Project 3 FYS4150
Ordinary differential equations

Audun Tahina Reitan and Marius Holm

October 24, 2018

1 Abstract

We give a brief introduction to differential equations and how we can discretize such equations in order to solve them using computer algorithms. We then introduce the most important equations we meet in our project, namely Newton's law of gravitation, and Newton's second law of motion on differential form. In table 1 we present the masses and distance from the Sun of all the planets in our solar system. We choose to scale our equations in such a way that the mass of the Sun is $M_\odot \overset{\text{def}}{=} 1$, and as such we also need the masses of the planets given in the same unit.

More of our results can be found in the GitHub repository linked in the implementation section.

2 Introduction

In this project we'll develop code for simulating the solar system, using the Verlet algorithm for solving coupled ordinary differential equations. For this project we will focus on the use of classes in our code, which makes it a lot easier to reuse larger parts of our code for problems of similar character. In our case of the solar system we can create a class describing a planet, a moon, or some other astronomical body which we want to include in our solar system.

In order to test that our algorithm works, we start by looking at a simple hypothetical system consisting of only the Earth orbiting the Sun. We then investigate the stability of our algorithm and plot the position of the Earth orbiting the Sun. Furthermore we consider the initial velocity needed for the planet to escape the gravity of the Sun. Then we'll consider a three-body system consisting of the Earth, the Sun, and Jupiter. We then expand our three-body model to the entire solar system including Pluto. In the final part of our project we look at the perihelion precession of Mercury.

3 Methods

3.1 Differential equations

The order of an ordinary differential equation (ODE) is given by the highest order of derivative found in the left-hand side of the equation. A first order differential equation is typically of the form

$$\frac{dy}{dt} = f\big(t, \frac{dy}{dt}, y\big) \tag{1}$$

where f is an arbitrary function. A well known second order differential equation is Newton's second law

$$m\frac{d^2x}{dt^2} = -kx \tag{2}$$

where k is the force constant. All ODE's depend on one variable only, compared to partial differential equations which can depend on several variables. PDE's are widely used, but won't be of any concern in this project.

3.2 Newton's law of gravitation

The first part of our project we take a look at a simplified solar system consisting of the Sun and the Earth. Newton's law of gravitation then takes the following form:

$$F_G = \frac{GM_\odot M_{Earth}}{r^2}, \tag{3}$$

where $M_\odot$ is the mass of the Sun and M_{Earth} is the mass of the Earth. G is the gravitational constant and r is the distance between the Earth and the Sun.

We assume that the orbit of the Earth around the sun is co-planar, and we let this be the xy-plane. We can then use Newton's second law of motion which gives us two differential equations, one for each coordinate.

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{Earth}}, \tag{4}$$

and

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{Earth}}, \tag{5}$$

where $F_{G,x}$ and $F_{G,y}$ are the x and y components of the gravitational force.

3.3 Useful constants

During our project we will use astronomical units (AU) as a measure of length. 1 AU is defined as the average distance between the Sun and the Earth, that is 1 AU $= 1.5 \times 10^{11}$ m. The mass of the Sun is $M_{\text{sun}} = M_\odot = 2 \times 10^{30}$ kg. We will also need the mass of the planets we include in our code, which are given in table 1. In our code we have defined the unit mass to be equal to $M_\odot$, i.e. $M_\odot = 1$. We therefore also present the planetary masses given in the unit $M_\odot$.

Table 1: Mass and distance from the Sun for each planet in our solar system.

