Project3

Daniel Johannessen

https://github.com/daniehj/

02.11.17

Project 3 - FYS3150

Abstract

This project is looking in to the solarsystem and how to solve some ODE's for gravitational forces of attraction between the planets and the sun. The two different solvers used are Euler and Verlet, the Euler is a lot faster, but the Verlet is more accurat. The mass of the different planets as well as the velocities is of big importance in the fact of holding all the planets in their respective orbits in our system. To show how velocities and massses matter, the Energy conservation law is used to calculate the escape velocity, and by giving Jupiter more mass we will see that the orbits would look a lot different.

1. Introduction

From Newtons 2. law, we build two differrent ODE solvers, starting with Euler, and then going on to Verlet. Building a planet class inn python, that handles the different planet and star properties. For the different properties that is needed in this project, the NASA Horizon telnet interface is used by building a script that downloads the diffrerent properties from the server. After downloading, the properties is sendt to al local database for easy local use of data. The properties we will use is name, mass, position and velocity, with those properties we can calculate and plot the orbits using the ODE solvers.

2. Theory

To build a model of the solarsystem, the sammenhng of the gravitational forces

and the attraction between the celestial bodies are needed. We have from Newton's law of gravitation that reads

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

where G is the gravitational constant, $M_\odot$ is the mass of the sun, M_{Earth} is the mass of earth, and r is the distiance between them. Using that the orbit is almost circular, and the centripetal formula

$$F = \frac{mv^2}{r}$$

we rewrite

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

Again we can rewrite

$$v^2 r = GM_\odot = 4\pi^2 \frac{AU^2}{year^2} AU = 4\pi^2 \frac{AU^3}{year^2}$$

seeing that

$$v^2 = \frac{AU^2}{year^2}$$
$$r = AU$$
$$G = 4\pi^2$$

Perihelion prescission

$$F_G = \frac{GM_\odot M_{Mercury}}{r^2}\left[1 + \frac{3l^2}{r^2 c^2}\right]$$

where

$$l = \|\vec{r} \times \vec{v}\|$$

and

$$\tan\theta_P = \frac{y_p}{x_p}$$

3. Implementation

We need the acceleration, we find this from Newtons second law

$$F = ma$$

$$F_G = \frac{GM_\odot M_{Earth}}{r^2}\frac{\vec{r}}{r} = M_{Earth} a$$
$$a_{(x,y,z),i} = \frac{GM_\odot}{r_i^3}\vec{r}$$
$$r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$$

Euler

$$x_{i+1} = x_i + v_{x,i}\Delta t$$
$$y_{i+1} = y_i + v_{y,i}\Delta t$$
$$z_{i+1} = z_i + v_{z,i}\Delta t$$
$$v_{x,i+1} = v_{x,i} + a_{x,i}\Delta t$$

$$v_{y,i+1} = v_{y,i} + a_{y,i}\Delta t$$
$$v_{z,i+1} = v_{z,i} + a_{z,i}\Delta t$$

Verlet method

$$x_{i+1} = x_i + v_{x,i}\Delta t + \frac{\Delta t^2}{2}a_{x,i}$$
$$y_{i+1} = y_i + v_{y,i}\Delta t + \frac{\Delta t^2}{2}a_{y,i}$$
$$z_{i+1} = z_i + v_{z,i}\Delta t + \frac{\Delta t^2}{2}a_{z,i}$$
$$v_{x,i+1} = v_{x,i} + \frac{\Delta t}{2}(a_{x,i+1} + a_{x,i})$$
$$v_{y,i+1} = v_{y,i} + \frac{\Delta t}{2}(a_{y,i+1} + a_{y,i})$$
$$v_{z,i+1} = v_{z,i} + \frac{\Delta t}{2}(a_{z,i+1} + a_{z,i})$$

$$\Delta t = \frac{t_{end} - t_{start}}{n}$$

To find the escape velocity we use follwing formulas

$$E_k = \frac{1}{2}mv^2$$
$$E_p = -\frac{GMm}{r}$$

Where v is the velocity, G is the gravitational constant, M the mass of the larger object, m the mass of the smaller object, and r the distance between the objects.

