Project 3 in FYS3150

Bendik Steinsvåg Dalen, Ulrik Seip

October 24, 2018

https://github.com/UlrikSeip/Projects/tree/master/prosjekt3

1 ABSTRACT

In this project we simulate the orbits of all the 8 planets in the solar system, and Pluto. Comparing the Forward Euler and the Velocity Verlet methods we find the Velocity Verlet method to be preferable due to its conservation of energy. We then test the Velocity Verlet method against the analytically derived escape velocity and perihelion of Mercury.

2 INTRODUCTION

Our solar system is littered with asteroids, planets and moons. This plethora objects floating around in space makes for a perfect exercise in solving multi body differential equations in 3 dimensions.

When simulating orbits for several celestial bodies with high accuracy, the computation can be expensive, and so it is paramount to strike a balance between efficiency and accuracy. To explore this balance, we will run simulations using the Velocity-Verlet integration method, and comparing with the Forward-Euler method. Having found the optimal way to simulate the orbits, we move on to test the effect of the gravitational pull between planets, and complete a full model for all planets of the solar system, and Pluto. Finaly we look at the perhelion precession of Mercury to see the stability of our algorithm.

3 METHOD

3.a Newtons law of gravitation

One of the most common representations of newtons law of gravitation on Earth is

$$\boldsymbol{F}_G = \frac{M_{\text{Earth}} v^2}{r}\hat{\boldsymbol{r}} = \frac{GM_\odot M_{\text{Earth}}}{r^2}\hat{\boldsymbol{r}}, \tag{1}$$

where $G = 6.67 \cdot 10^{-}11\frac{m^3}{kgs^2}$ is the gravitational constant, m_1 and m_2 are the masses of the bodies exerting a force upon each other, F_G is said force, and r is the distance between the bodies. Using Keplers laws this can be further simplified to

$$\boldsymbol{F}_G = \frac{M_\odot M_{\text{Earth}} 4\pi^2}{r^2}\frac{\text{AU}^3}{\text{yr}^2}\hat{\boldsymbol{r}}. \tag{2}$$

We can then rewrite this for a point mass and acceleration as

$$\boldsymbol{a} = \frac{M_\odot 4\pi^2}{r^2}\frac{\text{AU}^3}{\text{yr}^2}\hat{\boldsymbol{r}}. \tag{3}$$

3.b The Forward Euler method

To use equation 3 for the Forward Euler method we need an expression for $\Delta\boldsymbol{v}$. We therefore introduce a time step dt. We also define $\hat{\boldsymbol{r}} = \cos(\theta)\hat{\boldsymbol{i}} + \sin(\theta)\hat{\boldsymbol{j}} + \cos(\phi)\hat{\boldsymbol{k}}$. This gives us

$$\frac{d\boldsymbol{v}}{dt} = \frac{v_{i+1} - v_i}{dt} = -4\pi^2\frac{M_\odot}{r^2}\hat{\boldsymbol{r}}$$

$$\boldsymbol{v}_{i+1} = -4\pi^2\frac{M_\odot}{r^2}dt\hat{\boldsymbol{r}} + \boldsymbol{v}_i = -\boldsymbol{a}_i dt + \boldsymbol{v}_i. \tag{4}$$

We can do something similar to find $\Delta\boldsymbol{x}$:

$$\frac{d\boldsymbol{x}}{dt} = \frac{x_{i+1} - x_i}{dt} = v_{i+1}\hat{\boldsymbol{r}} = \boldsymbol{v}_{i+1}$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{v}_{i+1}dt + \boldsymbol{x}_i = \boldsymbol{v}_{i+1}dt + \boldsymbol{x}_i. \tag{5}$$

3.c The Velocity Verlet method

From [1] we know that the Verlet formula for a specific x_i is

$$x_{i+1} = 2x_i - x_{i-1} + h^2 x_i^{(2)} + O(h^4), \tag{6}$$

where h is the timestep, $x^{(2)}$ is function 3, and O is the truncation error. We also know that the velocity is

$$x_i^{(1)} = \frac{x_{i+1} - x_{i-1}}{2h} + O(h^2). \tag{7}$$

Unfortunaley function 6 is a bit difficult to use as we only know the initial position, and thus can't find x_1 or x_2, and so forth. To help with this we can rewrite them into

$$x_{i+1} = x_i + h x_i^{(1)} + \frac{h^2}{2} x_i^{(2)} \tag{8}$$

and

$$x_i^{(1)} = x_{i-1}^{(1)} + \frac{h}{2}\left(x_i^{(2)} + x_{x-i}^{(2)}\right), \tag{9}$$

see section 6.a for more details.