Planet	Mass in kg	Mass in unit M_{Sun}	Distance to sun in AU
Earth	$M_{\text{Earth}} = 6 \times 10^{24}$ kg	3.04×10^{-6}	1 AU
Jupiter	$M_{\text{Jupiter}} = 1.9 \times 10^{27}$ kg	9.54×10^{-4}	5.20 AU
Mars	$M_{\text{Mars}} = 6.6 \times 10^{23}$ kg	3.23×10^{-7}	1.52 AU
Venus	$M_{\text{Venus}} = 4.9 \times 10^{24}$ kg	2.45×10^{-6}	0.72 AU
Saturn	$M_{\text{Saturn}} = 5.5 \times 10^{26}$ kg	2.86×10^{-4}	9.54 AU
Mercury	$M_{\text{Mercury}} = 3.3 \times 10^{23}$ kg	1.66×10^{-7}	0.39 AU
Uranus	$M_{\text{Uranus}} = 8.8 \times 10^{25}$ kg	4.37×10^{-5}	19.19 AU
Neptune	$M_{\text{Neptune}} = 1.03 \times 10^{26}$ kg	5.15×10^{-5}	30.06 AU
Pluto	$M_{\text{Pluto}} = 1.31 \times 10^{22}$ kg	7.40×10^{-9}	39.53 AU

We include Pluto, even though it is technically not regarded as a planet anymore.

3.4 Discretizing differential equations

We assume that Earth's orbit is circular around the Sun. For circular motion we have

$$F_G = \frac{M_{\text{Earth}} v^2}{r} = \frac{G M_\odot M_{\text{Earth}}}{r^2}, \tag{6}$$

where v is Earth's velocity. By simple algebraic manipulation we find the relationship

$$v^2 r = G M_\odot = 4\pi^2 \,\text{AU}^3/\text{yr}^2 \tag{7}$$

By defining $M_\odot = 1$ we have $G = 4\pi^2 \,\text{AU}^3/\text{yr}^2$.

3.4.1 Euler's forward algorithm

Suppose we have an initial function value $y(t_0) = y(t = t_0)$ for a function $y(t)$. The function is defined in the well-behaved domain $[a, b]$. By splitting the domain into N sub intervals we find the step size h as

$$h = \frac{b-a}{N}. \tag{8}$$

Using the step size, h, and the derivative of y we can find an expression for the function at the next time step as $y_1 = y(t_1 = t_0 + h)$. We can generalize this method which gives us a general procedure for finding the step y_{i+1} in terms of the previous time step.

$$y_{i+1} = y(t = t_i + h) = y(t_i) + h\Delta\big(t_i, y_i(t_i)\big) + O(h^{p+1}), \tag{9}$$

where $O(h^{p+1})$ represents the truncation error. Using Taylor expansion we find

$$\Delta(t_i, y_i(t_i)) = (y'(t_i) + \cdots + y^{(p)}(t_i)\frac{h^{p-1}}{p!} \tag{10}$$

By defining $y'(t_i) = f(t_i, y_i)$ and truncating Δ at the first derivative, we have

$$y_{i+1} = y_i + hf(t_i, y_i) + O(h^2) \tag{11}$$

which combined with $t_{i+1} = t_i + h$ gives us Euler's forward algorithm. [Hjort-Jensen, 2015]

3.4.2 Euler's method specific

Introducing $x = r\cos\theta$, $y = r\sin\theta$ and $r = \sqrt{x^2+y^2}$ we can rewrite

$$F_{G,x} = -\frac{GM_\odot M_{\text{Earth}}}{r^2}\cos\theta = -\frac{GM_\odot M_{\text{Earth}}}{r^3}x \tag{12}$$

Similarly for y-direction we get

$$F_{G,y} = -\frac{GM_\odot M_{\text{Earth}}}{r^2}\sin\theta = -\frac{GM_\odot M_{\text{Earth}}}{r^3}y \tag{13}$$

This gives us the four coupled differential equations

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x, \tag{14}$$

$$\frac{dx}{dt} = v_x, \tag{15}$$

$$\frac{dv_y}{dt} = -\frac{GM_\odot}{r^3}y, \tag{16}$$

$$\frac{dy}{dt} = v_y, \tag{17}$$

The four equations can then be discretized using Euler's forward algorithm, and we find the four discretized equations as

