PROJECT 3: MODELING THE SOLAR SYSTEM
FYS4150 - COMPUTATIONAL PHYSICS

MARKUS LEIRA ASPRUSTEN MAREN RASMUSSEN METIN SAN

https://github.com/MetinSa/FYS4150/tree/master/Project_3

Abstract. This project involves creating a flexible object-oriented framework intended to simulate the Solar System. The model starts by looking at the Sun-Earth system, but is gradually expanded to include the entire Solar System. The planetary orbits are computed using both the forward Euler and velocity Verlet methods. We find that the velocity Verlet proves to be the superior integration method as it produces more accurate results at a negligible runtime increase. The framework is then used to study interesting properties of planetary systems such as the escape velocity of Earth which is found to be $\sqrt{8}\pi$ AU/yr, and the perihelion precession of Mercury found to be 6.74 ± 0.004 arcseconds per year.

1. Introduction

This project has two main purposes. The first is to illustrate some of the advantages with object-oriented programming. Object orientation leads to flexible and reusable code which is easily expandable. The second goal is to study popular methods for solving differential equations. The two methods we will look into are the forward Euler and velocity Verlet methods and how these compare in terms of stability and energy conservation.

We will attempt to reach these goals by creating a model of the Solar System. We will start by studying the two-body problem between the Sun and the Earth. Further we will include Jupiter, and look closer at this three-body problem. The Sun-Earth-Jupiter system will then be extended to represent the entire Solar System, which is easily achieved using object orientation. This model will allow us to study many of the interesting properties of the Solar System, one being the escape velocity in two or three-body problems. Another property concerns the energy and angular momentum in such systems, and whether or not the conservation of these quantities depend on the methods used. Finally we will look at the perihelion precession of Mercury. Mercury's orbit had troubled scientists for a long time and was recognized as a big problem to celestial mechanics prior to the introduction of Einstein's General Relativity.

In addition to this introduction, the report will consist of a theory section where we give a background for the study and the necessary physics which mainly consist of classical gravity and some relativistic corrections. We will then present the numerical methods used in our model. The section following this considers the object orientation of the code where we elaborate on the flow of data and its importance. And after that the results of our Solar System study is presented, and the report is wrapped up with a discussion of the results and some final remarks.

The code is written in `C++` and, to a small extent, `python`, and can be found by following the GitHub URL located beneath the author names on the front page.

Date: October 24, 2018.

2. Theory

2.1. **Newtonian Kinematics.** There are essentially two fundamental equations in classical celestial mechanics. The first is Newton's law of gravitation which describes the gravitational force between two bodies of mass. The second equation is Newton's second law which relates the acceleration of an object to the forces working on it. Combined, these two laws lie the foundation of our Solar System model.

Newton's law of gravitation is given as

$$F_G = G\frac{M_1 M_2}{r^2}, \tag{1}$$

where G is the gravitational constant, M_1 and M_2 are the masses of the two bodies and r is the distance separating the two masses. We will consider a three-dimensional model of the Solar System in which the gravitational force becomes a vector expressed as

$$\mathbf{F}_G = G\frac{M_1 M_2}{r^3}\mathbf{r}, \tag{2}$$

where $\mathbf{r}$ is the three-dimensional position vector in Cartesian coordinates and r is the length of $\mathbf{r}$.

Newton's second law in three-dimensions states that the sum of the forces is expressed as

$$\mathbf{F} = M\mathbf{a}, \tag{3}$$

where M is the mass of the object acted upon, and $\mathbf{a}$ is the three-dimensional acceleration vector of that object. When working on large scales as in celestial mechanics, the only force worth considering is the gravitational force, and the total force is then solely dependent on gravity.

2.2. **Unit Scaling.** When working on large scale systems such as the Solar System, it is often wise to rescale quantities. If we now consider the two-body Sun-Earth system, Newton's second law now states

$$F_G = G\frac{M_\odot M_\oplus}{r^2}, \tag{4}$$

where $M_\odot = 2 \times 10^{30}$kg is the mass of the Sun, and $M_\oplus = 6 \times 10^{24}$kg is the mass of the Earth. If we assume that Earth's orbit around the Sun is perfectly circular, the acceleration of Earth is given by the centripetal acceleration

$$a = \frac{v^2}{r}, \tag{5}$$

where v is Earth's velocity. If one inserts this acceleration into Newton's second law, we find that the following relation between the two force expressions

$$G\frac{M_\odot M_\oplus}{r^2} = \frac{M_\oplus v^2}{r},$$

which can be solved the constants

$$GM_\odot = v^2 r. \tag{6}$$

The purpose of the rescaling is to redefine the constants of equation (6) to a simpler form, allowing us to work with easier numbers. We start by redefining the mass of the Sun to $M_\odot = 1$. This means that the mass of the other planets are now given as fractions of the Solar mass, where for instance $M_\oplus = 3 \times 10^{-6}$. The distance between the Sun and Earth is on average $r = 1.5 \times 10^{11}$m. This distance

is defined as 1 astronomical unit AU. Adopting this definition means that we now have $r = 1\text{AU}$. We know that the Earth completes one orbit per year meaning that it has a frequency of $f = 1/\text{yr}$. These definitions allow us to redefine Earth's velocity through its angular velocity

$$\omega = 2\pi f = \frac{2\pi}{\text{yr}}, \qquad v = \omega r = \frac{2\pi\text{AU}}{\text{yr}}. \tag{7}$$

If we insert the new definition of the Solar mass $M_\odot$ and Earth's velocity v and distance r into equation (6) we find that the Gravitational constant G can be rescaled to read

$$G = 4\pi^2 \frac{\text{AU}^3}{\text{yr}^2}. \tag{8}$$

2.3. **Energy.** The gravitational force is a conservative force. This means that the total energy in the system should remain constant over time. This conservation will play a role in the stability analysis of our model. A system with a non-conserved energy can reflect the quality of the methods used, or simply indicate coding errors.

The sum of the kinetic energy in the system for N given objects is expressed as

$$K = \sum_{i=1}^{N} \frac{1}{2} M_i v_i^2, \tag{9}$$

where M_i and v_i are the mass and velocity of object i. Similarly, the total potential energy in the system is found by summing over all the objects and getting all the individual interactions

$$U = \sum_{i=1}^{N} \sum_{j=i+1}^{N} G \frac{M_i M_j}{r_{j,i}}, \tag{10}$$

where i and j now sums over the different objects, and $r_{j,i} = |\mathbf{r_j} - \mathbf{r_i}|$.

