Computational Physics
Project 3

Tommy Myrvik
Halvor Nafstad

October 2016

Abstract: *We have gradually built a model of the full solar system by mainly using the Velocity Verlet integration method. Along the way we found that varying the mass of Jupiter results in disturbances of the earth's orbit, even leading it to being thrown out of the system in some cases. We have confirmed that the Newtonian gravitational model indeed cannot accurately calculate the path of Mercury and by using real data from NASA we have observed that the planets in the solar system moves in almost circular paths in the xy-plane.*

Contents

1 Introduction

In this project we are going to build a full model of the solar system using object oriented code in C++. To accomplish this, we are going to use two different algorithms that solve second order differential equations; The Forward Euler and the Velocity Verlet. We're going to look at how the gravitational pull of each body in the solar system influence each other, meaning we need to find the differential equations describing this type of motion for the integration methods above to solve. We're going to start easy with a two-body system only containing the Earth and the Sun, and from there expand our simulations to include also Jupiter, and at last the whole solar system. We're going to study the stability and precision of the integration methods, as well as testing the speed.

We'll see what happens to the three-body system (the Sun, the Earth and Jupiter) when gradually increasing Jupiter's mass, and pay especially close attention to what happens with the Earth in this case. Finally we're going to run a simulation where we implement a relativistic correction to the motion of the planets by checking if we can find the perihelion precession of Mercury predicted by the theory of general relativity.

2 Methods

The two methods used, forward Euler and velocity Verlet method will first be introduced in general, then we introduce the application of this to different *N-body systems.* Both methods is numerical solvers of second order DE's.

2.1 Forward Euler method

In general the forward Euler method is obtained using two first order taylor expansions $x'_i = (x_{i+1} - x_i)/h + O(h^2)$ and $v'_i = (v_{i+1} - v_i)/h + O(h^2)$, where $v_i = x'_i$ and $v'_i = a_i$. Giving $x'_i \approx \frac{x_{i+1}-x_i}{h}$ and $v'_i \approx \frac{v_{i+1}-v_i}{h}$, resulting in

$$x_{i+1} = x_i + hv_i \tag{1}$$

$$v_{i+1} = v_i + ha_i, \tag{2}$$

where a_i in this project has a given form, which we will see later.

The algorithm for the forward Euler method is quite straight forward; set up initial conditions, loop over time ranging from $t_{start} \rightarrow t_{end}$. For each iteration first calculate the acceleration using the values obtained in the previous loop, then the position using the previous velocity, and then the velocity using the calculated acceleration. This is one of the simpler integration methods we have, using only $4 \times dim$ (the dimensions we're working in) floating point operations (FLOPS) per iteration (given that the step length h doesn't change), and gives an error proportional to h^2 as stated above. This error has a tendency to become a little too significant when calculating the motion for more complex systems, giving a lower precision in the results than one maybe would want. For simpler systems however, it can give a good approximation to how the system behaves, saving both FLOPS and time compared to using other more complex methods. The systems scientists are looking at nowadays however are rarely very simplistic, and for this reason the Forward Euler method is not used a lot in the scientific community.

2.2 Velocity Verlet method

To obtain find the velocity Verlet method we continue in the line of the forward Euler method, but instead of first order expansions we go to the second order

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

Now we do a taylor expansion of the acceleration $a_{i+1} = a_i + ha_i' + O(h^2) \rightarrow ha_i' = a_{i+1} + a_i$ to insert to the second degree expansion of the velocity, which takes the same form as for the position, to find

$$v_{i+1} = v_i + \frac{h}{2}(a_{i+1} + a_i). \tag{4}$$

The algorithm for the velocity Verlet method is much the same as for the forward Euler method, but note that a can be a function of x, thus a_{i+1} is a function of x_{i+1}, so: We start with setting up the initial conditions, then we loop over time ranging from $t_{start} \rightarrow t_{end}$. For each iteration we calculate a_i (this can be saved from the last iteration as we need to calculate a_{i+1} for each iteration anyways), then x_{i+1}, a_{i+1} and finally v_{i+1}, which concludes the general velocity Verlet method. This method has $9 \times dim$ FLOPS per iteration, but given that the step length h doesn't change, the number of FLOPS can be reduced to $7 \times dim$, but still giving an error proportional to h^3. This trade off between a higher precision for a few extra FLOPS makes this method a pretty efficient way to calculate complex systems, making it a widely used method.

2.3 Why Object Orientation?

To implement these methods into a well-flowing code, we have coded object oriented, using two C++ classes; Planet and Solver. The Planet class creates a planet object containing all relevant information about each planet, like mass, it's kinetic energy, and so on. This object can then be used to create another object through the class Solver, which then gathers all planet objects given. This in turn can be used to calculate properties of the entire system, e.g. the planets motions (using the ForwardEuler/VelocityVerlet methods), total kinetic energy of the system etc etc. So when dealing with say the whole solar system (which we will do later), it's much easier to have all properties of each planet contained in a single object instead of having a whole lot of different variables floating around in the code. This also goes for functions within the classes, where the objects is also tied to every function within it's class, making it easy to call the functions to act on the (objects) we are interested in. This makes the code very structured, where the manual defining of variables and data are minimized, and all functions in a certain class kind of "belong together" by being able to act on the same objects.

