MODELLING THE SOLAR SYSTEM WITH ORDINARY DIFFERENTIAL EQUATIONS

SHAFAQ NAZ SHEIKH
MOONIS ALI KHALID
HTTPS://GITHUB.COM/MOONNESA/FYS3150-2020

Abstract. The main aim of this project is to simulate the solar system using coupled ordinary differential equations. We solve these ODE's numerically by implementing two algorithms, Euler's Forward algorithm and the Velocity Verlet algorithm. In addition we want to object orient our code, this is because there are several coupled ordinary differential equations where the basic equations, except for the various physical constants and variables, are rather similar, so writing the code once, and running it many times, seems like a natural option. We find that the velocity verlet method is superior, when compared with Euler's Forward method.

Contents

Date: 22nd October 2020.

1. Introduction

Differential equations are an essential tool in wide range of applications. This is because many physical phenomena can be modelled by a relationship between a function and its derivatives. Many naturally occurring quantities can be represented as mathematical functions. This includes physical quantities like position, velocity and acceleration, which may vary in both space and time. Thus one of the many applications of differential equations to physics, is the modelling of the solar system. The basic equations which govern the solar system are rather simple, a set of coupled equations that are based on Newton's law of motion due to the gravitational force. Therefore we have to compute the orbits of the planets. In order to do this we study and implement two algorithms, the Forward Euler method and the velocity verlet method. These methods will help us solve the equations numerically. First we introduce the equations that govern the motion of the solar system, then we explain the algorithms that we are going to use. We then give an overview of object orientation, as this is something we wish to utilize in this project. Finally we present our results and discuss our findings before giving a conclusion.

2. Theory

2.1. **The differential equations.** In order to study the solar system, we have to first consider the gravitational force. This is given by Newton's law of gravitaion. The equation for Newton's law of gravitation is:

$$F = G\frac{m_1 m_2}{r^2} \tag{1}$$

where F is the gravitational force acting between two objects, m_1 and m_2 are the masses of the objects, r is the distance between the centers of their masses, and G[1] is the gravitational constant. By using Newton's second law of motion, we get the the following componential equations, in three dimensions, we would get three equations, as we are working with x, y and z in cartesian coordinates.

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{m_i}, \quad \frac{d^2y}{dt^2} = \frac{F_{G,y}}{m_i}, \quad \frac{d^2z}{dt^2} = \frac{F_{G,z}}{m_i} \tag{2}$$

where $F_{G,x}, F_{G,y}$ and $F_{G,z}$ are the x, y and z components of the gravitational force and m_i is the mass a particular celestial body i. The equations shown in (2) are second-order ordinary differential equations. However, we can rewrite them as two coupled first-order ordinary differential equations, in each coordinate x, y and z.

In order to do this, we have to consider Newton's second law, we know that $\sum \vec{F} = ma$. We get an expression for the relationship between the acceleration of an object, and the force acting on it,

$$\vec{a_i} = \frac{\vec{F_i}}{m_i}$$

[1]In SI units G is approximately $6.674 \times 10^{-11} m^3 kg^{-1} s^{-2}$

where i is the particular body we are looking at. In addition, we have to look at the relationship between acceleration, velocity and position. Let a denote the acceleration, v denote the velocity and x denote the position, then in general we have that

$$\frac{dv}{dt} = a, \quad \frac{dx}{dt} = v$$

This is equivalent to

$$v(t) = x'(t)$$
$$a(t) = v'(t) = x''(t)$$

In three dimensions, we get these six coupled ordinary differential equations, we get two equations for each coordinate.

$$a_x = \frac{dv_x}{dt}, \quad v_x = \frac{dx}{dt} \tag{3}$$

$$a_y = \frac{dv_x}{dt}, \quad v_y = \frac{dy}{dt} \tag{4}$$

$$a_z = \frac{dv_x}{dt}, \quad v_z = \frac{dz}{dt} \tag{5}$$

Here we calculate the acceleration from Newton's law of gravitation. In vector form, in three dimensions, it becomes

$$\vec{F}_{ij} = G\frac{m_i m_j}{r^3}\vec{r} \tag{6}$$

This is similar to equation (1), here $\vec{F}_{ij}$ is the gravitational force between the two bodies i and j, and the vector $\vec{r}$ is the position of the body in cartesian coordinates.

