Project 3 - FYS4150

Oda Langrekken, Trude Hjelmeland and Jostein Brændshøi

Department of Physics
University of Oslo

October 28, 2017

Abstract

We use the Euler and Velocity Verlet algorithms to simulate the motion of the Earth around the Sun, and later we use the latter method to model the entire Solar System. We discover that the Verlet method is by far superior to the Euler method regarding precision, conservation of energy and angular momentum, and that it nearly matches the Euler method in terms of time efficiency. Experimenting with our model, we also discover that changing the properties of the planets or the distance-dependency of the gravitational force, has dramatic consequences for our Solar System.

The address `https://github.com/jostbr/FYS4150-Projects/tree/master/Project3` on GitHub is associated with this project. Here one can find all code used in the project. This includes C++ source files containing the core of the project, but also a Python plotting script for result visualization. There are also available benchmark results from running the code as well as LaTeX source for this PDF. In addition some animations (generated with the python script) are uploaded to YouTube are associated with this project and the specifics links are provided below in the report when relevant.

Contents

1 Introduction

The motion of any classical object of constant mass experiencing some force, can be described by Newton's second law

$$F = ma \tag{1}$$

All high school student taking a course in physics, will have solved this equation for a number of simple systems with constant acceleration (the motion of an object moving on a frictionless plane is perhaps the most well-known example). If the acceleration is not constant, but rather some function of position, the situation becomes more complex as one will have to solve a second order differential equation to calculate the motion. Sometimes the equation can still be solved analytically, but more than often this is not possible. Luckily we have tools to find approximate solutions to these equations, namely numerical algorithms.

The aim of this project is to simulate the evolution of our Solar System by solving coupled ordinary differential equations using numerical methods in an object oriented manner. By first utilizing the forward Euler method we will see that the velocity Verlet is a more fitting algorithm to employ as the latter conserves the energy of the system, and thus provides a more stable solution.

First we will discuss the theory used in this project, most notably Newton's Second Law for many body systems. We then move on to explain how the equations can be discretized and solved using the Euler and Verlet method. Next we introduce some of the tests we have performed on our model to test the precision. We then present our results and how they can be interpreted. Finally we offer some reflections and personal insight gained from working with this project.

2 Theory

2.1 Newton's Second Law

Newton's second law states that the force $\mathbf{F}$ acting on a body is equal to the mass, m, multiplied by the acceleration, $\mathbf{a}$, of that body [6];

$$\mathbf{F} = m\mathbf{a} = m\frac{d^2\mathbf{r}}{dt^2} \tag{2}$$

This equation is valid for all systems where the masses are constant, which we will assume is the case in our project. This is, however, a truth with some modifications (i.e. the Sun is reducing it's mass by a fraction of 10^{-13} every year [2]).

The above equation (2) is a second order ordinary differential equation that can be rewritten as two coupled first order ordinary differential equations [4] by;

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}(\mathbf{r}, t) \tag{3}$$

$$\frac{d\mathbf{v}}{dt} = \mathbf{a}(\mathbf{r}, t) \tag{4}$$

To solve a set of coupled ordinary differential equations we need to know the initial values at some initial time t_0 for both position and velocity, namely $\mathbf{r}(t_0)$ and $\mathbf{v}(t_0)$ which we will discuss more in section 3.

2.2 Earth-Sun System

In the first part of our project we will simplify our system by limiting ourselves to only one planet orbiting a star, namely the Earth orbiting the Sun.

The motion of the Earth is governed by the universal law of gravitation, formulated by Sir Isaac Newton [6]. The law states states that two point masses are attracted to each other by a force that is directly proportional to the masses of the bodies and inversely proportional to the distance between their two mass centers, r, squared;

$$\mathbf{F} = G\frac{m_1 m_2}{r^2}\hat{\mathbf{r}} \tag{5}$$

where $\hat{\mathbf{r}}$ is the unit vector pointing from the body experiencing the force to the object exerting the force. Thus the gravitational attraction is a central force, meaning that the force is directed along a straight line passing through the centers of the interacting bodies [7].
The gravitational constant, G, was found through experiments to be $6.67 \times 10^{-11}\frac{m^3}{kg \times s^2}$.
Because the size of the planets is very small compared to the distance between them, we will approximate the planets and the Sun as point particles.

