fys3150 Project 3

The Solar System

Gaute Holen — Github Repo

October 2020

Abstract

To explore many bodied systems in a gravitational field, a program is developed to simulate aspects of the solar system. Both Euler and Verlet velocity method is used to calculate the position and velocity of the bodies, with the option to account for general relativity and a gravitational force proportional to $\frac{1}{r^\beta}$, $\beta \in [2, 3]$. The perihelion precession of mercury's elliptical orbit, escape velocity of the earth, the effect of a very massive Jupiter on the earth's orbit and the difference between a static and dynamic sun is explored.

1 Introduction

Many problems in physics consists of many bodies in a system, all interacting with each other in different ways. With such many bodied problems, it can be very tedious to calculate things by hand, so object oriented simulations of the system is implemented. A simple example of such a system, is our solar system with the sun and planets. Here, the planets only interact with each other through gravity, assuming no collisions, and there's a relatively small number of bodies in the system.

To explore the planets of the solar system and their interactions through the force of gravity, an object oriented simulation is built using the Verlet velocity method. The program can add static and dynamic bodies to the system, and bodies can also be "graviationally invisible", ie. they won't create a gravitational pull on other bodies. Additional options are available when running the simulation, such as whether the gravitational force should be perfectly $\frac{1}{r^2}$, or if general relativity is taken into account.

Every run of the simulation creates a text file with all the position vectors of all the planets, in addition to a system properties file with the initial conditions of that run, all put in a unique folder. There's also developed a plotting tool in python, that by default makes 4 different plots: positions on the xy-plane, positions in xyz, positions of the xy-plane with time on the z-axis, and the orbital radius r with respect to time. Running the simulaiton many times will create lots of position vector files, and lots of plots. However, the only insight into whether the simulation is doing what it is supposed to is through these plots, and one often does not know which plots are more helpful until all of them have been looked at.

A selection of useful plots that demonstrates certain aspects have been picked out for the report, however there were lots of unused plots created.

2 Theoretical Background

Here some the derivation of some of the methods used in the simulation that have their basis in theory of how an orbiting system behaves are described. The implementation may be slightly different, however the equations found here laid the foundation for the implementation.

2.1 Units

To make the program a lot simpler, the standard SI-units are not used. The following units are used instead:

1. Mass is measured in solar masses $M_\odot$
2. Distance is measured in astronomical units AU
3. Time is measured in years
4. Velocity is measured in AU/years

As a result the gravitational constant and the speed of light are different in these units.

2.2 Finding the orbital speed of the earth

With our units, with the assumption that the earth has a circular orbit, we can find it's speed by computing the circumference its orbit

$$s = 2\pi r$$

Where $r = 1AU$, and we know that it needs 1 year to complete this, such that $v = 2\pi AU/year$

2.3 Orbit around center of mass

The center of mass of many bodied system with mass m_i and position r_i, the center of mas is given as R

$$R = \frac{1}{M}\sum m_i r_i \tag{1}$$

Where $R = 0$ because we want the center of mass to be at the origin

$$0 = \frac{1}{M}\sum m_i r_i + \frac{1}{M} r_\odot m_\odot$$

$$-\frac{1}{m_\odot}\sum m_i r_i = r_\odot$$

Where $m_\odot = 1$ so that

$$\sum m_i r_i = r_\odot$$

The next step is to find the linear initial velocity, simply found by taking the total linear momentum of the system without the sun equal to the linear momentum of the sun

$$\sum p = p_\odot$$

Where the $p = mv$ and the mass of the sun is 1 so that

$$v_\odot = \sum p$$

2.4 Finding angular momentum

The angular momentum of a body is given as

$$L = r \times p$$

Where p is the linear momentum $p = mv$. For a whole system, the L vetors of all the bodies are simply added.

2.5 Conserving energy

The kinetic energy in the system must remain constant for orbits to remain stable. The same thing goes for the potential energy, or gravity in this case. Testing the angular momentum at the beginning of the simulation and at the end of the simulation, ensures that both the kinetic and potential energy is conserved, as changing either of them would result in differnt angular momentum at the beginning and end.