2.2. **Units.** Since we are working with a physical system, we have to consider the units. We will be using astronomical units(AU) to measure length. One astronomical unit of length, known as 1 AU, is the average distance between the sun and the earth, that is $1AU = 1.5 \times 10^{11}m$. When it comes to time, we will use years instead of seconds, as this matches much better with the time evolution of the solar system. We will express the mass of the celestial bodies as a fraction of the mass of the sun. The mass of the sun is given by $M_{sun} = M_{\odot} = 2 \times 10^{30}kg$. For example, if we consider Earth, then the mass of the Earth is given by $6 \times 10^{24}kg$. Then the mass of the Earth relative to the sun will be given by $M_{Earth} = 6 \times 10^{-6}$.

We can find the units for the gravitational constant G, by looking at the Earth-Sun system. We assume that the orbit of the Earth is almost circular around the sun. For circular motion we know that the force must obey the following relation

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

Rearranging the constants, we get

$$v^2 r = GM_{\odot} = 4\pi^2 AU^3/yr^2$$

Thus, the units for the gravitational constant are;

$$G = 4\pi^2 AU^3/(yr^2 M_{\odot})$$

Using the later equation we can easily find the velocity for which the Earths orbit around the Sun will be Circular, doing so we find;

$$v = \frac{2\pi}{\sqrt{r}} AU/yr$$

for the Earth to have a Circular orbit around the Sun.

2.3. **Discretization.** Before we present the algorithms used to solve the ODE's, we have to say something about the differential equations. In order to solve the equations (3-5) numerically, we have to discretize them first. This is done in the same way as in the previous projects. However, we now have an initial value problem.

We discretize the time steps, we first have to define a value for T_0, which is the initial time, and T_{max}, which is the final time. We have to choose some N, which is the total number of integration points. Then the discretized time values are given by,

$$t_i = T_0 + hi \tag{7}$$

for $i = 0, 1, 2, ..., N$. Where h is the step size, it is defined as,

$$h = \frac{T_{max} - T_0}{N} \tag{8}$$

Since this is an initial value problem, we have to provide initial values before we can start to compute the solutions. We have to provide the initial position for the x, y and z coordinates, and we have to provide the initial velocity for each coordinate. We have to provide six initial conditions,

$$\text{initial positions} = (x_0, y_0, z_0)$$
$$\text{initial velocity} = (v_{x_0}, v_{y_0}, v_{z_0})$$

We wish to compute the orbits of the planets in our solar system as accurately as possible. We extract our initial conditions from data provided by NASA[2]. We can get initial conditions in all three dimensions. At the mentioned website, we need to change from OBSERVER to VECTOR and then write in the planet we are interested in. The generated data contains the x, y and z values, as well as their corresponding velocities.

2.4. **Energy Conservation.** We want to study Newton's law of gravitation in the form shown below, that is, we want to study what happens as β varies.

$$\vec{F}_{ij} = G\frac{m_i m_j}{r_{i,j}^{\beta}}\vec{r} \tag{9}$$

If we study the Earth-Sun system, and we approximate the sun to be the center of mass, then the only force acting in the system is the gravitational force. More specifically, it is the gravitational force acting on the earth from the sun.

$$\vec{F} = G\frac{M_{\odot} M_{Earth}}{r^{\beta}}\vec{r} \tag{10}$$

Here, $\vec{r}$ is the position vector of the Earth. When studying the system in two dimensions, we can represent the system with two coordinates, namely

[2]`https://ssd.jpl.nasa.gov/horizons.cgi#top`

x and y. The kinetic energy minus the potential energy of the system is given by equation (9):

$$E_{total} = \frac{1}{2}M_{Earth}\dot{r}^2 - V_{eff}(r) \tag{11}$$

Here $V_{eff}(r)$ is called the effective potential, it is defined as

$$V_{eff}(r) = \frac{1}{2}\frac{l^2}{M_{Earth}}\frac{1}{r^2} - \frac{1}{\beta - 1}\frac{GM_{\odot}M_{Earth}}{r^{\beta-1}}$$

In order for this effective potential to have a stable equilibrium, we must have $\beta < 3$. The orbit of the Earth is effected by these β values, and for $\beta < 3$ the obit will be stable.

