FYS4150 - Computational Physics
Project 3

Fred Marcus John Silverberg

24 October 2018

Abstract

The project will investigate the numerical Euler and Verlet algorithms for simulating a complex system of multiple bodies in space, described by Newtonian laws. Detailed observations of the Newtonian gravitational force and certain critical velocities, as well as the relativistic event of Mercury's perihelion precession are also covered.

I found that the Verlet algorithm yielded more detailed models and satisfactory solved the complex problem. The computational time compared to Euler's algorithm was approximately 15% slower, and require in general 2.75 more flops. The time interval [dt] needed to produce acceptable results were in the order of 10^{-4} and the escape velocity for Earth was found to be $\sqrt{8\pi^2}$. Further, the initial velocity needed for Earth to orbit the Sun was identified as $V_{initial} = 2\pi$.

At last, the theory of general relativity was correctly calculating the perihelion precession of 43".

Contents

1 Introduction

In the field of astrophysics, the complexity of finding analytical solutions escalates as the number of bodies increases. This project will proceed from a simple two body system, to a three body system and finally into a full solar system. Details such as initial and escape velocities will be inspected as well.

Numerically solving these problems using the Euler's algorithm, are to be compared against the Verlet-Velocity algorithm. The comparison will focus on the precision as well as the computational time. In order to validate the given solutions, conservation of physical quantities such as angular momentum, potential energy and kinetic energy will checked.

Further, the Newtonian gravitational force will be investigated, with deeper focus on the radius exponent governing orbital shape. Finally an relativistic addition is to be made for an analysis of the perihelion precession event, related to the Mercury experiment that confirmed the theory of general relativity.

The problems will be handled by making the relevant equations dimensionless and discretize the equations by Taylor expansion. Implementation will occur by an object oriented code in python, that execute the algorithms and visualize the results.

2 Theory

2.1 Newtonian gravity

Newtons law of gravitation describe force as:

$$Fg = \frac{GM_1M_2}{r_{21}^2}\,, \tag{1}$$

Where G is the universal gravitational constant, M_1: mass of bodies(1), M_2: mass of bodies(2). r_{21}^2 is the distance between the center of the two bodies.

Newtons second law of motion describe force as:

$$F = Ma\,, \tag{2}$$

Where [M] is the mass of the body, and [a] is the acceleration

By combining (1) and (2) a set of second order differential equations arise:

$$\frac{d^2x}{dt^2} = \frac{Fg_{\mathrm{x}}}{m_1}, \qquad \frac{d^2y}{dt^2} = \frac{Fg_{\mathrm{y}}}{m_1}\,, \tag{3}$$

Fg_x and Fg_y describe the gravitational force in corresponding direction. The equations are analogous for three dimensions as well, but throughout of this project only two dimensions will be evaluated, representing the xy-plane. The movement in z-direction is neglected.

By decoupling (3) we can identify a set of coupled first order differential equations. This is done by defining the velocity as [v] and the acceleration as [a][1]

$$v_{\mathrm{x}} = \frac{dx}{dt}, \qquad a_{\mathrm{x}} = \frac{dv_{\mathrm{x}}}{dt} = \frac{Fg_{\mathrm{x}}}{m_1}\,, \tag{4}$$

$$v_{\mathrm{y}} = \frac{dy}{dt}, \qquad a_{\mathrm{y}} = \frac{dv_{\mathrm{y}}}{dt} = \frac{Fg_{\mathrm{y}}}{m_1}\,, \tag{5}$$

2.2 Perihelion precession of Mercury

The Newtonian gravity can describe the system sufficiently well, but due to increased gravitational potential close to the Sun, Mercury's orbit is notably affected in a way the Newtonian theory is unable to describe. By introducing Einstein's general theory of relativity, we can describe the effect by adding a correction term. Leading to alternations of equations (1),(4) and (5):

$$Fg = \frac{GM_{\mathrm{s}}M_{\mathrm{m}}}{r_{ms}^2}(1 + \frac{3L^2}{r_{ms}^2 * c^2})\,, \tag{6}$$

$$a_{\mathrm{x}} = \frac{dv_{\mathrm{x}}}{dt} = \frac{Fg_{\mathrm{x}}}{m_1}(1 + \frac{3L^2}{r_{ms}^2 * c^2})\,, \tag{7}$$

$$a_{\mathrm{y}} = \frac{dv_{\mathrm{y}}}{dt} = \frac{Fg_{\mathrm{y}}}{m_1}(1 + \frac{3L^2}{r_{ms}^2 * c^2})\,, \tag{8}$$

