The Solar System: an Exercise in Numerical Integration and Object oriented design

Kosar Nozari Mirarkolaei

October 2018

Abstract

In this project we build a flexible and easily expandable object oriented framework which we use to simulate the solar system. In order to calculate orbits of celestial bodies we implement the Euler-Cromer and the Velocity Verlet numerical integration methods. The Verlet method proves superior. The model is expanded gradually from a two-body system to include all planets of the solar system. The source code for this project can be found at my github[1]

1 Introduction

In this project the goal is to develop a code for simulating the motion of objects in the Solar System, using the velocity Verlet algorithm. The Verlet algorithm will be compared to the well-known forward-Euler algorithm for the initial parts of the project. To generalize and simplify the structure of the program, the code will be object oriented, so repeatable parts and operations are written as classes. The project will progress by first developing a code for the two-body Earth-Sun system, and then expanding to a three-body system, before finally completing the solar system with all planets. The implementation of the algorithms will be tested by making checks for the conservation of energies and angular momentum. Initial conditions for the objects in the solar system are found on NASA's web-pages [2].

2 Theory

2.1 Forward Euler method

The forward Euler method is a well known first-order numerical procedure for solving ODEs with a given initial value. It can be derived in a number of ways,

[1] `https://github.com/Kosarnm/FYS4150`
[2] http://ssd.jpl.nasa.gov/horizons.cgi

but the most common one is the taylor expansion of a function y around a point t_0. For the ODE

$$y^{(1)}(t) = f(t, y(t))$$

this gives:

$$y(t_0 + h) = y(t_0) + hy^{(1)}(t_0) + \frac{h^2}{2}y^{(2)}(t_0) + O(h^3) \tag{1}$$

Substituting $y^{(1)}(t) = f(t, y(t))$ and ignoring the higher order terms, and introducing a step size h such that $t_i = t_0 + ih$ where $i \in [0, n]$ and $h = \frac{t_n - t_0}{n}$, gives:

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

2.2 Velocity-Verlet method

2.2.1 General Verlet

This is a popular algorithm for solving Newton's equations of motion, and is also used in molecular dynamics simulations and computer graphics. In this case the algorithm will be applied to the gravitational force between to bodies in a simulated solar system. The problem is described by Newton's second law of motion, which will be the example for outlining the method.

$$m\frac{d^2x}{dt^2} = F(x, t) \tag{3}$$

Rewriting in terms of two coupled differential equations gives:

$$\frac{dx}{dt} = v(x, t)\frac{dv}{dt} = \frac{F(x, t)}{m} = a(x, t) \tag{4}$$

Taylor expanding x gives:

$$x(t + h) = x(t) + hx^{(1)}(t) + \frac{h^2}{2}x^{(2)}(t) + O(h^3) \tag{5}$$

Adding the corresponding Taylor expansion for $x(t - h)$, and discretizing the expressions, the following is obtained:

$$x(t_i \pm h) = x_{i\pm1}$$

$$x_i = x(t_i)$$

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

2.2.2 Velocity-Verlet

In the above section it is found that velocity is not directly included in the equation, since the function $x_i^{(2)} = a(x,t)$ is supposed to be known. If one would like to keep an eye on, say, conservation laws for energy, the velocity is needed to find kinetic energy. It can be computed using:

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

Note that the algorithm for position depends on x_{i-1}, which is rather ficticious for $i = 0$. This is where velocity Verlet comes in.
Taylor expanding the velocity gives:

$$v_{i+1} = v_i + hv_i^{(1)} + \frac{h^2}{2}v_i^{(2)} + O(h^3) \tag{6}$$

Through Newton's second law an analytical expression for the derivative of the velocity can be obtained:

$$v_i^{(1)} = \frac{d^2x}{dt^2}|_i = \frac{F(x_i, t_i)}{m} \tag{7}$$

Adding the corresponding Taylor expansion for the derivative of the velocity:

$$v_{i+1}^{(1)} = v_i^{(1)} + hv_i^{(2)} + O(h^2) \tag{8}$$

Since the error goes as $O(h^3)$ only terms up to the second derivative of the velocity are retained, to reach the following expression:

$$hv_i^{(2)} \approx v_{i+1}^{(1)} - v_i^{(1)} \tag{9}$$

This allows for rewriting the Taylor expansion for the velocity as

$$v_{i+1} = v_i + \frac{h}{2}\big(v_{i+1}^{(1)} + v_i^{(1)}\big) + O(h^3) \tag{10}$$

The final expressions for position and velocity become:

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

$$v_{i+1} = v_i + \frac{h}{2}\big(v_{i+1}^{(1)} + v_i^{(1)}\big) + O(h^3)$$

It is important when implementing this method that the term $v_{i+1}^{(1)}$ depends on the position x_{i+1}, which means the position must be calculated at t_{i+1} before the next velocity can be computed. Additionally, the derivative of the velocity at the time t_i used when calculating the updated position can be reused in the velocity update.