3 Algorithms

The algorithms described are used to update the position and velocity of celestial bodies in a solar system by funding the total gravitational pull. The program will also plot make files with positions for each body each time step.

Note that equality signs are used somewhat liberally here, similar to how it is used to assign values to variables in programming, even though it is not technically equal. This is because we are using it to describe algorithms in this section.

3.1 Eulers method

The first, and simplest algorithm implemented for computing the positions of the celestial bodies, is Euler's method. It works by updating each position according to the current velocity, such that

$$x_{i+1} = x_i + v_i h \tag{2}$$

Where h is the time step defined as

$$h = \frac{t_n - t_0}{n} \tag{3}$$

Where t_0 is the starting time, t_n is the final time and n is the number of steps. The velocity is then updated

$$v_{i+1} = \frac{F_i}{m} \tag{4}$$

Where F_i is the force at the current time step and m is the celestial body's mass. Note that the velocity and position is updated using the previous step's values, not the current ones.

3.2 Verlet velocity method

The velocity verlet method is an improved version of the euler method. Essentially, it uses updated values to find the updated values, insted of using previous step values to find updated values.

The position is updated with a two-term taylor expansion of x where

$$x_{i+1} = x_i + hv_i + \frac{1}{2}h^2 a_i \tag{5}$$

And the velocity is updated with

$$v_{i+1} = v_i + \frac{1}{2}h(a_i + a_{i+1}) \tag{6}$$

Where

$$a_i = \frac{F_i}{m} \tag{7}$$

Note that since both a_i and a_{i+1} is used, F_i and F_{i+1} has to be found for one update. However, the next update $i + 2$ can use F_{i+1} if it is stored in memory to only have to compute F_{i+2}. While this method does require more flops, it is also more accurate.

3.3 Finding the force of gravity

Both algorithms use a force F_i to calculate the change in velocity and position. In general, the total force F acting upon a celestial body is the sum all the forces acting upon it

$$F_{total} = \sum F \tag{8}$$

In our case, the only force acting is gravity, however many things can contribute to the total gravitational pull of each body. If each body is treated as a point mass, then the force from that body contributes to the gravitational pull on another body. The contribution of a body b on a body a is given by

$$F_{b\to a} = -G\frac{m_a m_b(x_a - x_b)}{r^3} \tag{9}$$

Where m is the mass, r is the distance between the bodies and x is the position vectors of the bodies. If multiple bodies make contributions, then they are summed up as in *Equation* 8. The program has a boolean parameter in each celestial body where it can either be affected by gravitational forces from other bodies, or not. This is useful for simple simulations where you don't really care if the earth moves the sun a little bit or not, so the sun can be set to be unaffected by gravity and static.

3.3.1 Modification to r^2 law

As mentioned already, there are many things that make contributions to the gravitational pull, such as bodies not being point masses and tidal effects. To account for this, the value β is introduced, such that

$$F_{b\to a} = -G\frac{m_a m_b(x_a - x_b)}{r^{1+\beta}} \tag{10}$$

Where $\beta \in [2, 3]$.

3.3.2 Accounting for general relativity

Another thing that affects the gravitational force is general relativity. A correction term is introduced where

$$F_{b\to a} = -G\frac{m_a m_b(x_a - x_b)}{r^3}\left[1 + \frac{3l^2}{r^2c^2}\right] \tag{11}$$

Where $l = r \times v$.

3.4 The Body Class

The Body class is the abstraction of celestial bodies. It has some notable parameters that are useful for setting up the different scenarios, that is worth mentioning:

1. A boolean paramater determining whether this celestial body should be affected by the gravitational pull of other bodies. This is useful for creating a "static" sun fixed in the middle of the system that won't feel any gravitational pull from other bodies.

2. A boolean parameter determining whether this celestial body should contribute to the gravitational pull felt by other bodies. This is useful for instance in the scenario for finding escape velocity, where many bodies are initiated at the same position with different initial velocities, where we don't want those bodies to affect each other, only to feel the gravitational pull of the sun.