2.4. **Angular momentum.** Angular momentum is a quantity which describes the rotational state of a system. If there are no external forces acting on a system, then the angular momentum is said to be conserved. This is the case for our Solar System model, which means that the total angular momentum should be conserved. The total angular momentum is given by the following expression

$$\mathbf{L} = \sum_{i=1}^{N} M_i \mathbf{r}_i \times \mathbf{v}_i.$$

If we again consider the two-body Sun-Earth system discussed during the unit scaling section, Earth's angular momentum per mass becomes $|\mathbf{L}/M| = l = 2\pi$.

2.5. **Escape Velocity.** In celestial mechanics, the minimum velocity required for an object to escape the gravitational influence of a massive body is called the escape velocity. The escape velocity is given as

$$v = \sqrt{\frac{2GM}{r}}, \tag{11}$$

where M is the mass of the body one tries to escape, and r is the distance to that body. This formula is derived by considering energy conservation in $K = U$, where K is the kinetic energy seen in equation (9), and U is the potential energy from equation (10). This equality is then solved for the velocity in K resulting in the escape velocity.

When considering the Sun-Earth system in our model, we find that the escape velocity of Earth has an exact solution. By inserting for the definition of G from

equation (8) in addition to the other rescaled quantities from section 2.2, we find that the escape velocity of Earth is

$$v = \sqrt{8}\pi \frac{\text{AU}}{\text{yr}}. \tag{12}$$

2.6. **The Perhelion Precession of Mercury.** Closed elliptical orbits are a direct result of the $1/r^2$ factor in the Newtonian Gravity force. If one were to even slightly changes this factor, the orbit would not be closed, meaning that an object would not return to its exact position after completing a full orbit. An example of this phenomena is Mercury which has a perihelion precession of 43 arcseconds per century.

We will study the Sun-Mercury system and the perihelion precession in our model. We will do so by adding a relativistic corrective term to the force expression seen equation (1). The new expression is then given as

$$F_G = G\frac{M_\odot M_☿}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right], \tag{13}$$

where $M_☿$[1] is the mass of Mercury in solar fractions, c is the speed of light in vacuum and $l = |\mathbf{r} \times \mathbf{v}|$ is the magnitude of Mercury's orbital angular momentum. We are interested in studying the precession of Mercury. We define the perihelion angle θ_p to be given as

$$\tan\theta_\text{p} = \frac{y_\text{p}}{x_\text{p}}, \tag{14}$$

where x_p and y_p is the x and y position of Mercury at perihelion, i.e the point where Mercury is at its closest to the Sun.

3. Method

3.1. **Integration.** The trajectory of an object in the solar system is constantly affected by all the other objects through Newton's law of gravitation. This means that we are looking at a coupled system, and computing the orbits therefore require stepwise integration of all the objects simultaneously. We will look to solve this problem through the two popular integration methods, the Forward Euler and the Velocity Verlet methods.

3.1.1. *Forward Euler.* The forward Euler method is a first order integration method with a local error that is proportional to the step size squared $O(h^2)$. Considering our system in 1-dimension, the forward Euler algorithm is given by the following recursive relations

$$v_{n+1} = v_n + a_n dt, \tag{15}$$

$$x_{n+1} = x_n + v_n dt, \tag{16}$$

where dt is the stepsize, v_n is the velocity, a_n the acceleration and x_n the position of the object after n steps. This algorithm allows us to compute the position of an object as long as we know its acceleration and its initial position and velocity. As mentioned in the theory section, the acceleration on an object in our system is found by relating the Newtonian gravitational force with Newton's second law.

If the acceleration calculation is excluded, the required floating point operations (FLOPS) for the forward Euler scheme are a simple 4 per timestep, making it the fastest possible integration method. It should be mentioned that calculating the force in our system through the laws of Newton require a single FLOP. This means

[1] ☿ is the official symbol for Mercury. See NASA's Solar System symbols for more information.

that the effective number of FLOPS per timestep is 5, and the total is then $5N$, where N is the total number of timesteps.

In addition to its simplicity and speed, the algorithm is also very intuitive in nature which makes it a go-to method for educational purposes. It is however somewhat lacking in terms of its accuracy. As we will see later, this method does not necessarily conserve the energy in a system, which is a necessity when trying to achieve realistic Solar System models.

3.1.2. *Velocity Verlet.* A more accurate calculation of the orbits can be achieved using the velocity Verlet method. This method is a part of the Verlet family which is a set of integration methods designed to effectively integrate Newton's equations of motion. Considering our system again in 1-dimension, the velocity Verlet algorithm is given by the following recursive relations

$$v_{n+1} = v_n + \frac{1}{2}(a_{n+1} + a_n)dt, \tag{17}$$

$$x_{n+1} = x_n + v_n dt + \frac{1}{2}a_n dt^2. \tag{18}$$

This method has a local error that is proportional to the stepsize cubed $O(h^3)$, which by it self might motivate its use over the forward Euler method. This is however not the primary reason for its use. The velocity Verlet method is a symplectic integration scheme, meaning that the energy remains mostly constant during integration. However, the conservation of energy comes with a price.

The velocity Verlet algorithm has 10 FLOPS per timestep (assuming that the acceleration is known) compared to the 4 of the forward Euler method, making it more than twice as demanding in terms of computational cost. If we consider its application to our system the method requires two individual acceleration calculations at step n and $n+1$. This increase the number of effective FLOPS to $12N$. Usually when one considers the numerical stability and conservation of energy provided by the velocity Verlet algorithm, the additional FLOPS are well justified. The velocity Verlet will therefore be our main method of integration throughout this study.

3.2. **Object Orientation.** As mentioned in the introduction, object orientated programming introduces a large set of benefits to both the programmer and the user. An object-oriented code can easily be reused through inheritance and cooperation between the different classes and subclasses. This removes the necessity to copy code from a program to another. The programs can also be recycled as they are written on the most general form. Another advantage of object orientation is the flexibility it provides through polymorphism. This is when a single function is capable of adapting to be used in a variety of settings and classes. Nevertheless, the biggest benefit of object orientation is perhaps the modularity. The functionality of a program is separated into several independent modules targeting the different aspects of the program. Each individual module or object is self-contained as they have their own functionality and leave the rest of the code to itself. This increases the maintainability and troubleshooting of the code as it easier to deal with individual objects rather than a single program spanning thousands of lines.

It can be wise to test the methods and functions one wants to use before one object orientate the code. This is usually done in order to make sure that everything is running correctly, and is often a key part of the object orientation planning process as it provides some insight into how the flow of data needs to be structured. We have included a simple Sun-Earth two-body problem in a the SunEarth_without_OO folder found on our GitHub.

