FYS4150
Project 3, deadline October 25.

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

Abstract

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 suns position will be fixed. This approximation is okay to make, as the earths mass only is three millionths of the suns 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 newtons second law of motion it it is easy to derive the acceleration.

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

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

Eulers 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:

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

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

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

The same algorithm applies for y. 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. Once again, the algorithm is viable for x- and y-dimension.

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

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

¡¡¡¡¡¡¡ HEAD

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

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

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

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

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.

$$F_{Earth-Jupiter} = G\frac{M_{earth}M_{jupiter}}{d_{Earth-Jupiter}}$$

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

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

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 headache. 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 members in code language, like initial position and velocity as well as mass. Many more members could be assigned to the planet and the code would still work.

Each planet was made as an object, passing initial values into our body class. This class takes the 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. 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 and 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. By conservation of energy:

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

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 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 see clearly if the 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 neighboring 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.

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 see clearly if the 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 neighboring 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.

This project is well suited for an object oriented code since we have planets with given properties, or members in code language, like initial position and velocity as well as mass. Many more members could be assigned to the planet and the code would still work.

Each planet was made as an object, passing initial values into our body class. This class takes the 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. 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 and 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.

Results

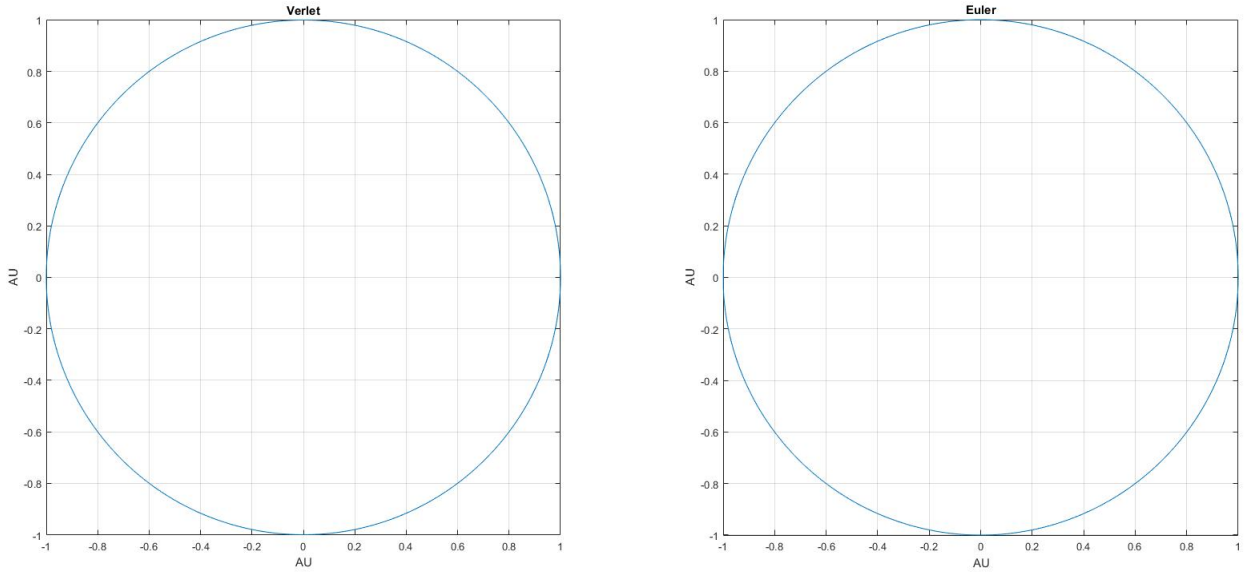


Figure 1: The planetary orbit of Earth around the sun, with no other interfering forces. Calculated over 100 years with 10000 integration steps per year. This figure was created by using the velocity verlet algorithm

Figure 1 is showing two models for the Earth-Sun binary system. Both are showing nearly perfect circles, but by playing around with $\triangle t$, 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.

