Project 3 - FYS3150*

Andreas G. Lefdalsnes
Student: University of Oslo, Department of Physics
email-address: andregl@student.matnat.uio.no

Tellef Storebakken
Student: University of Oslo, Department of Physics
email-address: tellefs@student.matnat.uio.no
(Dated: October 24, 2016)

In this project we use object orientation to model the solar system. We implement the solar system using a C++ class based approach, and solve the differential equations given by Newton's laws to obtain the full motion. We solve the discretized version of the differential equations using Euler's method and the Velocity Verlet method and compare the efficiency and numerical stability. We also check the accuracy of our results using physical laws such as conservation of energy and momentum. We find that the Velocity Verlet method is more efficient for us in this project. The timing if the methods shows no significant difference, but the accuracy of the Verlet higher than Eulers method.

I. INTRODUCTION

In this project we apply the principles of Object Orientation to a system of several objects whose interactions are governed by the gravitational force. We study the solar system consisting simply of the Earth and the sun, and the full solar system consisting of all 8 planets in addition to Pluto and the sun. We model each celestial object as a point particle in 3-dimensional space, and Newton's laws give us a simple ordinary differential equation governing the time evolution of each celestial object. A class based approach in C++ allows us to easily extend the code to obtain the full motion of the solar system. We solve the differential equations numerically using a length and time scale appropriate for our problem.

In particular, we will be implementing Euler's method and the Velocity Verlet method. Before comparing the results, we will note that Euler's method, though simple, can be quite numerically unstable. The Velocity Verlet method is well suited to our problem, and gives a good tradeoff betweeen numerical efficiency and accuracy.

Finally we will examine the perihelion precession of Mercury. One of the early successes of the general theory of relativity was to explain the precession of Mercury as it orbited the sun, after all other pure newtonian effects had been accounted for. We will be modelling this by adding a relativistic correction, and checking to see whether this correction gives us an appropriate perturbation of the orbit.

* Computational Physics, autumn 2016, University of Oslo

II. THEORY AND METHODS

A. Newton's law of gravitation

Newton's law of gravitation states that for two objects of mass m_1 and m_2, the gravitational force on object 1 from object 2 is given by [**?**],

$$\boldsymbol{F_{1,2}} = \frac{Gm_1m_2}{r^2}\boldsymbol{u_r} = \frac{Gm_1m_2}{r^3}\boldsymbol{r} \tag{1}$$

where G is the gravitational constant and $\boldsymbol{u_r} = \boldsymbol{r}/r$ is a radial unit vector. $\boldsymbol{r}$ is a radial vector pointing at object 2 and $r = |\boldsymbol{r}|$ is the distance. Newton's third law gives us that the force on object 2 from object 1 is $\boldsymbol{F_{2,1}} = -\boldsymbol{F_{1,2}}$. Newton's third law gives us the differential equation governing the motion of object 1

$$\ddot{\boldsymbol{r}}(\boldsymbol{t}) = \boldsymbol{a}(\boldsymbol{t}) = \boldsymbol{F_{1,2}}(t, \boldsymbol{r}(\boldsymbol{t}))/m_1, \tag{2}$$

where $\boldsymbol{a}$ is the acceleration, and we can solve this equation to find the motion $\boldsymbol{r}(\boldsymbol{t})$. For a two-body system this equation will produce closed elliptical orbits around a common center of mass.
If we assume that the orbit of object 2 around object 1 is circular we know that the force obeys the following equation

$$F_{2,1} = \frac{Gm_1m_2}{r^2} = \frac{m_1v_1^2}{r}, \tag{3}$$

which implies that

$$v_1^2 r = Gm_2. \tag{4}$$

Introducing 1 AU $= 1.5 \cdot 10^{11}$m; the distance from the Earth to the sun, we let object 1 be the sun and object 2 the Earth, and we obtain

$$v_{earth}^2 r = GM_\odot = 4\pi^2 AU^3/yr^2, \tag{5}$$

where I have used that the circular velocity of the Earth is $v_{earth} = 2\pi r/T = 2\pi AU/yr$. This lets us rewrite our differential equation in a more natural length scale for the solar system. For a solar system with 1 sun and 9 planets (including Pluto) we may write the differential equation governing the motion of each celestial object i as

$$\begin{aligned} \ddot{\boldsymbol{r}}_{\boldsymbol{i}} = \frac{\boldsymbol{F_{tot}}}{M_i} &= \sum_{j\neq i} \frac{\boldsymbol{F_{i,j}}}{M_i} \\ &= -\sum_{j\neq i} \frac{4\pi^2 M_j}{r^3} \boldsymbol{r_{i,j}} \quad , \end{aligned} \tag{6}$$

where the acceleration $\boldsymbol{a} = \ddot{\boldsymbol{r}}$ is measured in units of $4\pi^2 AU/yr^2$. The minus sign stems from the fact that the position vectors point in the opposite direction of the force vectors. We may also measure the mass of each celestial object by solar mass, by setting $M_\odot = 1$.