3.2.1. *Class Hierarchy.* Here follows an overview of the class structure, the function of each object and the flow of data in our class hierarchy. Our code consists of 6 individual classes. The first class `Vec3` is a modified version of the three component vector class found through the course webpage[2]. We are mainly working with three dimensional vectors so making a custom vector class can be practical as it allows us to define and overload spesific operations. These definitions are used to vectorize the code, resulting in a shorter and cleaner programs. `vec3` is also an example of polymorphism as it can be used with different arguments.

The two next classes are `PlanetaryBody` and `SmallObjects` which creates the different planetary objects we want to study. In `PlanetaryBody`, each object instance is given a mass, name and a position and velocity vectors through the `vec3` class. The class is also responsible for computing object spesific quantities such as the kinetic energy and angular momentum. `SmallObjects` is very similar to `PlanetaryBody`. It is responsible for initializing smaller objects such as comets, asteroids or satellites whereas `PlanetaryBody` is meant to deal with stars, planets and moons. The objects created by `SmallObjects` is similar to those of `PlanetaryBody` with the exception that they have no mass. This approximation is implemented in order to simplify and shorten computation time, and is well justified as the forces exerted by smaller objects are insignificant compared to the forces exerted by moons, planets and stars.

What follows is the `Gravity` class which deals with the above mentioned force. `Gravity` takes in two objects as arguments and calculates the gravitational force between them using Newtons law of gravitation seen in equation (2). The class is also an example of polymorphism as it can be called with an additional argument indicating the use of the relativistic force expression seen in equation (13). It is also responsible for computing the potential energy between the two objects, as this quantity is closely related to the force.

`SolarSystem` is the master class of the program. It initializing any particular planetary system one wants to study. It accepts one argument, which is a text file containing the initial conditions of the bodies one wants to study. We have also created a program called *SystemFinder.py* which extracts the initial conditions of any desired object in our Solar System through the use of NASA's official Horizon tool[3]. These conditions are then dumped to a text file on a format readable by `SolarSystem`. This given planetary system is then integrated over a desired time range through the use of the `Solver` class which we will touch on shortly. The final data is then written to a text file which can further be studied through the various plotting functions provided on our GitHub.

The `Solver` class implements the forward Euler and velocity Verlet integration methods described in section 3. It takes in two arguments, the first being the total integration time desired in units of years, and the second being the stepsize dt which directly affects the precision of the integration.

4. Results and Discussion

The object-oriented model will now be used to study different planetary systems. We will start by looking at the Sun-Earth system, then move on the the Sun-Earth-Jupiter system and finally finish with a model of the complete Solar System.

4.1. **The Sun-Earth System.** Here follows the results of the analysis considering the two-body problem of the Sun-Earth system. Most of the analysis in this section

[2]The `vec3` class can be found on the course webpage.

[3]More information about the HORIZONS tool can be found here.

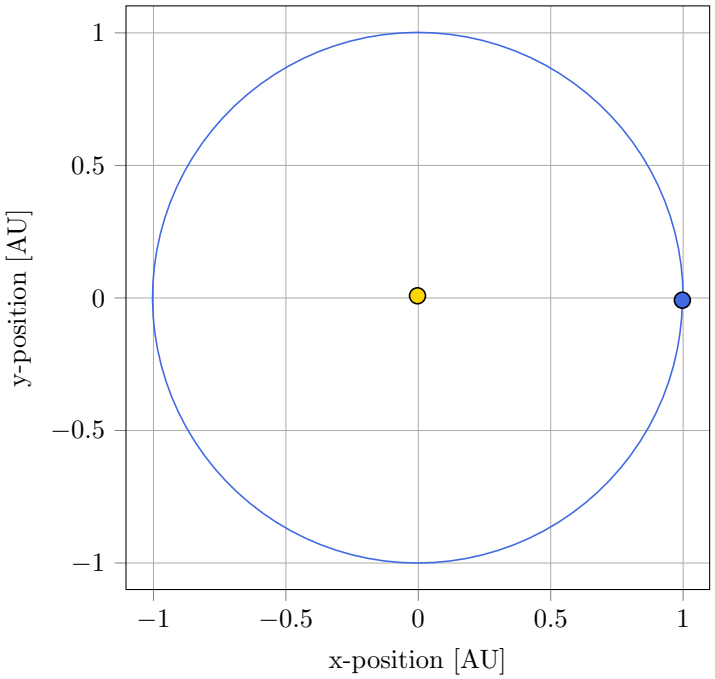


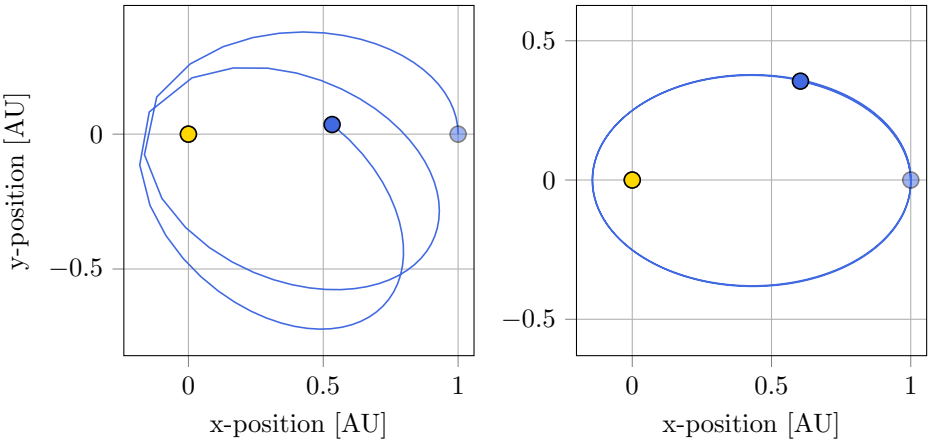
FIGURE 1. The Earth-Sun system with initial conditions $v = 2\pi$ AU/year and $r = 1$ AU simulated using velocity Verlet over a period of 1 year with $dt = 0.01$. The yellow dot represents the Sun and the blue represents the Earth.

is done with the Sun-Earth system for simplicity even though it would be valid for any planetary system.

4.1.1. *Velocity And Eccentricity Analysis.* When discussing the two-body Sun-Earth system in section 2, we found that the trajectory of Earth placed at a distance of 1 AU from the Sun would only result in a perfect circular orbit if its initial velocity satisfied equation 7. We have tested this statement by simulating a simple Sun-Earth system using the velocity Verlet method over a period of 1 year with a stepsize $dt = 0.01$. By setting the initial velocity to $v = 2\pi$ AU/year, we got the results seen in figure 1. The yellow dot represents the Sun while the blue dot is the Earth. As expected, this results in a perfect circular orbit.