2.5. **Escape velocity.**

Let us consider the earth-sun system once again. Let earth begin at a distance $1AU$ from the sun. Then we want to find the initial velocity, such that, starting at this velocity, will lead the earth to escape from the sun, that is, we want to find the escape velocity. Escape velocity is defined as the minimum speed needed for the earth to escape from the gravitational influence of the sun, to achieve an infinite distance from it.

The escape velocity can be found by setting the total energy of the earth to be greater than zero. We set up the inequality and solve it for v,

$$E_{earth} = \frac{1}{2}M_{Earth}v^2 - G\frac{M_{\odot}M_{Earth}}{r^2} > 0 \tag{12}$$

Where M_{Earth} is the mass of the earth, $M_{\odot}$ is the mass of the sun and G is the gravitational constant.

$$\begin{aligned}
\frac{1}{2}M_{Earth}v^2 - G\frac{M_{\odot}M_{Earth}}{r^2} > 0 & \\
\frac{1}{2}M_{Earth}v^2 &> G\frac{M_{\odot}M_{Earth}}{r^2} \\
v^2 &> \frac{2GM_{\odot}}{r^2} \\
v &> \sqrt{\frac{2GM_{\odot}}{r^2}} \\
v &> \sqrt{8\pi^2} = 2\pi\sqrt{2}
\end{aligned}$$

By inserting the appropriate units for time, length and masses, we get $v > 2\pi\sqrt{2}$. We use $G = 4\pi^2$, the mass of the sun relative to itself is 1, so $M_{\odot} = 1$, and $r = 1AU$. In order for the earth to escape the gravitational pull of the sun, the inital velocity must be equal to $2\pi\sqrt{2}AU/yr$.

2.6. **Three Body System.**

It is rather straightforward to add a new planet, say Jupiter to a two body system and make it a three body system. Jupiter has mass $M_J = 1.9 \times 10^{27}kg$, and distance from the Sun of $5.2AU$.

The additional gravitational force the Earth feels from Jupiter in the x-direction is

$$F_x^{EJ} = -\frac{GM_JM_E}{r_{EJ}^3}(x_E - x_J)$$

where E stands for Earth, J for Jupiter, r_{EJ} is distance between Earth and Jupiter.

$$r_{EJ} = \sqrt{(x_E - x_J)^2 + (y_E - y_J)^2}$$

and x_E ad y_E are the x and y coordinates of earth, respectively, and x_J and y_J are the x and y coordinates of Jupiter, respectively. The x-component of the velocity of the earth changes thus to

$$\frac{dv_x^E}{dt} = -\frac{GM_\odot}{r^3}x_E - \frac{GM_j}{r_{EJ}^3}(x_E - x_J)$$

we can rewrite this to

$$\frac{dv_x^E}{dt} = -\frac{4\pi^2}{r^3}x_E - \frac{4\pi^2 M_J/M_\odot}{r_{EJ}^3}(x_E - x_J)$$

where we used

$$GM_J = GM_\odot(\frac{M_J}{M_\odot}) = 4\pi^2(\frac{M_j}{M_\odot})$$

Similarly, for the velocity in y-direction we have,

$$\frac{dv_y^E}{dt} = -\frac{4\pi^2}{r^3}y_E - \frac{4\pi^2 M_J/M_\odot}{r_{EJ}^3}(y_E - y_J)$$

And we can find the velocity in z-direction the same way. Similar expressions apply for Jupiter. The equations for x and y derivatives are unchanged. These equations are similar for all other planets and as we will see later, it will be convenient to object orient this part when we program the full solar system.

2.7. **Object orientation.** One of the aims of this project is to use object oriented programming. The concept of classes and object-oriented programming first appeared in the Simula programming language in the 1960s. Simula was invented by the Norwegian computer scientists Ole-Johan Dahl and Kristen Nygaard, and the impact of the language is particularly evident in C++.

