

# FYS4150

## Project 3, deadline October 25.



Authors: Robin D. Kifle, Sander W. Losnedahl and Vemund S. Thorkildsen  
University of Oslo, Autumn 2017

## Abstract

We have developed a program that simulates the Solar System. The program is object oriented, and hence can be used to simulate various scenarios where celestial bodies interact via gravitational forces. In other words, the code could be written once and used many times! Four different scenarios have been studied. The first scenario was an idealized Earth-Sun binary system. The second scenario was a three body system, consisting of Earth, Jupiter and the Sun. The third scenario was for the entire Solar System, and the final scenario was for a binary system consisting of the Sun and Mercury. Various quantities were studied and we found that the kinetic and potential energy in addition to angular momentum was conserved for the binary system. By changing the mass of Jupiter in the three body system, we found that Earth's orbit initially is not affected much by Jupiter. When we increased the mass by a factor of 10 and 1000, Earth's orbit was much more affected. In the last model, the perihelion precession was studied. The precession is small, but was found to be  $\approx 43$  arcseconds per century.

# Introduction

In this report we will build a model for our Solar System. The emphasis will be on creating an object oriented code and calculating the orbits of the planets. The system can be described by using a series of coupled ordinary differential equations. In the first part of the report, a simplified version of the problem will be looked at. This is the hypothetical Earth-Sun binary situation. Another hypothetical situation will also be looked at, namely a three body system, with Earth, Jupiter and the Sun. The equations are rather simple up to three planets, but as the end goal is a system of ten celestial bodies, it will become vital to object orient the code. Preservation of kinetic and potential energy will be discussed, in addition to preservation of angular momentum. The last thing that will be looked upon is the perihelion precession of Mercury.

Two different algorithms will be tested. The Euler forward algorithm, and the velocity Verlet algorithm. Both of them are numerical integration algorithms, and will be discussed later. The code can be found [here](#).

## Method

For the Earth-Sun binary system, a simple algorithm was made. In this example, the Sun's position will be fixed. This approximation is viable, as the Earth's mass only is three millionths of the Sun's mass. In this system there is only one force:

$$F_G = \frac{GM_{\odot}M_{\text{Earth}}}{r^2}$$

To use the Euler or velocity Verlet algorithm, the acceleration must be known. Using Newton's second law of motion it is easy to derive the acceleration.

$$F_G = M_{\text{Earth}} * a$$

$$M_{\text{Earth}} * a = \frac{GM_{\odot}M_{\text{Earth}}}{r^2} \Rightarrow a = \frac{GM_{\odot}}{r^2}$$

Euler's forward method consists of three steps. Firstly the acceleration must be known, as well as initial conditions for position and velocity. As shown above, the acceleration can be calculated easily for this case, but it needs to be decomposed into x- and y-coordinates.

$$a_x = -G \frac{M_{\odot}}{r^2} \cos(\theta) = -G \frac{M_{\odot}x}{r^3}$$
$$a_y = -G \frac{M_{\odot}}{r^2} \sin(\theta) = -G \frac{M_{\odot}y}{r^3}$$

Using that  $x = r * \cos\theta$  and  $y = r * \sin\theta$ . The three steps shown for x:

$$a_t = \frac{F_G}{M_{\text{Earth}}}$$

$$v_{t+\Delta t} = v_t + a_t \Delta t$$

$$x_{t+\Delta t} = x_t + v_{t+\Delta t} \Delta t$$

The same algorithm applies in y and z-direction. This algorithm is simple, and only consists of 5n FLOPS, but this has its effects on the accuracy. The velocity Verlet algorithm can be summarized in four steps. This algorithm is viable in all spacial directions, and the acceleration, velocity and position would be calculated in exactly the same way (Hjorth-Jensen, 2015).

$$v_{t+\frac{1}{2}\Delta t} = v_t + \frac{1}{2}a_t \Delta t$$

$$x_{t+\Delta t} = x_t + v_{t+\frac{1}{2}\Delta t} \Delta t$$

$$a_{t+\Delta t} = a_{t+\Delta t}$$

$$v_{t+\Delta t} = v_{t+\frac{1}{2}\Delta t} + \frac{1}{2}a_{t+\Delta t} \Delta t$$