4 Results

Running the algorithms and plotting their results gives insight into whether they succesfully have modelled the solar system as intended. Here, we explore different algorithms and different scenarios to learn more about the solar system and our program.

The methods for obtaining the plots here are either described in the assignment text [3], described directly here or described in the section for Theoretical Background or Algorithms. In an attempt to make the report concise, only the results themselves are discussed here. Please see the README file in the github repo for more details.

The initial velocities and positions are either found with work shown here, gotten from the assignment[3], or from NASA's webtool for orbital data in the solar system[2].

4.1 Tests

On every compile, the program runs through a default scenario with the earth-sun system, and makes sure that angular momentum is conserved.

4.2 Runtimes and mesh points

The program runs rather efficiently, at least compared to the other projects, and can do up to 10^6 time steps in around 2 seconds for two bodied systems. However, any higher than that, and my computer starts to run out of memory when trying to analyse the files. The runtime scales linearly to the mesh-points.

The major limitation to the algorithm is actually the plotting tools, as the lists containing the positional values for plotting in python will max out the memory on the computer at around 10^6 in my specific case. This could probably have been worked around, however it was not the focus here.

4.3 Euler vs Verlet Velocity

Both the Euler and the verlet velocity methods can be used to simulate such systems. The strength of the Euler method is that it requires fewer FLOPS, however Verlet is more accurate and stable.

From *Equation* 2 and 4 we get 2 equations, and in 3 dimenstions that results in 6 equations for Euler's method. Counting FLOPS from the implementation, there are 23 with Euler's method per update.

With the Verlet Method, we use *Equation* 5 and 6, and counting FLOPS from the program there are 37.

Obviously, the implementation here is not optimized, and there are probably better ways to do this. However, there's only about twice as many FLOPS for Verlet compared to Euler, which might me surprising as it's so much more accurate.

In *Figure* 1, it is apparent that using the Euler method results in an unstable system. As such, the Verlet method is much better and will be used from here onwards.

4.4 Escape velocity

The escape velocity from a body is given as

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

Where $M = 1M_\odot$, $r = 1AU$ and $G = 4\pi^2 \frac{AU^3}{M_\odot years^2}$. So that

$$v_e = \sqrt{2 \cdot 4\pi^2} = \sqrt{8}\pi \approx 8.89 \frac{AU}{year}$$

The above property is derived from setting the kinetic energy equal to the potential energy, and solving for v. If the kinetic energy is greater than the potential energy, then the system will lose the planet. One could compute this to find the escape velocity numerically, but that doesn't seem to be the intention of this part, as it's more or less equivalent to just computing it with the formula for escape velocity.

Instead, a simulation where earth is moving radially away from the sun towards positive x with velocity v is done over a century. If

$$x_{i+1} < x_i$$

at any point, then the initial velocity is less than the escape velocity.