The Earth simulated in figure 1 has an eccentricity of 0. Eccentricity is a measure of how much a planetary orbit deviates from being perfectly circular. In reality, Earth has an eccentricity of 0.0167 [1] which closely represents a perfect circle. The true velocity of Earth is therefore not exactly $v = 2\pi$ AU/year, but very close. Using NASA's HORIZONS tool, we find that the Earth has an initial velocity of $v = 6.3112$ AU/year.

We can further study the eccentricity and stability of Earths orbit by altering the initial velocity to $v = \pi$ AU/yr. This is then simulated for the two scenarios seen in figure 2, one with a stepszise of $dt = 0.01$ seen in subfigure (A), and one with $dt = 0.001$ seen in (B), both over a period of 1 year. We see that the velocity is too low to keep Earth in a circular orbit, and the planet accelerates towards the Sun resulting in elliptical orbits. The simulation in (A) has a low precision, and results in an unstable orbit where the perihelion of Earth changes with every orbit.



(A) Stepsize $dt = 0.01$ (B) Stepsize $dt = 0.001$

FIGURE 2. The Earth-Sun system with initial conditions $v = 2\pi$ AU/year and $r = 1$ AU. The system is simulated using velocity Verlet over a period of 1 year, with a different stepsize for each subfigure.

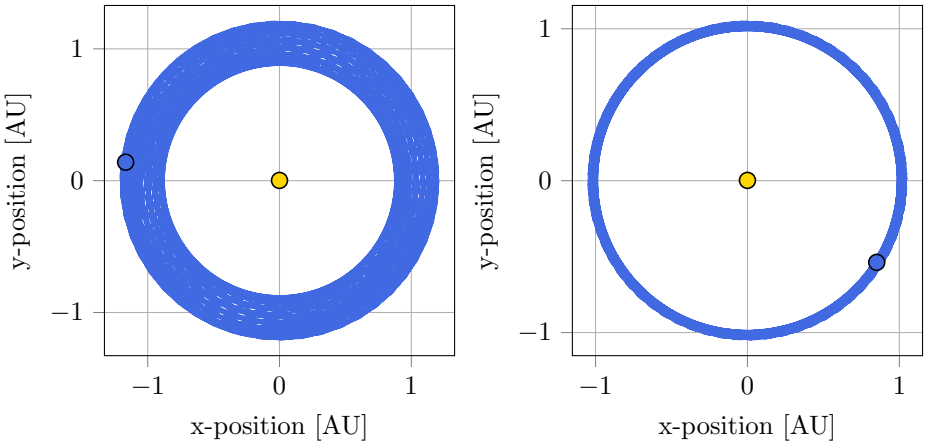
We also see that the orbital period is affected, as Earth completes more that one orbit per year.

The simulation in (B) with better precision results in a more stable elliptical orbit. This suggests that the stepsize should be picked with care, as the outcome of the simulation can be drastically different depending on your choice. As mentioned in section 3, the error in the velocity Verlet method is proportional to the stepsize cubed, meaning that the simulation in (B) represents a more realistic scenario than the simulation in (A). The orbital period is still altered however as we see the earth continuing past its initial position (denoted as a faint blue marker).

4.1.2. *Comparing the forward Euler and velocity Verlet methods.* The Sun-Earth system can act as a good proving ground for the implemented integration methods. We will now apply both integration methods to the same Sun-Earth system which we simulated in figure 1. We test both methods by integrating over 100 years, with a stepsize of $dt = 0.05$. This stepsize is large enough to visually display the resulting difference in the two methods. The results of the comparison can be seen in figure 3.

It is clear that the velocity Verlet simulation in (B) is more precise compared to the forward Euler simulation in (A) as Earth's orbit is stable. The orbit in (A) is unstable as the radius of the orbits grows with increasing periods. These results come as no surprise and coincide well with the presented material in section 3.

Another important aspect of the comparison is the runtimes of each method, which we now will put to test. The average runtimes for both algorithms are shown in table 1. These results are aquired using a ASUS (2018) computer with Ubuntu. The stepsize is again $dt = 0.05$ and we have tested for a varying set of years. The results seen in the table are very similar. The velocity Verlet method has on average a few percentages longer runtimes compared to the forward Euler method. As the forward Euler algorithm has less than half the number of FLOPS as the velocity Verlet algorithm, one would expect a greater runtime difference between the two methods. However, from previous projects we know that some floating point operations require more computing time than others.



(A) Forward Euler (B) Velocity Verlet

FIGURE 3. The Earth-Sun system with initial conditions $v = 2\pi$ AU/yr and $r = 1$ AU, calculated using forward Euler in (A) and velocity Verlet in (B) with stepsize $dt = 0.05$ yr over 100 years.

TABLE 1. Average runtimes for the forward Euler and velocity Verlet methods. The results are aquired through a ASUS (2018) computer with Ubuntu.

Final time (with timestep $dt = 0.05$)	Forward Euler	Velocity Verlet
10 years	0.01272 s	0.03273 s
10^2 years	0.05399 s	0.05824 s
10^3 years	0.30503 s	0.318808 s
10^4 years	3.2482 s	3.3395 s
10^5 years	47.423 s	48.259 s

4.1.3. *Conserved Quantities.* The first conserved quantity of interest is the total energy in the system. As mentioned in section 2.3, we know that energy conservation directly reflects the stability and quality of the integration methods we are using. Computing equations (9) and (10) for the Sun-Earth system, allows us to study time evolution of the energy in the system. These energy calculations are done separately for the forward Euler and velocity Verlet methods over 10 years with a stepsize of $dt = 0.01$. The results can be seen in figures 4 and 5

The uppermost plot in figure 4 shows the kinetic, potential and total energy in the system in units of $M_\odot \text{AU}^2$ year^{-2}. The lowermost plot shows the absolute relative energy $|E_{\text{tot}} - E_0|$. We note that the kinetic and potential energy oscillates with periods shifted so that that they almost cancel each other out. The total energy is of orders 10^{-4} in value, while the change in energy as the system evolves is of order 10^{-7}. Even though the numbers seem low they are non negligible because of their units.