Assuming circular orbits, the acceleration of the Earth is described by the centripetal acceleration. Inserting the centripetal acceleration in Newton's Second Law (2), using gravitation (5) as the force and solving for the Earth's acceleration in this simplified two body Solar System yields;

$$a_E = \frac{v_E^2}{r} = \frac{F}{M_E} = \frac{GM_S}{r^2} \tag{6}$$

where M_E and M_S is the mass of the Earth and the Sun respectively.
This leads to the relationship $v^2 r = GM_S$.

To avoid working with very large numbers (which easily leads to loss of numerical precision) we scale our equations. Lengths will be measured in units of astronomical units (AU) where 1 AU is the mean distance between the Earth and the Sun. For the velocities we choose to work in units of AU/year.

Assuming circular orbits, we have $r = 1$ AU and $v = 2\pi$ AU/year as 2π AU is the circumference of a full orbit. Plugging this back into $v^2 r = GM_S$ we see that $GM_S = 4\pi^2\frac{(\text{AU})^3}{(\text{year})^2}$ for our simplified system.

To further simplify our system, we decompose the acceleration of the Earth. For the x-direction , this leads to the expression

$$a_x = a\cos\theta = -\frac{4\pi^2\cos\theta}{r^2} = -\frac{4\pi^2 x_E}{r_{ES}^3} \tag{7}$$

where we have used that the x-component of the position of the Earth, x_E, can be expressed as $x_e = r_{ES}\cos\theta$ where r_{ES} is the distance between the Earth and the Sun. Similar expressions can be found for the y- and z-directions, where x_e will be replaced by y_e and z_e respecitvely.

2.3 The Three Body System

The Solar System does not, however, consist only of the Earth and the Sun. Proceeding with our project we want to include more planets in our solar system, and we will start by expanding our system to include the planet Jupiter. If we continue on with our simplified system described in the section above (section 2.2), we can include the gravitational force on the Earth from Jupiter by simply expanding the expression in equation (7) with a additional term, namely;

$$a_x = -\frac{4\pi^2 x_E}{r_{ES}^3} - \frac{4\pi^2 \frac{M_J}{M_S}(x_E - x_J)}{r_{EJ}^3} \tag{8}$$

where we have used

$$GM_J = G\frac{M_S}{M_J} = 4\pi^2\frac{M_J}{M_S}$$

This needs to be calculated for all the spatial directions and we need to add a similar set of equations for Jupiter's acceleration.

2.4 The Solar System

To finalize our project we want to include all $N = 9$ planets of the Solar System in the simulation. We do this by expanding the calculation of the accelerations as shown in equation (8) to include all planets and thereby all pair of planets. To simultaneously make the calculation as close to reality as possible, we want to include the motion of the Sun. This would imply that the sun is no longer some special case fixed at the origin, but rather treated as a planet just like the other planets. To be even more general we may think of this purely as an N-body system. In an N-body system we have N bodies (in our case planets) all interacting with each other with some force (in the following discussion we will use gravity, however the approach stated below applies to any force that is inversely porportional to r^2, e.g. Coulomb force etc.). In particular the gravitational vector force between body i and body j can be, following (5), expressed as

$$\mathbf{F}_i = G\frac{m_i m_k}{|\mathbf{r}_i - \mathbf{r}_k|^3}(\mathbf{r}_i - \mathbf{r}_k) \tag{9}$$

and Newtons second (2) law for body i reads

$$\mathbf{F}_i = m_i\frac{d^2\mathbf{r}_i}{dt^2} \tag{10}$$

and combining (9) with (10) by summing up all the forces acting on body i gives the second order vector differential equation

$$m_i\frac{d^2\mathbf{r}_i}{dt^2} = \sum_{k=1}^{N} G\frac{m_i m_k}{|\mathbf{r}_i - \mathbf{r}_k|^3}(\mathbf{r}_i - \mathbf{r}_k) \tag{11}$$

for body i. We have one such equation for each of the N bodies. Also the vector equation implies one equation for each direction, i.e. three equations per body, resulting in a total of $3N$ second order differential equations.

Another interesting aspect with such an N-body system is its mass center. By using a weighted average of the position vectors of each planet with each planet position weighted by its respective mass we can write

$$\mathbf{r}_M = \frac{1}{M}\sum_{i=1}^{N} m_i \mathbf{r}_i \tag{12}$$

where

$$M = \sum_{i=1}^{N} m_i \tag{13}$$

is the total mass of the system and $\mathbf{r}_i$ is the position vector of planet i. We will utilize this in the result section to examine how good of an approximation it is to treat the Sun as fixed in the origin vs fully accounting for its motion.