$$x_{i+1} = x_i + hv_{x,i}, \tag{18}$$

$$v_{x,i+1} = v_{x,i} - h\frac{4\pi^2}{r_i^3}x_i, \tag{19}$$

$$y_{i+1} = y_i + hv_{y,i}, \tag{20}$$

$$v_{y,i+1} = v_{y,i} - h\frac{4\pi^2}{r_i^3}y_i. \tag{21}$$

When adding more planets these equations have to be modified. We will show how this is done in the case of adding Jupiter to our system. The additional gravitational force on Earth from Jupiter in the x-direction is

$$F_x^{EJ} = -\frac{GM_J M_E}{r_{EJ}^3}(x_E - x_J), \tag{22}$$

where M_E is the mass of the Earth, M_J is the mass of Jupiter, and

$$r_{EJ} = \sqrt{(x_E - x_J)^2 + (y_E - y_J)^2}. \tag{23}$$

x_E and y_E are the x and y coordinates of Earth, while x_J and y_J are the x and y coordinates of Jupiter. Thus the velocity of Earth changes as

$$\frac{dv_x^E}{dt} = -\frac{GM_\odot}{r^3}x_E - \frac{GM_J}{r_{EJ}^3}(x_E - x_J) \tag{24}$$

By using $GM_J = GM_\odot\left(\frac{M_J}{M_\odot}\right) = 4\pi^2\frac{M_J}{M_\odot}$ we can rewrite the equation as

$$\frac{dv_x^E}{dt} = -\frac{4\pi^2}{r^3}x_E - \frac{4\pi^2\frac{M_J}{M_\odot}}{r_{EJ}^3}(x_E - x_J). \tag{25}$$

Similarly for the y-direction we get

$$\frac{dv_y^E}{dt} = -\frac{4\pi^2}{r^3}y_E - \frac{4\pi^2\frac{M_J}{M_\odot}}{r_{EJ}^3}(y_E - y_J). \tag{26}$$

These equations also apply to Jupiter, and very similar equations apply to the other possible planets we might add to our system. [Hjort-Jensen, 2018]

3.4.3 Velocity Verlet method

We have the Taylor expansion for the position given by

$$x_{i+1} = x_i + hx_i^{(1)} + \frac{h^2}{2}x_i^{(2)} + O(h^3). \tag{27}$$

The corresponding expansion for the velocity is

$$v_{i+1} = v_i + hv_i^{(1)} + \frac{h^2}{2}v_i^{(2)} + O(h^3). \tag{28}$$

From Newton's second law we have an analytical expression for the derivative of the velocity

$$v_i^{(1)} = \frac{d^2x}{dt^2}|_i = \frac{F(x_i, t_i)}{m}. \tag{29}$$

By adding this to the corresponding expansion for the derivative of the velocity

$$v_{i+1}^{(1)} = v_i^{(1)} + hv_i^{(2)} + O(h^2), \tag{30}$$

and keep the terms up to the second derivative because our error is of order $O(h^3)$, we get

$$hv_i^{(2)} \approx v_{i+1}^{(1)} - v_i^{(1)}. \tag{31}$$

We can then rewrite the Taylor expansion for the velocity as

$$v_{i+1} = v_i + \frac{h}{2}\left(v_{i+1}^{(1)} + v_i^{(1)}\right) + O(h^3) \tag{32}$$

The final equations for position and velocity are then given by

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}v_i^{(1)} + O(h^3), \tag{33}$$

and

$$v_{i+1} = v_i + \frac{h}{2}\left(v_{i+1}^{(1)} + v_i^{(1)}\right) + O(h^3). \tag{34}$$

Note that the term $v_{i+1}^{(1)}$ depends on x_{i+1}, and as such we have to calculate the position at t_{i+1} before computing the velocity at the updated time. [Hjort-Jensen, 2018]

3.5 Implementation

In our implementation of the abovementioned methods we use a more general scheme which allows us to implement any given formula for the acceleration. Thus we are not limited to the specified acceleration derived from circular motion, but have the opportunity to add other terms as well.

All our code, calculations, and plots used can be found in Auduns GitHub repository.

4 Results

4.1 Object oriented code

Specify and explain important parts of the object oriented code we have implemented.