Write once and run many times, is one of the central points of object orientation. Different people put different meanings into the term object-oriented programming, some use the term for programming with objects in general, while others use the term for programming with class hierarchies. Here, we apply the second meaning. We can put related classes together in families such that the family can be viewed as one unit. This idea helps to hide details in a program, and makes it easier to modify or extend the program. A family of classes is known as a class hierarchy. As in a biological family, there are parent classes and child classes. Child classes can inherit data and methods from parent classes, they can modify these data and methods, and they can add their own data and methods. This means that if we have a class with some functionality, we can extend this class by creating a child class and simply add the functionality we need. The original class is still available and the separate child class is small, since it does not need to repeat the code in the parent class. Although we could write the code for this project without classes, the introduction of classes enables us to write code that is either more elegant, or easier to extend at a later stage.

3. Algorithms

3.1. **Euler's Forward algorithm.**

Euler's Forward method uses the first two terms of the Taylor expansion to approximate the next step. The method can be written as,

$$\begin{aligned}\vec{x}_{i+1} &= \vec{x}_i + h\vec{x'}_i \\ &= \vec{x}_i + h\vec{v}_i \\ \vec{v}_{i+1} &= \vec{v}_i + h\vec{v'}_i \\ &= \vec{v}_i + h\vec{a}_i\end{aligned}$$

Euler's Forward method is given by:

$$\vec{x}_{i+1} = \vec{x}_i + h\vec{v}_i \tag{13}$$

$$\vec{v}_{i+1} = \vec{v}_i + h\vec{a}_i \tag{14}$$

where h is the step size, as defined by (8), $\vec{x_i}$ is the position, $\vec{v_i}$ is the velocity and $\vec{a_i}$ is the acceleration. The algorithm can be written in psuedo code as it is shown in algorithm 1.

Algorithm 1 Euler's Forward method

```
define $h$
define the initial conditions $\vec{x}_0$ and $\vec{v}_0$
for i = 0, 1, 2, ..., N do
    find the acceleration $\vec{a}_i$
    $\vec{x}_{i+1} = \vec{x}_i + h\vec{v}_i$
    $\vec{v}_{i+1} = \vec{v}_i + h\vec{a}_i$
```

The number of FLOP's this algorithm uses is $4N$, this does not include the calculation of the acceleration. The local error in the algorithm is $O(h)$, but the global error in $O(h)$. Euler's Forward method does not conserve energy.

3.2. **Euler-Cromer's algorithm.**

Another variation of Euler's method is Euler-Cromer's method. Although we did not implement this method in our code, it is still useful to take a look at it. The Euler-Cromer method is given by

$$\vec{v}_{i+1} = \vec{v}_i + h\vec{a}_i \tag{15}$$

$$\vec{x}_{i+1} = \vec{x}_i + h\vec{v}_{i+1} \tag{16}$$

The pseudo code is quite similar to Euler's Cromer method, it is given in algorithm 2. Euler-Cromer's method also has a global error of $O(h)$, however this method is energy-conserving.

Algorithm 2 Euler-Cromer's method

```
define $h$
define the initial conditions $\vec{x}_0$ and $\vec{v}_0$
for i = 0, 1, 2, ..., N do
    find the acceleration $\vec{a}_i$
    $\vec{v}_{i+1} = \vec{v}_i + h\vec{a}_i$
    $\vec{x}_{i+1} = \vec{x}_i + h\vec{v}_i$
```

3.3. **Velocity Verlet algorithm.** The velocity verlet method belongs to a family of verlet methods. These methods are widely used as they are numerically stable and easy to implement. The velocity verlet method is given by,

$$\vec{x}_{i+1} = \vec{x}_i + h\vec{v}_i + \frac{h^2}{2}\vec{a}_i$$
$$\vec{v}_{i+1} = \vec{v}_i + \frac{h}{2}\Big(\vec{a}_i + \vec{a}_{i+1}\Big)$$

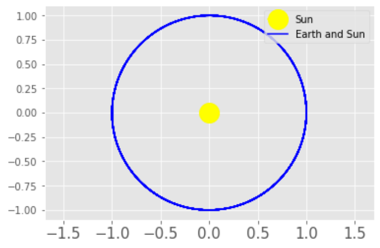
The pseudo code for the algorithm is given in algorithm 3. This method