The results in figure 5 show that the total energy seen in the upper most plot matches those of the forward Euler method. However, this time around there are no oscillations present in the kinetic and potential energy. 4. As a result, The total relative energy is of orders 10^{-10}, which means that the energy is approximately constant. This suggests that the velocity Verlet method does conserve the energy to

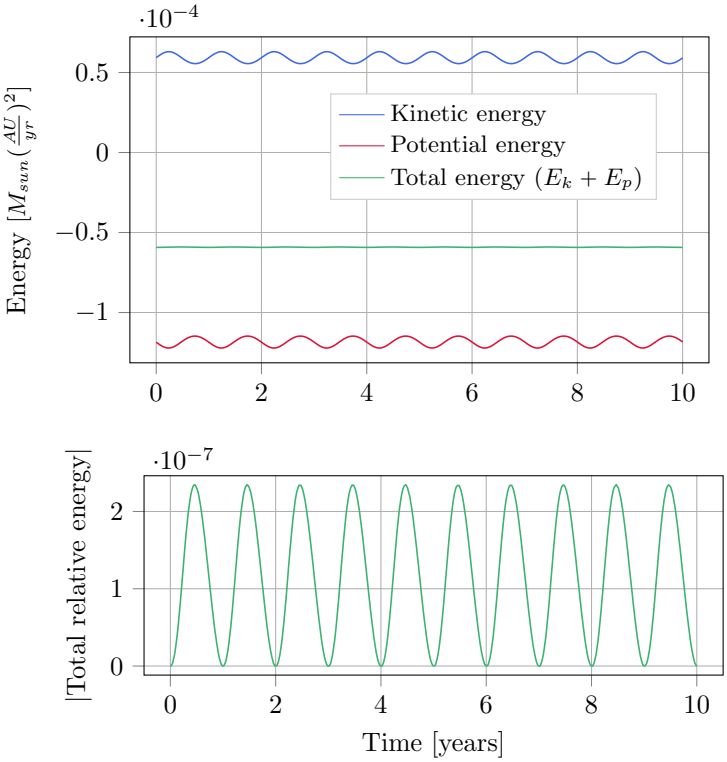


FIGURE 4. Time evolution of the energy in the Sun-Earth system over 10 years with $dt = 0.01$ using forward Euler. The uppermost plot shows the kinetic energy, potential energy and total energy in the system, while the lowermost plot shows the absolute total relative energy.

at least some extent. It should be noted that these results come from a simulation with a relatively high stepsize meaning that the specific energy values results found above would likely somewhat improve for lower stepsizes.

It is also interesting to see if the angular momentum in the Sun-Earth system is conserved. By computing the total angular momentum per mass in the system over a period of 10 years with a stepsize of $dt = 0.01$, we produce the results seen in figure 6 which consist of two subfigures. Both subfigure shows the total angular momentum per mass, but we have manually fixed the position of the Sun to the origin during the integration in subfigure (B). We see that the angular momentum oscillates about $l = 2\pi$ with increasing oscillations in (A), while it remains a constant $l = 2\pi$ in (B). The constant 2π is the desired result as discussed in section 2.4. The oscillations in (A) is perhaps a result of the motion of the Sun. During integration, the Sun drifts ever so slightly away. This becomes more obvious during long integration. Instead of hard fixing the Sun's location one could try to subtract the linear momentum of the centre of mass (the Sun) in order to produce a more constant angular momentum.

4.2. **Escape velocity.** The escape velocity of the Earth in the considered system was derived in section 2.5 to be $\sqrt{8}\pi$. It is however interesting to see how different initial velocities affects the trajectory of the Earth. By integrating for a long time

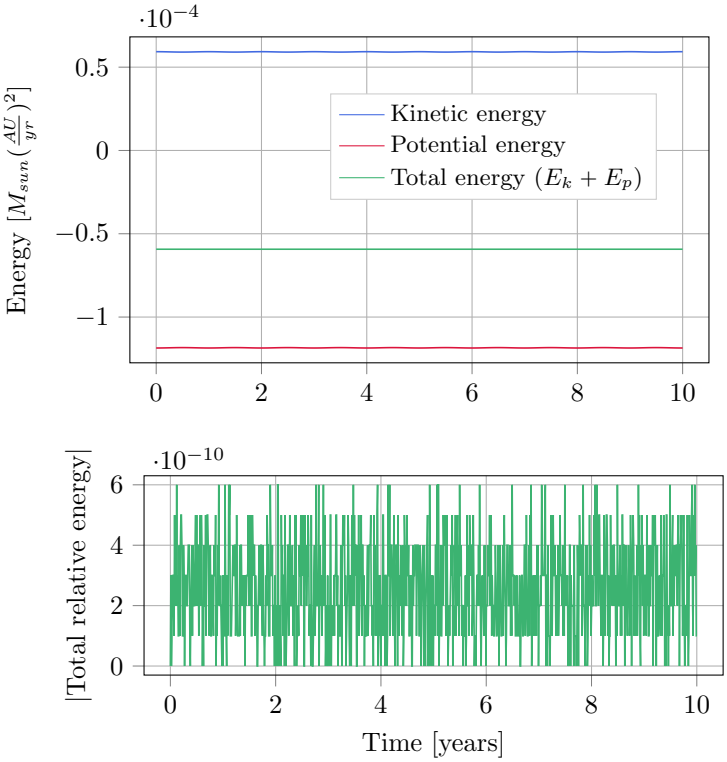
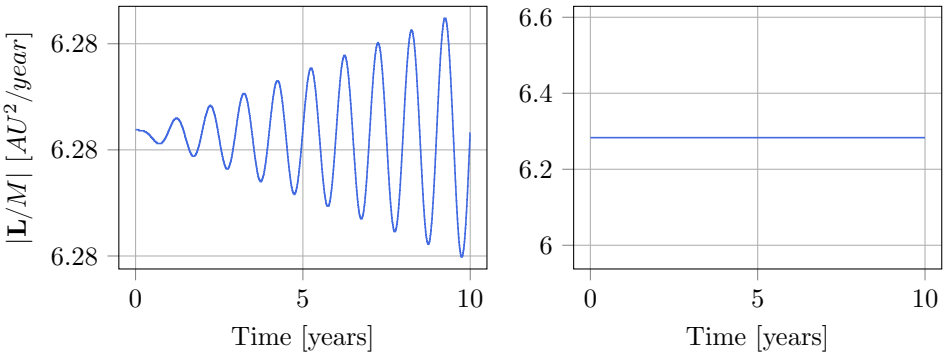


FIGURE 5. Time evolution of the energy in the Sun-Earth system over 10 years with $dt = 0.01$ using velocity Verlet. The uppermost plot shows the kinetic energy, potential energy and total energy in the system, while the lowermost plot shows the absolute total relative energy.