4.2 FLOPS

If we have N_b celestial body in our solar system the number of computations done to calculate the Force for all is:

$$((N_b - 1) + \cdots + 1)\,(O(\mathbf{F}_G)), \tag{35}$$

where $O(\mathbf{F}_G)$ is the number of FLOPS done to calculate the force between two bodies.

$$((N_b - 1) + \cdots + 1) = \frac{(N_b - 1) + 1}{2}(N_b - 1) = \frac{1}{2}(N_b^2 - N_b). \tag{36}$$

$$\mathbf{F}_G = G\frac{m_i m_j}{r^2}\frac{\mathbf{r}}{r} = 7\ FLOPS, \tag{37}$$

where $r = |\mathbf{r}| = \sqrt{r \cdot r}$ which for a body in 3 dimensional space takes 6 FLOPS. Here we count the square root as a single FLOP for simplicity's sakes. $\mathbf{r} = \mathbf{x}_i - \mathbf{x}_j$ which is 3 FLOPS (in 3D).

$$O(\mathbf{F}_G) = 16\ FLOPS \tag{38}$$

$$((N_b - 1) + \cdots + 1)\,(O(\mathbf{F}_G)) = 8(N_b^2 - N_b)\ FLOPS. \tag{39}$$

Per timestep the number of floating points operations done is:

$$8(N_b^2 - N_b)\ FLOPS + O(IM) + O(\mathbf{a}), \tag{40}$$

where $O(\mathbf{a}) = O(\mathbf{F}_G/m) = 3N_b\ FLOPS$ is the number of FLOPS done to calculate the accelearation of the bodies in the system. $O(IM)$ is the number of FLOPS the integration method takes. For Euler's method in 3 dimensions we have:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i = 3 \cdot 2\ FLOPS = 6\ FLOPS, \tag{41}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + h\mathbf{a}_i = 3 \cdot 2\ FLOPS = 6\ FLOPS, \tag{42}$$

which sums up to 12 FLOPS. For the velocity Verlet method we have:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}a_i = 3 \cdot 6\ FLOPS = 18\ FLOPS, \tag{43}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{2}(\mathbf{a}_{i+1} + \mathbf{a}_i = 3 \cdot 5\ FLOPS = 15\ FLOPS, \tag{44}$$

which sums up to 33 FLOPS. For the case with $N_b = 2$ bodies in the solar system, we have that for each time step the ration between Euler's and velocity Verlet method is:

$$\frac{(16 + 6 + 12)\ FLOPS}{(16 + 6 + 33)\ FLOPS} \approx 0.62. \tag{45}$$

In reality Euler's method would be using more time then ≈ 0.62 of the velocity Verlet method because we are also measuring potential energy, kinetic energy, and the angular momentum which also requires a few FLOPS. So a ratio higher than ≈ 0.62 is expected. Multiple rounds with the `timemesure` program gives an estimate of the ratio ≈ 0.8.

4.3 Escape velocity

We can find an analytical result for the escape velocity needed for a planet, starting 1 AU from the Sun, to escape the Suns gravitational force. We use conservation of energy in order to solve this problem. The only significant force on the planet is the gravitational force from the Sun. As such we have conservation of energy according to

$$(K + U_g)_i = (K + U_g)_f, \tag{46}$$

where $K_f = 0$ as the final velocity is equal to zero, and $U_{g,f} = 0$ because the distance between the Sun and the planet will go to infinity. Thus we have

$$\frac{1}{2} M_{planet} v_e^2 + \frac{-GM_\odot M_{planet}}{r} = 0 + 0 \tag{47}$$

$$v_e = \sqrt{\frac{2GM_\odot}{r}} = \sqrt{8\pi^2} = 2\sqrt{2}\,\pi \,\frac{\mathrm{AU}}{\mathrm{yr}} \approx 42\,\mathrm{km/s} \tag{48}$$

Try changing

$$F_G = \frac{GM_\odot M_E}{r^2} \tag{49}$$

$$F_G = \frac{GM_\odot M_E}{r^\beta} \tag{50}$$

where $\beta \in [2, 3]$. Comment results.