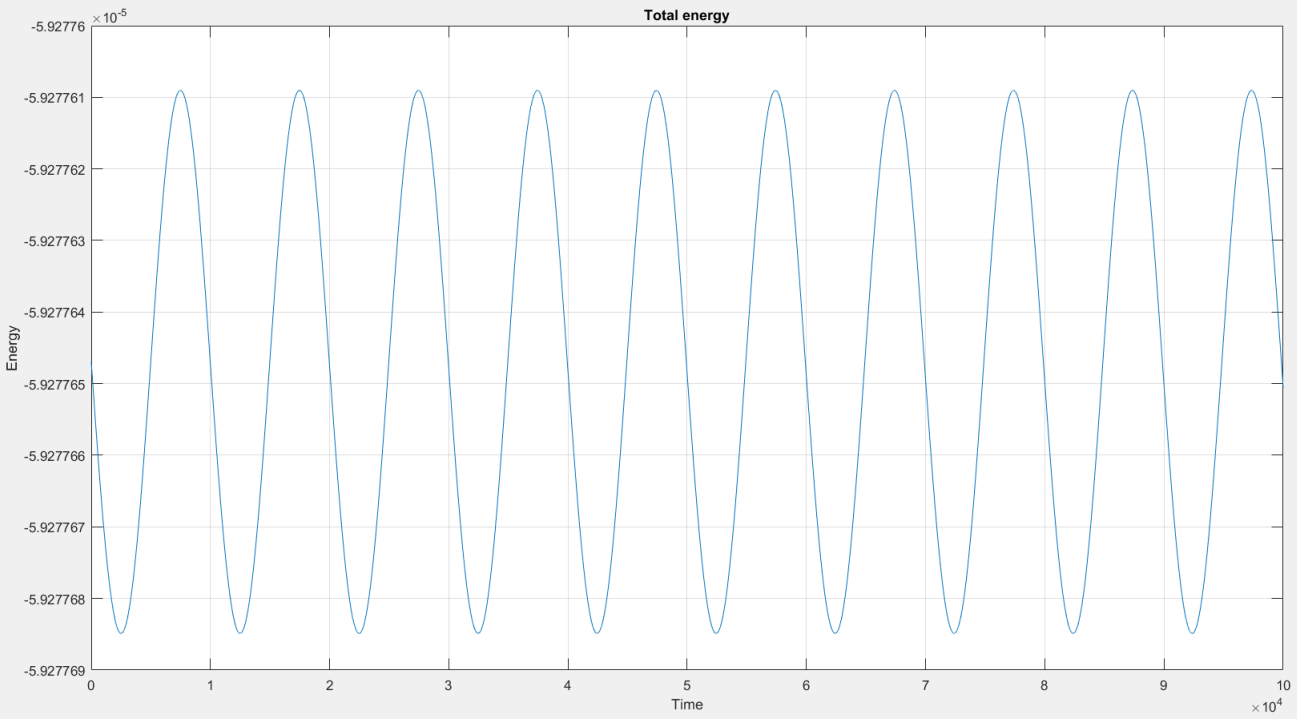


Figure 2: The total energy over 10 years, using Eulers forward algorithm.

As there is a circular orbit, the kinetic and potential energies should be conserved, but there will be small small oscillations in energy.

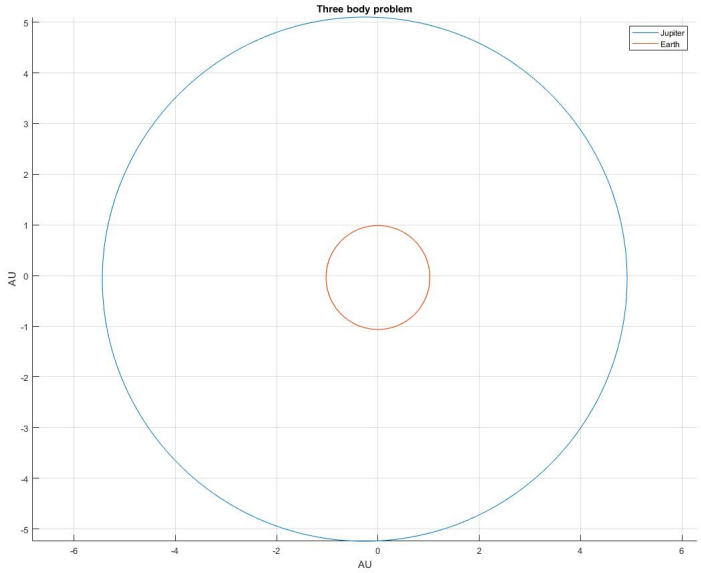
The exact escape velocity of Earth was found to be $v = 2\sqrt{2}\pi = 8.8857$. By trial and error, the escape velocity was found to be ≈ 8.8, which is pretty close to the actual velocity. By replacing the gravitational force from