In this project we have used Morten Hjorth-Jenssens example on an object oriented code for solving these types of problems (that was given prior to this project) as framework. This example can be found at this GitHub repository: https://github.com/mortele/solar-system-fys3150

2.4 Two-body model

Our first task is to simulate a two-body system, where the lucky participants are the Sun and the Earth. In this system, the gravitational force, in Newtonian mechanics, between the two bodies is given by

$$F_G = \frac{GM_\odot M_{earth}}{r^2} = \frac{M_{earth}v^2}{r} \tag{5}$$

, where v is the velocity of the Earth, and we have used that $F_G = M_{earth}a = M_\odot M_{earth}\frac{v^2}{r}$ for circular motion. From this expression we also see that

$$v^2 r = GM_\odot = \frac{4\pi^2 AU^3}{yr^2}, \tag{6}$$

with $v = \frac{2\pi AU}{yr}$ where we use astronomical units AU. From eq. (5) we see that the acceleration of the earth (in a given direction) is given by

$$a_x^{earth} = \frac{F_{G,x}}{M_earth} \tag{7}$$

The same is valid in the other directions giving us three coupled equations for the acceleration.

For a system like this we have several physical properties that must be fulfilled. Some of these are that the total kinetic energy, potential energy and angular momentum of the system is conserved. These should be conserved as we're considering our system to be closed, with no external forces acting on it. This actually means that it is the total mechanical energy (the sum of the total kinetic and total potential energies) that should be conserved, but since we're looking at a stable system of two bodies which we assume move in perfect circles, the kinetic and potential energies will be conserved separately as well. Similarly with the conservation of angular momentum, where we have no external torques acting on the system, which in turn means that the total angular momentum of the system must be conserved. Angular momentum can be exchanged between the bodies in the system, but then the increase of ang.mom. for one body will be equal to the decrease of the ang.mom. of the other.

The kinetic energy of two bodies will be conserved if the kinetic energy of each body always sums up to the exact same number. Bodies that move in perfect circular motions will always have the same speed at any point in their paths, since the radius from the points they're rotating about will be constant. In our Earth-Sun system we have that the Sun's position is fixed, hence the Earth is the only body with kinetic energy. It will therefore be enough to check that the kinetic energy for the Earth is preserved.

$$K_{earth} = \frac{1}{2} m_{earth} v_{earth}^2 \tag{8}$$

The potential energy of the system is simply found from:

$$U_{earth,sun} = -\frac{GM_{\odot} m_{earth}}{|r_{sun} - r_{earth}|}, \tag{9}$$

hence it's the only quantity to check that is conserved.

Since the Earth is the only body that moves in the system, hence has kinetic energy and angular momentum, the total angular momentum is given by:

$$\vec{L}_{earth} = \vec{r}_{earth} \times m\vec{v}_{earth} \tag{10}$$

and is also here the only quantity we need to check is conserved.

2.5 Three-body model

We will now add Jupiter to our binary system. To do this we need to add the force between earth and Jupiter to eq. (5), we get

$$F_G^{earth} = \frac{GM_{\odot} M_{earth}}{r_{e\odot}^2} + \frac{GM_{jup} M_{earth}}{r_{ej}^2}, \tag{11}$$

where $r_{e\odot} = r_{earth} - r_{\odot}$ and $r_{ej} = r_{earth} - r_{jupiter}$. This can be expressed in each direction by

$$F_{G,x}^{earth} = \frac{GM_{\odot} M_{earth}}{r_{e\odot}^2} \frac{x_{e\odot}}{r_{e\odot}} + \frac{GM_{jup} M_{earth}}{r_{ej}^2} \frac{x_{ej}}{r_{ej}}, \tag{12}$$

where $x_{e\odot} = x_{earth} - x_{\odot}$ and $x_{ej} = x_{earth} - x_{Jupiter}$. This applies to the other directions (y, z) in the same manner. With acceleration, again, given by

$$a_x^{earth} = \frac{F_{G,x}^{earth}}{M_{earth}}. \tag{13}$$

The same equations can be applied to find the forces that Jupiter experiences. In our code, ll of this functionality is taken care of by the function Gravitiation-alForce in the solver class, combined with a loop in the velVerlet function that loops over all celestial bodies and adds the force on the current bodies from all the other bodies in the system. This functionality will also allow us to do our calculations in a center of mass (CM) system rather than only a system where the sun is in the center. Thus we are allowed to generate a realistic system. Note that these calculations will have to be done for all bodies, not just for the earth.