When treating the Sun as a planet in the coming simulations below, we wish to give it an initial velocity such that the total momentum $\mathbf{p}_{tot}$ of the Solar System is equal to zero which will make sure the Solar System doesn't drift away, i.e. keeping the mass center (12) fixed. In order to find this initial velocity we decompose the total momentum of the system into

$$\mathbf{p}_{tot} = \sum_{i \neq sun} m_i \mathbf{v}_i + m_{sun}\mathbf{v}_{sun} = 0 \tag{14}$$

which by rearrangement gives

$$\mathbf{v}_{sun} = \frac{1}{m_{sun}} \sum_{i \neq sun} m_i \mathbf{v}_i \tag{15}$$

which would be the initial velocity we would give the Sun in the Solar System simulations when we take motion of the Sun into account instead of keeping it fixed at the origin.

2.5 Escape Velocity

The escape velocity is defined as the minimum velocity a celestial body must have to escape the gravitational influence of a massive body. In our project we seek to find this velocity both analytically and through trial and error with our numerical model.

To find the escape velocity analytically we look at the specific energy (energy per mass unit) for the body in question. The specific energy is composed of two components, namely the specific potential- (E_p) and kinetic energy (E_k) which is given as; $E_p = -\frac{GM}{r}$ and $E_k = \frac{v^2}{2}$ where G is the gravitational constant, M is the mass of the body generating the gravitational field and r is the distance between the body generating the field and the body experiencing the field. [7].
For the body to escape the gravitational field, it must have a net positive mechanical energy, i.e. the body's kinetic energy must be larger than the absolute value of its potential energy.

$$\frac{v^2}{2} \geq \frac{GM}{r} \tag{16}$$

This implies that to escape a body with mass M, the escaping body need a velocity, v_e given as:

$$v_e \geqslant \sqrt{\frac{2GM}{r}} \tag{17}$$

Later we will compare this analytical result with numerical simulations.

2.6 Mercury's perihelion

So far we have used Newton's law of universal gravitation (5) to express the forces acting between the planets. A special feature of the Newtonian force [5] are closed elliptical orbit, i.e. after completing one full orbit the planet will be at the same point as where it started. This is, however, not the case for Mercury. Observations have shown that the perihelion of Mercury (the position where Mercury is at its closest to the Sun) changes by 43" per century more than what would be expected due to attraction from other planets [5]. This effect was explained by the theory of general relativity, which proposes a correction term to Newton's law of gravitation [5]:

$$F_G = \frac{GM_{Sun}M_{Mercury}}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right] \tag{18}$$

where $l = |\mathbf{r} \times \mathbf{v}|$ is the magnitude of Mercury's angular momentum (per unit mass) and c is the speed of light in vacuum.

3 Methods

3.1 Discretization

In order to discretize (3) and (4), we introduce an upper time limit t_{max} and subdivide the continuous time interval from $t_0 = 0$ up to t_{max} into n grid points as

$$t_i = t_0 + ih \ , \qquad n = \frac{t_{max} - t_0}{h} + 1$$

for $i = 1, ..., n$ and where h is the chosen step size between time points. For e.g. the x-position, this means that the continuous variable is now evaluated at the discrete time points t_i, i.e., $x(t_0 + ih) = x(t_i) = x_i$ and similarly for the velocity and acceleration. Moving forward we will focus on the position, velocity and acceleration in one spatial dimension for one body (planet) for simplicity and readability, however the methods described below apply similarly to the two other dimensions and more bodies as shown later.

Next we wish to use finite difference approximations for the derivatives in order to obtain discretized versions of (3) and (4) and, with finite difference methods in general, they are based on