The energy cannot be conserved if the object is to escape the gravitational pull, this yelds

$$\frac{1}{2}mv^2 - \frac{GMm}{r} = 0$$

$$v = \sqrt{\frac{2GM}{r}}$$

For the Sun-Earth system we then can calculate

$$r = \sqrt{(1.285 \times 10^{11})^2 + (7.6455 \times 10^{10})^2 + (-2.3466 \times 10^{7})^2} = 1.4952 \times 10^{11}$$

$$v = \sqrt{\frac{2 * 6.67428 \cdot 10^{-11} m^3 (kg)^{-1} s^{-2} \cdot 1.98892 \cdot 10^{30} (kg)}{1.4952 \times 10^{11} m}} =$$

$$42138 \frac{m}{s} : 1000 \frac{m}{km} = 42.138 \frac{km}{s} * \frac{(60 \cdot 60 \cdot 24 \cdot 365) \frac{s}{(year)}}{149597871 \frac{(km)}{(AU)}} = 8.8829 \frac{AU}{year}$$

4. Result

The program
`Horizon2SQL.py`
is ran by entering name of the planets/stars in the command-line and the properties is downloaded to the datbase created in the script. It can also update the properties by doing a rerun with the same planets/stars it is relevant to use in furher projects.

When the database is initiated with data, the script

```
C:\FYS\FYS3150>calcnplot.py
Connection to DB OK, SQLite version: 2.6.0
Plotting...
Plotting...
Plotting...
Plotting...
```

This program is testing the two algorithms Euler and Verlet and plots the orbit for one year

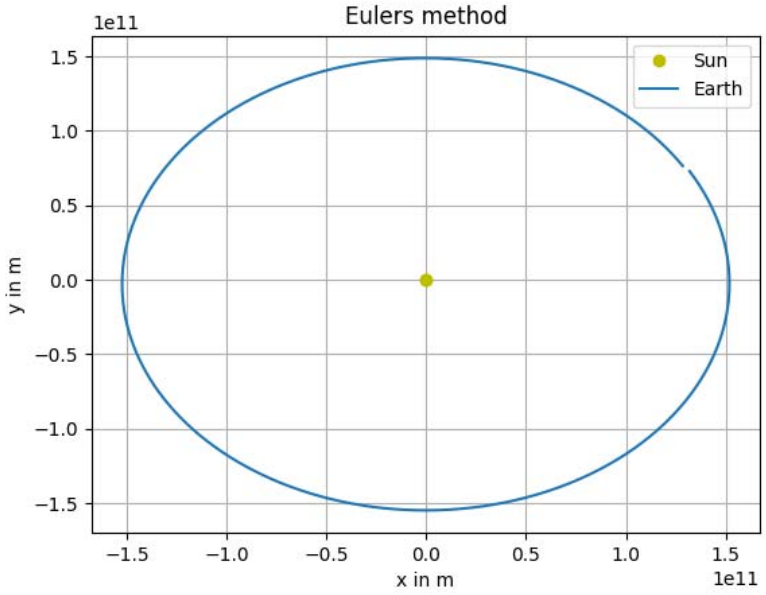

Figure 1 - Earth orbit around sun using Eulers-method

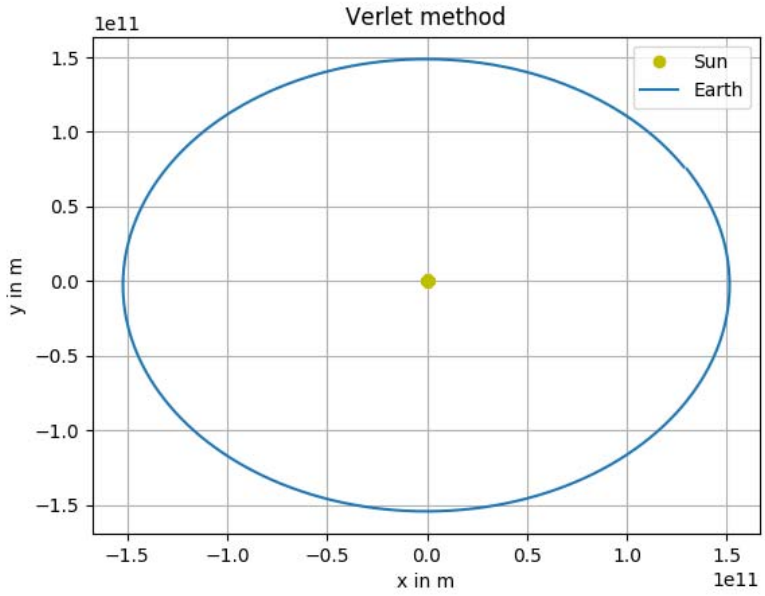

Figure 2 - Earth orbit around sun using Verlet-method

Further on from the more objectorientated script

```
C:\FYS\FYS3150>solarsystem.py 1 euler
Solvinng for planets:['Sun', 'Earth'] with ODE: euler

dt=1day/1000.0
Time solving: 15.617s
```