2.6 Full Solar system model

All the functionality needed to generate a model of the entire solar system is discussed in section 2.4 or previous, but it is worthwhile to note that the expression for the force given in eq. (8) for an n-body system in general look like this

$$F^j_{G,x} = \sum_{i=0, i\neq j}^{n} G\frac{M_i M_j}{(r_j - r_i)^3}(x_j - x_i), \tag{14}$$

for a body j being acted on by all other bodies in the system. The acceleration of a body j has the familiar form $a^j_x = F^j_{G,x}/M_j$.

To make our simulation even more realistic we use actual data from ssd.jpl.nasa.gov/horizons.cgi as initial values for all the bodies in the solar system (only bodies classified as planets), this is actually done already for the model with three celestial bodies.

2.7 Mercury perihelion precession

The last task at hand is to study the perihelion precession of Mercury predicted by General Relativity, and see if we can simulate this effect. The prediction says that if we disregard all disturbances on the orbit of Mercury caused by the other planets, the perihelion of the orbit will move about 43" (arc seconds) each century. This effect is not found when applying Newtonian gravity on this system (again with no influences from any other planets), as the orbit will just follow the same elliptical path to the end of times. By modifying this by accounting for relativity however, we can get the expression:

$$F_G = \frac{M_{sun}M_{mercury}}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right], \tag{15}$$

where l is the magnitude of Mercury's angular momentum, and c is the speed of light.

By using this expression for gravitational force instead of the pure Newtonian one, we should be able to see the precession. The effect is very small however, so a small time step in our code is required to actually be able to calculate it precisely. We have implemented a test that checks at what times Mercury is at it's perihelion, and what it's position is at these times. We set the initial position to be on the x-axis, where it's initially on it's perihelion. This means that we can find how many degrees the perihelion has moved by simply calculating:

$$\theta_p = atan\left(\frac{y_p}{x_p}\right), \tag{16}$$

where θ_p is just this angle, and y_p, x_p is the calculated positions. This angle will be in radians, and can be converted to arc seconds by multiplying by $60 * 60 * \frac{180}{\pi} \approx 206265$. By simulating the system for at least 20 years we should be able see how the position of the perihelion moves over time, and see if it corresponds with the prediction by general relativity.

3 Results

3.1 Two-body system

The main results of the system containing only the earth and the sun is shown in figures 1 and 2. The initial position and velocity used to obtain figure 1 and 2 was $(x, y, z) = (1, 0, 0)$ and $v(x, y, z) = (0, 2\pi, 0)$, respectively.

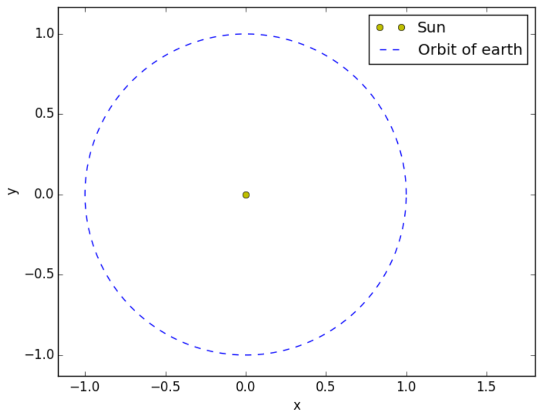


Figure 1: *Orbit of the earth around the sun using the velocity Verlet method. Number of integration points* $N = 10000$. *earthvv.png*

Comparing figure 1 and 2 (below) we see that the velocity Verlet method is more accurate than the forward Euler method, as we should expect as the error goes errors go as h^3 and h^2 for velocity Verlet and forward Euler respectively

.

Although the difference in figure 1 and figure 2 are minimal, there is a slight difference in the final/start position (1,0) in figure 2 that is not present in figure 1. This difference is more apparent if we zoom in around (1,0) as is done in figure 2 (right).
About the algorithms (velocity Verlet and forward Euler) we can tell by comparing figure 1 and 2, especially the right image in figure 2, that the velocity Verlet method is much more precise than the forward Euler method. Although this precision comes at the cost of FLOPS, there will be $3N \times dim$ more FLOPS using velocity Verlet than forward Euler, and thus time consumption, the precision provided by velocity Verlet is well worth it. The time usage is

```
terminal>system
Time spent on algorithm: 0.060000 seconds.
```

for the forward Euler algorithm, and

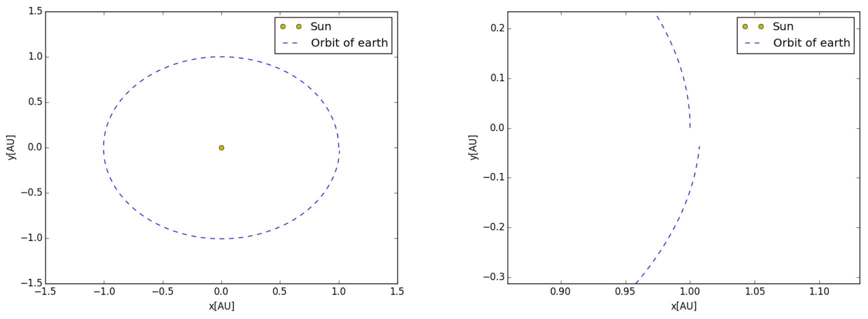


Figure 2: *Orbit of the earth using the euler forward method. Number of integration points $N = 10000$. Left: Full orbit of earth around the sun. Right: Closer look at what happens after one full orbit is supposed to be done, at (1,0). earthfe.png, this file was lost but a zoom in earthfe.png will provide it*

```
terminal>system
Time spent on algorithm: 0.113000 seconds
```

for the velocity Verlet algorithm, using 10000 integration points. If you are still not convinced that velocity Verlet is an algorithm with considerably more precision than forward Euler we encourage you to take a look at figures 10 and 11 in Appendix A