Taylor series. We then do a Taylor expansion of $x(t+h), v(t+h), a(t+h)$ around t:

$$x(t_i+h) = \sum_{k=0}^{\infty} \frac{1}{k!}\frac{d^k x}{dt^k}\Big|_{t_i} h^k = x(t_i) + h\frac{dx}{dt}\Big|_{t_i} + \frac{h^2}{2}\frac{d^2x}{dt^2}\Big|_{t_i} + \mathcal{O}(h^3) \tag{19}$$

$$v(t_i+h) = \sum_{k=0}^{\infty} \frac{1}{k!}\frac{d^k v}{dt^k}\Big|_{t_i} h^k = v(t_i) + h\frac{dv}{dt}\Big|_{t_i} + \frac{h^2}{2}\frac{d^2v}{dt^2}\Big|_{t_i} + \mathcal{O}(h^3) \tag{20}$$

$$a(t_i+h) = \sum_{k=0}^{\infty} \frac{1}{k!}\frac{d^k a}{dt^k}\Big|_{t_i} h^k = a(t_i) + h\frac{da}{dt}\Big|_{t_i} + \frac{h^2}{2}\frac{d^2a}{dt^2}\Big|_{t_i} + \mathcal{O}(h^3) \tag{21}$$

Rearranging these equations in terms of the first order derivative (which is the one appearing in the equations we are studying) gives

$$\frac{dx}{dt}\Big|_{t_i} = \frac{x(t_i+h)-x(t_i)}{h} - \frac{h}{2}\frac{d^2x}{dt^2}\Big|_{t_i} + \mathcal{O}(h^2) \tag{22}$$

$$\frac{dv}{dt}\Big|_{t_i} = \frac{v(t_i+h)-v(t_i)}{h} - \frac{h}{2}\frac{d^2v}{dt^2}\Big|_{t_i} + \mathcal{O}(h^2) \tag{23}$$

$$\frac{da}{dt}\Big|_{t_i} = \frac{a(t_i+h)-a(t_i)}{h} - \frac{h}{2}\frac{d^2a}{dt^2}\Big|_{t_i} + \mathcal{O}(h^2) \tag{24}$$

We are now in a position to discuss the two algorithms central in this project. Even though the methods below can be applied generally to second order ODE initial value problems, the discussion below will be slightly skewed towards our case in which we are dealing with Newtons second law. However generalization is attempted.

3.2 Euler's method

To derive Euler's algorithm for second order differential equations, we use (22) and (23) explicitly to first order, i.e. absorbing the second derivative terms into the order term implying the order term becomes $\mathcal{O}(h)$. Then inserting into (3) and (4) for the derivatives produces

$$\frac{x(t_i+h)-x(t_i)}{h} + \mathcal{O}(h) = v(t_i) \tag{25}$$

$$\frac{v(t_i+h)-v(t_i)}{h} + \mathcal{O}(h) = a(t_i) \tag{26}$$

which through rearrangement, neglecting the order term (truncating the taylor series) and using the numerical notation introduced earlier, gives

$$x_{i+1} = x_i + hv_i \tag{27}$$

$$v_{i+1} = v_i + ha_i \tag{28}$$

where $a_i = F(x_i, t_i)/m$ is found through Newtons second law of motion (or another right-hand-side of a second order ODE if one is not talking about Newtons second law). To start the Euler solver, one needs initial conditions x_0 and v_0.

As mentioned above, we could have done this derivation for the position, velocity and acceleration in either of the three dimensions. Thus, using vector notation, we can write the Euler method for all three dimensions as

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i \tag{29}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + h\mathbf{a}_i \tag{30}$$

where $\mathbf{x}_i = (x_i, y_i, z_i)$, $\mathbf{v}_i = (v_i^x, v_i^y, v_i^z)$, $\mathbf{a}_i = (a_i^x, a_i^y, a_i^z)$ and $\mathbf{F}(\mathbf{x}_i, t_i)$ is some force with components in all three directions. This describes a recipe for finding x and v at the next time point using the values of x and v at the previous time point and is the essence of Euler's method. Note that this is a general approach for any force and can thus be applied for any system described by Newtons second law.

Regarding the discretization, we note that the truncation error done in approximating the derivatives is $\mathcal{O}(h)$. In other words, one of the drawbacks of this method is the "large" truncation error and thus not as accurate solutions as one might wish. In the next section we shall see that there exists improvements on this matter.