[1] http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html

Here the [s,m] notation represent the Sun and Mercury. While [L] is the magnitude of Mercury's angular momentum and [c] is the speed of light in vacuum. The position of Mercury where it is closest to the Sun is called the perihelion point. Observations show that during a period of 100 earth-years, the angle $[\theta_{\mathrm{p}}]$ of which the perihelion precession changes is measured to be 43" (arc-seconds).[2]

$$tan(\theta_{\mathrm{p}}) = \frac{y_{\mathrm{p}}}{x_{\mathrm{p}}}\,, \tag{9}$$

3 Method

3.1 Dimensionless equations

By scaling a differential equation it reduces down to a more simpler form. This is done by identifying the dimension variables and the parameters of the function. The goal is to convert the dimensional variables into new dimensionless variables by relating them in terms of the parameters.

First by using polar coordinates we can do:

$$x = rcos(\theta), y = rsin(\theta), r = \sqrt{x^2 + y^2}\,, \tag{10}$$

$$Fg_{\mathrm{x}} = -\frac{GM_1M_2}{r_{21}^2}cos(\theta) = -\frac{GM_1M_2}{r_{21}^3}x = a_{\mathrm{x}}\,, \tag{11}$$

$$Fg_{\mathrm{y}} = -\frac{GM_1M_2}{r_{21}^2}sin(\theta) = -\frac{GM_1M_2}{r_{21}^3}y = a_{\mathrm{y}}\,, \tag{12}$$

By using the astronomical unit 1 [AU] = $1.5 * 10^(11)$ for length and measure mass in terms of Sun masses $[M_1]$ and time in years [Y], we have:[3]

$$V_{\mathrm{circulation}} = \frac{2\pi * AU}{Y}\,, \tag{13}$$

$$G = V_{\mathrm{circulation}}^2 r = 4\pi^2(AU^3/yr^2)\,, \tag{14}$$

3.2 Discretizing the equations

By defining time as evenly time steps $[t_{\mathrm{i}}]$ in the interval $[t_0,t_{\mathrm{end}}]$:

$$t_{\mathrm{i}} = t_0 + ih\,, \tag{15}$$

$$h = \frac{t_{\mathrm{end}} - t_0}{N}\,, \tag{16}$$

Where [h] is the step length and [i] represent the step point, both governed by [N], the number of step points such that [i] = [0,1,2...N].[4]
Further:$[(x,y)_{\mathrm{t[i]}}] = [(x,y)_{\mathrm{i}}]$,$[(vx,vy)_{\mathrm{t[i]}}] = [(vx,vy)_{\mathrm{i}}]$,$[(ax,ay)_{\mathrm{t[i]}}] = [(ax,ay)_{\mathrm{i}}]$

[2] https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Projects/2018/Project3/pdf/Project3.pdf
[3] http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html
[4] http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html

3.3 Euler's algorithm

The method is a numerical algorithm that solves differential equations by using information about the derivatives at the beginning of the time interval. The method become more accurate as the the number of points [N] increases. It yields a local error of h^2 and a global error of h.

First, by approximate our derivatives with Taylor expansion:

$$\frac{df(t)}{dt} \approx \frac{f(t)_{\mathrm{t}+1} - f(t)_{\mathrm{t}}}{h}\,, \tag{17}$$

Second, with knowledge of the initial position $[x_0, y_0]$ and velocity $[vx_0, vy_0]$:

$$x_{\mathrm{i}+1} = x_{\mathrm{i}} + hvx_{\mathrm{i}} + O(h^2)\,, \tag{18}$$

$$y_{\mathrm{i}+1} = y_{\mathrm{i}} + hvy_{\mathrm{i}} + O(h^2)\,, \tag{19}$$

$$vx_{\mathrm{i}+1} = vx_{\mathrm{i}} + hax_{\mathrm{i}} + O(h^2)\,, \tag{20}$$

$$vy_{\mathrm{i}+1} = vy_{\mathrm{i}} + hay_{\mathrm{i}} + O(h^2)\,, \tag{21}$$

Pseudo code of algorithm:[5]

```
initial_values = []
for i in range(0,N-1)
   x[i+1] = x[i] + h*vx[i]
   y[i+1] = y[i] + h*vy[i]
   vx[i+1] = vx[i] + h*ax[i]
   vy[i+1] = vy[i] + h*ay[i]
return(x,y,vx,vy)
```

The number of flops are: 4N*D with $N = \frac{1}{h}$ and D stands for dimension.