By setting the program to simulate the system for one year (an orbit of the Earth) with 100000 integration points, and making it print out these (hopefully) conserved quantities mentioned above 10 times through the year, we get this printout:

Time	K_{tot}	U_{tot}	L_{tot}
0.000000	5.921763e-005	-1.184353e-004	1.884956e-005
0.090910	5.921763e-005	-1.184353e-004	1.884956e-005
0.181820	5.921763e-005	-1.184353e-004	1.884956e-005
0.272730	5.921763e-005	-1.184353e-004	1.884956e-005
0.363640	5.921763e-005	-1.184353e-004	1.884956e-005
0.454550	5.921763e-005	-1.184353e-004	1.884956e-005
0.545460	5.921763e-005	-1.184353e-004	1.884956e-005
0.636370	5.921763e-005	-1.184353e-004	1.884956e-005
0.727280	5.921763e-005	-1.184353e-004	1.884956e-005
0.818190	5.921763e-005	-1.184353e-004	1.884956e-005
0.909100	5.921763e-005	-1.184353e-004	1.884956e-005

Table 1: Total kinetic energy (K_{tot}), potential energy (U_{tot}) and angular momentum (L_{tot}) for the earth moving in a circular orbit.

where the time has unit yr, the energies unit $M_\odot AU^2 yr^{-2}$, and the angular momentum $M_\odot AU^2 yr^{-1}$. By plugging in the initial values for the Earth in the equations above, we see that we get the exact values in the printout. We also see that the values are conserved with a very high precision, so we can easily say that our algorithm pass these physical unit tests.

Escape velocity

By trial and error

Looking at the trajectory while finding the escape velocity gave some insight into how large a circular orbit must be for a comet that visits the inner solar system once every hundred, thousand or even more years. The escape velocity is found numerically to be approximately

$$v_{escape} = 2.828\pi \frac{AU}{yr} = 8.884 \frac{AU}{yr},$$

by manually varying the initial velocity of the earth.

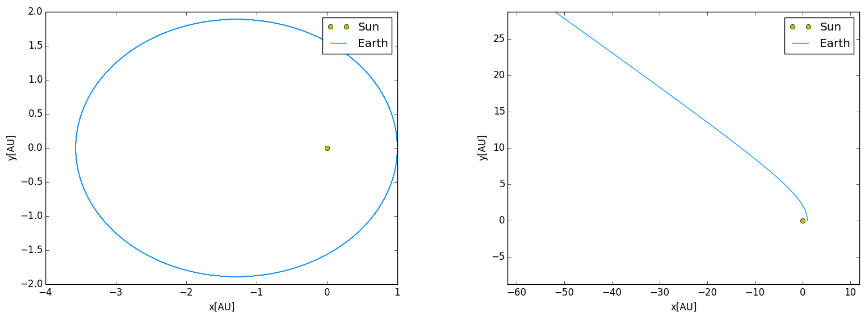


Figure 3: *Left: Orbit of the earth if the initial velocity is* $v = 2.5\pi AU/yr$. *Right: Trajectory of the earth if the initial velocity is* $v = 2.9\pi AU/yr > v_e scape$. *earthsun2-5escape.png, earthescape29.png*

Analytical

From conservation of energy we have that:

$$\begin{gathered}(K + U_g)_i = (K + U_g)_f \\ \frac{1}{2} m_{planet} v_e^2 + \frac{-GM_\odot m_{planet}}{r} = 0 + 0,\end{gathered} \tag{17}$$

where the sum of the kinetic energy K and potential energy U_g at a given initial point (distance r from the Sun) must be the same at the final point where the two bodies don't interact anymore. At this point the final velocity is 0, since we're finding the limit where the planet *exactly* escapes the gravitational pull of the Sun, hence the potential energy at this point is also 0. Solving for the escape velocity v_e and using that $GM_\odot = 4\pi\frac{AU^3}{yr^2}$ at distance $r = 1AU$, we get that:

$$v_e = \sqrt{\frac{2GM_\odot}{r}} = 8.886\frac{AU}{yr}, \tag{18}$$

which is almost impressively close to what we found numerically by trial and error.