3.3 Velocity Verlet method

Now we move on to derive another algorithm for solving second order ordinary differential equations, most notably Newtons second law. Again we start our discussion for the position, velocity and acceleration in one dimension. As before we use the Taylor series (22) and (23) (but now keep the second order h term explicit) and substitute into first and second equation in (3) and (4) to get

$$\frac{x(t_i + h) - x(t_i)}{h} - \frac{h}{2}\frac{d^2x}{dt^2}\Big|_{t_i} + \mathcal{O}(h^2) = v(t_i) \tag{31}$$

$$\frac{v(t_i + h) - v(t_i)}{h} - \frac{h}{2}\frac{d^2v}{dt^2}\Big|_{t_i} + \mathcal{O}(h^2) = a(t_i) \tag{32}$$

and then using $d^2x/dt^2 = a$ in (31) as well as the Taylor series (24) (explicitly to first order) for $d^2v/dt^2 = da/dt$ in (32) we get

$$\frac{x(t_i + h) - x(t_i)}{h} - \frac{h}{2}a(t_i) + \mathcal{O}(h^2) = v(t_i) \tag{33}$$

$$\frac{v(t_i + h) - v(t_i)}{h} - \frac{1}{2}[a(t_i + h) - a(t_i)] + \mathcal{O}(h^2) = a(t_i) \tag{34}$$

which by rearrangement, truncation of Taylor series and applying numerical notation yields

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}a_i \tag{35}$$

$$v_{i+1} = v_i + \frac{h}{2}(a_{i+1} + a_i) \tag{36}$$

where $a_i = F(x_i, t_i)/m$ as for the Euler method and $a_{i+1} = F(x_{i+1}, t_i)/m$ can be obtained after having computed x_{i+1} in (35). As for the Euler case above, we have similar time-stepping for

position and velocity for all three dimensions giving the vector equation

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\mathbf{a}_i \tag{37}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{2}(\mathbf{a}_{i+1} + \mathbf{a}_i) \tag{38}$$

with $\mathbf{x} = (x, y, z)$, $\mathbf{v} = (v_i^x, v_i^y, v_i^z)$ and $\mathbf{a} = (a_i^x, a_i^y, a_i^z)$. Needed to start solving the equations are initial conditions for position and velocity in all three directions.
We have now reached the Velocity Verlet algorithm. We reiterate, as above, that (37) and (38) allow for a general force to compute $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ so this method (as with Euler's method) is by no means limited to a gravitational study done in this project and can be applied to e.g. a classical molecular dynamics problem.

As can be seen from the used approximations of the derivatives, we have here a truncation error of $\mathcal{O}(h^2)$, that is an improvement by order h compared to the Euler method. This tells us that the Verlet algorithm is more accurate. Also this higher accuracy comes only at a marginal increase in the number of floating-point operations as will be discussed below.

3.4 Algorithm for the N-body system

Next we wish to utilize the above discussed Euler and Verlet methods to develop an algorithm for the general N-body system. Since the Verlet algorithm later turns out to be the method of choice we will here focus the discussion on this method.

When we go about solving (11) for each body, we must first loop over each time step, then loop over all bodies, then loop over all dimensions x, y, z, then loop over all bodies again to compute the acceleration for each body (we need forces from all other bodies). This puts us in a position then to update the velocity and position of each body an accordance with (37) and (38). A more detailed description can be seen in algorithm 1 based on an object-oriented setup for the bodies. We reiterate that the below algorithm will can be applied to any N-body system that's modeled using Newtons second law of motion regardless of the type of force we use.

3.5 Algorithm complexity

Let us consider the efficiency of algorithm 1. Although this algorithm is stated for the Verlet case, the efficiency discussion below will include both the Euler and Verlet methods for comparison. In doing this we will specialize to the case of the force being gravity as this is relevant for the implementations in this project. First we look at the Euler method. Observing (29) and (30) we conclude that $3 \cdot 4 = 12$ FLOP (all three directions) is needed for computing $\mathbf{x}$ and $\mathbf{v}_{i+1}$ in addition to $5N$ (N bodies in the simulation) FLOP for computing the needed $a_i = F(x_i, t_i)/m_i$ for F given by equation (5). This is repeated $n-1$ times (excluding initial time) to compute the solution for all time points and for each of the N-bodies, thus resulting in a total cost of

$$\text{COST}_{euler} = 3N(4 + 5N)(n-1) = (12N + 15N^2)(n-1) \qquad \rightarrow \qquad \mathcal{O}(n, N^2) \tag{39}$$

Moving on to the Verlet algorithm (37), (38) we conduct a similar argument. Here we need $3 \cdot 9 = 27$ FLOP to compute $\mathbf{x}_{i+1}$ and $\mathbf{v}_{i+1}$. It is also needed to compute $\mathbf{a}_{i+1}$ ($\mathbf{a}_i$ is known from