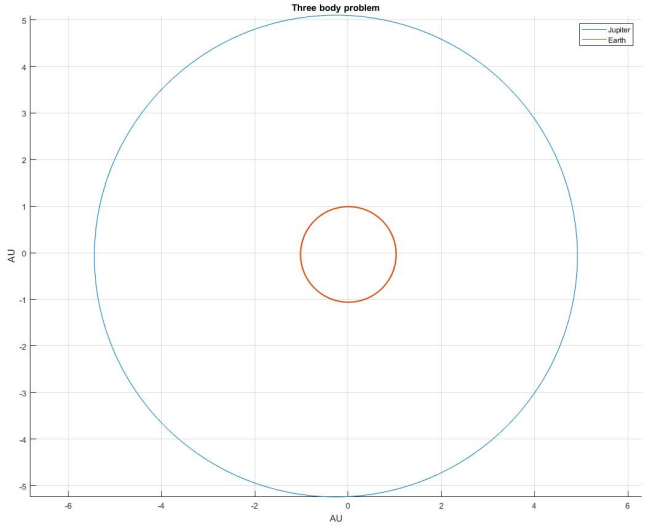
$$F_G = \frac{GM_\odot M_{\text{Earth}}}{r^2}$$

to

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

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





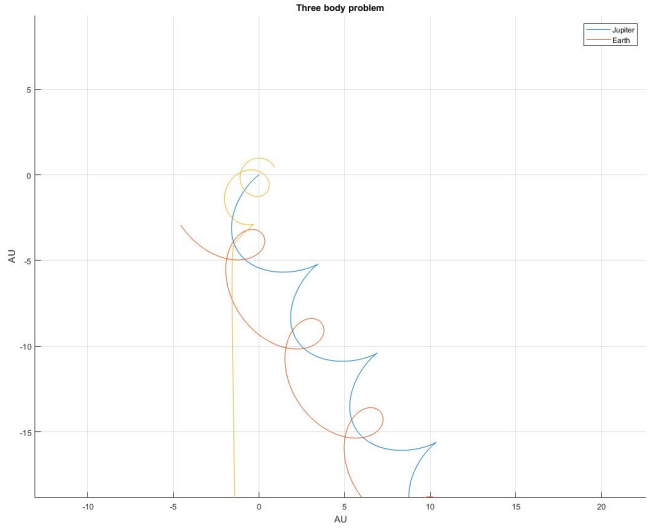


Figure 3: The planetary orbits of Earth and Jupiter around the sun, with no other interfering forces. Calculated over 50 years with 10000 integration per year. In the last picture the sun is not fixed as center.

The orbit of Earth is affected by the gravitational force of Jupiter. The change is slight for the 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 $M_{jup} * 1000 \approx 0.95 M_{\odot}$. The sun is not fixed at the center in the last picture, as we wanted to see the development of the free system (figure 3).