Now looking at the difference between the brute force escape velocity and the analytical, it is, as mentioned, almost impressively accurate. There is of course a small difference, but this may be due numerical or human errors. Note that the analytical v_e is slightly bigger than the one found by trial and error, this should lead us to believe that the velocity found by trial is not actually a velocity with which the earth totally escapes the suns gravitational pull. Now considering figure 3 which compares two different initial velocities of the earth, neither of which is the analytical or trial escape velocity, we may reason that the velocity found by trial just leads to an incredibly huge orbit, or it may be a result of numerical inaccuracy.

As mentioned earlier regarding the escape velocity, finding this with trial and error gave some (minor) insight into the orbit of comets that rarely visit the inner solar system. Of course such objects will most likely have an elliptical orbit, but nevertheless they wander both extremely close to the sun and incredibly far away and are still being kept in orbit, which is fascinating.

3.2 Three-body system

In figure 4 (left) we see the orbit of the earth and Jupiter around the sun with the sun fixed in the center. Adding Jupiter to the system gave some, but very little, effect on the orbit of the earth. Some effect is expected when you add such a massive body to a binary system because it will provide a gravitational pull on the earth as can be seen from eq. (5). Increasing the mass of Jupiter to nearly that of a light brown dwarf (see Appendix: A on brown dwarfs) makes little to no change in the system, also seen in figure 4 (right). There should be some effect by adding a small brown dwarf to the system, but we suspect it is too far away from the biggest body to make a real difference after only one orbit.

Now increasing Jupiters mass to that of the the sun, as seen in figure 5, gives significant results. We can think of this as placing the earth between two stellar bodies with the same mass $M_\odot$. We see that earths trajectory is rather messy before it goes almost straight at the sun and gets slung out of the system. This is because after one year Jupiter is has just moved one fifth of its path, and at some point the gravitational pull on earth is incredibly (not like a black hole or neutron star) high and accelerates it in one direction, which happens again and again until the earth gets too much velocity and escapes the system.

This gives insight into life in "compact" binary star systems; a planet with orbit

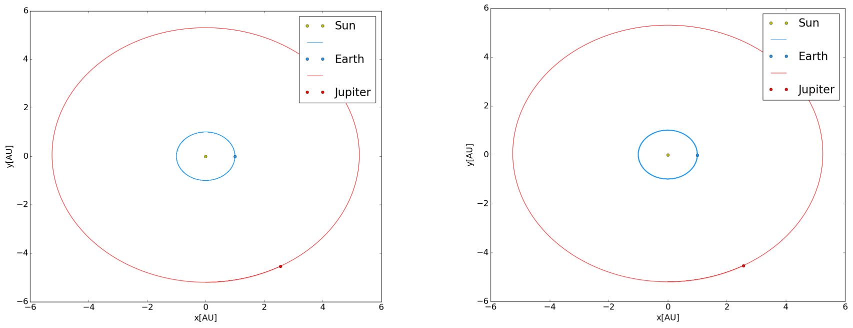


Figure 4: *Trajectory of earth when we vary the mass of Jupiter. Left: Proper mass and trajectory. Right:* $M^{trial}_{Jupiter} = 10M_{Jupiter}$*. In this and figure 5 the earth is initially at* $(x, y, z) = (1, 0, 0)AU$ *with velocity* $(v_X, v_y, v_z) = (0, 2\pi, 0)AU/yr$*, while Jupiter is at* $(x, y, z) = (0, -5.2, 0)AU$ *with velocity* $(v_x, v_y, v_z) = (\frac{5.2}{11.8} \times 2\pi, 0, 0)AU/yr$*. Jupiters velocity* v_x *is roughly one orbit per Jupiter year, which is logical. earthsunjupcirc.png, earthsunjup10circ.png*

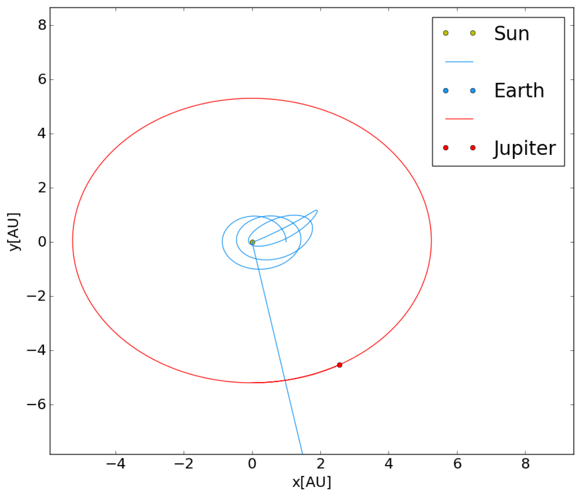


Figure 5: *Earths path in a system with, in effect, two stars. earthsun-jup1000circ.png*

in the habitable zone of one of the stellar bodies, again with mass $M_\odot$, varying the mass of the bodies obviously change this statement, would not stay in orbit for very long (see Appendix A on habitable zones). It is worth to note that a planet in such an orbit would not come to exist unless one of the stellar bodies is a wanderer that gets caught by the gravitational pull from the other stellar body.