The new acceleration is computed in the third line by using the updated position. The velocity Verlet algorithm uses  $9n$  FLOPS, hence is more time consuming than the Euler forward algorithm. The forces acting upon Earth in the three body problem is more complex. In this case, the force acting upon earth from the sun can be described in the exact same fashion. The new element is the gravitational force of Jupiter.

$$F_{Earth-Jupiter} = G \frac{M_{Earth} M_{Jupiter}}{d_{Earth-Jupiter}^2}$$

where  $d_{Earth-Jupiter}$  is the distance between the two planets. The sum of forces acting on earth in this problem:

$$\sum F = F_{Earth-Jupiter} + F_{Earth-Sun} = M_{Earth} G \left( \frac{M_{Jupiter}}{d_{Earth-Jupiter}^2} + \frac{M_{\odot}}{r^2} \right)$$

The same relations act upon Jupiter, but then the mass of Earth is swapped for the mass of Jupiter. The acceleration in all dimensions can be found by dividing with respective masses and decomposing using the relations showed earlier in this chapter. Continuing on this path will only lead to long equations, and headaches. Consider the model for the entire solar system. The forces acting upon earth will be the sum of nine gravitational forces. The forces acting upon the remaining nine celestial bodies can also be expressed as the sum of nine gravitational forces. This underlines the importance of object orientation once again.

This project is well suited for an object oriented code since we have planets with given properties, or member variables in code language, like initial position and velocity as well as mass. Many more members could be assigned to a given planet and the code would still work. Again we see the strength of object oriented programming. Write once, use many times!

Each planet was made as an object, passing initial values into our "body" class. This class takes the initial values and turn them into member variables, making them available to all classes in the code. The "solarSystem" class takes these member variables and does operations on them, such as calculating energy and force for each integration step. The integration itself is encoded in the "verlet" class, which actually contains both the Euler forward method and the velocity Verlet method. Which method to use is up to the user. By having planets as objects, one simply have to calculate the force between each planet, and not rely on the equation above. This is done by creating a vector containing all the planets and letting a loop run through this array, each time calculating the force using the two planet force equation. Another feature of the "solarSystem" class is the printing function where the position, angular momentum, perihelion angle and distance and total energy is printed to .txt files.

The positions and velocities are vectors which are based on the "components" class. The class most of all allows for vector based operations such that one can define the position and velocities in terms of their spacial components. This makes the vectors much easier to work with. Other vector operations are also possible with this class such as finding the length of vectors, the cross product of vectors and basic algebraic operations.

By using conservation of energy, it is possible to derive a formula for escape velocity. The initial speed will be equal to the escape velocity  $v_e$ . At a later state, the planet has escaped and is positioned an infinite distance away from the sun and the speed will be close to zero. We will

only account for kinetic energy,  $E_k$ , and gravitational potential energy,  $E_p$ , assuming there are no other forms of energy conversion. By conservation of energy:

$$\begin{aligned}(E_k + E_p)_i &= (E_k + E_p)_\infty \\ \Rightarrow \frac{1}{2}M_\otimes v_e^2 + \frac{-GM_\odot M_\otimes}{r} &= 0 + 0 \\ \Rightarrow v_e &= \sqrt{\frac{2GM_\odot}{r}} = 2\sqrt{2\pi} \frac{AU}{yr}\end{aligned}$$

where  $M_\otimes$  = mass of Earth,  $G$  = the gravitational constant,  $M_\odot$  = mass of Sun, and  $r$  = distance between the Earth and the Sun.

The center of mass is given by:

$$centerofmass = \frac{\sum M_i * positions_i}{\sum M_i}$$

If the system is centred around its center of mass, one would expect the entire solar system to move as the sun gets an initial velocity from not being the center of the system. If the total momentum of the system stays at exactly zero, the planets would orbit the sun as usual despite the movement of the entire solar system.

The perihelion precession of Mercury is an important test for the general theory of relativity. The orbit of Mercury is not a perfect circle, but an ellipse. The perihelion of Mercury is not at the same place for each orbit, but is slightly shifted. This slight shift is only possible to find precisely if there are fine integration steps. To find the perihelion, one must find the point, in each orbit, that is closer to the sun than both its immediate neighbouring integration points. This is only true for one point per orbit. By using finer and finer integration partition, the precision of the perihelion will get better.