Algorithm 3 Velocity Verlet method

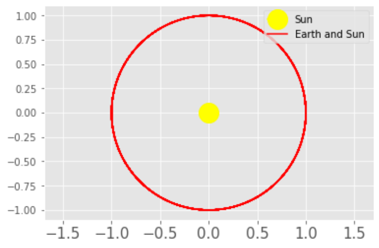
```
define $h$
define the initial conditions $\vec{x}_0$ and $\vec{v}_0$
for i = 0, 1, 2, ..., N do
    find the acceleration $\vec{a}_i$
    $\vec{x}_{i+1} = \vec{x}_i + h\vec{v}_i + \frac{h^2}{2}\vec{a}_i$
    find the acceleration in the next time step $\vec{a}_{i+1}$
    $\vec{v}_{i+1} = \vec{v}_i + \frac{h}{2}(\vec{a}_i + \vec{a}_{i+1})$
```

is energy conserving, it is much better than Euler's Cromer method. The number of FLOP's used in this algorithm is around $11N$. This method has a higher computational cost, compared to Euler's Cromer method, this is because we have to compute the acceleration twice, so more computations are needed.

3.4. **Object oriented implementation.** As i mentioned in the theory section, one of the aims of this project is to use object oriented programming, and that we have done. By object orienting our code we can reduce the number of FLOPS, one would use without Object oriented code. Our program is divided in three classes; class vec3, for defining a vector (this class was taken directly from the lecture slides), class Wanderer, where we define a wanderer object(planets and sun) with initial positions and velocities and the respective forces and accelerations and the distances, and a class Solvers, where we have the two solvers euler and verlet. By making a class wanderer, we are calculating the distances, forces and accelerations separately. And in class Solvers, we have a method euler, where we are calculating just the new positions and velocities, while in verlet, in addition to calculating the positions and velocities, we calculate acceleration one more time to get the respective acceleration value to the new positions that we have calculated



(A) Circular orbit with Euler Forward algorithm



(B) Circular orbit with Velocity Verlet algorithm

FIGURE 1. Circular orbit of the Earth with velocity 2π around the Sun at rest, time; 10 years and integration points; 1000, with both Euler Forward(A) and Velocity Verlet(B) algorithms

in the earlier step. And last, we have the main method in a different file, where i simulate the solar system with many wanderer objects, and solvers.

4. Results

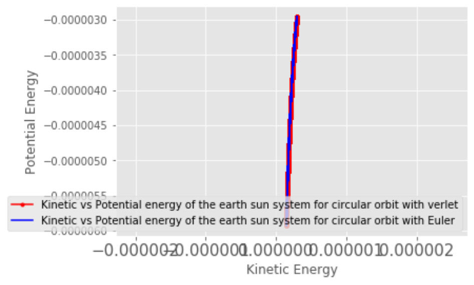
Now we will talk about my results from our program.

4.1. **For The Two Body System,** we found out, theoretically, that the analytical velocity for which earth should have a circular orbit is 2π AU/yr. In our program, we set earth's initial position vector equal to (1,0,0), means 1 on x-axis and 0 on y- and z-axis, and the initial velocity vector to be (0,2π,0) and we got a circular motion for the two body system. In figure 1 the reader can see for which time value and how many iterations the circular orbit was plotted with, in 2D. We also find out that the kinetic and the potential energies are conserved for both methods with a very small error.If we look at Figure 2(A). We can see that there is almost no difference between the kinetic and potential values we get from Euler and Verlet methods for a circular orbit. And since the values are almost the same for all distances we can say that these energies are conserved.

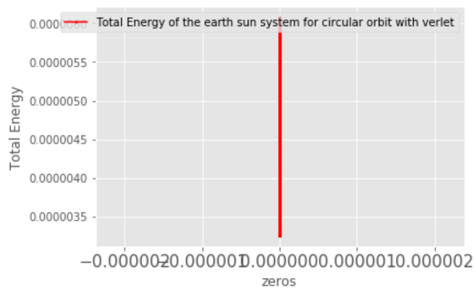
The total Energy of the Earth Sun system, we found with verlet algorithm, with time 10 years and 10000 iterations for velocity 5 AU/yr and distance 1 AU from the Sun at rest, comes out to be an infinitesimal number(a number very close to zero) but varying with every iteration with a very infinitesimal difference of 10^-6. We can say that its conserved. In Figure 2(B) we can see that the total energy values vary but with a very small difference.

When β creeps towards 4, for the Earth Sun System with distances other than 1, the whole system falls apart.