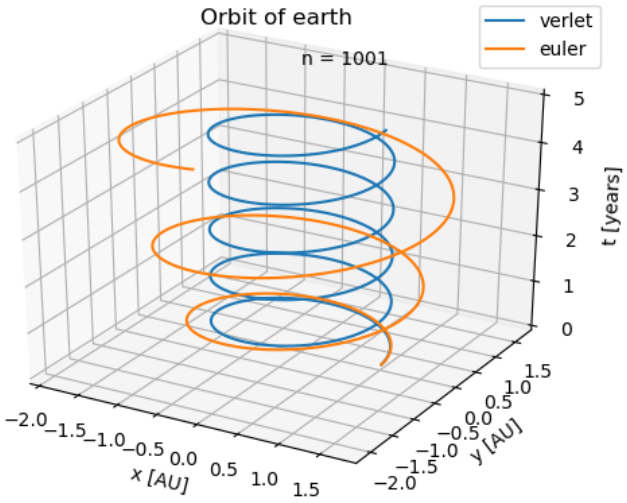
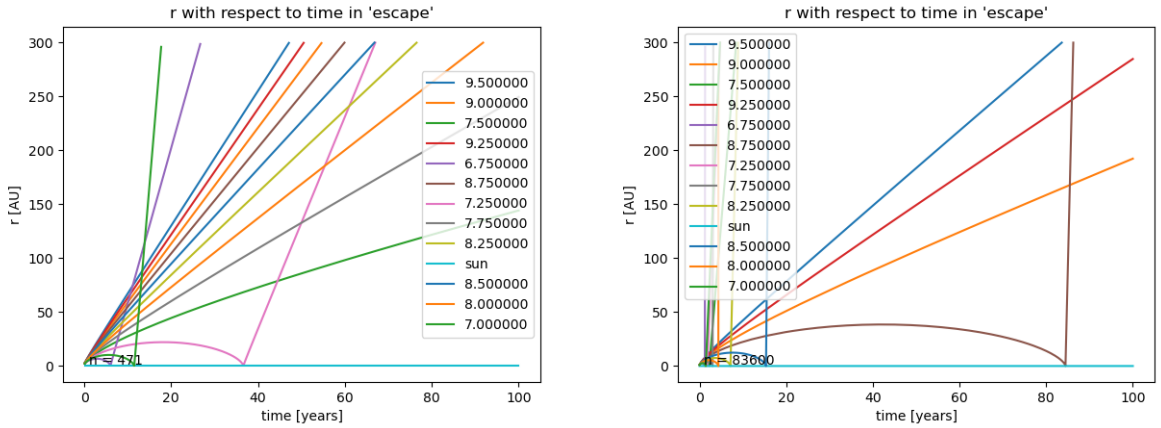


Figure 1: The orbit of the earth simulated over a period of 5 years with 1001 mesh points using the Verlet velocity and Euler algorithm



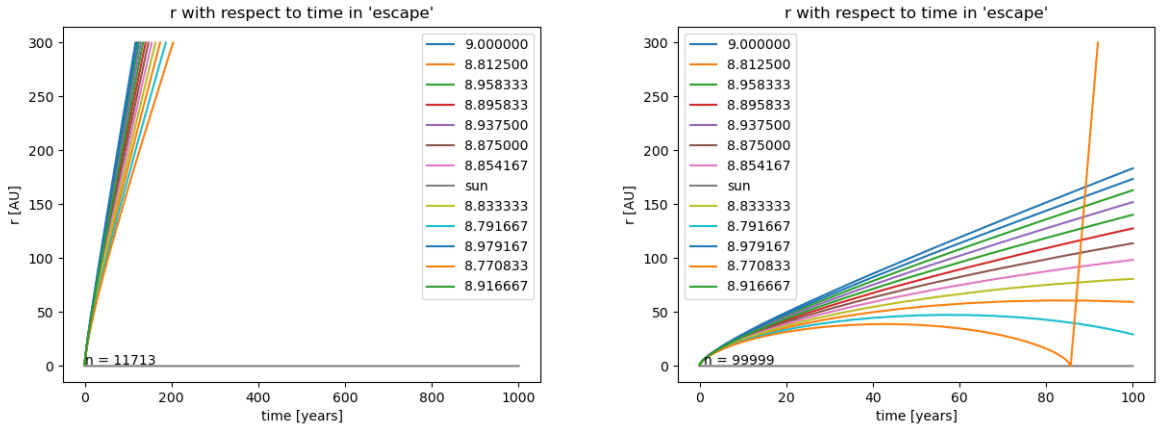
(a) (b)

Figure 2: The distance to the sun for objects traveling radially away from the sun with different initial velocities. In (a) there are 10^4 mesh point and in (b) there are 10^6 mesh points

The way it is done here though, is to simply plot the radius r with respect to time for a range of initial velocities as seen in *Figure* **??**.