# Results

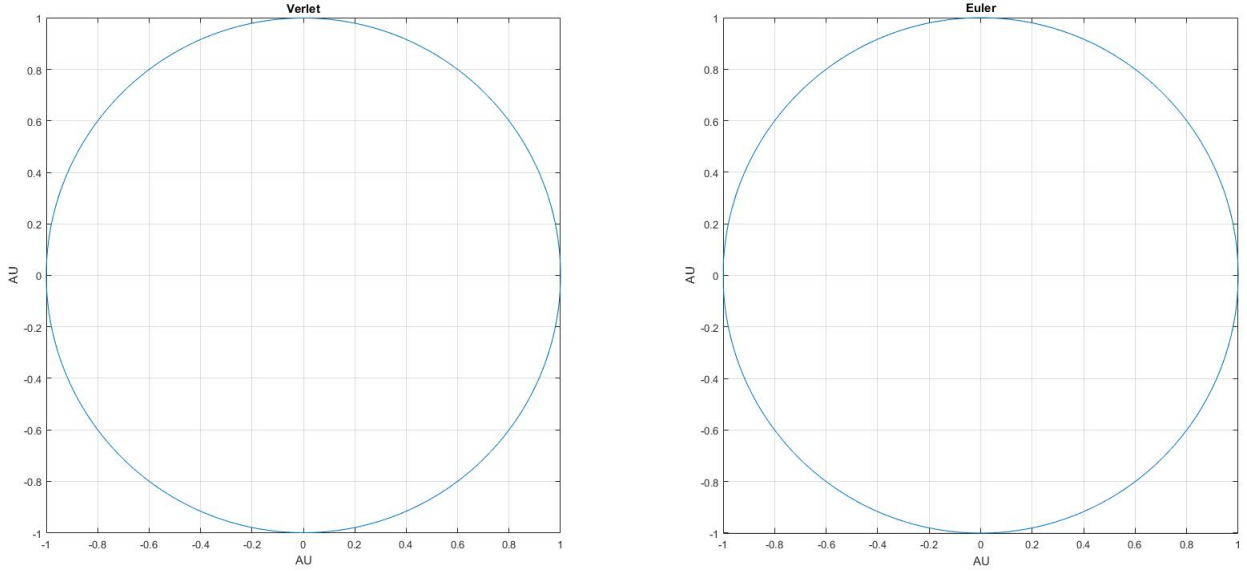


Figure 1: The planetary orbit of Earth around the Sun, with no other interfering forces. Calculated over 100 years with 10000 integration steps per year, created with the Velocity Verlet and the Euler algorithm.

Figure 1 is showing two models for the Earth-Sun binary system. Both are showing nearly perfect circles, but by playing around with  $\Delta t$ , one of the algorithms prove much more solid. As stated earlier, the Euler method has fewer FLOPS, but this takes a toll on the accuracy. The velocity Verlet algorithm is heavier for the computer to handle, but it gives a much better approximation for each iteration.

n	Euler	Velocity Verlet
10	0.001s	0.001s
$10^2$	0.001s	0.001s
$10^3$	0.004s	0.004s
$10^4$	0.021s	0.018s
$10^5$	0.163s	0.178s
$10^6$	1.614s	1.785s
$10^7$	15.864	17.553
$10^8$	159.762s	177.366

Table 1: Timing of the algorithms for the Earth-Sun binary system. Euler is faster than velocity Verlet for high n-values. Both algorithms show a linear increase in computing timing.