B. Ordinary differential equations

We discretize the differential equation by setting the number of time steps n, the number of years we want to evolve the system y and the length of each time step $\Delta t = y/n$. The simplest way of solving equation (6) numerically is Euler's method [**?**]:

$$\begin{aligned} &\boldsymbol{v_{i+1}} = \boldsymbol{v_i} + \Delta t \cdot \boldsymbol{F_{tot}}/\boldsymbol{M_i} \qquad i = 0, 1, 2, .., n-1 \\ &\boldsymbol{x_{i+1}} = \boldsymbol{x_i} + \Delta t \cdot \boldsymbol{v_i} \end{aligned} \tag{7}$$

A significant improvement would be the Velocity Verlet algorithm. For a second order differential equation of the form

$$\frac{d^2}{dt^2}\boldsymbol{r} = \boldsymbol{F}(\boldsymbol{r}, t), \tag{8}$$

(such as ours!) we may rewrite our differential equations as:

$$\begin{aligned} \boldsymbol{r_{i+1}} &= \boldsymbol{r_i} + \Delta t \cdot \boldsymbol{v_i} + \frac{(\Delta t)^2}{2}\boldsymbol{a_i} \\ \boldsymbol{v_{i+1}} &= \boldsymbol{v_i} + \frac{\Delta t}{2}(\boldsymbol{a_{i+1}} + \boldsymbol{a_i}). \end{aligned} \tag{9}$$

Note that the velocity depends on the acceleration at time t_{i+1}. This is a function of $\boldsymbol{r_{i+1}}$ which needs to be calculated first. The error in this method goes like $\mathcal{O}(\Delta t^3)$ locally, which means we get a global error $\mathcal{O}(\Delta t^2)$, compared to Euler's method which has a total error $\mathcal{O}(\Delta t)$.

C. Conservation laws

In Newtonian mechanics, for a system of objects without any external forces we have three fundamental conservations laws: The conservation of energy, the conservation of momentum and the conservation of angular momentum.
For the solar system, if we neglect any forces other than the pure Newtonian gravitational force (and the celestial objects are taken to be point masses) we may write the total energy as

$$E_{tot} = \sum_{i\neq j} \frac{1}{2} m_i v_i^2 - \frac{G m_i m_j}{r_{i,j}}, \tag{10}$$

This is a conserved, or time invariant quantity; that is to say it must be the same in our calculations for all times t_i.
The total momentum of the system (no external forces applied) is also a conserved quantity and can be given as

$$\boldsymbol{p_{tot}} = \sum_i m_i \boldsymbol{v_i}. \tag{11}$$

If we wish to study our system from the gravitational center of mass at the origin with $\boldsymbol{p_{tot}} = \boldsymbol{0}$ we require

$$\begin{aligned} m_1\boldsymbol{r_1} + ... + m_N\boldsymbol{r_N} &= \boldsymbol{0} \\ m_1\boldsymbol{v_1} + ... + m_N\boldsymbol{v_N} &= \boldsymbol{0}. \end{aligned} \tag{12}$$

This gives us $N-1$ degrees of freedom to choose the initial positions and velocitites. We can then fullfill these conditions by requiring the sun to have an initial position and velocity

$$\begin{aligned} \boldsymbol{r_\odot} &= \frac{1}{M_\odot}(m_2\boldsymbol{r_2} + ... + m_N\boldsymbol{r_N}) \\ \boldsymbol{v_\odot} &= \frac{1}{M_\odot}(m_2\boldsymbol{v_2} + ... + m_N\boldsymbol{v_N}). \end{aligned} \tag{13}$$

As Newton's second law gives us conservation of momentum, a rotational analog of Newton's second law gives us conservation of angular momentum. For a system unaffected by external torque, the total angular momentum is conserved.

$$\boldsymbol{L_{tot}} = \sum_i \boldsymbol{r_i} \times \boldsymbol{v_i} \tag{14}$$

Note that momentum, energy and angular momentum will constantly be exchanged between the objects of the system via the gravitational forces, and these conservation laws are only valid when examining the system as a whole. If we didn't model the celestial objects as point particles there could also be several extra degrees of freedom in which energy and momentum could be distributed, but this is negligible for our purposes.