2.3 Gravity

Newton's law of gravitation is given by a force F_G

$$F_G = \frac{GM_1M_2}{r^2} \tag{11}$$

Where M_1 and M_2 are the masses of the objects, G is the gravitational constant, and r is the distance between the objects. Assuming the motion is co-planar in the xy-plane, the position of object 2 relative to object 1 is given by a set of second-order differential equations on the form:

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_2}$$
$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_2}$$

where $F_{G,x}$ and $F_{G,y}$ are the x and y components of the gravitational force. These equations are the basis for the N-body problem that this project aims to solve.

2.4 General relativity and the perihelion precession of Mercury

Closed elliptical orbits are a special feature of the Newtonian $1/r^2$ force. In general, any correction to the pure $1/r^2$ behaviour will lead to an orbit which is not closed, i.e. after one complete orbit around the Sun, the planet will not be at exactly the same position as it started. If the correction is small, then each orbit around the Sun will be almost the same as the classical ellipse, and the orbit can be thought of as an ellipse whose orientation in space slowly rotates. A general relativistic correction to the Newtonian gravitational force can be added. For the Sun - Mercury system, the force becomes

$$F_G = \frac{GMM_{Mercury}}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right] \tag{12}$$

where $l = |\vec{r} \times \vec{v}|$ is the magnitude of Mercury's orbital angular momentum per unit mass, and c is the speed of light in vacuum. This allows for testing how general relativitys prediction for the perihelion precession of Mercury compares to the observed value. The observed value when all classical effects are subtracted is 43 (43 arc seconds) per century.
The precession of Mercury's orbit is not easy to spot simply by plotting the planets orbit, but the angle of the perihelion[3] can be calculated easily enough by the following equation

$$\theta_p = \arctan\left(\frac{y_p}{x_p}\right) \tag{13}$$

[3]The point in the orbit closest to the sun.

2.5 Two-body problem

Initially in this project, the two-body problem is solved, with Earth orbiting the Sun. Actually the problem of Earth orbiting the Sun can be approximated as a one-body problem, as the mass of the Sun is a lot bigger than the mass of Earth, and so we can view it as Earth orbiting a stationary Sun.

A planet orbiting a star, generally has an elliptical orbit, here we want to look at the case where the Earth has a circular orbit. For this to be the case, the centripetal force must be the same as the gravitational force. This means that

$$\frac{v^2}{r} = \frac{GM}{r^2} \tag{14}$$

where v is the speed of the Earth, r is the distance between the Earth and the Sun, M is the mass of the Sun and G is the gravitational constant. Solving for v, we get

$$|v| = \sqrt{\frac{GM}{r}} \approx 2\pi \frac{AU}{year} \tag{15}$$

When calculating this, we get that for the planet's orbit around the Sun to be circular, the magnitude of the velocity must be $2\pi \frac{AU}{year}$

2.6 n-body problem

The one-body problem can be solved analytically, and so can the two-body problem by splitting it up to two one-body problems. The three-body problem, however, can not be solved analytically. More generally, the n-body problem with $n > 2$ cannot be solved analytically, and we depend on numerical tools to solve them. The n-body problem is solved by using numerical integration methods, i.e the forward-Euler or the Verlet method.

2.7 Conservation laws

The angular momentum of a system is constant, unless there is a torque acting on the system. The torque is defined as

$$\tau = \mathbf{r} \times \mathbf{F} \tag{16}$$

In this case, $\mathbf{r}$ and $\mathbf{F}$ are parallel, so $\tau = 0$ As there is no torque acting in the system, the angular momentum of the particles should be conserved.

Also, both the kinetic energy and the potential energy should be conserved. From Newton's third law, we know that the action on one particle on the other is met by an equal and opposite reaction. Thus:

$$\sum_i \mathbf{F}_i = 0 \tag{17}$$

We know that

$$\sum_i \mathbf{F}_i = \sum \frac{d\mathbf{p}_i}{dt} \tag{18}$$

And so, the sum of the total momentum must be a constant. For non-relativistic particles $\mathbf{p} = m\mathbf{v}$, and so, also the total velocity must be a constant. As the kinetic energy, $K = \frac{1}{2}mv^2$, this means that the total kinetic energy of the system is conserved. The total energy must be conserved, and so also the total potential energy of the system must be conserved.

2.8 Escape velocity

Considering a planet with an initial distance from the sun of 1 AU, the escape velocity can be found using conservation of energy (assuming the planet - sun system can be treated as isolated). Define kinetic energy as

$$E_k = \frac{1}{2}m_p v_p^2$$

and potential energy in the gravitational field as

$$E_p = -\frac{GMm_p}{r}$$

where r is the distance between the center of mass of the planet and the sun, and m_p is the mass of the planet. Define also the potential energy to be 0 when the planet is at an infinite distance away from the sun. The escape velocity can then be seen as the initial velocity required for the planet to continue increasing its distance from the sun until it comes to a stop when the distance reaches infinity. This gives the following equation:

$$E_{k,1} + E_{p,1} = E_{k,2} + E_{p,2} \tag{19}$$

where the left-hand side is the initial case and the right-hand side is the infinite-distance case. Inserting the expressions for kinetic and potential energy, and the conditions for the infinite-distance energies, gives:

$$\frac{1}{2}m_p v_{esc}^2 - \frac{GMm_p}{r} = 0$$

$$\frac{1}{2}m_p v_{esc}^2 = \frac{GMm_p}{r}$$

$$v_{escape} = \sqrt{\frac{2GM}{r}}$$

Inserting $r = 1AU$ and $M = 1$ gives

$$v_{esc} = \sqrt{2G} \approx 8.88\frac{AU}{yr} \tag{20}$$

where the converted gravitational constant is $G = 39.42\frac{(AU)^3}{M(yr)^2}$

2.9 Units

In order to make the system easy to work with and the calculations easier as well, a change of units is warranted. In this study we will therefore express the mass of any body as a fraction of solar masses, that is $M_\odot = 2 \times 10^{30} kg$. This means that the Earth[4], for instance will have a mass of $M_{Earth} = 3 \times 10^{-6}$.

Additionally, the units for distance will be astronmical units AU. This is the mean distance between the earth and the sun. For a simple system and a particular coordinate system with the sun at origin, the initial position for the Earth could simply be $\mathbf{r}_{Earth} = (1, 0, 0)$.

The unit for time will be Earth years, yr. This means that velocity will be in units AU/yr.

Changing the variables will have consequences for the gravitationan constant G and velocities v as well. This can be deduced easily enough by picking a sample system consisting of the earth and the sun. Inserting in equation **??** gives

$$F_G = \frac{GM_\odot M_{Earth}}{r^2} \tag{21}$$

Furthermore, assuming the orbit of the earth is perfectly circular, will give the following relation

$$F_G = \frac{M_{Earth} v^2}{r} \tag{22}$$

Combining equations 21 and 22 yields the following

$$v^2 r = gM_\odot = 4\pi^2 AU^3 yr^{-2} \tag{23}$$

Because the mass of the sun as a fraction of solar masses is $M_\odot = 1$ and the distance between the earth and the sun is $r = 1AU$ we land at

$$G = 4\pi^2 AU^3 yr^{-2} \quad v_{Earth} = 2\pi AU yr^{-1} \tag{24}$$

3 Algorithms and implementation

3.1 Object orientation

Object orientation allows for more general code to be written and for easy reuse. Moreover, object orientation makes sense for humans, who tends to classify objects we see and interact with in everyday life. In this study we make use of all the advantages of object oriented programming by implementing several classes.

The Planet class contains all the information about the initial conditions of the Planet, which help us to implement different object of it in our Solar System class and easily extend our problem to n-body problem.

[4] In SI-units, $M_{Earth} \approx 6 \times 10^{24} kg$

The Solar System class contains a method for setting up particles. Depending on what system one wants to model. I also implement methods for calculating kinetic and potential energy and also my integrator method EulerCromer and VelocityVerlet.

3.2 Euler-Cromer

Analytically the Euler-Cromer method, or the semi-implicit Euler method, can be expressed in one dimension by the following recursive relations

$$v_{n+1} = v_n + a_n dt \tag{25}$$
$$x_{n+1} = x_n + v_{n+1} dt, \tag{26}$$

where a_n is the acceleration, v_n is the velocity and x_n is the position, after a certain number of steps n. dt is the time step and $t_n = t_0 + ndt$ is the time after n steps. One sees that the algorithm is relatively cheap, with $4n$ FLOPS. In Python code EulerCromer method looks like this

```
    def EulerCromer(self,planet,h,n):
        N      = int(n)
        x      = np.zeros(N)
        y      = np.zeros(N)
        vx     = np.zeros(N)
        vy     = np.zeros(N)
        x [0]  = planet.x_position()
        y [0]  = planet.y_position()
        vx[0] = planet.velocity_x()
        vy[0] = planet.velocity_y()
        for i in range(0,N-1):
            ax,ay   = self.accelaration_x_y(x[i],y[i])
            vx[i+1] = vx[i] + ax*h
            vy[i+1] = vy[i] + ay*h
            x [i+1]  = x[i] + vx[i+1]*h
            y [i+1]  = y[i] + vy[i+1]*h

        return x,y
```

Notice that the acceleration are computed within the SolarSystem class.

The Euler Cromer method is a first-order integrator which means that it commits a global error of the order of dt. This means that by decreasing the time step one will get a more accurate result.

3.3 Velocity Verlet

The Verlet method is an integration method designed to integrate Newton's equation of motion. The method provides good numerical stability and more precise results compared with a regular method like the Euler-Cromer method.

The reason for this is that it is symplectic, which means that it conserves state-space volume.