In *Figure* 2, the objects with too small initial velocity will fall back towards the sun, and then get slingshot super far. This would potentially act differently if the simulation accounted for general relativity, as speeds potentially became relativistic, however it didn't in this run. Maybe it would have to account for special relativity as well. Regardless, in (a) in *Figure* 2, the escape velocity appears to be somewhere between 7.25 and 7.5 AU/year, something which is way too low. In (b) in *Figure* 2 however, it is somewhere between 8.75 and 9 AU/year, which is a correct range. The difference between the two is the number of mesh points in the simulation, reaching 10^6 in (b).

In order to obtain a very accurate answer, one would have to run the simulation towards infinite time, which is not possible. As such, the only additional run to try to find a better approximation done here is to run it from 8.75 to 9, over 100 years abd 1000 years as seen in *Figure* 3.



(a) (b)

Figure 3: The distance to the sun for objects traveling radially away from the sun with different initial velocities. There are 10^6 mesh points in both, but (a) goes from 0-1000 years while (b) goes from 0-100 years

In *Figure* 3, the resolution in (a) is too bad close to the sun to actually get a good gravitational pull on the bodies, so here the escape velocity is lower. Perhaps a linear time step is not ideal, where there are more time steps closer to the sun, and less further out. In (b) however, we get a closer approximation. Yet, we don't know which lines come back down to earth, and which diverge. Increasing the total time with the same n will, as seen in (a), change the escape velocity, so that's not going to help. An guess is that it's somewhere between 8.8 and 8.9 form (b), however, as already explaied, this is not really the best way to do such an estimate.

4.5 Different forms of the force of gravity

Although the force of gravity according to Newton is proportional to $\frac{1}{r^2}$, this is only valid for point masses. There are things such as tidal effects, that will effect it so that it's not a perfect $\frac{1}{r^2}$. To explore this, a variable β is introduced, so that gravity is proportional to $\frac{1}{r^\beta}$, $\beta \in [2, 3]$.

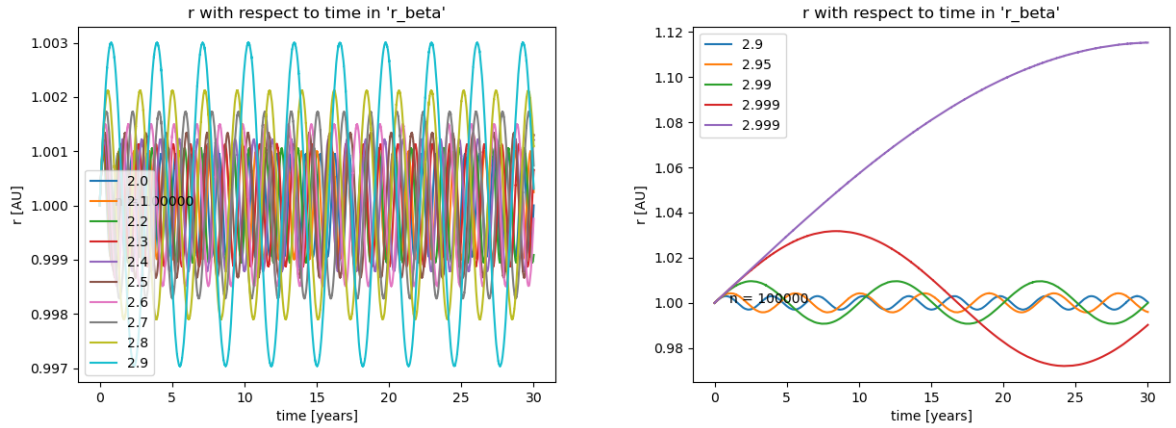
In *Figure* 4, we can observe how the orbital radius of the earth varies with β. It's a periodic cycle, meaning that the orbit is becoming more and more eliptical. Relating this to Kepler, who describes orbits as ellipses, this very much increases the accuracy of the simulation. The earth's orbit actually deviates from 1 AU with about +/- 0.02AU[1], so something closer $\beta = 3$ is more accurate to earth's actual orbit.

In *Figure* 4 (b) and 5, β approaches and is equal to 3. When β is getting close to 3, the period and the magnitude of the periodic behaviour becomes so great, that it almost seems like the orbit is unstable and the radius is just increreasing. When β is 3, the orbit is unstable and the radius is increasing steadily.