D. Celestial mechanics

For a planet in orbit around a much more massive celestial object of mass M, we may find the escape velocity of the planet by finding the minimum kinetic energy required to move the planet infinitely far away. We thus acquire an analytical expression:

$$v_{esc} = \sqrt{\frac{2GM}{r}}. \tag{15}$$

When considering the Earth-sun system this is simply $v_{esc} = \sqrt{8\pi^2}$, when expressed in AU/yr.

For a planet (mass m) in uniform circular orbit around a much more massive object of mass M we have that the distance between the objects r = constant. From equation (10) this implies that the potential energy is a conserved quantity. By studying for example the expression for the centripetal acceleration $a = F(r)/m = v^2/r$ it is easy to see that the speed must be constant and therefore the kinetic energy must be conserved.
From vector calculus we find for circular motion that $\boldsymbol{r} \perp \boldsymbol{v}$, and from equation (14) we obtain conservation of angular momentum.
As the velocity is constantly changing, the vector momentum $m\boldsymbol{v}$ is not conserved. The scalar quantity mv is however (as speed is constant).
The speed $v = |\boldsymbol{v}|$ is easily obtained for the Earth-sun system, $v = 2\pi r/T = 2\pi AU/yr$.

E. Mercury's perihelion precession

In classical mechanics, using Newtons law of gravity, we will get closed elliptical orbits for a planet moving around the sun. The perihelion-position (the point where the planet is closest to the sun) for the planet will appear at the same place every orbit. In real life, this does not happen. What we see is that the perihelion-position will move slightly for every orbit for a planet around the sun. There are different factors in the universe that causes this to happen. One factor is the theory of relativity.
The perihelion-precession is the difference between the postulated classical and the observed movement of the perihelion-position. When subtracting all factors that causes the shift of the perihelion-position except for the theory of relativity, the value of the perihelion precession is $43''$ per century. [**?**]

When simulating this in our project, we add a relativistic correction to the classical force, so we get

$$F_{1,2} = \frac{GM_1M_2}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right] \tag{16}$$

where $l = |\vec{r}\times\vec{v}|$ is the magnitude of the planets orbital angular momentum per unit mass and c is the speed of light in vacuum. We looked at a simulation of Mercury and the sun for a century, and found all the perihelion-positions. We could then calculate the angle θ_p by

$$\tan\theta_p = \frac{y_p}{x_p} \Rightarrow \theta_p = \arctan(\frac{y_p}{x_p}) \tag{17}$$

where y_p and x_p are the x and y coordinates at the perihelion. The perihelion precession is then given by

$$\theta_{p,classical} - \theta_{p,relativistic} \tag{18}$$

F. Object Orientation

A object oriented approach to our problem in short allows us to write the code once, and use it many times [**?**]. As we are interested in celestial objects characterized by their position, velocity and mass; a class based approach lets us easily implement the properties and interactions of each celestial object in the solar system. In addition, it makes it quite easy to extend the code to include and calculate other properties such as the force on a single object or the total angular momentum of the system. We can also include other functionality such as writing to file or choosing the method of numerical integration without interfering with other parts of the code.

G. Testing

As mentioned above, we have severial conserved quantities to consider as we evolve our system in time. A test of whether the total energy of the solar system is conserved in time, or whether the angular momentum of a circular orbit remains zero at all times, is a good check to make sure our calculations are correct and sufficiently precise.