3.4 Velocity-Verlet algorithm

A numerical method for solving differential equations, especially related to Newton's equations, with similarities to Euler's algorithm but with an extra term. The stability of the method increases such that the local error become h^3 and the global h^2. Note that the forward velocity depends on the forward acceleration.

$$x_{\mathrm{i}+1} = x_{\mathrm{i}} + hvx_{\mathrm{i}} + ax_{\mathrm{i}}\frac{h^2}{2} + O(h^3)\,, \tag{22}$$

$$y_{\mathrm{i}+1} = y_{\mathrm{i}} + hvy_{\mathrm{i}} + ay_{\mathrm{i}}\frac{h^2}{2} + O(h^3)\,, \tag{23}$$

$$vx_{\mathrm{i}+1} = vx_{\mathrm{i}} + hax_{\mathrm{i}} + \frac{dax_{\mathrm{i}}}{dt}\frac{h^2}{2} + O(h^3)\,, \tag{24}$$

$$vy_{\mathrm{i}+1} = vy_{\mathrm{i}} + hay_{\mathrm{i}} + \frac{day_{\mathrm{i}}}{dt}\frac{h^2}{2} + O(h^3)\,, \tag{25}$$

[5] http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html

Pseudo code of algorithm:[6]

```
initial_values = []
for i in range(0,N-1)
   x[i+1] = x[i] + h*vx[i] + 0.5*ax[i]*h^2
   y[i+1] = y[i] + h*vy[i] + 0.5*ay[i]*h^2
   ax[i+1] = -x[i+1]*(4pi^2/r[i+1]^3)
   ay[i+1] = -y[i+1]*(4pi^2/r[i+1]^3)
   vx[i+1] = vx[i] + 0.5*h*(ax[i+1]+ax[i])
   vy[i+1] = vy[i] + 0.5*h*(ay[i+1]+ay[i])
return(x,y,vx,vy,ax,ay)
```

The number of flops are: 11N*D with $N = \frac{1}{h}$ and D stands for dimension.

3.5 Verification of algorithms

Earth need a specific initial velocity in order to have stable circular orbits obeying the algorithms above. The initial velocity can be obtained by setting equation (1) against Newtons second law $[F = ma]$:[7]

$$aM_2 = \frac{GM_1M_2}{r_{21}^2}\,, \tag{26}$$

And with knowledge of $[ar = v^2]$ and from equation (14) we have:

$$v = \sqrt{\frac{GM_1}{r_{21}}} \qquad => \qquad v = 2\pi\sqrt{\frac{AU^3}{rY}}\,, \tag{27}$$

Using that the distance from Earth to Sun is $[r_{21} = 1AU]$:

$$v_{\text{initial}} = 2\pi\,, \tag{28}$$

As the algorithms precision depend on the step length and the size of the values, there is also implemented a stability test for different step lengths.

For further verification of the algorithms ability to predict our solar system, the physical quantities of potential energy, kinetic energy and angular momentum are tested by concluding conserved, for a defined zero $[10^{-6}]$. Angular momentum should be conserved throughout the calculated interval, since the system is free from external torque and forces. The case stands for the potential and kinetic energies as well, since no external forces are active.

[6] http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html

[7] https://www.khanacademy.org/science/physics/one-dimensional-motion/kinematic-formulas/a/what-are-the-kinematic-formulas

3.6 Implementation of methods

Initial data regarding the bodies are obtained from Nasa Horizons website.[8]

The calculations and visualizations are preformed in python with usage of the numpy library. Further, the project is touched by object oriented coding with classes handling information about the bodies and of calculations. A separate python file [methods.py] was also created, holding various methods.