4.2. **The escape velocity.** for the Earth to escape Suns gravitational pull was found analytically to be $2\pi\sqrt{2}$. But it was very hard to find it by trial and error with our program or to plot it. Somehow we always end up with a closed elliptical orbit for the Earth no matter how many iterations and years we plug in for. We tried plotting the motion of the Earth by plugging the analytical velocity directly in our code for time; 45000 years



(A) Axis: x= kinetic and y=potential for both Euler(blue) and Verlet (red) methods



(B) Axis: x= Total Energy of the system and y= zeros for V-Verlet method

FIGURE 2. Energy Plots. (A) Kinetic vs Potential (B) Total Energy

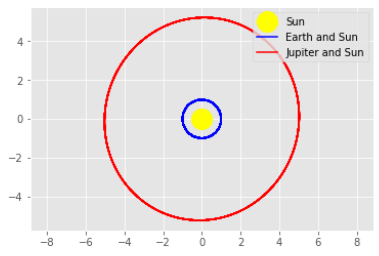
and 4500000 iterations but the plot always came out elliptical and without any breaks. We will talk about what this means for our program in the Discussion section.

4.3. **In The three body problem,** we add the forces between Earth and Sun and Earth and Jupiter together in computeTBGForce method in Wanderer class, and get the acceleration from the TBacceleration method in 'said' class, and use the TBverlet method i Solvers class to write the position values to a file and plot them with python. In Figure 4 the reader can see how the Earth is attracted by the Jupiter, with Jupiter's mass being 10 and 1000 times its original mass. We will discuss more in the Discussion section.

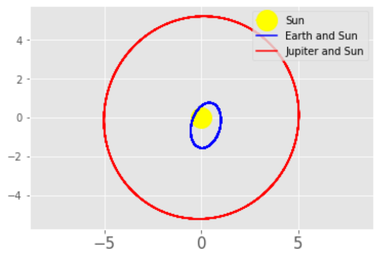
And at last, we plot the motion of all planets in the solar system with time; 100 years and 1000000 iterations. See Figure 5.

5. DISCUSSION

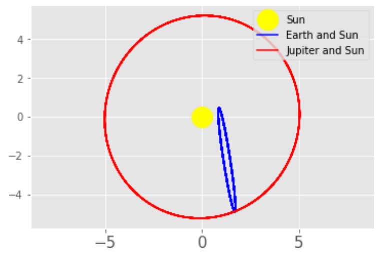
The Euler Forward and Velocity Verlet methods, both gave almost equal positions for iterations more then a 1000, but we can clearly see a difference between the precision they have when the iterations are under 50. In Figure 6 reader can see the Circular orbit of the Earth with 50 and 100 iterations and won't see much difference, but in Figure 7 the reader can easily spot the difference in precision for 10 iteration points. Euler Forward methods



(A) Using Velocity verlet algorithm with Jupiters mass 10 times bigger



(B) Using Velocity verlet algorithm with Jupiters mass 1000 times bigger



(C) Using Velocity Verlet algorithm with mass equal to Suns

FIGURE 3. Circular orbit of the Earth with velocity 2π around the Sun at rest, time; 10 years and integration points; 1000, with Jupiter attracting Earth and rotating around the Sun with mass 10 times (A), 1000 times (B) and 10000 times (C) its original size

gives a more elliptical orbit while Velocity Verlet still gives a circular orbit. But this precision comes at a price of 7n more FLOPS. Euler Forward only uses 4N FLOPS, while Velocity verlet uses 11N FLOPS.

Now lets talk about the energy conversion of the Earth sun system for both methods. As we saw in figure 2, both Kinetic energy and Potential Energy of the system changes for a circular orbit of the Earth Sun System. But the change is very small, can we then safely neglect this? Our guess is no! That must mean that our code is not the best code for simulating a very realistic Solar system. Since the total energies are not conserved either, this just points more and more to the fact that we suck at this.