4.4 Perihelion precession of Mercury

By adding a relativistic term to the Newtonian gravitational force we find

$$F_G = \frac{GM_\odot M_{Mercury}}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right], \tag{51}$$

where $M_{Mercury}$ is the mass of Mercury, r the distance between the Sun and Mercury, $l = |\vec{r} \times \vec{v}|$ is the magnitude of Mercury's orbital angular momentum per unit mass, and c is the speed of light in vacuum. We then check the value of the perihelion angle, θ_P, using

$$\tan\theta_P = \frac{y_P}{x_P} \tag{52}$$

where x_P (y_P) is the x (y) position of Mercury at perihelion, which is the point where Mercury is closest to the Sun. Simulating a system consisting of only the Sun and Mercury over a century we plot the perihelion precession of Mercury.

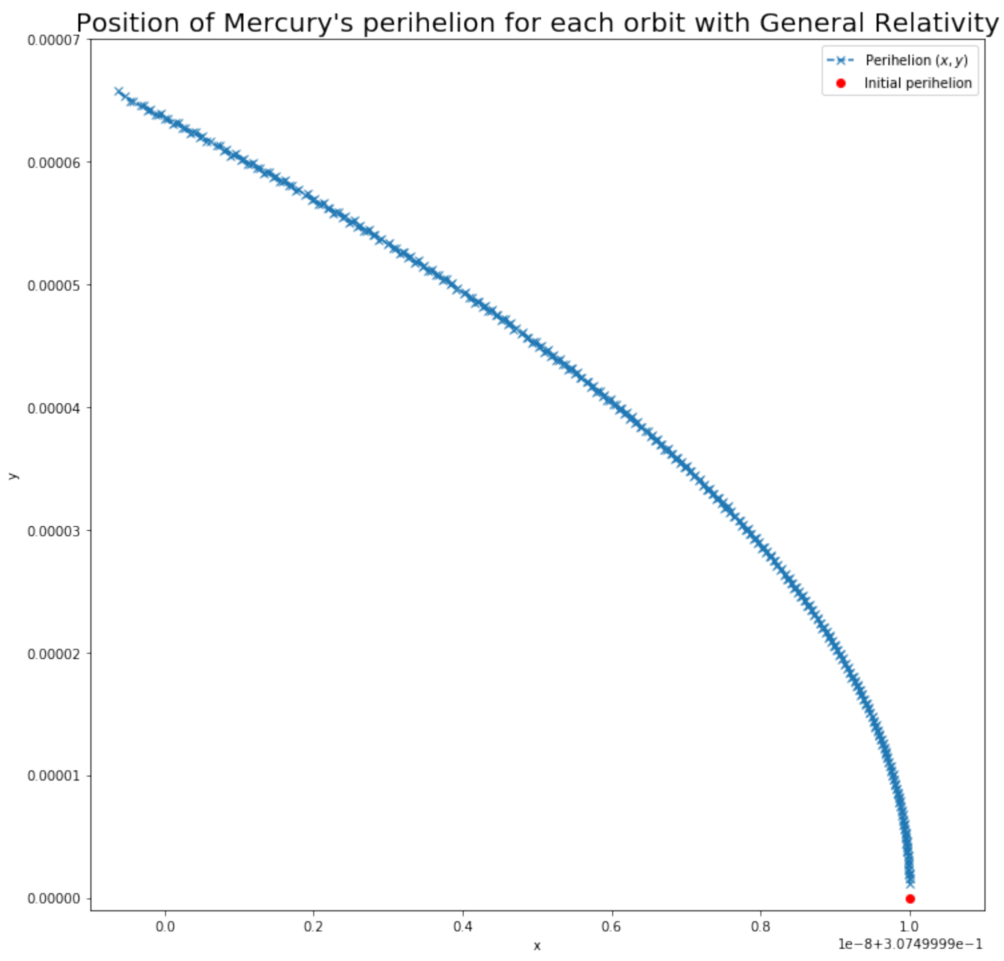

Figure 1: Plot of Mercury's position at perihelion using the general relativistic correction term.