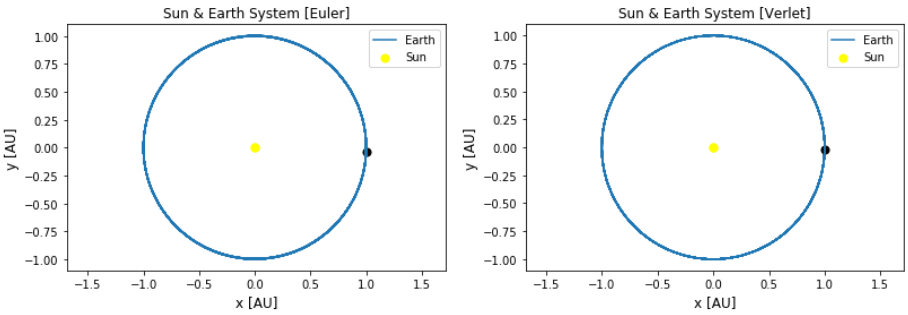
```
Classes:
* Bodies: Handle information about the bodies
* Solver_small: Execute calculations for small system
* Solver_large: Execute calculations for large system
```

4 Results

4.1 Earth and Sun System

Figure 1: Earth-Sun system

(a) Euler. (b) Verlet.



Calculated with [0.0002dt,100000n,20y].
The black point represents earth after 20 years.

Table 1: Conservation errors

Algorithm	Angular Momentum	Potential Energy	Kinetic Energy
Euler	$1.80 * 10^{-19}$	$8,83 * 10^{-11}$	$8.94 * 10^{-11}$
Verlet	$3.66 * 10^{-19}$	$4.12 * 10^{-17}$	$3.89 * 10^{-17}$

As seen in table [1], all quantities are conserved in terms of numerical expectation. Error calculation is made between initial values and final values.
Note that the Verlet algorithm yield less errors. [0.0002dt,100000n,20y]

[8] http://ssd.jpl.nasa.gov/horizons.cgi

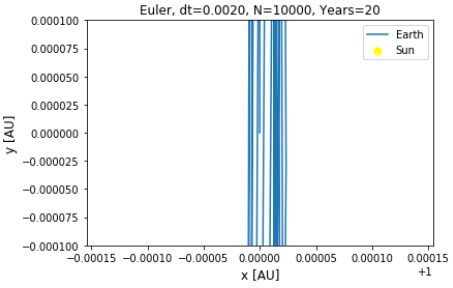
Table 2: Calculation time for 20 years [sec]

dt	Euler	Verlet
0.02	0.008	0.010
0.002	0.076	0.079
0.0002	0.650	0.750
0.00002	6.44	7.34
0.000002	66.2	76.0

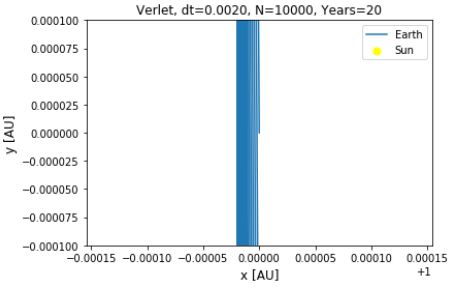
The difference in processing time are negligible for our scale.
Verlet is approximately 15 percent slower.

Figure 2: Stability plots, strongly zoomed.

(a) Euler.



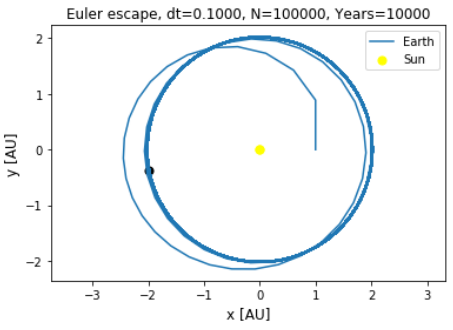
(b) Verlet.



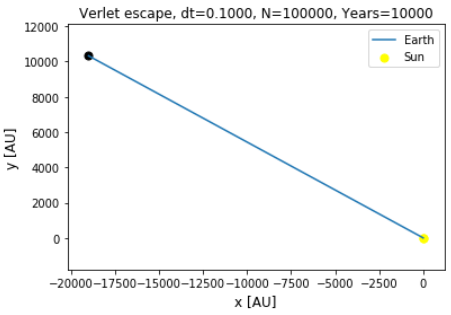
For [0.002dt,100000n,20y] both methods failed to make accurate calculations, as seen in figure [3]. For [0.0002dt,100000n,20y] both methods made stable calculations, the radius between Sun and Earth was stable at 1.0 AU.