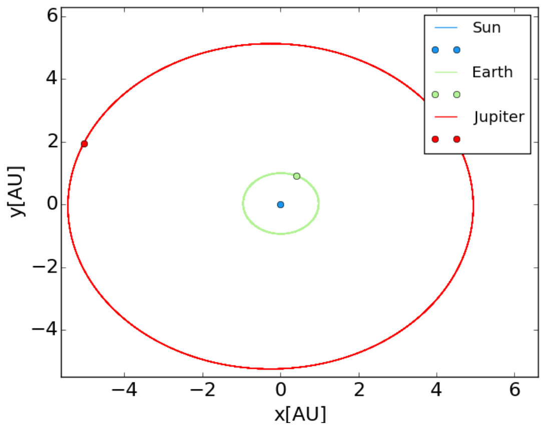


Figure 6: *Orbit of the earth, sun and Jupiter around the center of mass. Number of integration points* $N = 100000$*, initial data gathered from jpl, as mentioned in section 2.5, at 00.00 on October 23. 2016. earthsunjupCM.png*

In figure 6 we see the trajectory of earth, Jupiter and the sun, no longer keeping the sun in the center of the system, but using a CM system. In figure 7 (left) we see that this movement clearly has effect on the earths orbit and makes it more uneven, but still quite consistent. In figure 7 (right) we have zoomed in to get a better view of the trajectory of the sun.

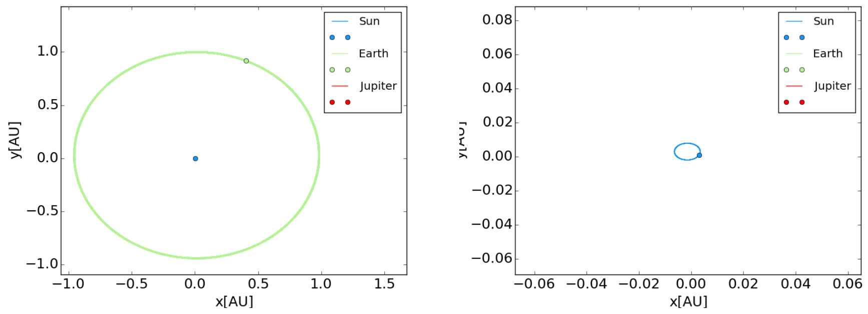


Figure 7: *Left: Closer look at the orbit of the earth and sun around a CM, calculated with Jupiter as well. Right: An even closer look at the suns movement. earthsunVjupCM.png, sunVearthjupCM.png*

The fact that earths orbit is more uneven using a CMs that when pinning the sun in the center, is an effect of just that; the sun is stationary no longer. After one year the sun will have completed only a portion of its rotation, leading to a slightly different gravitational pull on the earth than what happened one year earlier, thus the trajectory moves slightly each year. We may argue that jupiters gravitational pull should counteract this effect, but the earth is so much closer to the sun, which is a thousand times more massive as well, than Jupiter. The effect is very slight, but it is evident from the thick line in figure 7 (left) indicating the movement of the earth.

3.3 Full Solar system

In figure 8 we see the final model of the full solar system. Since the inner solar system is so small compared with the outer solar system we have provided a closer look on the inner solar system in figure 9.

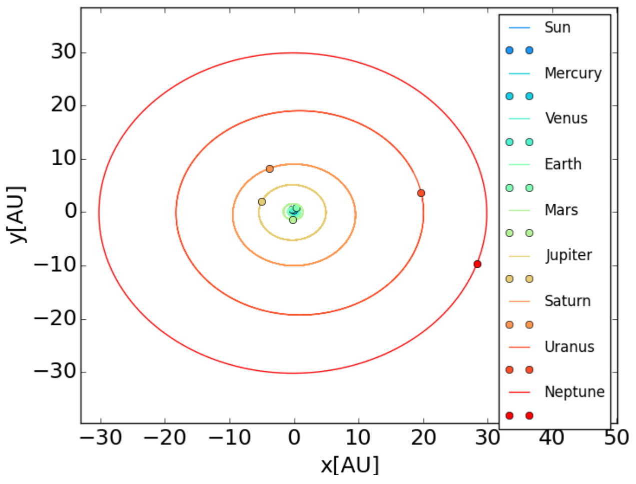


Figure 8: *A simulation of the full solar system where the outermost planet (Neptune) gets one full orbit. Again initial parameters are gathered from jpl at 00.00 on October 23 2016. fullsolarsystem.png*

We see from figure 8 that the outer planets moves in almost circular paths, but note that this is only in the xy-plane, in three dimensions the picture should look quite different with more elliptical paths for many of the planets. Looking at the inner planets in figure 9 we see that their orbits also behave in almost circular orbits as we should expect. Again it is worth to note that even if the orbit looks circular in the xy-plane this may not be the case in three dimensions.