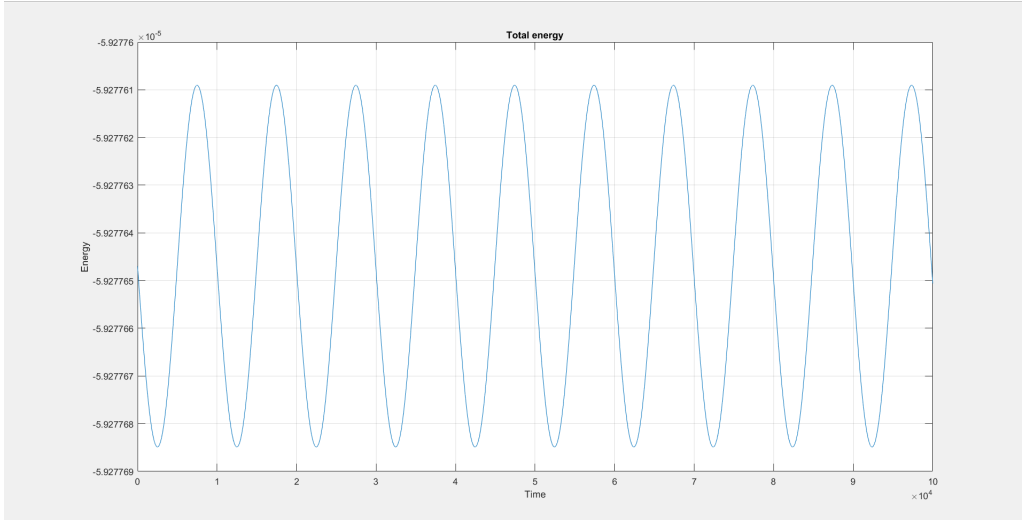


Figure 2: The total energy over 10 years, using the velocity verlet algorithm.

As there is a circular orbit, the kinetic and potential energies should be conserved, but there will be small oscillations in energy. The angular momentum is also conserved over one orbit.

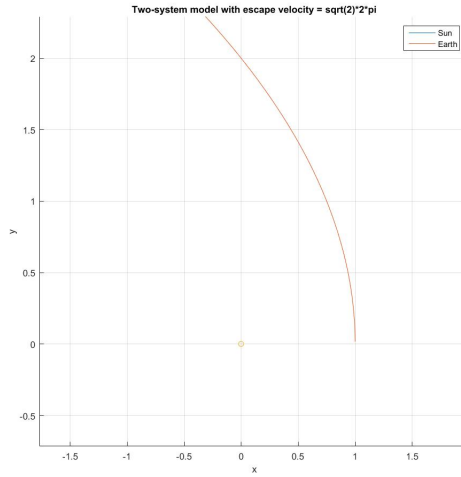


Figure 3: Plot showing the escape velocity of Earth

Figure 3 shows the trajectory of Earth with escape velocity. The exact escape velocity of Earth was found to be  $v = 2\sqrt{2}\pi = 8.8857$ . By trial and error, the escape velocity was found to be  $\approx 8.8$ , which is pretty close to the actual velocity. By replacing the gravitational force from

$$F_G = \frac{GM_{\odot}M_{\text{Earth}}}{r^2}$$

to

$$F_G = \frac{GM_{\odot}M_{\text{Earth}}}{r^{\beta}}$$

and letting  $\beta \in (2, 3)$ . The orbit becomes increasingly unstable.

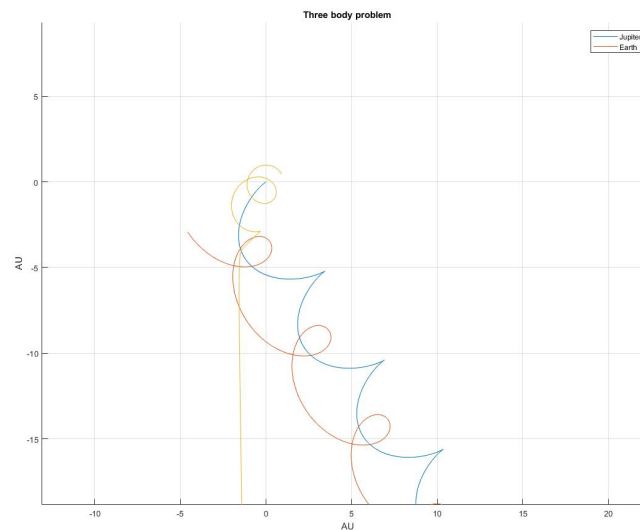
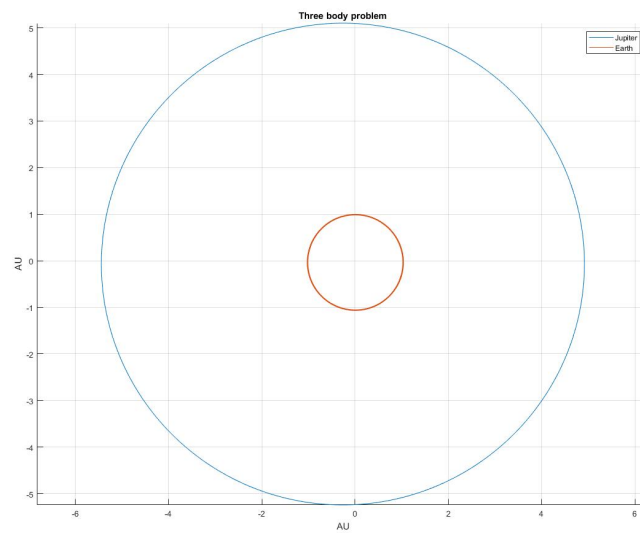
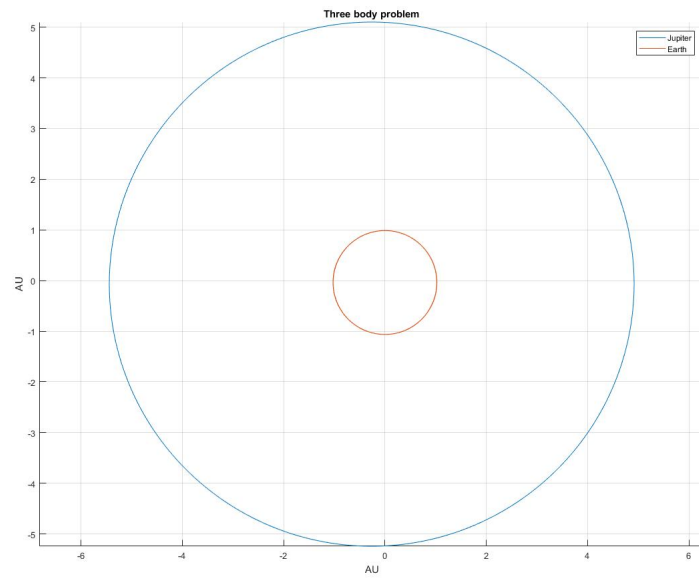