Algorithm 1 N-Body Velocity Verlet Algorithm

```
1: procedure NBODY_SOLVER
2:     while t<=t_max do
3:         for i=1 to num_bodies do
4:             for d=1 to num_dimensions do
5:                 body_new[i].r[d] = body_old[i].r[d] + h*body_old[i].v[d] +
   h*h/2*body_old[i].a[d]
6:
7:         for i=1 to num_bodies do
8:             for d=1 to num_dimensions do
9:                 for k=1 to num_bodies do
10:                    body_new[i].a[d] += body[i].compute_acc(body_new[k])
11:
12:                body_new[i].v[d] = body_old[i].v[d] + h/2*(body_new[i].a[d] +
   body_old[i].a[d])
13:
14:            for i=1 to num_bodies do
15:                body_old[i] = body_new[i]
```

previous step) meaning $5N$ FLOP here too. Again this is repeated for $n-1$ time points producing a cost of

$$\text{COST}_{verlet} = 3N(9+5N)(n-1) = (27N+15N^2)(n-1) \qquad \rightarrow \qquad \mathcal{O}(n,N^2) \tag{40}$$

While both algorithms are efficient and require floating-point operations first order in n and second order in N, we see that the Verlet algorithm needs slightly more computations, but definitely not much. In fact with e.g. $N = 9$ and $n = 1000$, the Verlet algorithm would only require 10% more floating point operations. Also recalling from section 3.3, the Verlet algorithm was significantly more accurate (smaller error) and thus this seems like the superior algorithm so far, especially considering one could afford using a fewer number of grid points n for the more accurate algorithm balancing out the need for more FLOP. More advantages of the Verlet algorithm will be discussed in the coming sections.

Also worth mentioning is that for many applications n (number of time steps) will be significantly larger than N (number of bodies). This is especially the case for us with N at most equal to the 9 planets in the solar system, and so the number of time steps will dominate the time consumption of the algorithm. However, in e.g. molecular dynamics when one simulates millions, maybe billions of particles, N might be similar to or much larger than n and so in this case the Euler and Verlet algorithm would converge towards the same time usage as N gets very big (due to the fact that most of the computations will take place on inter-particle force calculations). In this case the Verlet algorithm would be especially beneficial.

4 Testing

In this project we have been executing various tests on the implemented algorithm in order to make sure that the code functions as expected. These tests are largely based on physical and

mathematical properties of the system we are studying. We thus check if our code produces results consistent with some theoretical principles like e.g. conservation of energy. However we also look briefly at timing and numerical stability. All the tests below are done on the sun-earth system with the sun fixed at the origin for all times, unless the general solar system case is specified.

4.1 Circular orbits

One property we wish to check for is that of circular orbits. To do this we consider the two-body case of the sun-earth system with the Sun fixed in the origin and the initial position of the Earth as

$$\mathbf{r}_E(0) = (1\text{AU}, 0, 0) \tag{41}$$

We then ask the question; what initial velocity for the Earth would result in a circular orbit around the Sun? A possible test for this is visual examination upon the plots produced for various initial velocities. While this method is useful as a first approach, it's a qualitative approach. In order to obtain more quantitative results we may look at the time evolution of the distance

$$r_E(t) = |\mathbf{r}_E(t)| = \sqrt{x_E(t)^2 + y_E(t)^2 + z_E(t)^2}$$

between the Earth and Sun. Mathematically, a circular orbit would then correspond to

$$\frac{dr_E}{dt} = 0 \ , \qquad t \in [0, t_{max}]$$

or similarly that $r_E(t) = r_0$ where r_0 is the initial distance to the Sun. To check for this in the code, one could, for different initial Earth velocities, write $r(t)$ to file for all time steps and then plot the results. One would then look for the initial conditions that give $r(t) =$ constant. A more accessible approach is to just examine the values of r_E before and after the simulation, that is the initial r_E versus the final r_E. If these are within a tolerance of, say 0.01%, for arbitrary upper time limits, there is strong indication of circular orbits.