Figure 3: Escape situation

(a) Euler.



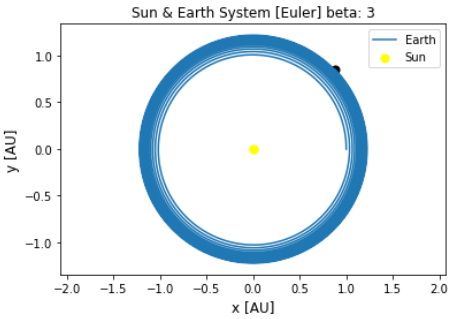
(b) Verlet.



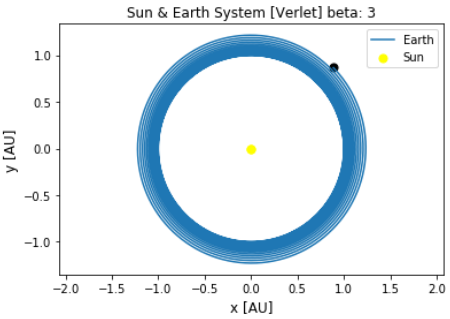
The velocity used is the calculated escape velocity: 8.88 AU/y, found by equating the formulas for kinetic energy and potential energy. It is noted that by Euler's method more extreme values was needed, a situation of $V_{initial} = 11$ AU/y with dt $= 10^{-6}$ and a timespan of 10000 years finally let Earth escape.

Figure 4: Beta variation.

(a) Euler.



(b) Verlet.

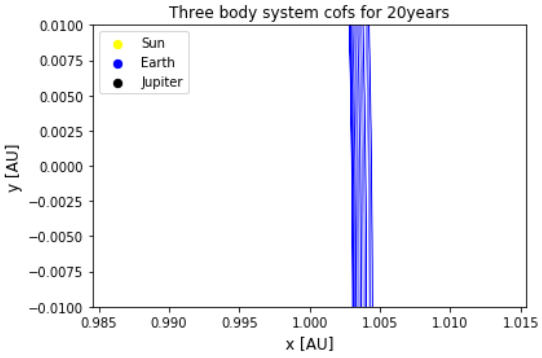


By allowing the exponent of the radius in eq [1] to vary between beta = [2,3], the results became gradually unstable. Especially when closing in on the value 3, where both potential energy and kinetic energy conservation did not hold.

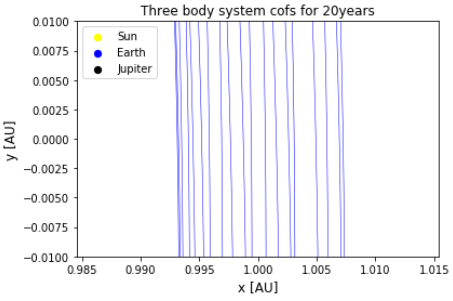
4.2 Earth, Jupiter and Sun System

Figure 5: Mass change of Jupiter [Verlet]