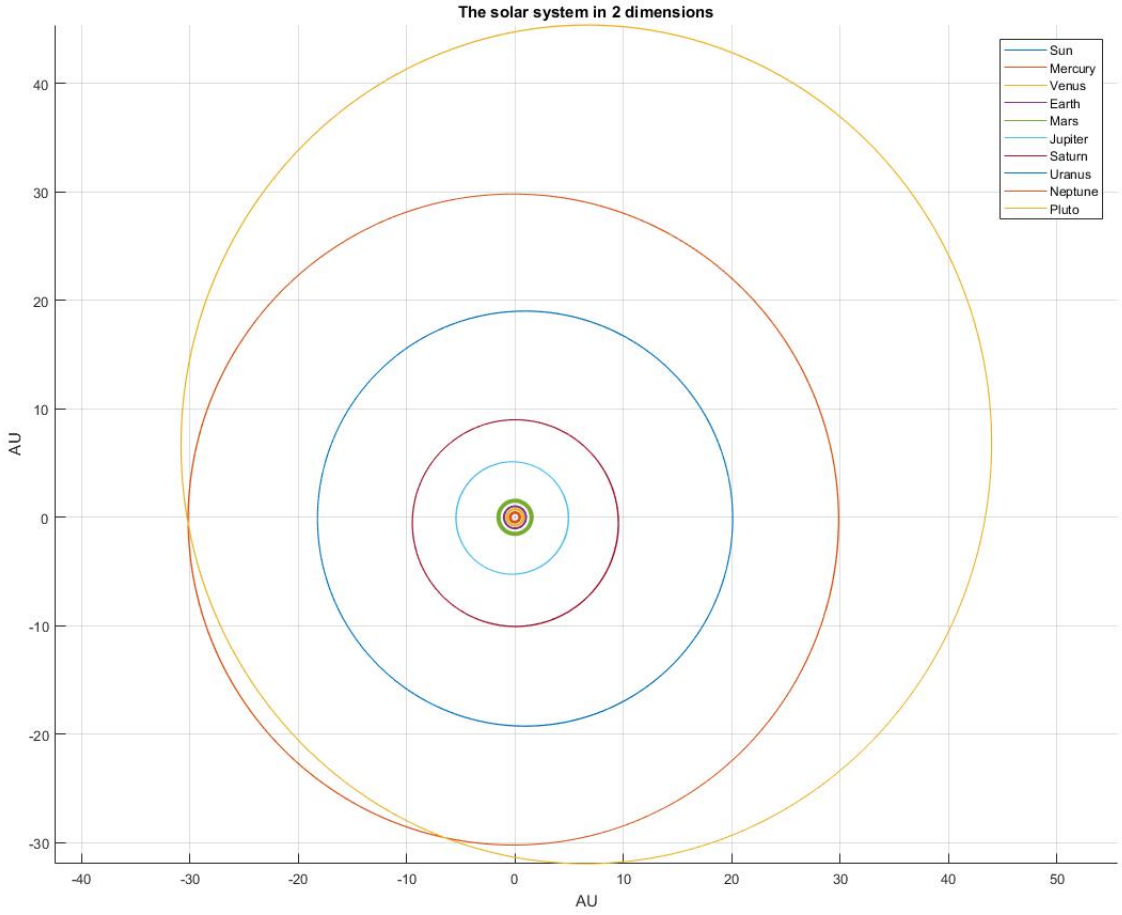


Figure 4: The planetary orbits represented in two dimensions for with 10^5 integration steps over 500 years.