3.d The code for the implementations

Both the Velocity Verlet and the Forward Euler algorithms were implemented using a combination of python and Julia. Python happens to be a convenient language for making an object oriented implementation of a solar system simulation, while Julia is in most cases more than 10 times faster than Python, and even rivals c++, when it comes to computation speeds.

3.e Testing the algorithms

To test our algorithm we can check if the escape velocity of our planets is correct. We find it analytically by using conservation of energy, and the fact that the kinetic energy, and the potential energy of our planets are defined with opposite signs. We therefore start with the equation:

$$E_k + E_p = \tfrac{1}{2} M_E v_{esc}^2 - G\frac{M_\odot M_E}{r} = 0$$

If we then isolate v_{esc} on the left side of the equation, we end up with the following equation:

$$V_{esc} = \sqrt{\frac{2GM_\odot}{r}} \tag{10}$$

To test our implementation against the analytical solution we simply run a simulation for an imaginary planet with the initial analytical velocity, and see if it has returned after several thousand years. We can also try simulating with a slightly lower initial velocity, and se if the planet does indeed stay in orbit with anything less than the analytical initial velocity.

Furthermore the algorithms can be tested against each other for both accuracy and consistency.

3.f The three-body problem

We now have a good basis to extend our algorithm to study the three-body problem, by adding Jupiter to the equation. The force between Earth and Jupiter is

$$\boldsymbol{F}_{\text{Earth-Jupiter}} = \frac{GM_{\text{Jupiter}} M_{\text{Earth}}}{r^2_{\text{Earth-Jupiter}}}\hat{\boldsymbol{r}}. \tag{11}$$

For simplicity's sake we will keep the Sun fixed in the centre of mass, or origo, for now. For each timestep we then calculate the force on Earth and Jupiter from the Sun, and the the force between Earth and Jupiter, and add them up. We then get an position array for both Earth and Jupiter.

To test the stability of our Verlet solver we also studied what effect increasing the magnitude of Jupiter by 10 and 1000 would have on the system.

3.g Final model for all planets of the solar system

We now almost have a working model of the solarsystem. Firstly we calculated the three-body problem of the Earth, Sun and Jupiter again, but now treating the Sun as an object, instead of fixing it in the centre of mass. We then get the path of the objects around the centre of mass. The initial values of the Sun was found by setting the posisiton in origo, and making sure the momentum of the Sun was the negative of the total momentum of the other planets.

Finaly we simply added the remaining planets, and Pluto, to the calculation as extra bodies. The acceleration on each body was found as above, by calculating the force on each object from the other objects.