(A) Standard simulation.

(B) Sun fixed at the origin.

FIGURE 6. The total angular momentum per mass in the Earth-Sun system simulated using the velocity Verlet method over 10 years with a stepsize $dt = 0.01$. Subfigure (B) shows the case where the Sun is hard fixed at the origin during integration.

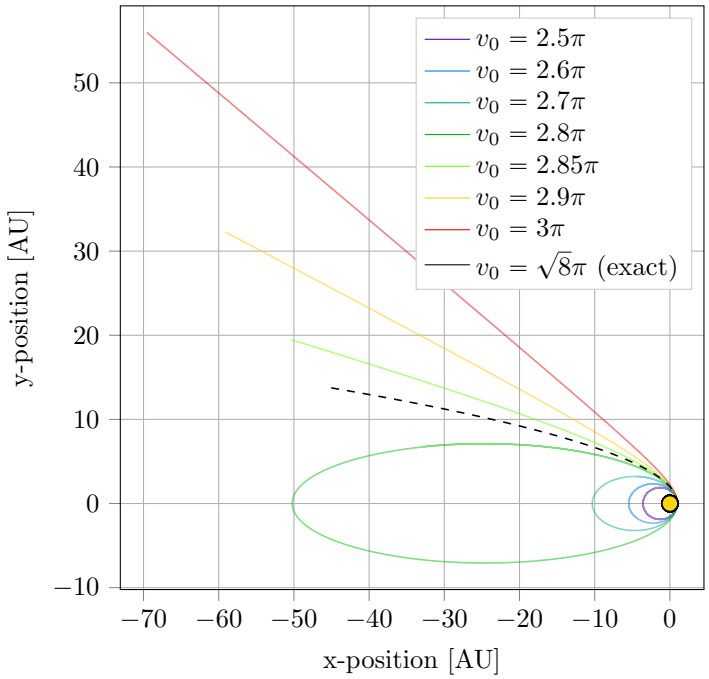


FIGURE 7. Earth's orbit/trajectory in a Sun-Earth system for different initial velocities. The analytic escape velocity found in equation 12 is denoted by the black dotted line.

period (total time is different depending on the intial velocity) with a stepsize of $dt = 0.01$, we find the following results seen in figure 7.

We see that for $v_0 \leqslant 2.8\pi$ AU/year, the orbits remain closed but at different eccentricities. For higher velocities the planets seem to escape at increasingly small angles. When studying escape velocity, one should have in mind that planets can end up in orbits with very large major axis without escaping. It all depends on the system. It is hard to know whether or not a trajectory is an escape trajectory or a bound orbit with a large major axis. From the numerical calculations it seem that initial velocities equal to or larger than $v_0 = 2.85$ AU/year result in escapes as the trajectory seem very linear. Numerical calculations therefore suggest that the escape velocity should be somewhere between 2.8π and 2.85π AU/year. From equation 12 we know that this is true and that the exact escape velocity would be $v_0 = \sqrt{8}\pi \approx 2.828\pi$ AU/year.

It would be nearly impossible to find an exact escape velocity by only using our program as we lack infinite precision and memory. However, one rarely need an exact escape velocity. In most cases we are not solely interested in escaping an object, but also reaching a goal which requires a higher initial velocity (e.g. within space exploration).

4.3. **Three-Body Problem (Sun-Earth-Jupiter).** We will now complicate our two-body system by including Jupiter as an object. Integrating for 30 years with a step size of $dt = 0.001$, results in the simulation seen in figure 8. The three-body system is stable as expected since it accurately represents the Solar System. Similar to the two-body problem, we have chosen to plot the figures in two-dimensions as all the orbits lie in the same plane.

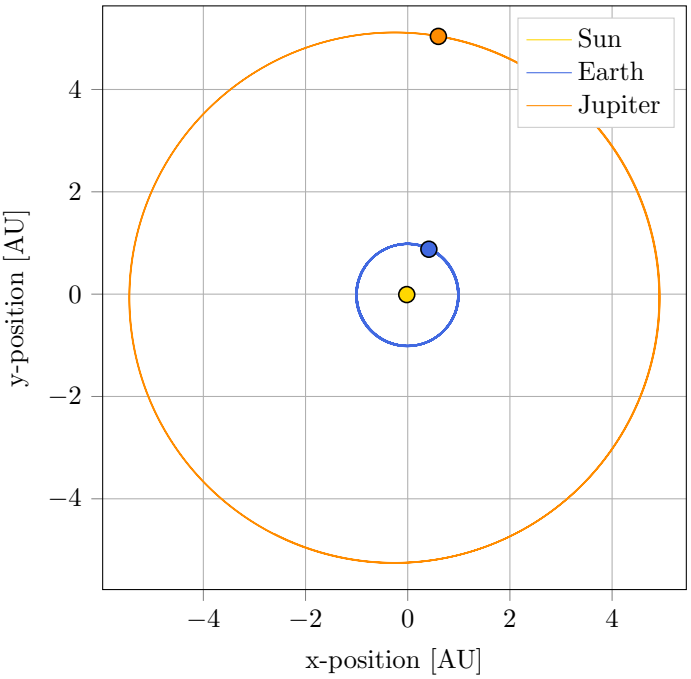
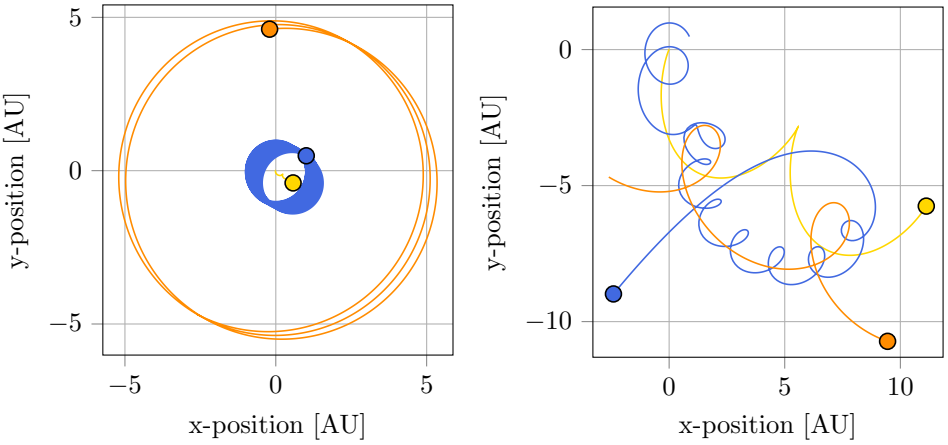


FIGURE 8. Three-body system of the Solar System model including the Sun, Earth and Jupiter. The orbits of the planets have been calculated for 30 years using the velocity Verlet method with a stepsize of $dt = 0.001$.