Figure 4: The planetary orbits of Earth and Jupiter around the Sun, with no other interfering forces. Calculated over 50 years with 10000 integration per year. In the last picture the Sun is not fixed as center.

The orbit of Earth is affected by the gravitational force of Jupiter. The change is slight for the case with normal mass. When multiplying Jupiters mass by ten, Earth still manages to stay in a slightly shifted orbit. When the mass is multiplied by 1000, Earths' orbit gets unstable. This is to be expected, as  $M_{jup} * 1000 \approx 0.95M_{\odot}$ . The Sun is not fixed at the center in the last picture, as we wanted to see the development of the free system (figure 4).

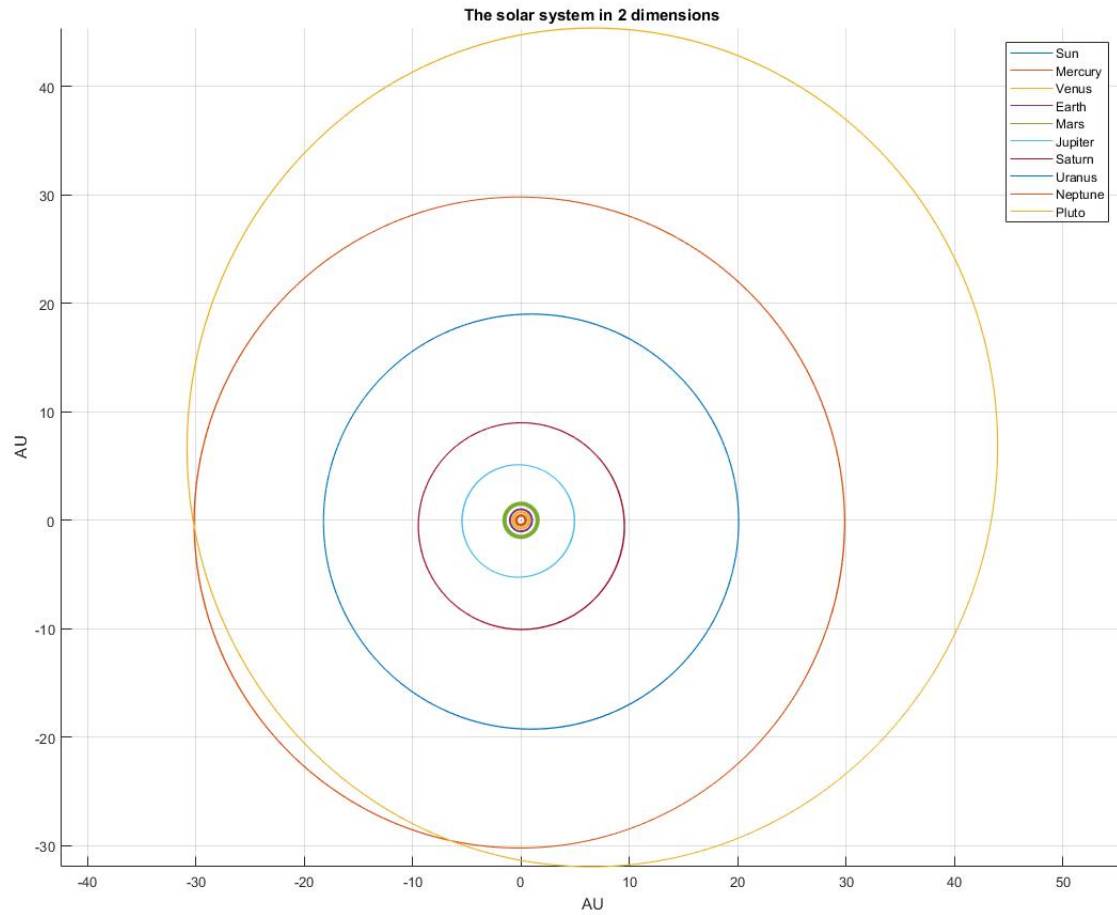


Figure 5: The planetary orbits represented in two dimensions for with  $10^5$  integration steps over 500 years.