The velocity Verlet method can also be represented by a (one-dimensional) recursive relation

$$x_{n+1} = x_n + v_n dt + \frac{1}{2} a_n dt^2 \tag{27}$$

$$v_{n+1} = v_n + \frac{a_n + a_{n+1}}{2} dt. \tag{28}$$

These equations tell us that the algorithm is slightly more expensive compared to the Euler-Cromer scheme at $11n$ FLOPS. Here follows an implementation of the algorithm in Python.

```
    def Velocity_Verlet(self,planet,h,n):
        N  = int(n)
        x  = np.zeros(N)
        y  = np.zeros(N)
        vx = np.zeros(N)
        vy = np.zeros(N)
        ax = np.zeros(N)
        ay = np.zeros(N)
        x[0]  = planet.x_position()
        y[0]  = planet.y_position()
        vx[0] = planet.velocity_x()
        vy[0] = planet.velocity_y()
        a_x0,a_y0  = self.accelaration(planet)
        ax[0] =  a_x0
        ay[0] =  a_y0
        for i in range(0,N-1):
            x[i+1] = x[i] + vx[i]*h + 0.5*ax[i]*h**2
            y[i+1] = y[i] + vy[i]*h + 0.5*ay[i]*h**2
            planet.update_position(x[i+1], y[i+1])
            a_x, a_y = self.accelaration(planet)
            ax[i+1]  = a_x
            ay[i+1]  =  a_y
            vx[i+1]  = vx[i] + 0.5*h*ax[i] +  0.5*h*ax[i
                +1]
            vy[i+1]  = vy[i] + 0.5*h*ay[i] +  0.5*h*ay[i
                +1]

        return x,y
```

One sees that it is necessary to compute forces and update acceleration once more in order to update the velocity, which means more operations must be conducted. Time will show if the extra cost of this algorithm is worth the increase in precision it promises.

4 Data

As we wish to determine the orbits of our solar system as accurately as possible, we use date from HORIZON Web-Interface, provided by the Jet Propulsion Laboratory (NASA) at the California institute of technology: `http://ssd.jpl.nasa.gov/horizons.cgi`. Here of data from several celestial bodies can be downloaded. It is simple to find both position and velocity for all particles in our bodies at high precision. Units of length can be set to AU and velocities to AU per day. The velocities must be converted for our simulation. We have faith that these data are accurate.

In an arbitrary system, the entire system will usually gain momentum and drift off in some direction over time. One must usually correct for such drift be substracting total linear momenum. Because the velocity data from NASA is very good, this is unnecessary for our system. Total linear momemtum for the solar system is already zero from our frame of reference.

5 Results

The object-oriented model for the solar system that has been built is very flexible, as the reader will soon come to know. We start by constructing some simple models with two and three bodies, then the entire solar system, and lastly, simulate the perihelion precession of Mercury, as discussed in the theory section.

5.1 Two-Body system: Earth and Sun

A two-body system is a meager representation of the solar system, but is the first important stepping stone towards a more interesting and realistic model. The two-body system has two instances of the Planet class, representing the sun and the Earth. The orbits are found by solutions from both the Euler-Cromer method and the Velocity Verlet method.

5.1.1 Precision of two algorithm

I have tested the two algorithm for different time step and figure 1 and figure 7 show the results for Sun and Earth system. It can be seen that the difference between the two methods is quite clear. The Velocity Verlet method have a much narrower "orbit band" and is therefore more precise.

5.1.2 Conservation of Energy and Angular momentum

Since we consider the Solar system as an isolated system in this project. According to the law of conservation of energy states that the total energy of an isolated system remains constant, it is said to be conserved over time.[1] This law means that energy can neither be created nor destroyed; rather, it can only be transformed or transferred from one form to another. Here the kinetic energy