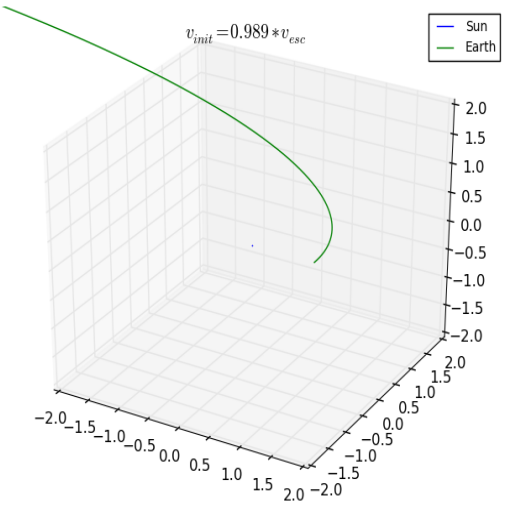
III. RESULTS AND DISCUSSION

In our final implementation it is somewhat difficult to make an assessment of the total number of floating points operations involved, and questionable if it would be of much use. This is because the program involves a large number of function calls, if tests, vector operations and recalculation of quantities. Some of these inefficiencies might be alleviated by an intelligent compiler. We do however find that the time usage scales roughly linearly with the number of time steps n.

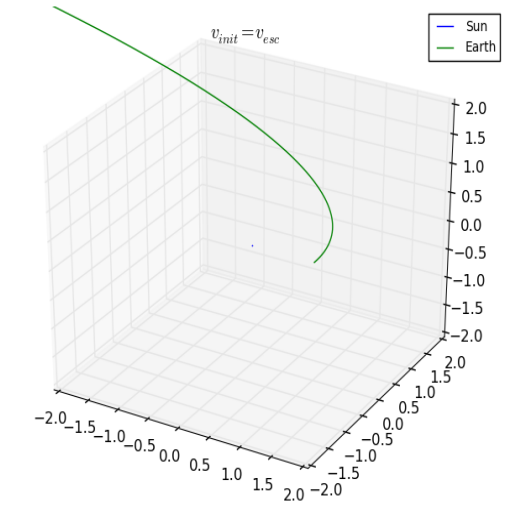
A. Earth-Sun system

B. Escape velocity

We looked at a planet which started at a position $1AU$ from the sun, and we were to find its escape velocity. Using equation 15 we found that the escape velocity for the Earth-Sun system should be $v = \sqrt{8\pi^2}$. When doing this numerically, we started by using the exact escape velocity and multiplying this by numbers close to 1 (see Figure 1)



(a)



(b)

FIG. (1) Eart-Sun system with Earth's escape-velocity

We can see that for the initial escape velocity we found theoretically, the planet does escape. As the figure shows, there is some numerical error here, since the planet also escapes with an initial value smaller than the theoretical escape-velocity.

C. Three-body problem

In Figure 2 we have included three dimensional picture of the three-body problem where we have used the Verlet method for integration.

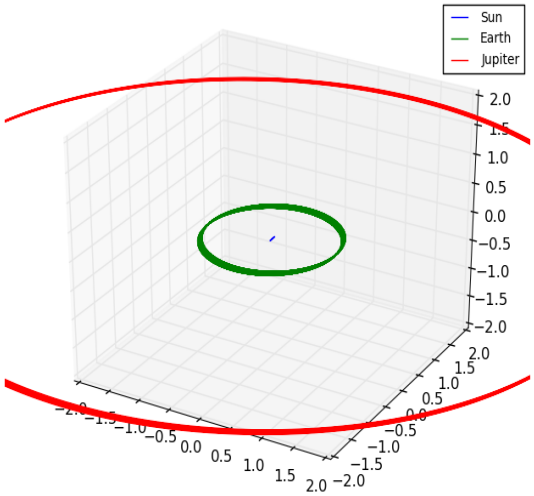
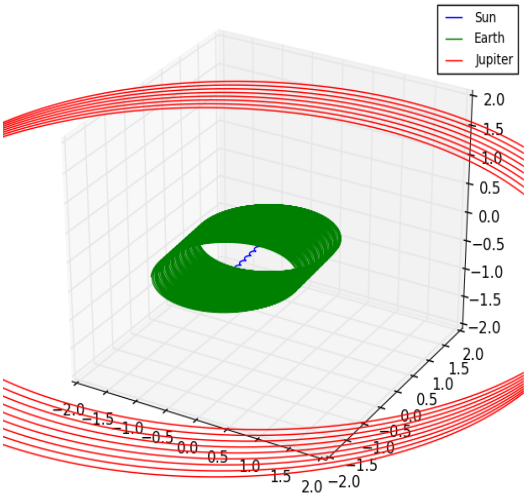
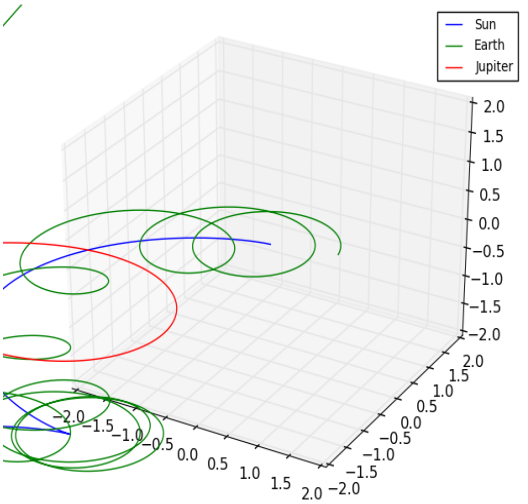