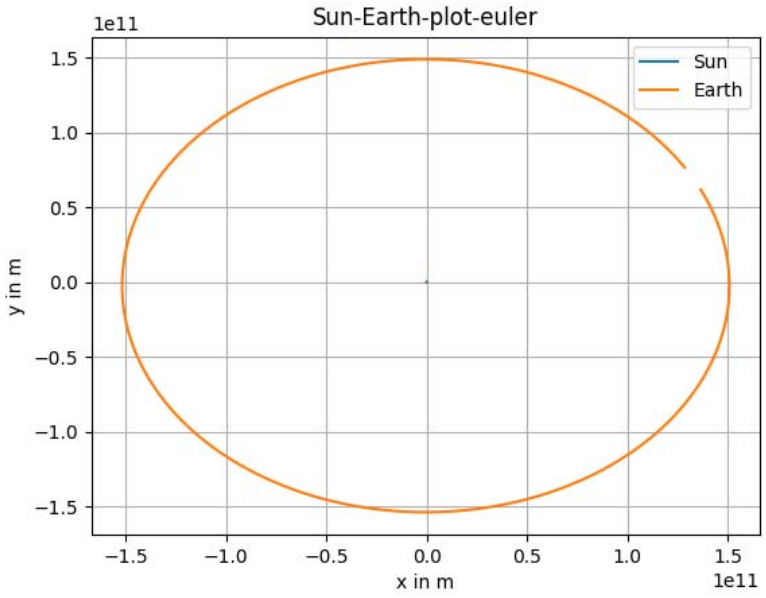

Figure 3 - Sun-Earth plot using Euler method

```
C:\FYS\FYS3150>solarsystem.py 1 verlet
Solvinng for planets:['Sun', 'Earth'] with ODE: verlet

dt=1day/1000.0
Time solving: 40.932s
```

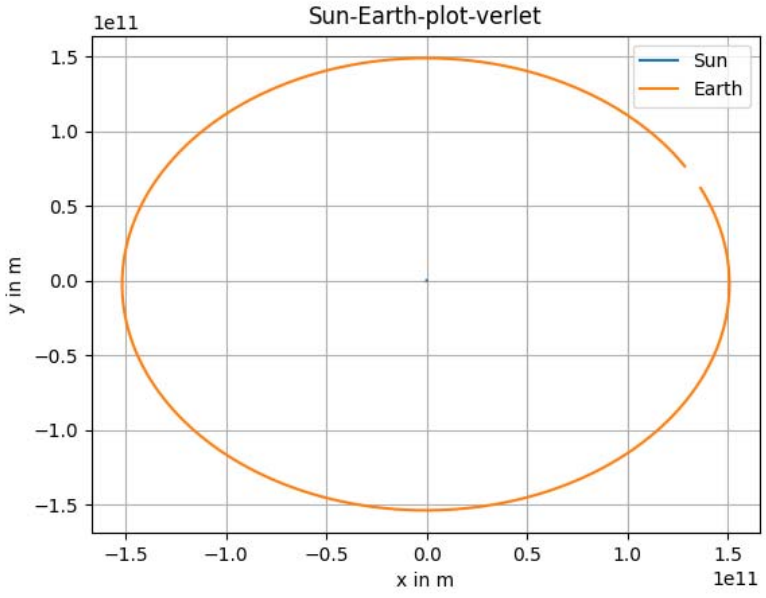

Figure 4 - Sun-Earth plot using Verlet method

Table 1 - Times for the ODE's to solve

ODE	$\Delta t[day]$	$Time[s]$
Euler	$\frac{1}{1000}$	15.617
Verlet	$\frac{1}{1000}$	40.932

Adding Jupiter to the plot as well, using Verlet from now because of accuracy

```
C:\FYS\FYS3150>solarsystem.py 2 verlet 1
Solvinng for planets:['Sun', 'Earth', 'Jupiter'] with ODE: verlet

dt=1day/30.0
Time solving: 54.555s
```