4.6 Perihelion precession of mercury

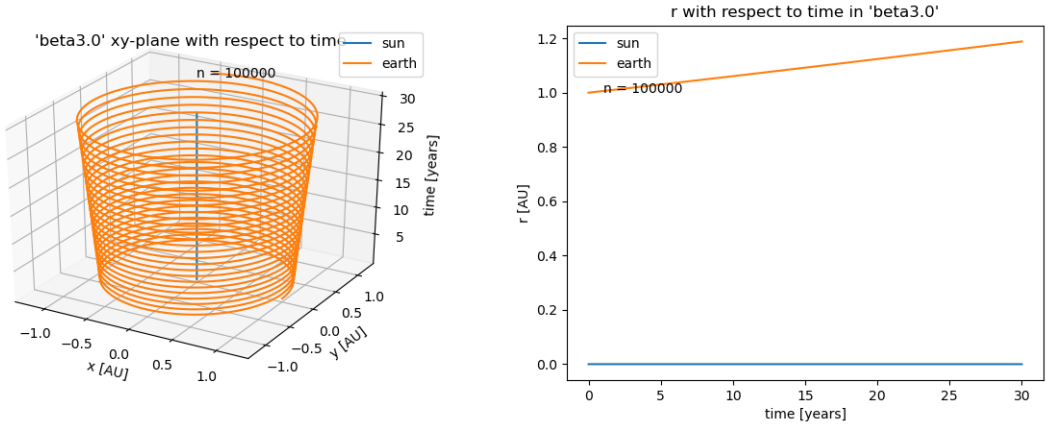
Another thing that affects gravity, and that Kepler couldn't explain, is the perihelion precession of Mercury. It's elliptic orbit appears to rotate more than it should, which is partially acredited to general relativity.

In *Figure* 6, (a) is done with relativity over a century, and (b) is done without relativity over a century. The rotation of the elliptic orbit is far greater than it should be, but the difference is $\Delta\theta \approx 0.021 DEG$ which is about 75", which is within the correct order of magnitude[3].



(a) (b)

Figure 4: The variation of the radius of the orbit of the earth for different values of β



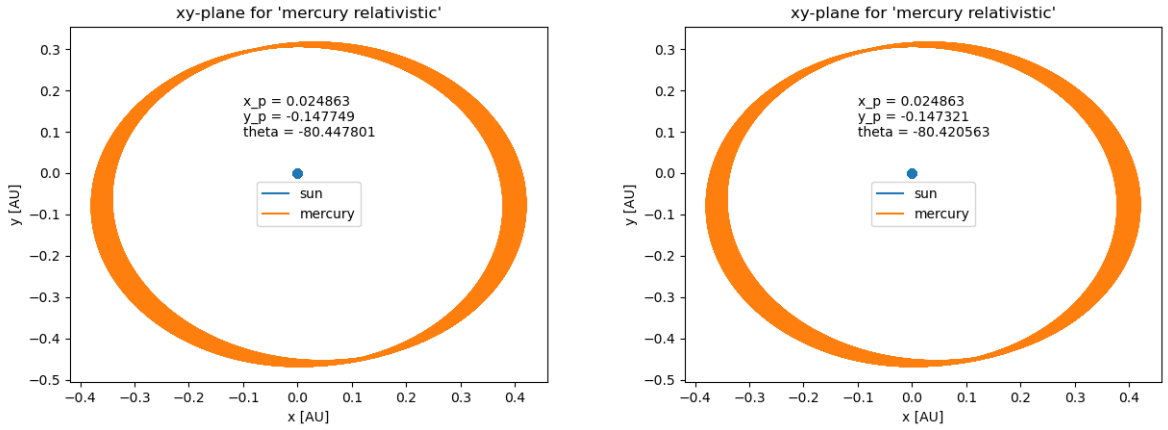
(a) (b)

Figure 5: The distance to the sun for objects traveling radially away from the sun with different initial velocities. In (a) there are 10^4 mesh point and in (b) there are 10^6 mesh points