Even though Jupiter is the most massive planetary body in the Solar System, we see that it poses no noticeable effects on the trajectory of Earth which remains circular. An interesting study is to increase the mass of Jupiter and look at the consequences this has for the three-body System. We simulate two scenarios, one where we multiply Jupiters mass by 10, and another where we multiply it by 1000. The results is seen in figure 9. In subfigure (A) we see the influence of a Jupiter with a mass 10 times that of its initial mass. All orbits are now altered and unstable. Even the massive sun is affected. Subfigure (B) shows the extreme case of a Jupiter with 1000 times its initial mass. This results in chaotic trajectories for all three objects. This system could represent a binary star system as Jupiter now has a similar mass to the sun. Stable orbits in binary star systems are harder to come by, especially with a system resembling ours where the two stars are very close.

4.4. **The Complete Solar System.** The program is now ready to be tested for the entire Solar System. We include all eight planets in addition to Pluto, and simulate the system for 250 years which is approximately the orbital period of Pluto. The result of this simulation is seen in figure 11. We have plotted the system in 3D to demonstrate that not all objects in the Solar System orbit in the same xy-plane. All orbits remain stable and accurately represent the real Solar System.

As mentioned earlier, working with a object orientated code written on a general form allow us to study whatever system of planetary bodies we want. To demonstrate this, we will look at the Jovian system consisting of Jupiter and its Galilean moons (Jupiter's four largest moons). This system is then simulated over roughly 9 days with a stepsize of $dt = 0.0001$. The results of this simulation is seen in



(A) Jupiter: 10× M. (B) Jupiter: 1000× M.

FIGURE 9. Influence of a more massive Jupiter. Subfigure (A) shows the evolution of the system with a Jupiter with a mass increased by a factor 10. Integration time: 30 years, $dt = 0.001$. Subfigure (B) shows the chaotic planetary system with a Jupiter with a mass increased by a factor 1000. Integration time: 10 years, $dt = 0.001$.

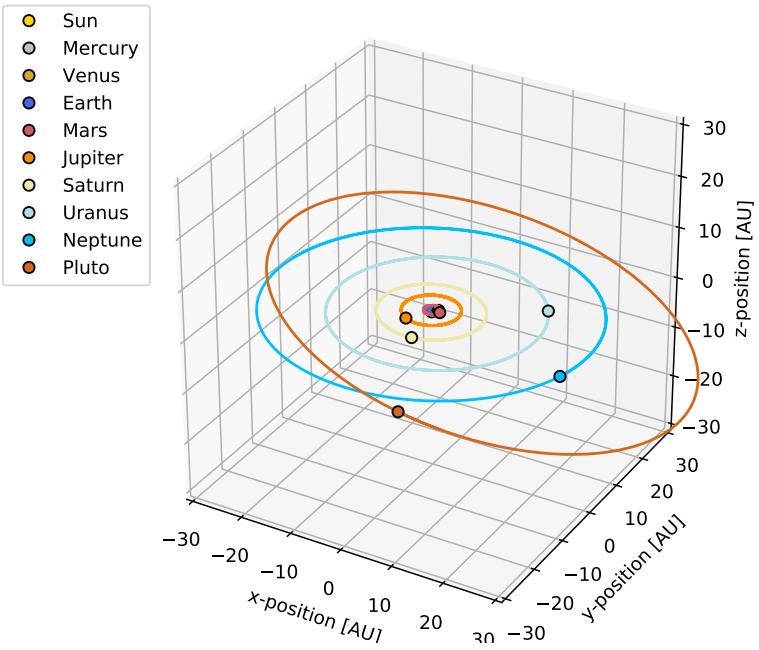


FIGURE 10. The Solar System. Simulation of all the planets in the Solar System including the dwarf planet Pluto. Integrated over 250 years which corresponds to Pluto's orbital period with a $dt = 0.001$.

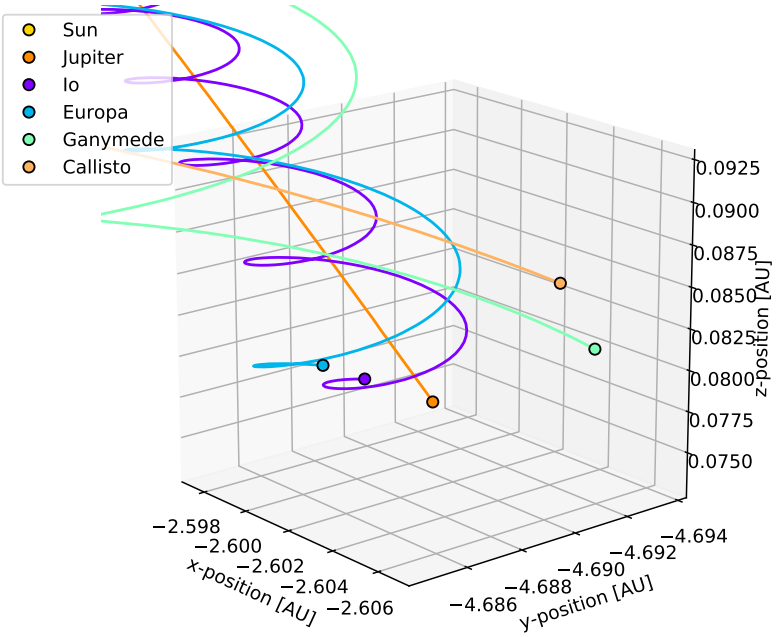


FIGURE 11. The Jovian system. Simulation of the Galilean moons orbiting Jupiter as Jupiter orbits the Sun. Total integration time: 0.025 years, $dt = 0.0001$.

figure 11. The simulation is done in the reference frame of the Sun with focus on Jupiter. The figure shows the trajectories of the natural satellites as they orbit Jupiter, while Jupiter orbits the Sun.

4.5. **Changing the law of Gravity.** In figure 12, we can see the motion of the four inner-most planets with the gravitational force substituted with

$$\mathbf{F}_G = G\frac{M_1 M_2}{r^\beta}, \tag{19}$$

where we have set $\beta = 3$. This modified force gives us two different behaviours depending on the initial conditions. We can see that the orbits of Mercury and Venus have become quite unstable and is spiralling towards the sun. We can observe the other type of behaviour by looking at Mars. Mars has a velocity larger than the escape velocity for this system. There would seem that only one type of orbit would be stable: a circular orbit with $r = 1$, as a different β would not affect the results since $1^\beta = 1$ is always true. So the near circular orbits at lower r with $\beta = 2$ would collapse inward with $\beta = 3$, and the near-circular orbits at higher r with $\beta = 2$ would escape further away with $\beta = 3$.