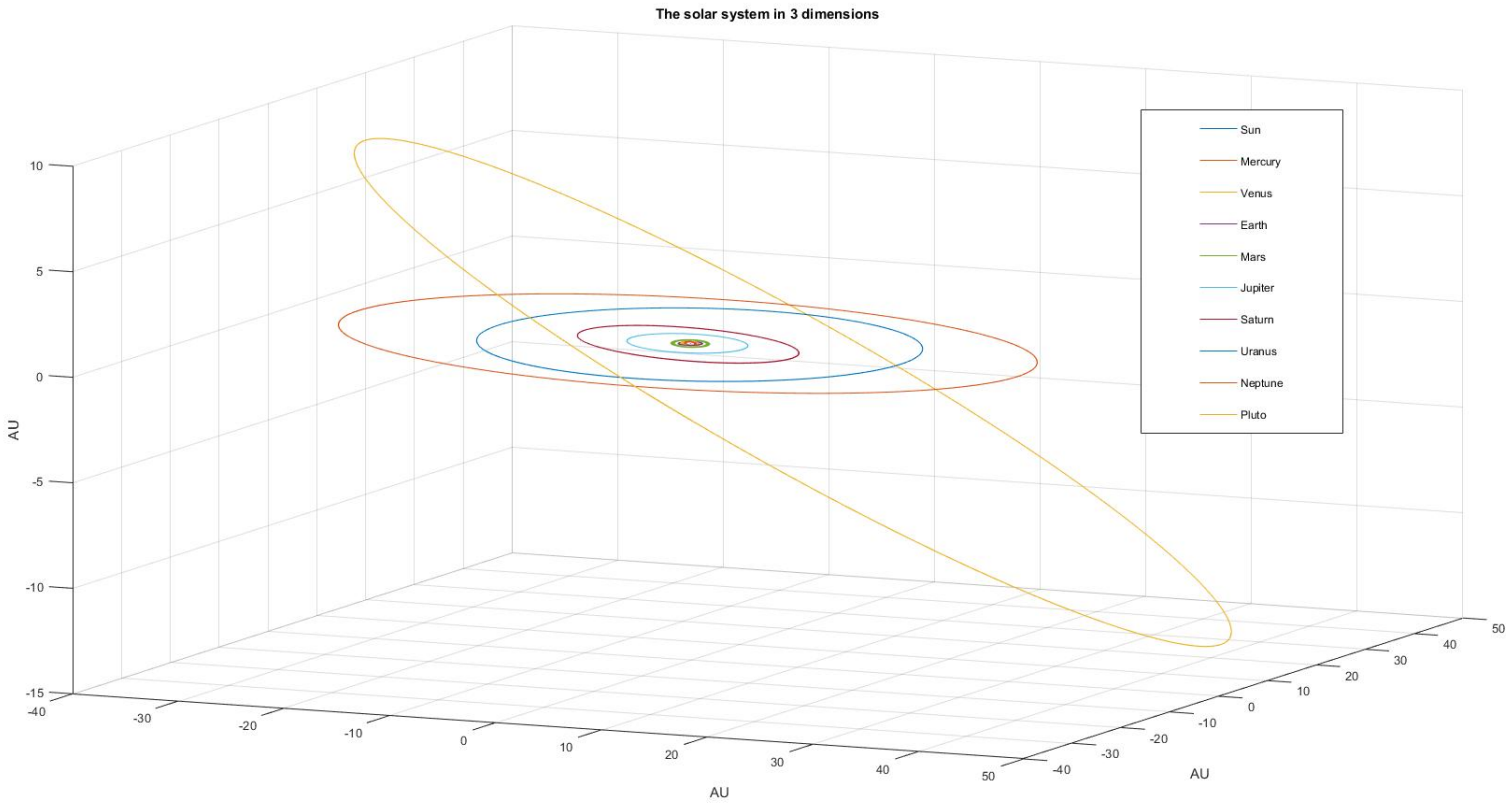


Figure 5: The planetary orbits represented in three dimensions for with 10^5 integration steps over 500 years.

In figure x and y, the solar system is depicted in both 2d and 3d. Almost all the planetary orbits are have stable orbits in the x-y plane. The biggest exception is Pluto (figure y).