4.7 3 body problem

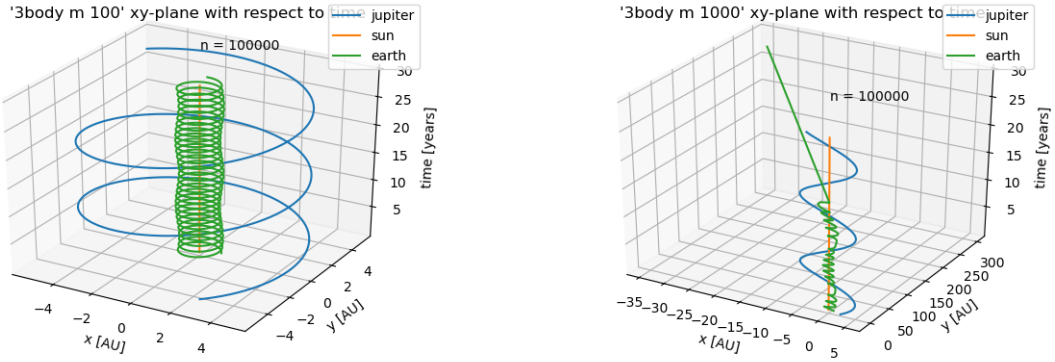
Until this point, the simulations have been with a static sun, and just one planet. But, in the real world, there are many planets, all affecting each other with their gravitational force. The most significant contribution to earth's orbit other than the sun, is Jupiter. To explore the three body earth-sun-jupiter system, a simulation over 3 years is done with a static sun, with Jupiter's mass being 10, 100 and 1000 times greater than it actually is. As you'd expect, when Jupiter's mass is unaltered, earth's orbit is not really affected much. Same for 10 times it's mass.

In *Figure* 7, Jupiter's mass is 100 and 1000 times greater. At 100 times the mass, the earth's orbit "wobbles" and follows Jupiter's orbit with great attraction. However, at 1000 times jupiter's mass, the earth's orbit is altered so much, that the earth get's ejected out of the solar system as seen in In *Figure* 7 (b).



(a) (b)

Figure 6: The orbit of mercury on the xy-plane around the sun over the course of 100 years using 10^6 mesh points. The left (a) figure has general relativity and the right (b) does not



(a) (b)

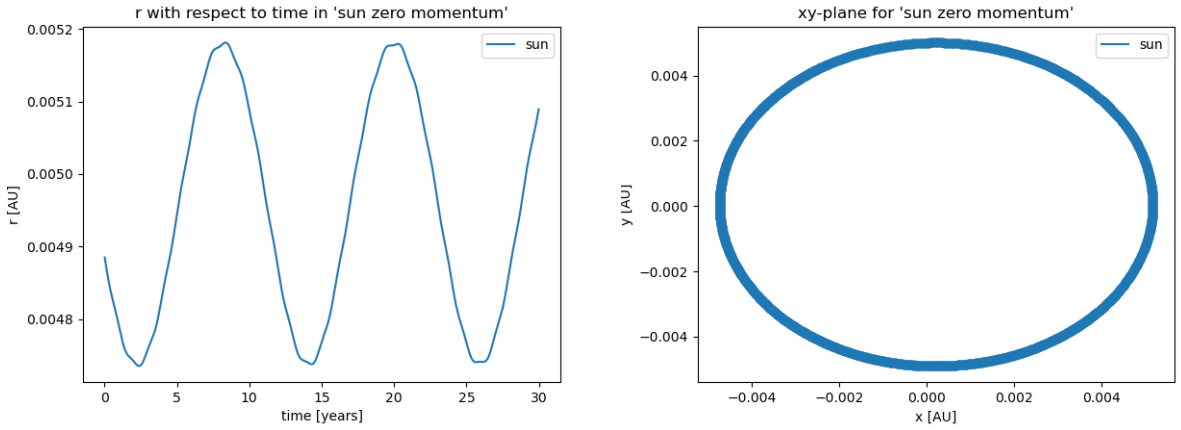
Figure 7: The sun fixed at the centre, with Jupiter at 100 (a) and 1000 (b) times it's actual mass orbiting the sun with the earth for a period of 30 years.