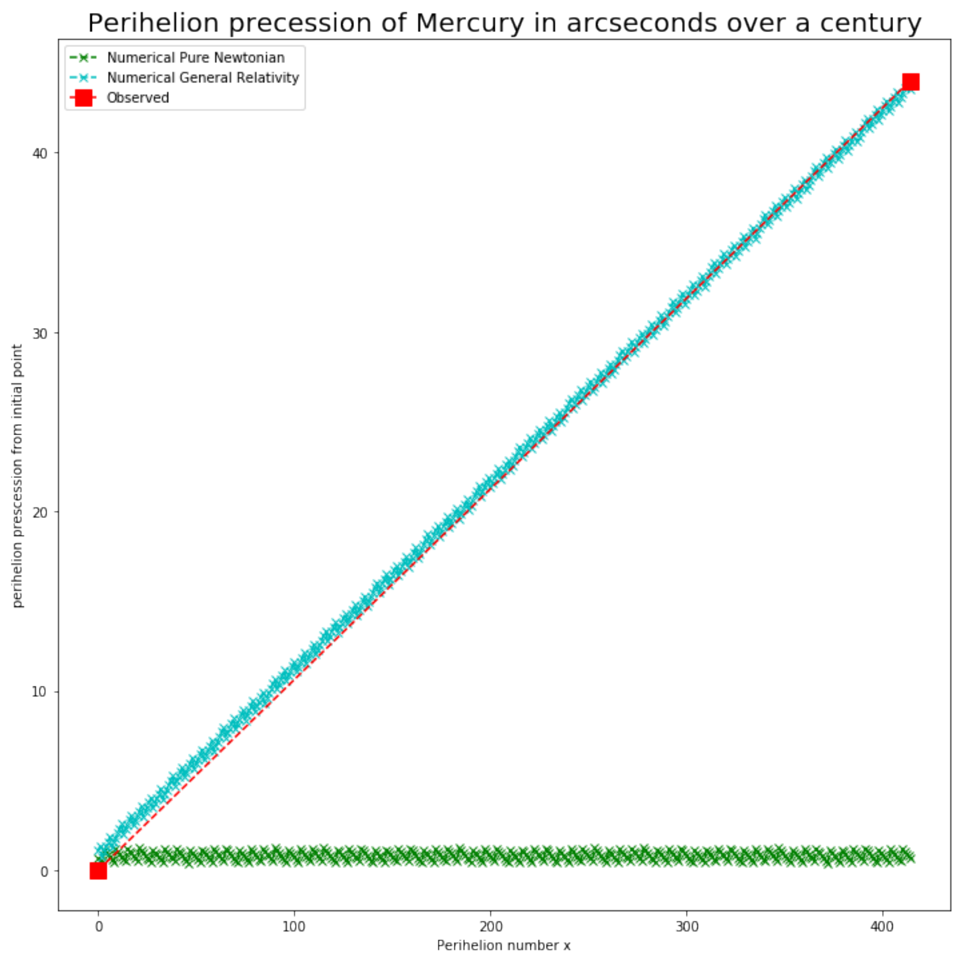

Figure 2: Perihelion precession of Mercury.