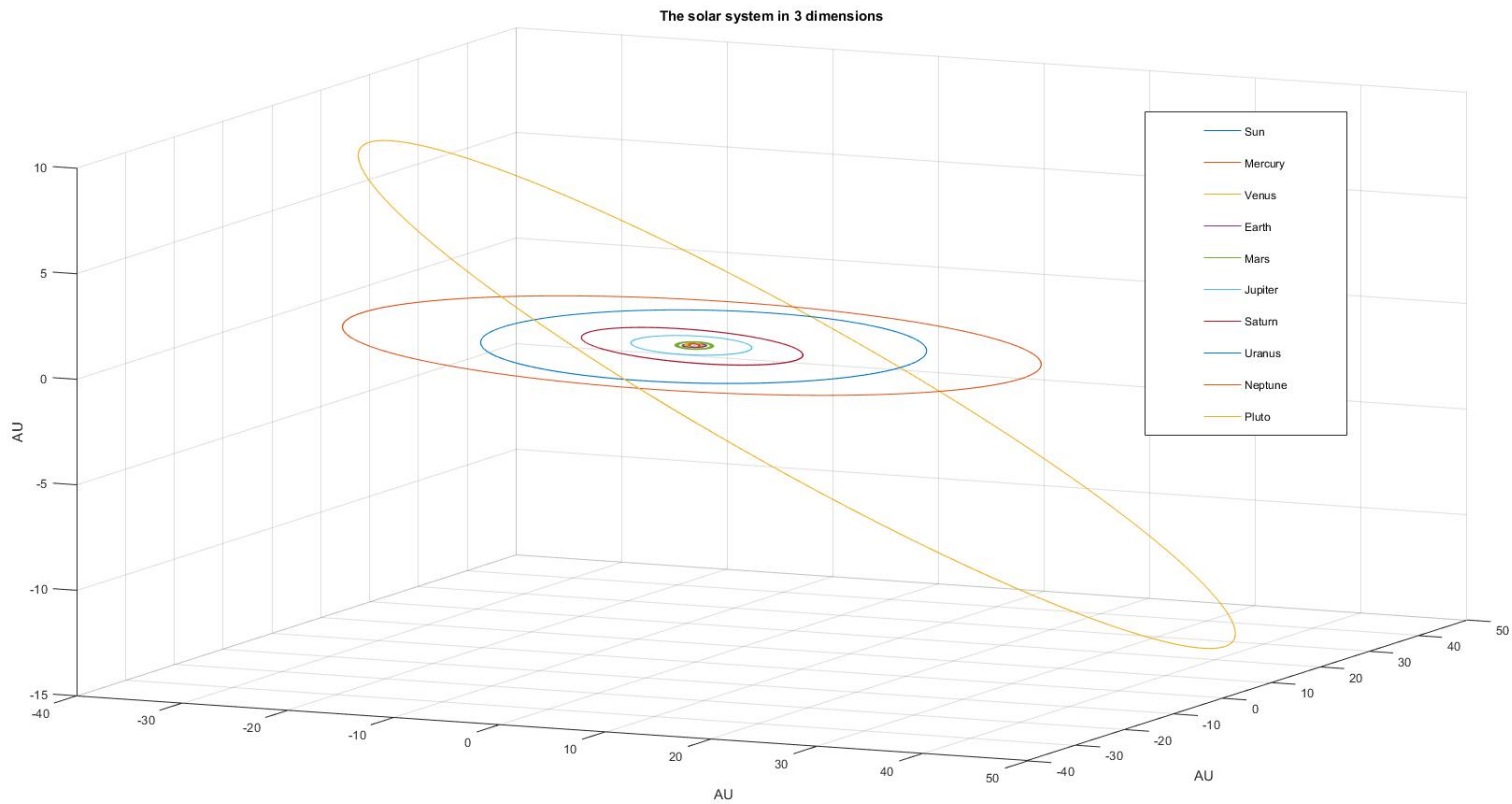


Figure 6: The planetary orbits represented in three dimensions for with  $10^5$  integration steps over 500 years.

In figure 5 and 6, the Solar System is depicted in both two and three dimensions. Almost all the planetary orbits have stable orbits in the x-y plane. The biggest exception is Pluto (Figure 6).

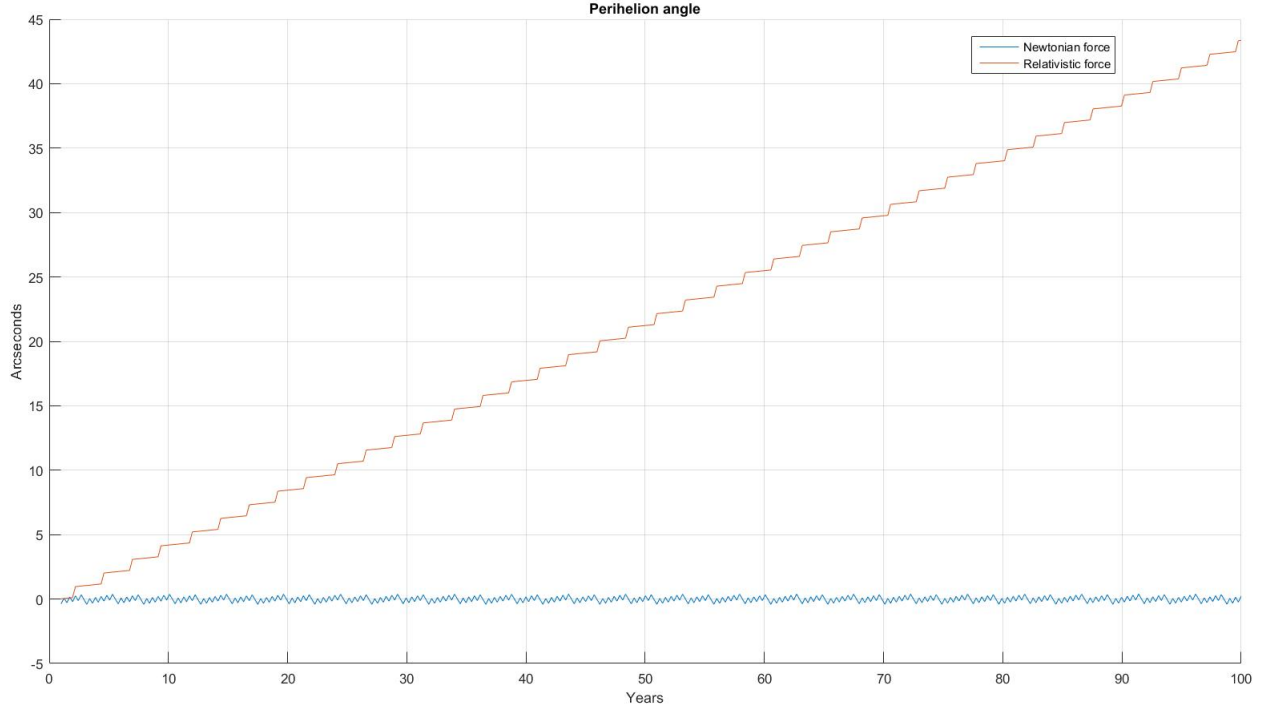


Figure 7: The perihelion precession of Mercury computed for both newtonian and relativistic forces over one century with 1 billion integration points

As stated earlier, the perihelion of Mercury is shifted slightly every year. The perihelion precession was calculated to be  $\approx 43$  arcseconds per century. To obtain an accuracy of this level, it was crucial to use many integration points. The perihelion of Mercury is not shifted if the forces used are Newtonian without relativistic correction. This is because a force depending on the pure Newtonian force will move in a perfect orbit, returning to the exact same point after one orbit.