3.h The perihelion precession of Mercury

4 RESULTS

4.a The Forward Euler method

4.b The Velocity Verlet method

4.c Testing the algorithms

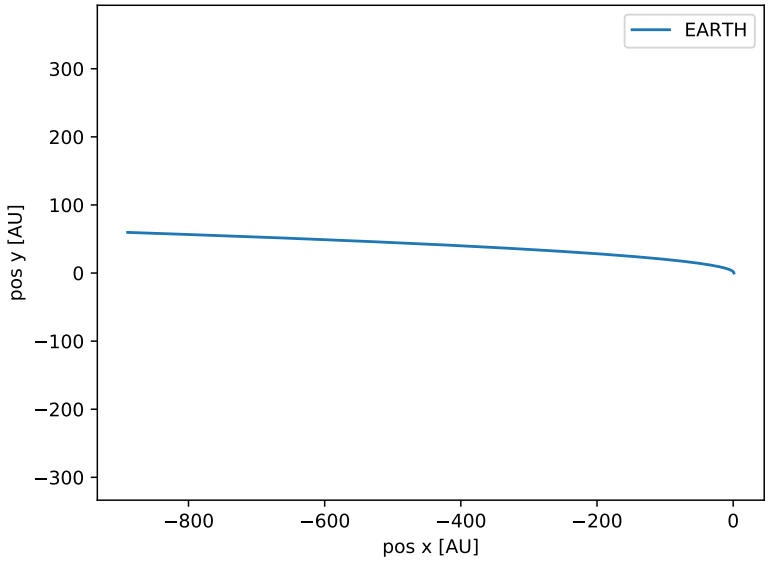

Figure 1: A plot of the simulated minimal escape velocity over 2000 years

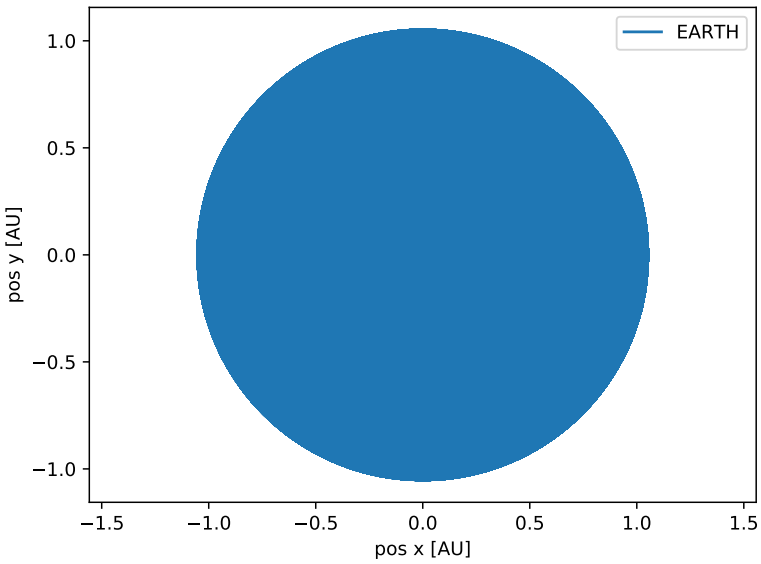

Figure 2: A plot of the simulated minimal escape velocity over 2000 years

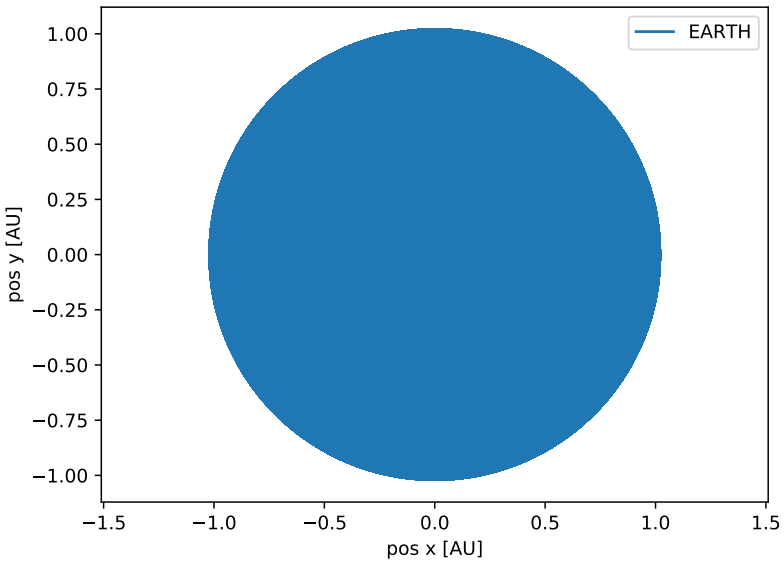

Figure 3: A plot of the simulated minimal escape velocity over 2000 years

5 CONCLUSIONS

6 APENDICES

6.a The Velocity Verlet method math

Firstly function 8:

$$x_i^{(1)} = \frac{x_{i+1} - x_{i-1}}{2h} \Rightarrow 2hx_i^{(1)} = x_{i+1} - x_{i-1} \tag{12}$$

$$x_{i-1} = x_{i+1} - 2hx_i^{(1)} \tag{13}$$

$$x_{i+1} = 2x_i - x_{i-1} + h^2 x_i^{(2)} = 2x_i - \left(x_{i+1} - 2hx_i^{(1)}\right) + h^2 x_i^{(2)} \tag{14}$$

$$2x_{i+1} = 2x_i + 2hx_i^{(1)} + h^2 x_i^{(2)} \tag{15}$$

$$x_{i+1} = x_i + hx_i^{(1)} + \frac{h^2}{2} x_i^{(2)} \tag{16}$$

Then function 9:

$$x_{i+1} = x_i + hx_i^{(1)} + \frac{h^2}{2} x_i^{(2)} \Rightarrow x_i = x_{i-1} + hx_{i-1}^{(1)} + \frac{h^2}{2} x_{i-1}^{(2)} \tag{17}$$

$$x_{i+1} = x_{i-1} + hx_{i-1}^{(1)} + \frac{h^2}{2} x_{i-1}^{(2)} + hx_i^{(1)} + \frac{h^2}{2} x_i^{(2)} \tag{18}$$

$$x_i^{(1)} = \frac{x_{i-1}^{(1)}}{2} + \frac{h}{4} x_{i-1}^{(2)} + \frac{x_i^{(1)}}{2} + \frac{h}{4} x_i^{(2)} = x_{i-1}^{(1)} + \frac{h}{2}\left(x_i^{(2)} + x_{i-1}^{(2)}\right) \tag{19}$$

7 REFERENCES

References

[1] Computational Physics, Lecture Notes Fall 2015, Morten Hjort-Jensen p.215-220