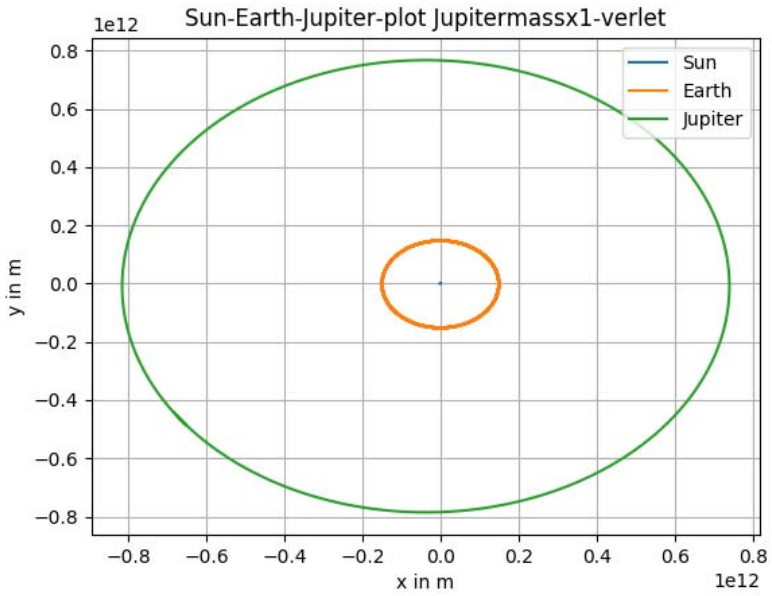

Figure 5 - Sun-Earth-Jupiter, Jupiter original mass running for $12 years$ to get Jupiter orbit

```
C:\FYS\FYS3150>solarsystem.py 2 verlet 10

Solvinng for planets:['Sun', 'Earth', 'Jupiter'] with ODE: verlet

dt=1day/30.0
Time solving: 53.450s
```

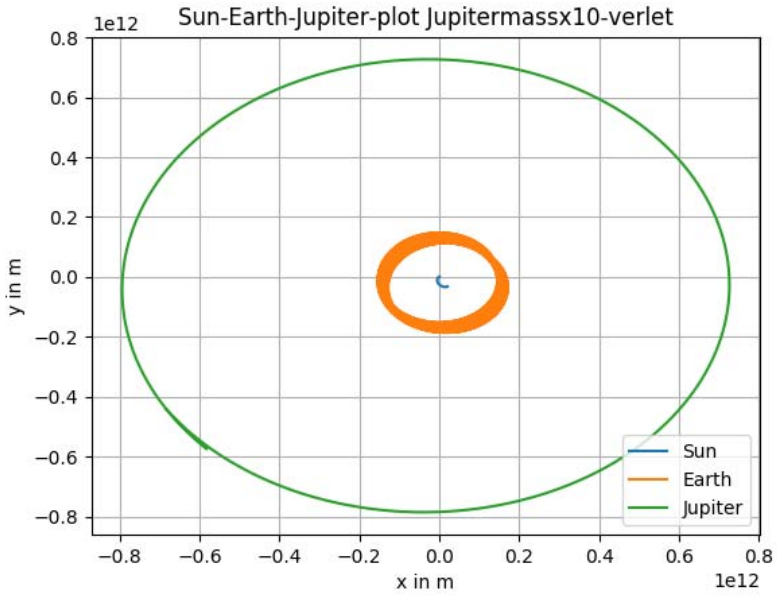

Figure 6 - Sun-Earth-Jupiter, Jupiter $10xM_{jupiter}$ running for $12years$ to get Jupiter orbit

```
C:\FYS\FYS3150>solarsystem.py 2 verlet 1000
Solvinng for planets:['Sun', 'Earth', 'Jupiter'] with ODE: verlet

dt=1day/30.0
Time solving: 54.871s
```