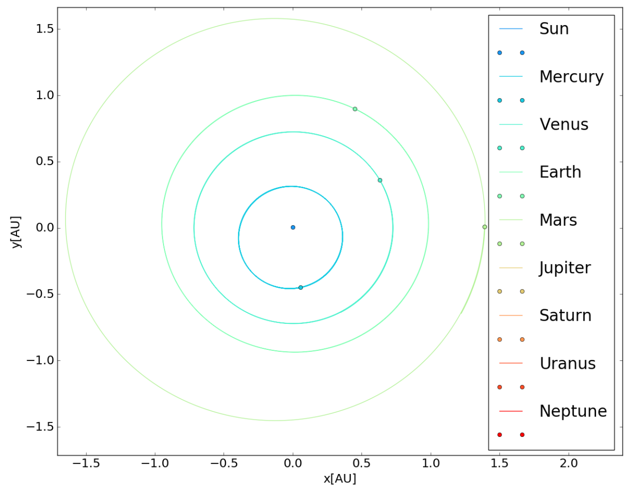


Figure 9: *A simulation of the inner solar system (closer to the sun than Jupiter), calculated with the full system of nine bodies, but only allowing all the inner planets to orbit at least once. innersolarwo.png*

3.4 Mercury perihelion

By running our program for the Sun and Mercury only over 20.5 years with 10^9 integration points, and with the relativistic correction in the calculations of the forces between them in place, we got the results printed to the file MercuryPerihelionRel.txt, which can be found in the GitHub repository linked in the Appendix. Here is some of the results:

Time (yr)	Angle (rads)	Angle (arc secs)
18.536342	0.000039	7.990871 "
18.777073	0.000039	8.054652 "
19.017805	0.000039	8.118433 "
19.258537	0.000040	8.353278 "
19.499268	0.000041	8.417060 "
19.740000	0.000041	8.480841 "
19.980732	0.000041	8.544623 "
20.221464	0.000043	8.779467 "
20.462195	0.000043	8.843251 "

Table 2: *Computed angles of the perihelion*

The first column shows at which time (in years) Mercury was calculated to be at perihelion, while the next shows the calculated angle in both radians and arc

seconds. This can be compared with the textfile MercuryPerihelionNoRel.txt, which shows the exact same calculations, but with no relativistic correction used. This case shows no signs of the perihelion changing position over time, other than computational errors.
By looking at the textfile for the relativistic case (and also the snippet above) we can see that the time between each recording is about 0.24 years, which correspond well to the actual orbital time of the planet, 0.240846 years.(ref: https://en.wikipedia.org/wiki/Mercury_(planet)). We also see the trend that the angle of the position of the perihelion is increasing linearly in time, meaning that we can predict at what angle the perihelion will be at when say a hundred years have passed. This is the reason we chose to only run it for 20 years, as it's very time consuming to run the program with 10^9 integration point (or even more). By running for fewer years we also get increased precision in our calculations, meaning that the linear approximation mentioned will be more accurate. From the snippet we see that Mercury doesn't hit it's perihelion at exactly $T = 20$ years, but if we use the closest one at time ≈ 19.98 and multiply the angle at this point with 5 to find what the estimated angle for $T \approx 100$ would be, we find:

$$8.544623'' \cdot 5 = 42.723115'' \approx 43'' \tag{19}$$

As we can see, our model also agrees with General Relativity in that the perihelion of the orbit of Mercury should move about 43 arc seconds per century. This effect in itself can be explained by the General Theory of Relativity. This theory states that everything that has mass bends the spacetime around it, making space and time itself behaving differently. The more massive the object, the more the spacetime is being bent, and it's this bending that's being perceived as gravity. The deeper you are in a gravitational field, the shorter distances will seem, and the slower time will seem to pass.
Mercury's orbit is located relatively close to the sun (perihelion about 0.3 AU), hence it's relatively deep in the Sun's gravitational field (compared to the other planets). At this "depth" the distance around the Sun will seem a little bit shorter than for an observer located farther from the Sun. So a full revolution around the Sun seen from Mercury will actually be a little MORE than one revolution seen from the observer. Hence in the time Mercury should have only done one full revolution, it has actually moved a longer distance while still having the same momentum, meaning it shoots out from the sun into the orbit again at a slightly different angle than it did in the last revolution. This is the effect that's seen when looking at the Mercury perihelion precession, confirming that the nature of motion isn't as straight forward as the Newtonian view would propose.

4 Conclusions

From this project we have seen that velocity Verlet integration method is a more precise, but slower method than forward Euler. Then it is reasonable to use forward Euler for simple tasks where precision may not be as important as just getting a feel for the system, but if we need precision we should (at least) use velocity Verlet. We have seen that the velocity Verlet gives high precision for very few integration points as well, making it really accurate considering how few extra FLOPS it uses per iteration. This method is therefore definitely a method we should consider using for future work.