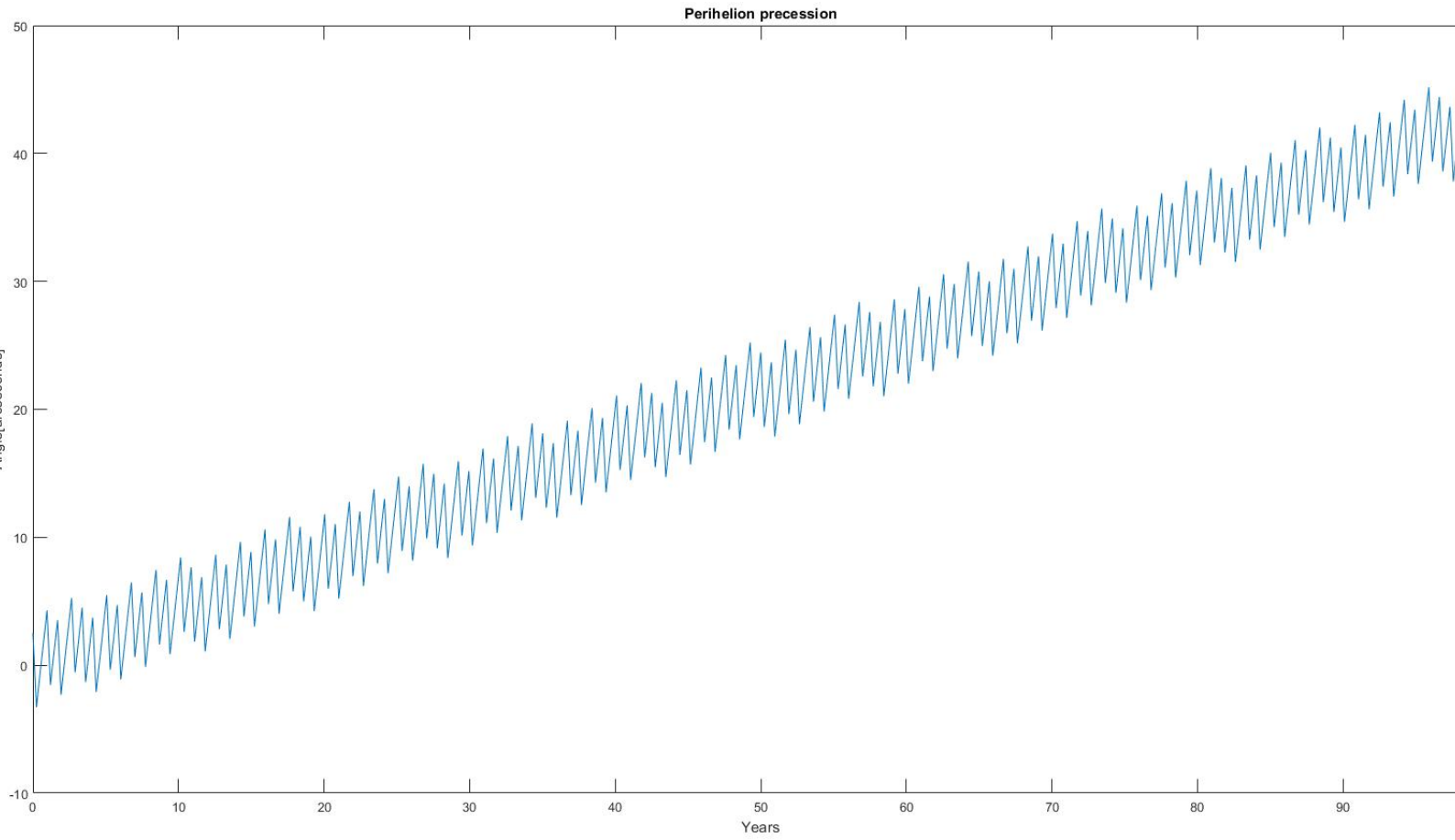


Figure 6: The perihelion precession of Mercury computed over one century with 100 million integration points

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 gets more out of each 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 Earths orbit per definition is 1 AU and the circumference is given by $2\pi r$, the velocity of Earth has to be $2\pi \frac{AU}{year}$ in order to do a full rotation per year. The potential and kinetic energy oscillates, but is generally conserved over one orbit(figure 2). The angular momentum is also conserved. These quantities should be conserved if a planet is an idealized circular orbit, as the radius and velocity is constant.

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 $r^2 \to r^\beta$, 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 can work like a slingshot-effect, making stable orbits harder to obtain.

Jupiter does not impact Earths orbit much with its original mass (Figure 3a). Nor does it destabilize earths orbit much when the mass is multiplied by 10, though there is a slight change (figure 3b). 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. It was due to this that the sun was chose to not be stationary in this plot(Figure 3c). 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.

Conclusion

References