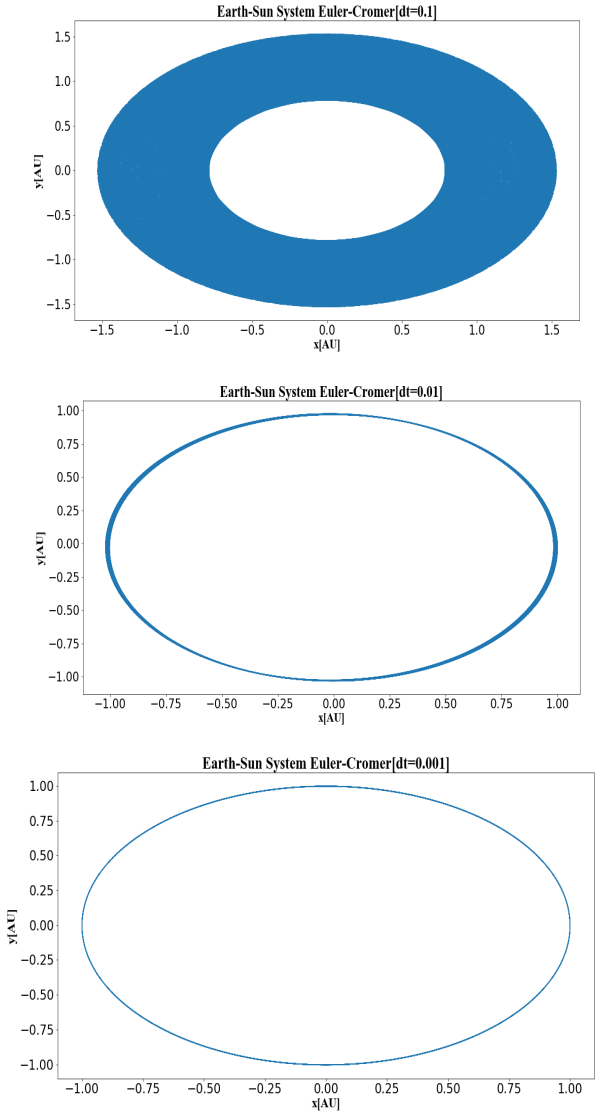


Figure 1: Precision of Euler-Cromer method for differenet dt

converted to the potential energy and vice versa. The angular momentum for earth sun system is also conserved. The conservation of angular momentum law states that as long as there is not an external torque the angular momentum will not change. Figure 4 and 3 shows these properties of the sun-earth system.

N	Euler-Cromer [t]	Velocity-Verlet [t]
10^3	0.008	0.019
10^4	0.194	0.082
10^5	1.9461	0.8430
10^6	19.7751	8.5084

Table 1: Timing Analysis of two algorithm

5.1.3 Escape Velocity

For calculating the escape velocity of the earth I used the trial and error to see in which velocity the earth would escape its orbit. Figure 5 shows the earth orbits in different velocities. As it can be seen the escape velocity is approximately $v = \sqrt{8}\pi AU/yr$.

5.1.4 Analyzing the Earth orbit by changing β

Changing the Gravitational force by replacing

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

with

$$F_G = \frac{GM_\odot M_{\text{Earth}}}{r^\beta},$$

As the plots in figure 6 show the orbit of the earth would not stable as the β reach 3.

5.1.5 Timing Analysis

Table 1 shows a comparison of how much time the two algorithm requires to handle a problem of increasing size. As we expected the Euler-Cromer method is faster than Velocity-Verlet method.

5.2 Three-Body system: Earth, Sun and Jupiter

A natural progression from a two-body system is a three-body system. We have implemented a model consisting of particles representing Jupiter, Earth, and the Sun. Here the effect of Jupiter mass on earth orbit has been studied. As the plot shows the orbit of the earth becomes unpredictable as the mass of Jupiter increases.

5.3 Multi-Body system: all planets

Finally a model of the entire solar system[5] Figure 8 shows a multi-body simulation representing the solar system, integrated using the Velocity Verlet algorithm. One can see that the algorithm handles the problem quite well.

[5]Excluding moons and other satellites, but including Pluto.

5.4 The perihelion precession of Mercury

Figure 9 shows the orbit of mercury around the sun adding general relativistic correction to the gravitational force. It shows Mercury's elliptical path around the Sun shifts slightly with each orbit such that its closest point to the Sun (or "perihelion") shifts forward with each pass. Newton's theory had predicted an advance only half as large as the one actually observed. Einstein's predictions exactly matched the observation. Figure 10 shows the angel of the perihelion of Mercury calculated by equation 13 in the theory section. One can see a clear trend in the angle of the perihelion as predicted. I get a slope of the perihelion angle versus number of orbits of approximately 5^{-4}. It is important to include enough time steps where this accounts to how precis the calculation of the angle is or else we will not possibly include the lowest point in the orbit.

.

6 Discussion

6.1 Object Orientation

This project has been a nice exercise in object oriented thinking.Object oriented programming is close to how we as humans usually thinks about all things in the world. A planet is something that has mass, a position, velocity and some other factors. However, when one learns how to program learns object orientation at a late stage of the learning process.
Most widely used programming languages today are object-oriented to a greater or lesser extent. Critique of object orientation usually emphasizes that design and modelling comes at the expense of other important aspects like the actual computation and algorithms.
Object Oriented Programming provides a clear modular structure for programs and makes it easy to maintain and modify existing code thereby reducing lowering programming "cost".

6.2 Integration Algorithm

We have seen that the Velocity Verlet algorithm is much more precise than the Euler-Cromer method . However the Euler method is a bit faster than Velocity-Verlet method. The discussion of an Euler method versus a Verlet method is a rather big one. Both methods are quite quick cheap which is what you want, and they are usually the two methods of choice if one does not need entirely exact results, but results that are "good enough".

6.3 Mercury

The result I've got for mercury perihelion is differet from calculations.This error may come from several sources. The Uncertainty of the integration methods is one likely source of error. It is also important to include enough time steps where

this accounts to how precis the calculation of the angle is or else we will not possibly include the lowest point in the orbit. Because of hardware limitation I could not calculate for smaller time steps.

7 Conclusion

In this study we have shown the advantages of building an object oriented framework and how it can easily be expanded upon and modified to its needs. We wanted to apply object oriented design on analysis of the solar system and have succeded. Moreover, a few of the classes, like the SolarSystem, Planet can be applied to other problem and in new fields, like molecular dynamics.

We have implemented two integrators, the Euler-Cromer and the velocity Verlet. The velocity Verlet proved to be more stable than the Euler-Cromer method. The reason for this is because Verlet integration methods are symplectic and therefore perfect for system with conserved angular momentum and energy. Additionally, the velocity Verlet method is more CPU costly than the Euler-Cromer method.