4.6. **The Perihelion precision of Mercury.** In figure 13, we can see the angular drift of the perihelion of Mercury as a function of time. What interests us here is the derivative, so we did a linear regression to find the slope of the graph to be 6.74 ± 0.04 arcsecond per year, where the uncertainty was found by finding the maximum difference between the data points and the prediction. This corresponds to a perihelion precession of 674 arcseconds per century. The actual perihelion precession is closer to 43 arcseconds per century. This points to an inaccuracy in our calculation of the relativity-corrected gravity, or other numerical errors in our program.

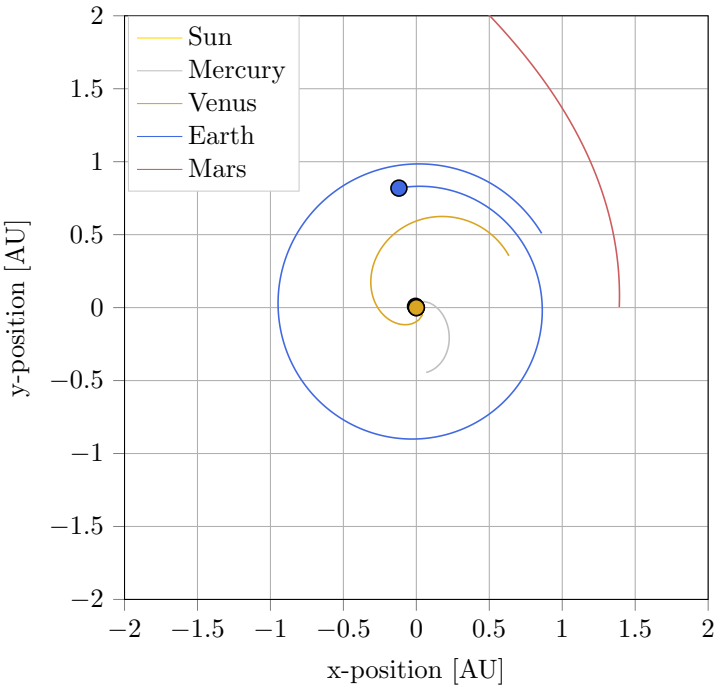


FIGURE 12. The four inner-most planets with gravitational force given by $\beta = 3$.

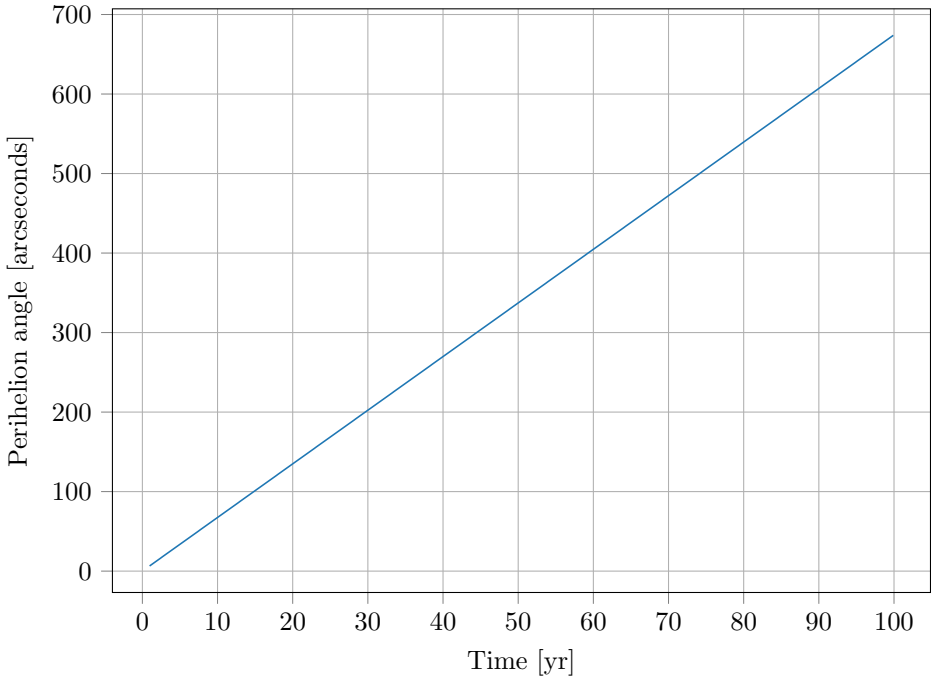


FIGURE 13. The changes in the Perihelion of Mercury over time with a relativistic correction to the gravitational equation. The calculation was done with dt = 10^{-8} yr to 100 yr.

5. Conclusion

We tested the functionality of both forward Euler and velocity Verlet, and found the Verlet method to be superior. Even though Euler requires fewer FLOPS than the Verlet method, we found that the difference in computational time for both method were negligible. I.e. the better conserved energy and greater accuracy of the velocity Verlet method in problems related to gravitationally bound systems, is more than enough to encourage its use over forward Euler even with a small increase in computational run time.

The calculated escape velocity, which we found through computational trial and error to be within a small margin of 2.825 AU/yr, matched very well with the analytical result of $\sqrt{8}\pi$ AU/yr, and is a sign that at least parts of our simulations are accurate.

Our model of the solar system is accurate and stable for most uses, and it is feasible to use it to calculate maneuvers for space flight quite accurately after further testing and calibrations, but it is not perfect. The perihelion precession we found of 674 arcseconds per century is very different to the expected 43 arcseconds per century. This might be an indicator of a larger problem with the implementation of our code.

Further works

Future work with our code should attempt to correct the perihelion precession, and further test the model to real data to validate and improve its accuracy. The increasing, oscillating angular momentum should also be investigated further.

Another possible continuation of this project would be to extend the model to include multiple solar systems. This would pose several new challenges to overcome, e.g. precision problems with both very large and very small values and what simplifications are acceptable.

References

[1] Williams, Dr. David R. (2017) *Earth Fact Sheet*, NASA, URL: `https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html` (2018.10.20)

[2] Hjorth-Jensen, M. (2018) *Overview of couse material: Computational Physics*, University of Oslo, URL: `https://compphysics.github.io/ComputationalPhysics/doc/web/course` (2018.10.20)