4.8 A model of a dynamic solar system

Finally, to accurately model the dynamic nature of our system, where every body in the system orbits a centre of mass in the middle of the solar system, the sun is added after the other planets with position and velocity as described in the Theoretical Background section about orbits around center of mass.

The sun is added so that the total momentum of the system is zero, causing the sun to orbit the center of mass at the origin. Only the earth and Jupiter is added to the system, and as seen in *Figure* 8, the sun's orbit now varies with periods equal to Jupiter's orbital period (the big variation) and the earth's orbital period (small variation).

With the data from NASA[2], one could add all the planets to this system, and build a replica of the solar system with all planets affecting each other dynamically. However, adding all of them would be rather tedious, and the plot with just 3 bodies and a dynamic sun demonstrates the same principle.



(a) (b)

Figure 8: The sun's orbit around the origin (a) and the variation of the radius of earth's orbit around the origin (b) in a zero angular momentum system with the earth, sun and jupiter

5 Discussion

With the observations this far, potential strengths and shortcomings of this investigations can be highlighted. This particular report was heavily focused on the visual plotting of positions in the systems created, and using the plots to gain insight into whether the simulation was sufficient or not. The reason for this was that it can quickly give feedback as to whether the simulation works as intended. The whole program and report was structured around this, as is seen in already. One downside of this focus, is that it has little focus on runtimes, comparing different runs other than visually, and perhaps a lack of focus on optimisation. That being said, the program did run rather smooth and quick, as already briefly mentioned in the algorithm section.

That being said, although the focus has been on visual feedback and the functionality of the simulation with all the different scenarios, some could have been explored further. For instance, the perihelion precession of mercury, and the general elliptic nature of the orbits, where observed to be there, but not really examined closely. The extent to which orbits are elliptical or not, and how that matches the real world data, was only briefly looked into.

Similarly, the fact that the program is trying to do so much at once, makes it difficult to really go into depth on each functionality, other than a macroscopic plot of the orbit and observing that it roughly acts as it should.

On the same note, investigating numerical errors, loss of precision, what mesh points and time intervals are ideal etc have not been done in depth. An observation without a thorough understanding of its uncertainty, is not worth much. Similarly, this is a great tool to look at how different aspects of gravity interact in the solar system. But, it should not be used to make any reliable predictions or calculations used in the real world. Yet, it is useful for getting a general understanding.

6 Conclusion

To conclude, this investigation into many bodied system yields a program that can simulate static and dynamic systems, with the option to change the force of gravity's proportionality to distance and to account for general relativity. The sun can either be set at the origin, or be automatically put so that the momentum of the system is zero with the sun orbiting the origin as the center of mass. With this functionality, Euler and Verlet's methods are implemented and compared, finding the Verlet method to be the best, despite having

more FLOPS. The escape velocity of earth is estimated through observations, and found to be approximately equal to its theoretical value. The perihelion precession of mercury's orbit is also explored, found to be within the order of magnitude of what it should be due to general relativity. Lastly, 3-Body systems with both a static and dynamic sun are explored, with varying masses of Jupiter, where the orbit of the earth around the sun and orbit of the sun around the origin is explored.

The investigation is heavily focused on visualization and plotting, and could be more centered around numerical accuracy, loss of precision, uncertainty, errors in general and deviations from observed behaviour from real data. Yet, it remains a powerful tool to model the solar system, or other many bodied systems, although it's results shouldn't be taken as extremely precise predictions, but rather an indication of how bodies behave.

References

[1] Table of earth's orbital data from wikipedia `"https://en.wikipedia.org/wiki/Earth%27s_orbit"`

[2] Data on positions and velocities of celestial bodies in the solar system `http://ssd.jpl.nasa.gov/horizons.cgi#top`

[3] The Assigment text of FYS3150 at UiO.