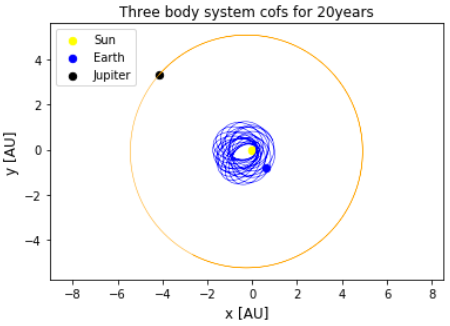
(a) Ordinary mass



(b) 10 times the mass



(c) 1000 times the mass

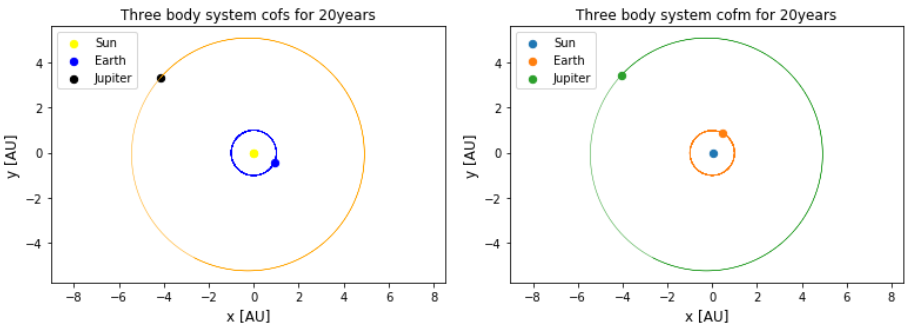


By increasing the mass of Jupiter earth become unstable, which is highly indicated on figure [5c]. Angular momentum, PE or KE are not conserved for any of the three situations. a) and b) are strongly zoomed for highlighting the effect.

Figure 6: Three Body System.

(a) Sun as center.

(b) Mass as center.

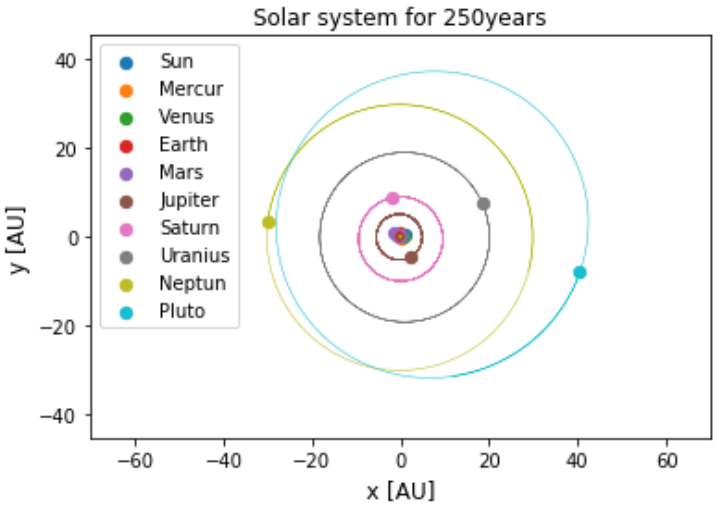


With the center of mass as the origin of the three body system, all physical quantities are conserved. And earth reaches a few distances longer in 20 years.

4.3 Solar System

Figure 7: Solar System

(a) Whole system.



The Verlet algorithm preform precision calculations for the whole system. There is four warnings regarding conservation of angular momentum, the values are listed in table [3] below. [0.00025dt,10^6n,250y]

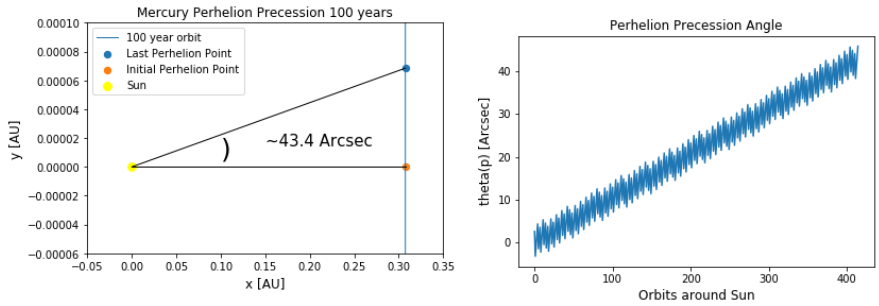
Table 3: Conservation warnings:

Planet	Angular Momentum [error]
Mars	0.111
Jupiter	0.016
Mercury	0.004
Saturn	0.001

4.4 The perihelion precession of Mercury

Figure 8: Perihelion precession.

(a) Euler. (b) Verlet.



The simulation is calculated with values: [10^{-6}dt,10^{8}n,100y]

5 Discussion

The simple model of only Earth and the Sun gives the opportunity to study the efficiency and precision of the algorithms. For larger time intervals [dt] the Euler method seems behave more unstable compared to the Verlet method. This can be observed in Figure [2], which also indicate that both methods decrease in precision for $dt > 0.002$. When precision are of concern it is wise to have a look at the invested time for the calculation, which for these two algorithms are negligible in comparison, see table [2]. With the number of flops in mind, this can probably be explained by the pre-calculation of certain steps in the algorithm for Verlet, reducing the number of flops.