4.5 Method stability

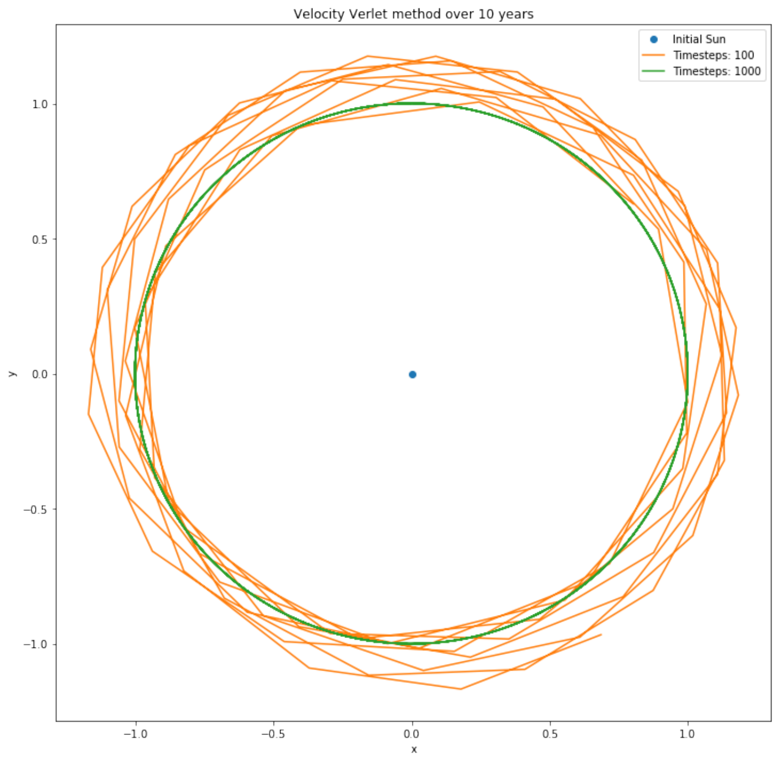

Figure 3: Mercury's orbit using velocity Verlet method over 10 years with 100 and 1000 time steps.

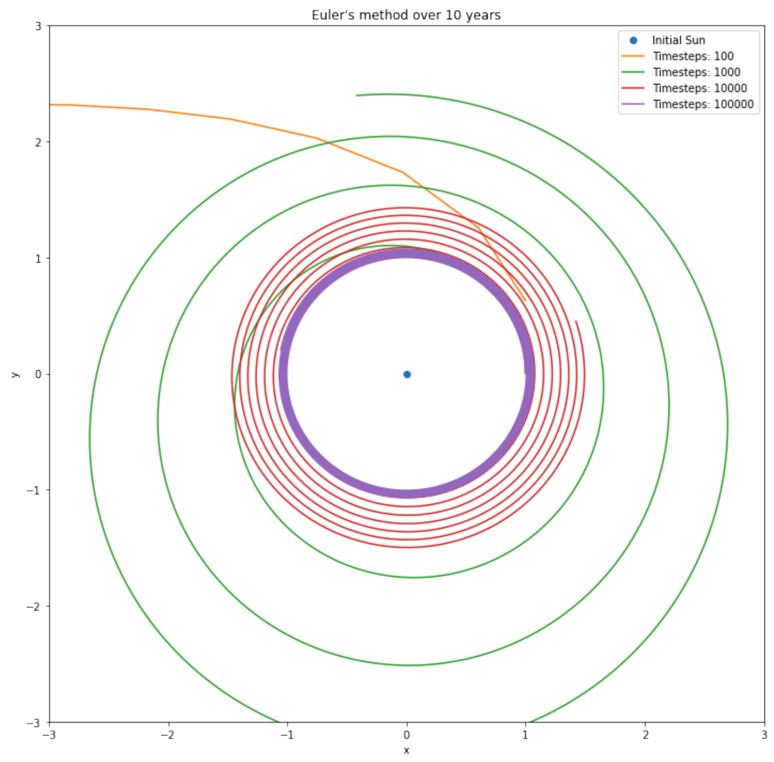

Figure 4: Orbit of Mercury using Euler's method over a time period of 10 years. We test for 100, 1000, 10 000, and 100 000 time steps.

As we can see from figures 3 and 4, we can achieve pretty nice results using both Euler's forward algorithm as well as the velocity Verlet algorithm. However we notice that in order to get a nice circular orbit we need 100 times more timesteps for Euler's method compared to velocity Verlet.

References

[Hjort-Jensen, 2015] Hjort-Jensen, M. (2015). Computational physics. lecture notes. Accessible at course github repository. 551 pages.

[Hjort-Jensen, 2018] Hjort-Jensen, M. (2018). Ordinary differential equations. lecture notes. Accessible at course github repository. 25 pages.