One thing that we used most of our time on in this project was trying to find the escape velocity of the Earth from its orbit around the Sun. We must have tried 1000 different velocities for god knows how many different number of iterations, all more then 10000, but every time we get an elliptical orbit of the earth without any breaks, in both 2D and 3D. Then we just gave in and plugged in the analytical escape velocity 2π AU/Yr, just to see how it would look seeing the earth leaving the orbit, and tried 100000 iterations with time = 45000 years. It took 3 minuets for the computer to calculate and write the position values for every 100th iterations to a file and this took python around 30 seconds to bring up a plot for, and guess what, the

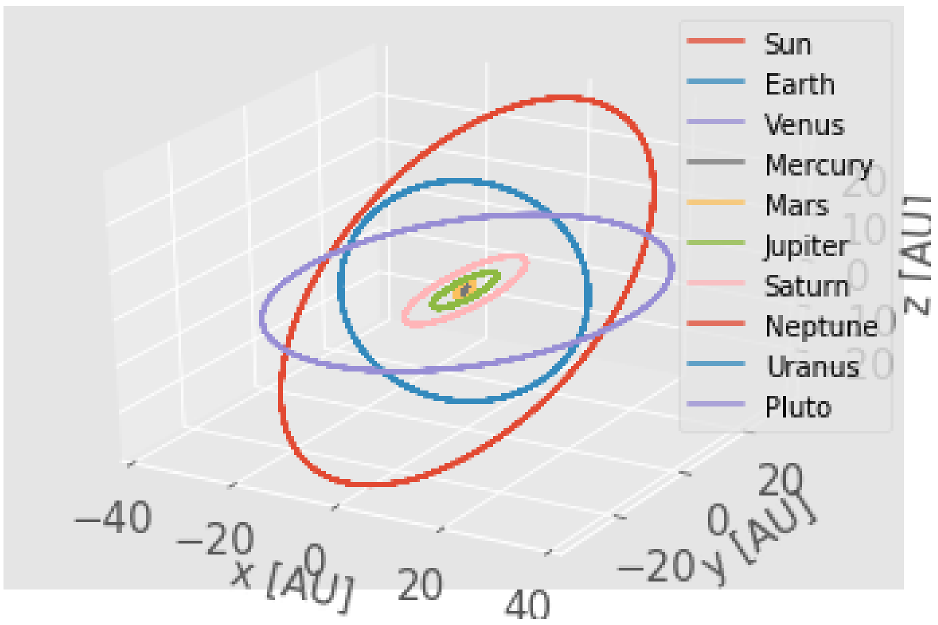
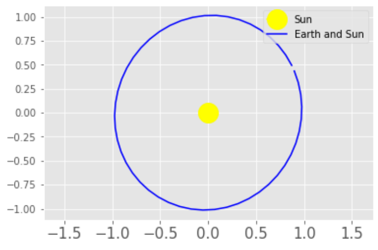


Figure 4. All Planets (with Pluto) with initial positions and velocities from the NASA site around the Sun at rest, time; 100 years and integration points; 1000000, with Velocity Verlet algorithm

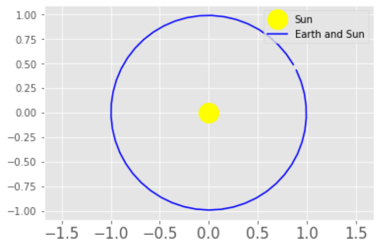
orbit was still elliptical. This can only mean one thing. Our code was not good enough for this calculation. We think the problem leis in how we find the distance between the wanderer objects.

In the Three Body problem we can begin to see in Figure 4 how Earth changes its orbit when Jupiter's mass is changed to 100 times its original mass. But here we can also spot a very clear error. Earth is only attracted to Jupiter in a vertical direction. The same happens when Jupiter's mass is 1000 and 10000 times its original. In a more realistic model with sun fixed as a the center of mass the Earth would go around both Jupiter and Sun before it comes to close to the Sun and crashes into the Suns surface. In another model where the Sun is not fixed the whole system will come apart and all the object would revolve around each other before crashing into each other. We can now safely say that this error occurs because of the same problem, the wrong calculation of the distance between the wanderer object in the distance method in the Wanderer class. But on the positive side, although the change in the earths orbit is just in one direction, we can clearly see that orbit changes when the Jupiter's mass is the same as the Suns and the Earth revolves around both Jupiter and The Sun.