As the initial velocity of the earth have implications on how the orbital shape will behave and if the body will stay in the system, it is of certain interest to find the critical escape velocity. By trial and error it became clear that the Verlet method was very accurate compared to the analytical [$V_{escape} = \sqrt{8\pi^2}$]. While the Euler's method was more sensitive regarding choice of [dt], number of years and $V_{initial}$, as can be further exploited in figure [3]

In the case where the gravitational force is altered by the exponent for the radius, the orbital shape never close. It is highlighted in figure [4], where Earth is

slowly disappearing from the origin. Note that the conservation of mechanical energy is broken as well, indicating disturbance in the force balance.

The effect of Jupiter on Earth is notable in the sense that adding the body yields again, disturbance in the force balance. Which can be identified by the result of unstable orbit and non-conserved physical quantities, see figure [5]. For an ordinary mass of Jupiter, Earths orbital shape is slightly altered, but for a mass 1000 times greater, Earth eventually crashes into the Sun. However, by correcting the system into an equilibrium state, where the origin is the center of the mass, the calculations behave as normal and the physical laws are obeyed.

This project was my first hands on experience with object oriented code, which essence became clear when the whole solar system was to be implemented. It was convenient to add bodies and let the Varlet algorithm construct an impressive model of the solar system. The results are found in figure [7], the attached table may hint about the drawback of numerical solutions, as the defined zero is beaten but still acceptable close to zero, regarding conservation of angular momentum.

Additionally the event of the changing perihelion point of Mercury was observed by implementing a relativistic term into the Newtonian gravitational force. This task required fine tuned time intervals for extracting the small variations in the angles. The produced result are seen in figure [8], to be highlighted is the perihelion precession value of 43.4" for the last perihelion points, which corresponds well with mentioned theory.

6 Conclusions

Both algorithms successfully constructed a model for the simple task that obeyed the physical laws by conserving the angular momentum, potential energy and kinetic energy, for sufficiently small time intervals, at least $[dt < 0.0002]$. This with the Verlet model being negligible slower, but outperforming the Euler model in detail, clearly identified by the superior precision of the critical escape velocity $[V_{escape} = \sqrt{8\pi^2}]$. Hence the tiny computational cost of using Verlet for the reward in precision, places the Verlet algorithm as an eminent choice for detail requirements and complex systems.

The addition of Jupiter had a disturbing effect on the equilibrium. It was highlighted that the system would search for a new balance, and hence unstable motions was to be observed. By correcting this in terms of a center of mass origin, the stability was acceptable restored. It must be mentioned that there is possibilities that an sharper resolution of the system would increase the instability regarding the conservation of physical quantities, since information may be lost as the calculations and values grows. During the project I had my concerns regarding the center of mass. In the future this would be the subject to investigate further, since the calibration may be needed along the calculations in order to hold the origin and conserve angular momentum.

Alternation of the Newtonian gravitational force by experimenting with the ra-

dius exponential, opened the insight to non-closed orbits, which in turn gave rise to the introduction of a small relativistic term. By adding this term an investigation into the perihelion precession of Mercury could be done and successfully calculated to approximately 43", which coincide with the observed event, confirming that the general relativity theory could explain the phenomena.

References

[1] Nasa Horizon. 2018. HORIZONS Web-Interface. [ONLINE] Available at: http://ssd.jpl.nasa.gov/horizons.cgi. [Accessed 24 October 2018].

[2] Morten Hjorth-Jensen. 2018. Computational Physics Lectures: Ordinary differential equations. [ONLINE] Available at: http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html. [Accessed 24 October 2018].

[3] Morten Hjorth-Jensen. 2018. Project 3 [ONLINE] Available at: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Projects/2018/Project3/pdf [Accessed 24 October 2018].

[4] Khan academy. 2018. Khan academy. [ONLINE] Available at: https://www.khanacademy.org/science/physics/one-dimensional-motion/kinematic-formulas/a/what-are-the-kinematic-formulas. [Accessed 24 October 2018].

Access to all material can be found at: https://github.com/silverberg89/FYS4150/tree/master/Project3