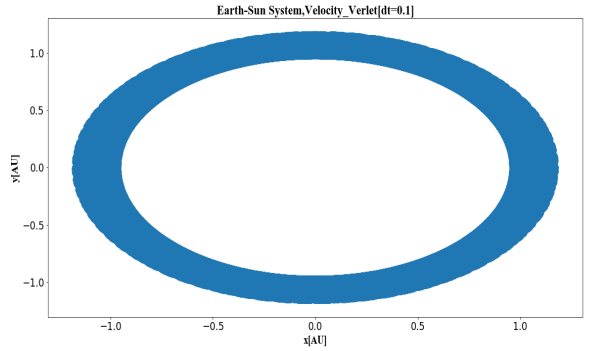
I tried to adjust Mercurys orbit for relativity. We found a clear trend in the perihelion angle of Mercury's orbit, but got disappointing results compared with previous observations and calculations.

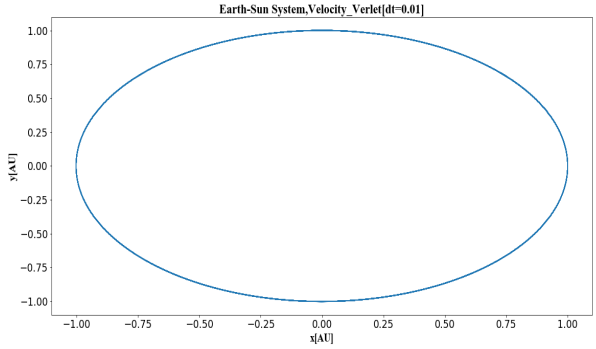
References

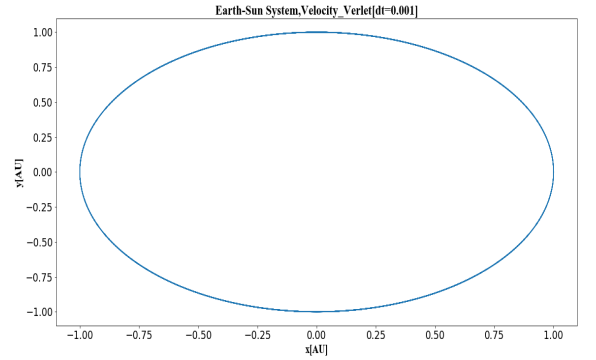
[1] Hjort -Jensen Morten, *Computational Physics*,
Lecture notes fall 2015, 2015.