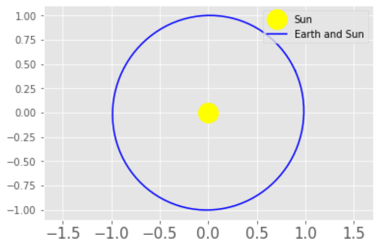
In the end, we were successful when it comes to modeling the movement of all the planets around the Sun fixed as the centre of mass.



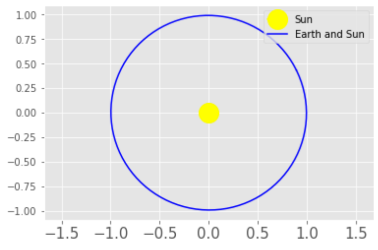
(A) Using Euler's Forward algorithm with time 1 year 50 iterations



(B) Using Velocity verlet algorithm with time 1 year and 50 iterations

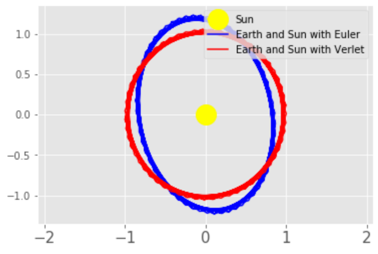


(C) Using Euler's Forward algorithm with time 1 year 100 iterations

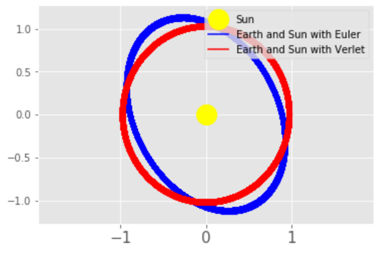


(D) Using Velocity Verlet algorithm with time 1 year and 100 iterations

FIGURE 5. Circular orbit of the Earth with velocity 2π around the Sun at rest, time; 1 year and integration points; 50 with Euler Forward(A), 50 with Velocity Verlet(B), 100 with Euler Forward(C) and 100 with Velocity Verlet(D), and distance of the Earth from Sun is 1 AU.



(A) Difference in Euler and Verlet with time 10 years and 10 iterations



(B) Difference in Euler and Verlet with time 1000 years and 10 iterations

FIGURE 6. Orbit of the Earth with velocity 2π around the Sun at rest, time; 10 years with Euler Forward(blue) and Velocity Verlet methods(red)(A) and time; 1000 years with Euler Forward(blue) and Velocity Verlet(red)(B), both with with 10 integration points and distance of the Earth from Sun is 1 AU.

6. Conclusion

For the two body system we found out that the earths orbit is circular for the initial velocity of 2π AU/Yr from a distance of 1 AU from the Sun. The total energy of the two body system is conserved for both methods but with a small tolerance of error, 10^-6. Euler Forward and Velocity verlet methods differ mostly when the iterations are less then 30. We found out that for 10 iterations Euler Forward gives an elliptical orbit for Earth around the Sun for an initial velocity 2π AU/yr, when its suppose to be circular, while Velocity verlet is still very Circular. The analytical escape velocity of the Earth around the Sun is $2\pi\sqrt{2}$ AU/yr, but we couldn't find it by trial and error nor were we successful in making a plot of earth leaving its orbit around the Sun, after plugging in the analytical escape velocity. For the three body problem, we found that when jupiters mass is 100 times more then its original mass, we can start to Earth orbit shifting towards Jupiter, and when Jupiter is 1000 times bigger, its a more prominent change in the earths movement. The Solar System was modeled using the data from NASA site and we saw all the planets revolving around the Sun fixed as the centre of mass in elliptical orbits.

7. References

- All codes can be found in the github repository `https://github.com/moonnesa/FYS3150-2020/tree/master/Project3`
- Langtangen, H., 2016. A Primer On Scientific Programming With Python. Heidelberg: Springer, pp.567,568.
- Mørken, K., 2017. Numerical Algorithms And Digital Representation. Blindern: Department of Mathematics, University of Oslo, pp.321-330.
- Morten Hjorth-Jensen. Computational Physics, Lecture notes, Fall 2015, chapter 8. 2015
- https://ssd.jpl.nasa.gov/horizons.cgiresults
- https://phys.libretexts.org/Bookshelves/University$_Physics/Bookhttp://www.sklogwiki.org/SklogWiki/index.php/Velocity_Verlet_algorithm$