FIG. (2) The three-body problem where Jupiter is the red graph and the earth is the green. The motion of the sun can barely be seen as the blue graph.

The plot in Figure 2 reproduces what we would expect to see in real life with the planets moving around the sun. Comparing this image to Figure 3 where we have increased the mass of Jupiter by 10 and 1000, we can clearly see that we have two unstable systems.



(a) $M_{Jupiter} \times 10$



(b) $M_{Jupiter} \times 1000$

FIG. (3) Earth-Sun-Jupiter three-body system for bigger values of $M_{jupiter}$

We can also see this by looking at the conserved energy. For the system where we don't change Jupiter's mass, we get the change in energy

$$\Delta_E = 2.0 \times 10^{-8} \tag{19}$$

which we can approximate as zero, because of round-off errors. This is also the case when Jupiter's mass is multiplied by 10, while for the case where Jupiter's mass is multiplied by 1000, we get the change in energy

$$\Delta_E = 0.042 \tag{20}$$

D. Mercury's perihelion precession

In this project we looked at Mercury's perihelion-precession. We then added a relativistic correction to the force (equation (16)) and compared the angle at perihelion (equation (17)) to the situation with classical Newtonian gravitational force.
To get the right precision in this part of the project it was crucial to have enough steps for integration. In Table I we have shown the time and the perihelion precession for two different numbers of integration-steps. The perihelion precession is calculated by Equation (18) for the θ_p values after a century.

TABLE (I) Perihelion precession for different timesteps using the Verlet solver

Integration steps n	**Time** $[s]$	**Perihelion precession** [”]
10^8	0.3	47.72
10^5	234.3	165.0

We expected the perihelion-precession to be $43''$. For $n = 10^8$ integration-steps we get an answer that is close enough to show that this is caused by relativistic correction. Also, if we were to to more integration-steps, our computers would spend too much time doing it. In Figure 4 we have included a plot for Mercury's perihelion precession. From this we can see that the precession between the classical case and the relativistic case gets bigger as time goes by.

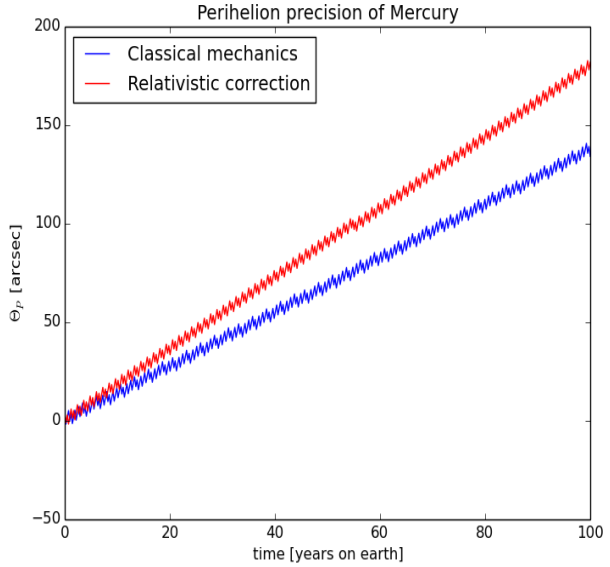


FIG. (4) Perihelion precession for Mercury around the sun for a century. The blue graph represent the classical case and the red graph represent the relativistic case. This is done with $n = 10^8$ integrationsteps

IV. CONCLUSION

Object orientation makes this mostly easy. Weird flow of the exercices? Implement unit testing? Inefficient IF tests? Time constraints?

V. APPENDIX

All code used is available at: The programs used in this project are listed in this section:

main.cpp: Program1

plot.py: Program2