[2] http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode-bs.html

[3] http://hyperphysics.phy-astr.gsu.edu







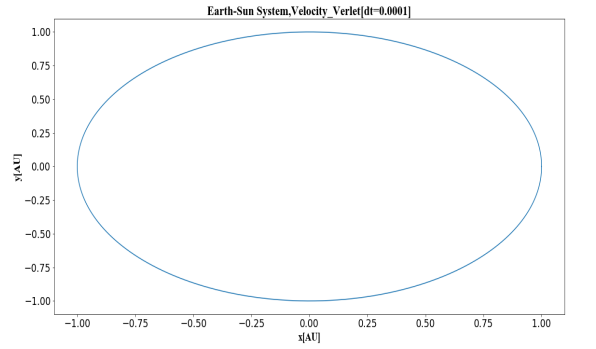


Figure 2: Precision of Velocity Verlet method for differenet dt

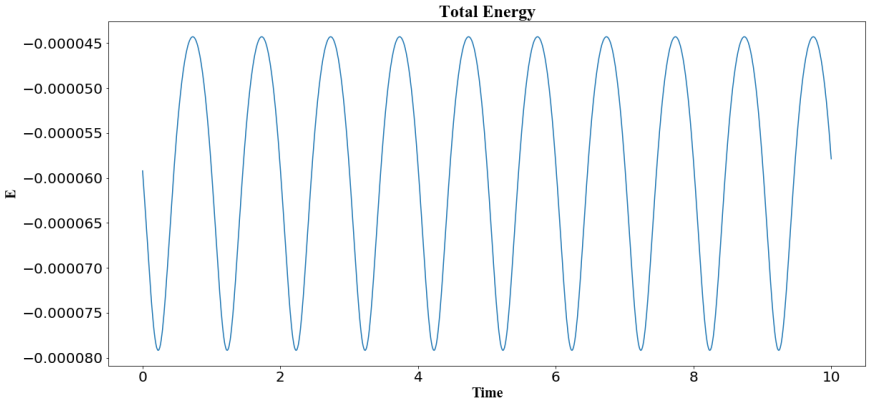


Figure 3: Total energy using Velocity-Verlet method

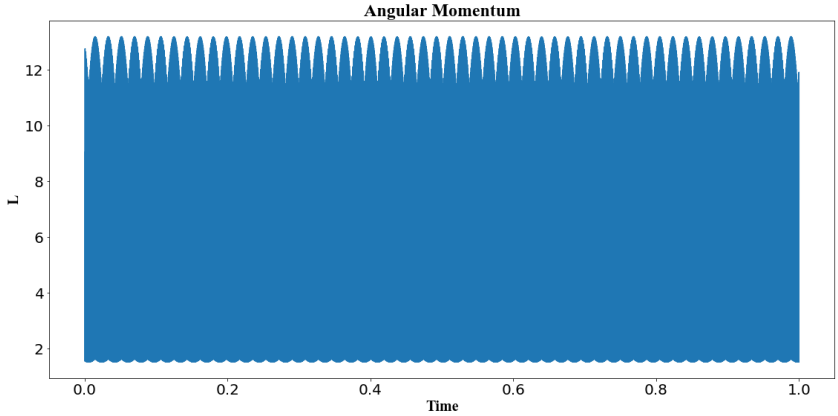


Figure 4: Angular momentum using Velocity-Verlet method

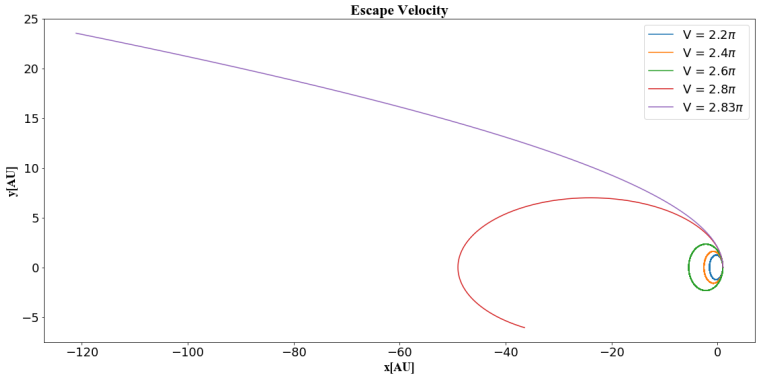