## Discussion

The Euler forward method is easy for the computer to handle, but it is not robust for few integration points. The number of FLOPS in the Euler forward algorithm is  $5n$ -FLOPS. Compared to the velocity Verlet algorithm, that uses  $9n$ -FLOPS, the Euler algorithm is faster. This is shown in table 1. The velocity Verlet is more precise per rotation, hence is more robust for fewer integration points. The velocity that makes a circular orbit is  $v = 2\pi$ . This is pretty logical given the choice of units. As the radius of Earth's orbit per definition is 1 AU and the circumference is given by  $2\pi r$ , the velocity of Earth has to be  $2\pi \frac{\text{AU}}{\text{year}}$  in order to do a full rotation per year. The potential and kinetic energy oscillates, but is conserved over one orbit (figure 2). The angular momentum is also conserved. These quantities should be conserved for a planet in an idealized circular orbit, as the radius and velocity is constant. The total angular momentum a closed system must be conserved. Angular momentum can be transferred to other objects in the closed system, but the total will remain constant (Crew, 1908)

The escape velocity of Earth is exactly  $v_e = 2\sqrt{2}\pi$ . This corresponded quite well with the velocity found by trial and error. When the gravitational force was changed by replacing  $r^2 \rightarrow r^\beta$ , where  $\beta \in (2, 3)$ , the orbit got unstable. The gravitational force will be stronger for low  $r$ , resulting in higher velocities. The force will decrease more rapidly at higher  $r$ . This works like a slingshot-effect by having a greater force when the planet is far away from a star and a lower force when the planet is closer to the sun. Stable orbits harder to obtain in such a system.

Jupiter does not impact Earth's orbit much with its original mass (Figure 4a). Nor does it destabilize Earth's orbit much when the mass is multiplied by 10, though there is a slight change (figure 4b). When the mass is multiplied by 1000, Jupiter has almost as high mass as the sun. Earth soon escapes for the Solar System when Jupiter's mass is this high. Given these starting parameters, Jupiter and the Sun developed an interesting binary system. This is the reason why we chose the Sun not be stationary in this plot (Figure 4c). The velocity Verlet solver showed impressing stability for low  $n$ -values. It did not show significant large scale weakness before  $n \approx 50$  per year.

The final payoff of this article are the models of the entire Solar System (Figure 5, 6). In this system, the Sun is also moving. The movement of the Sun is almost negligible compared to all the other planets. This makes total sense, as the mass of all the planets combined only adds up to about 1 percent of the solar mass.

Last but not least, the perihelion precession was found to be  $\approx 43$  arcseconds per century (Figure 7). To find this precession, it is important to use a high number of integration points. The detected precession of the perihelion is  $\frac{44}{3600}$  of one degree, so this explains the requirement of a high number of integration points.

## Conclusion

Velocity Verlet proved to be the best algorithm to use for this project. The improved accuracy more than weigh up the relative low difference in computing times. The escape velocity was found exactly, and by trial and error to be  $v_e = 2\sqrt{2}\pi$ , while the velocity giving a perfect circular orbit was  $v = 2\pi$ . Changing the Newtonian gravity proved to be a bad choice if we want a solar system with stable orbits. The presence of Jupiter does not alter Earths' orbit to a big extent. If the mass of Jupiter is increased by a factor of 10, earth still manages to stay in orbit, though shifted a bit. Increasing the mass by a factor of 1000 made the earth leave the solar system altogether. The perihelion precession was found to  $\approx 43$  arcseconds for one century. To find this quantity, the resolution had to be high, and was finally found by using 1 billion integration points over 100 years.

## References

Crew, Henry (1908). The Principles of Mechanics: For Students of Physics and Engineering. Longmans, Green, and Company, New York. p. 88

Hjort-Jensen,M., 2015. Computational physics, accessible at course github repository. 551 pages