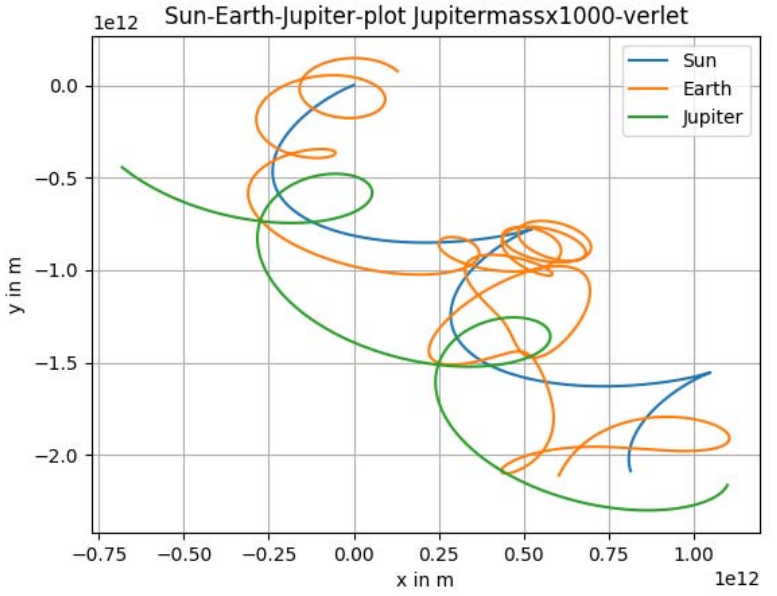

Figure 7 - Sun-Earth-Jupiter, Jupiter $1000xM_{jupiter}$ running for $12years$ to get Jupiter orbit

```
C:\FYS\FYS3150>solarsystem.py 3 verlet
Solvinng for planets: full-solarsystem with ODE: verlet

dt=1day/0.125
Time solving: 159.796s
```

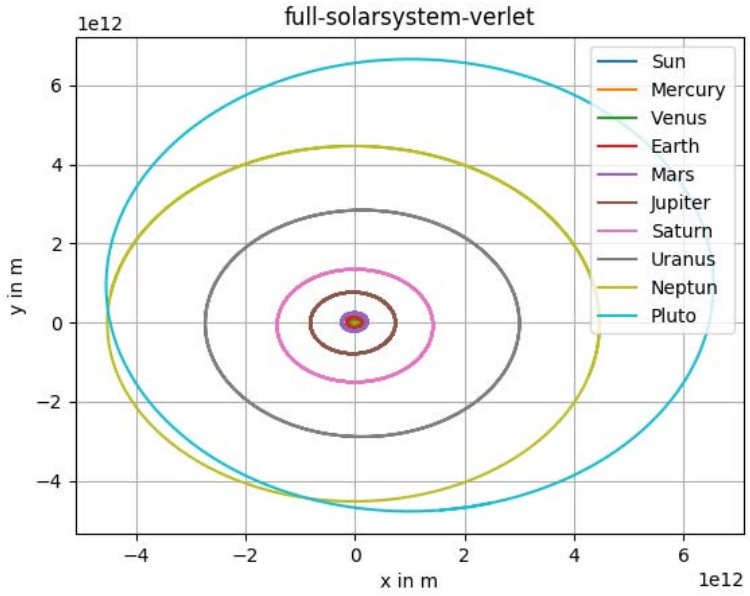

Figure 8 - Full solarsystem plot runinng for $250 years$ to get full Pluto orbit

```
C:\FYS\FYS3150>solarsystem.py 4
Solvinng for planets:['Sun', 'Mercury'] with ODE: perihelion-precission

dt=1day/1.0
Time solving: 23.194s
```

5. Conclusion

From the reults we can see that the two algorithms both are working fine, but that tha accuarcy is better on the Verlet ODE solver. This is because we update the forces twice per loop. This results in 4 more FLOPS per loop, en thereby a longer computation time. The approximation to circular orbits seems to work fine, but for the Euler solver we can se that it does not go the full orbit. Tampering with the

mass of Jupiter makes the orbits change a lot, we can see from figure 6, with Jupiter mass times $10,$ that the Sun now moves a little bit together with Jupiter's orbit. From figure 7 we can see that the Sun and Jupiter now orbits around eachother, they now almost have the same mass when Jupiter mass is 1000 times original mass.

6. References

[1] https://en.wikipedia.org/wiki/Verlet_integration
[2] https://en.wikipedia.org/wiki/Euler_method
[3] https://en.wikipedia.org/wiki/Centripetal_force
[4] https://en.wikipedia.org/wiki/Newton's_laws_of_motion
[5] https://www.jpl.nasa.gov