Figure 5: Earth orbit by changing the velocity - Escape Velocity

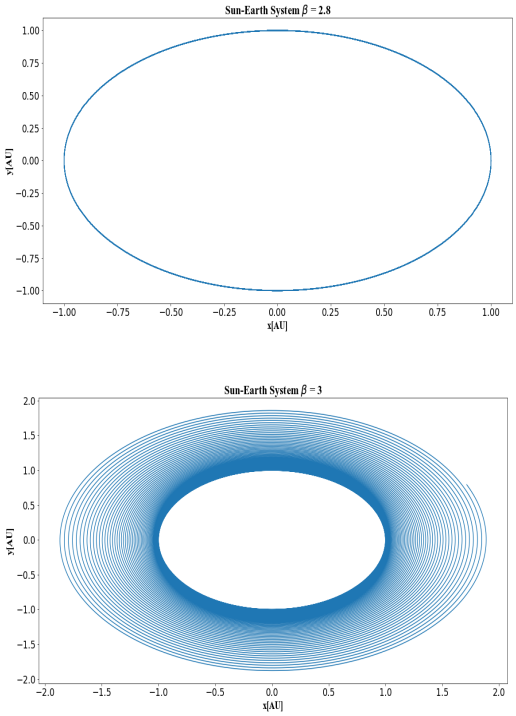


Figure 6: The stability of the system change when β reaches 3

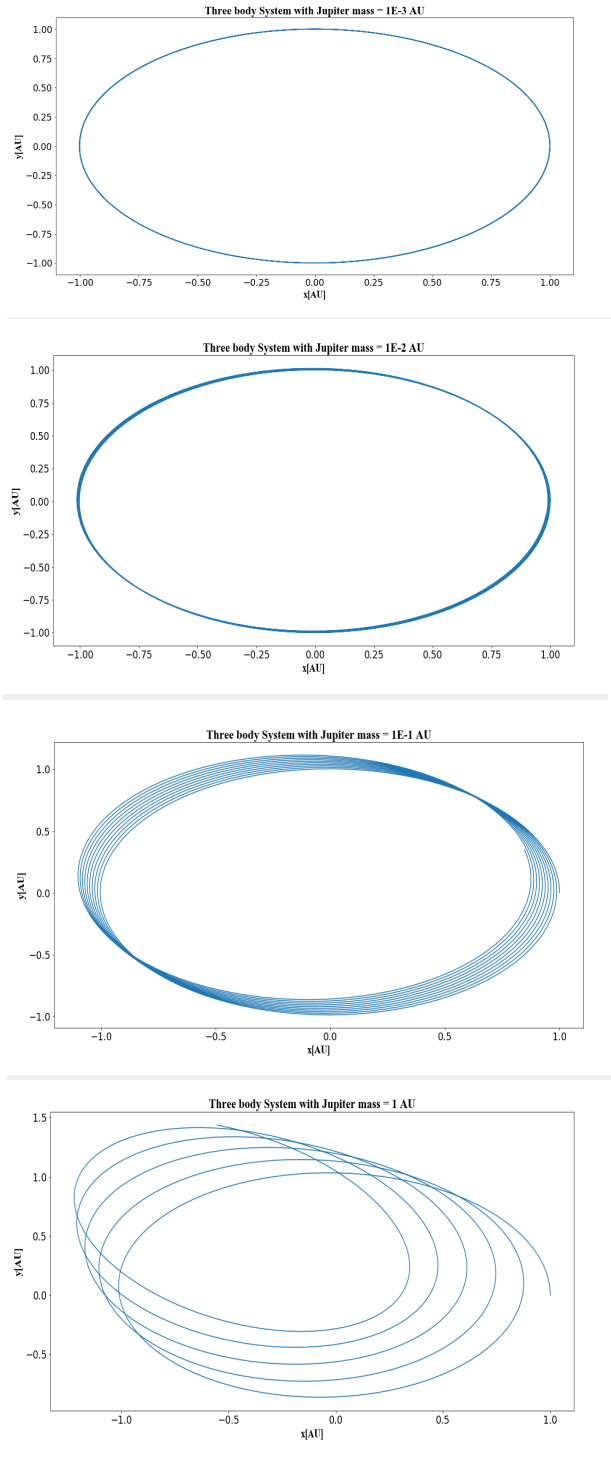


Figure 7: Effect of changing the mass of Jupiter on Earth orbit

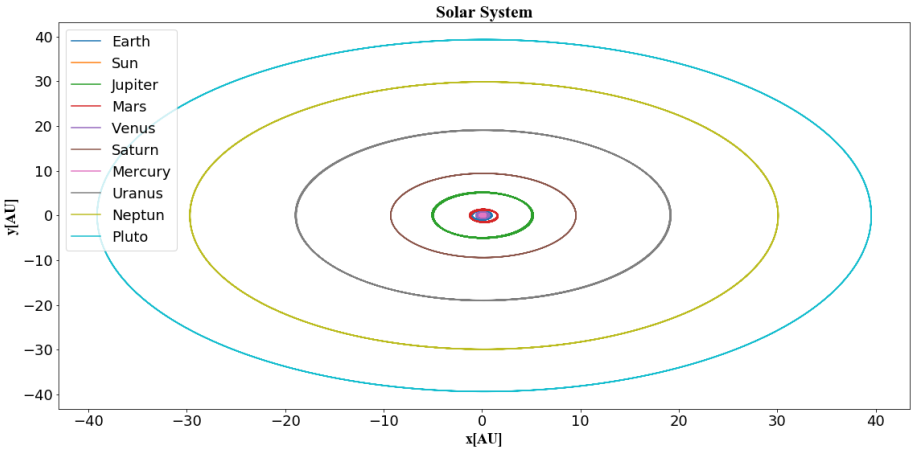


Figure 8: Final model of solar system

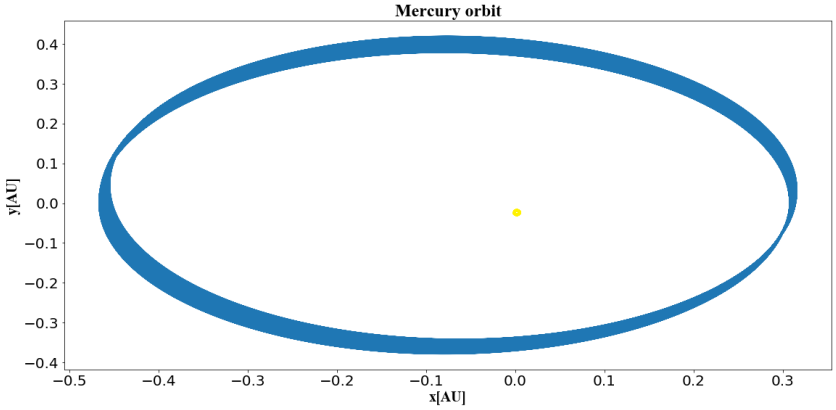


Figure 9: Mercury orbit

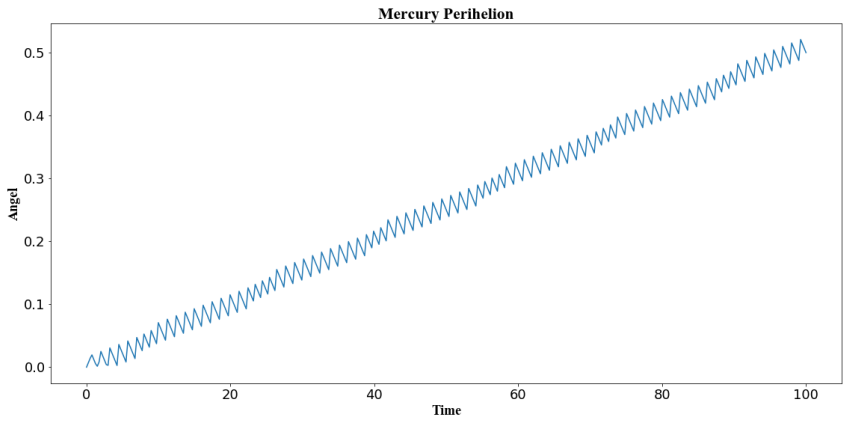


Figure 10: Mercury Perihelion dt = 1E-3