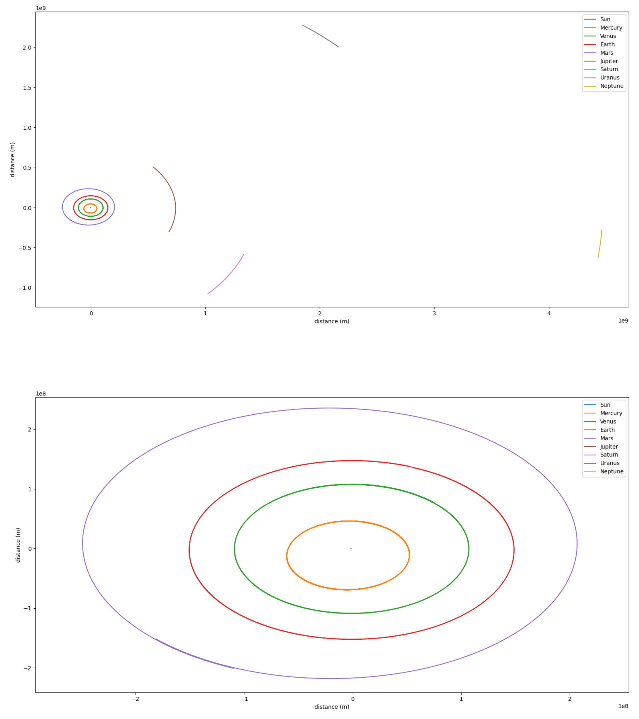
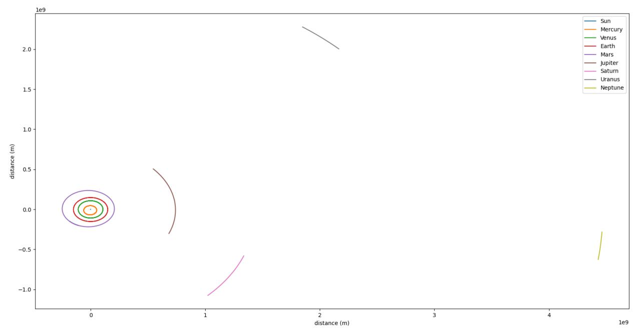


Figure 4: *Here are the trajectories of all the planets in our solar system, calculated through the Euler-Cromer method.*

Euler-Cromer is also a first order differential equation method, and hence has an error-propagation same as the Euler method, the step size. However, unlike the Euler method, Cromer does not increase the energy of the solution and has a more accurate solution. This scheme does still produce some artificial oscillations, but it can be shown that the energy of the system is conserved, and when averaged over time, equals zero. This conservation of Energy is what makes the Euler Cromer much more stable than the Euler method.

3.3 Verlet method



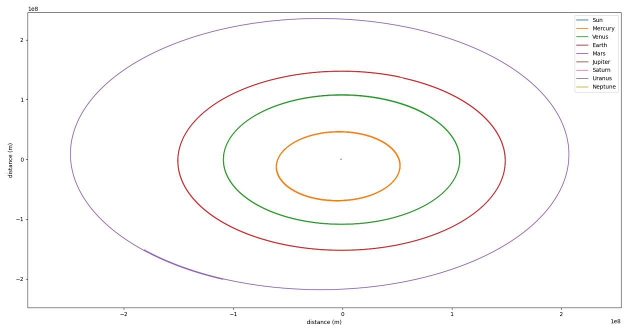


Figure 5: *Here are the trajectories of all the planets in our solar system, calculated through the Verlet method.*

Verlet is a higher order algorithm, with a local error propagation proportional to the cube of the step size in displacement and velocity, whereas the global error is proportional to the square of the step size in displacement and velocity. Verlet, like Cromer, is considered to be a good stable numerical approximation. The Verlet integration schemes are time reversible, which insures the conservation of energy.

4 Conclusion

This report focused on the simulation of an n-bodied solar system through python. The simulation was produced using various numerical approximations, namely Euler, Euler Cromer, and Verlet. Each of these approximations has produced a simulation that differs from the rest. It has been qualitatively shown that Euler was the least accurate approximation, whereas Verlet and Cromer produced far more accurate and stable orbits of the planets. Further work could include conducting qualitative tests to show that the conserved properties of the systems, such as energy and momentum, are indeed conserved.

References

[1] Drew Baden. "numerical integration, with examples". *UMD Physics.*

[2] Hugh D.Young and Roger A.Freedman. "the gravitational field". *University Physics with Modern Physics*, 14th Edition, 2016.

[3] Professor Branislov K.Nikolic. "verlet method". *Computational Methods of Physics PHYS 4660.*

[4] Matthew Pitkin. "gravity simulation". *PHYS281:Scientific Programming and Modelling Project.*

[5] Dr V.Hohmann. "simple methods". *Numerical methods for Physicists*, Version 1.0, April 2004.