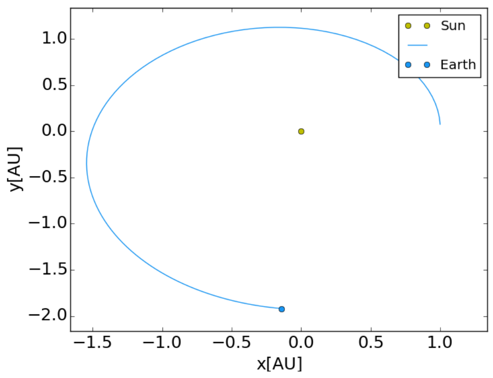
We have seen that changing the mass of Jupiter can have dramatic effect on the inner solar system. As the mass of Jupiter is increasing towards the mass of the sun, the more the system acts like a binary star system, with no planets being able to keep a stable orbit in between these bodies. This results in slinging the earth and the other planets of the inner solar system out of orbit and into space after a couple of years. So we can safely say that we should be happy that Jupiter isn't more massive than it already is.

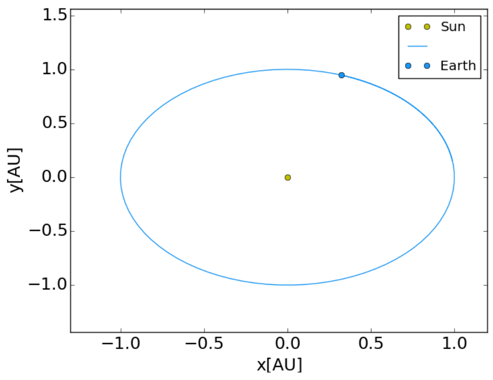
We have also seen that Newtonian mechanics cannot calculate the orbit of Mercury with much precision and that we need the theory of general relativity (GR) to calculate Mercury's orbit precisely. We have seen that corrections using GR provides quite good accuracy for Mercury's orbit, but again we may be susceptible to numerical errors.

5 Appendix

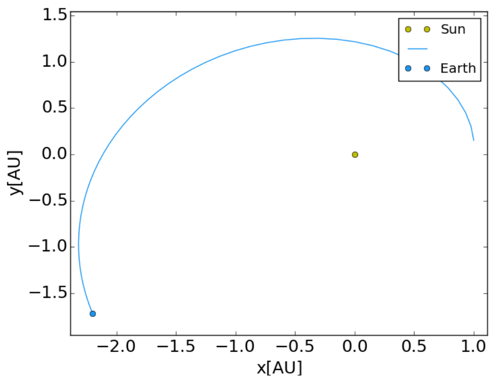
5.1 A

Precision and stability of forward Euler and velocity Verlet: In figure 10 and 11 we compare the precision and stability of forward Euler and velocity Verlet using rather big timesteps.





These figures can be found in the repository as Eu$_{s}tability100.png, VV_{s}tability100.png, Eu_{s}tability50.pngan$



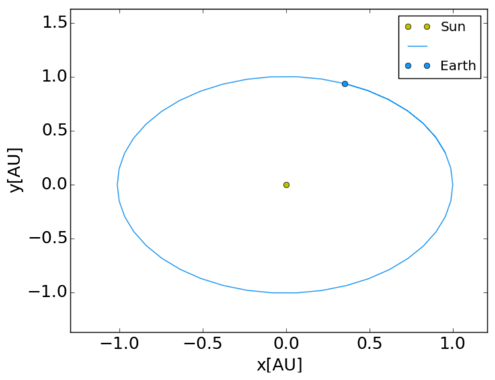


Figure 11: *How the Forward Euler (upper) and Velocity Verlet (lower) methods perform with $N = 50$ over the timespan of 1.2 yr.*

Brown dwarf: A brown dwarf is a very small and cold star, which makes it hard to observe. Brown dwarfs has mass in the range of $13 - 75(80)M_{Jupiter}$, this mass does not allow brown dwarfs to sustain nuclear fusion.
`https://en.wikipedia.org/wiki/Brown_dwarf` (24.10.2016)

Habitable Zones: The habitable zone around a star is the distance from the star that allows the planet to keep a habitable atmosphere, so that life can evolve. Estimates of the habitable zone for a star like our sun ranges from $0.725AU \rightarrow 2.0AU$
(`https://en.wikipedia.org/wiki/Circumstellar_habitable_zone` 24.10.2016). This is not taking into account the composition of the atmosphere, as we see from our own solar system with three planets in the habitable zone where only one has life (that we know at the moment).

5.2 GitHub Repository

All programs, textfiles and plots can be found at this
GitHub Repository: **https://github.com/mrnafstad/Project-3**

Relevant programs and textfiles:
system.cpp: main program
planet.cpp: program for the class 'planet'
solver.cpp: program for the class 'solver'
MercuryPerihelionRel.txt: Data for Mercury's perihelion, with relativity
MercuryPerihelionNoRel.txt: Data for Mercury's perihelion, without relativity

5.3 References

`https://en.wikipedia.org/wiki/Mercury_(planet)`
`https://en.wikipedia.org/wiki/Brown_dwarf`
`https://en.wikipedia.org/wiki/Circumstellar_habitable_zone` 24.10.2016