We adopted this final method, and after repeating the process for many different sets of initial velocities and upper time limit, we found that an initial velocity of

$$\mathbf{v}_E = (v_x^E, v_y^E, v_z^E) = (0, 2\pi \text{ AU/yr}, 0) \tag{42}$$

produce circular orbits. This was done for initial earth position $\mathbf{r}_E(0) = (1 \text{ AU}, 0, 0)$, however, due to the spherical symmetry of the gravitational force, it is reasonable to believe that any initial condition with $|\mathbf{r}_E| = 1\text{AU}$ would give the same results as long as the magnitude of the initial velocity is $|\mathbf{v}_E| = 2\pi\text{AU/yr}$ with tangential direction, i.e. $\mathbf{r}_E \cdot \mathbf{v}_E = \mathbf{0}$. The results in (42) correspond with the earth traveling along the circumference of a circle with radius 1AU and completing one cycle in one year, very much like we experience in real life.

4.2 Energy conservation

Another test we would like to do on the circular orbit sun-earth system regards energy conservation. So this test will use the initial condition (41) and (42). As discussed above, maintaining a circular orbit would mean a constant distance from the Earth to the Sun, and thus the Earth would also have constant potential energy E_P, in the sun's gravitational field. In other words we would expect

the potential energy to be conserved. Also since the gravitational force is spherically symmetric, we would expect the earth to travel the circular orbit with a constant velocity magnitude (not direction), and thus the kinetic energy E_K should also be conserved in this case. In order to test for this in our program, we compute the kinetic and potential energy of the system as

$$E_K = \frac{1}{2} m_E |\mathbf{v}_E|^2 \ , \qquad E_P = -G\frac{m_E m_S}{|\mathbf{r}_{ES}|} \tag{43}$$

where M_E and M_S are the masses of the Earth and the Sun, respectively. We check for the values of the kinetic and potential energy in the simulation similarly to how we did in section 4.1. We then find that both the kinetic and potential energy is conserved (down to about 0.00001% accuracy) by using the Verlet method, and so this algorithm conserves the energy as it should. However this is not the case when using the forward Euler method. In table 4.1 and 4.2 the energy conservation properties of the two algorithms can be more quantitatively viewed. The Euler method misses by almost 50% on the conservation of both kinetic and potential energy. The results of this test isn't the only way in which we evaluate the two methods, but here the Verlet algorithm clearly seems like the preferred one.

Euler	Initial value [kg AU^2/yr^2]	Final value [kg AU^2/yr^2]
Kinetic energy	$1.1843525 \cdot 10^{26}$	$8.2225093 \cdot 10^{25}$
Potential energy	$-2.3822416 \cdot 10^{26}$	$-1.6519664 \cdot 10^{26}$

Table 4.1: Kinetic and potential energy of the circular orbit Sun-Earth system at the initial and final time (after 74.31 years) when using the foward Euler algorithm and time step $h = 1$ hour.

Velocity Verlet	Initial value [kg AU^2/yr^2]	Final value [kg AU^2/yr^2]
Kinetic energy	$1.1843525 \cdot 10^{26}$	$1.1843521 \cdot 10^{26}$
Potential energy	$-2.3822416 \cdot 10^{26}$	$-2.3822412 \cdot 10^{26}$

Table 4.2: Kinetic and potential energy of the circular orbit Sun-Earth system at the initial and final time (after 74.31 years) when using the velocity Verlet algorithm and time step $h = 1$ hour.

In the above test we studied the circular orbit case. However, it is worth noting that we also compute the kinetic, potential and then total mechanical energy

$$E = E_K + E_P = \sum_{i=1}^{N} \frac{1}{2} m_i |\mathbf{v}_i|^2 + \sum_{i=1}^{N} \sum_{j>i} -G\frac{m_i m_j}{|\mathbf{r}_{ij}|} \tag{44}$$

before and after the simulation for any system of N planets we examine. Here we have $j > i$ as a condition for the inner loop in the potential energy term to make sure we only compute the potential energy once per planet pair and not twice.

Since the only force acting on the planets is gravity, and that is a conservative force, we know from fundamental physics that the total mechanical energy (44) should be conserved. Indeed this is what we observe as well in the simulation when continuing with the Verlet algorithm. As an example we may look at the system including all planets in the solar system, i.e. $N = 9$. Results are listed table 4.3 for a simulation over 5000 years. We see that while the kinetic and potential energy changes considerably throughout the simulation, the total mechanical energy stays fairly constant (down to about 0.1% accuracy).

Velocity Verlet	Initial value [kg AU^2/yr^2]	Final value [kg AU^2/yr^2]
Kinetic energy	$8.0184014 \cdot 10^{27}$	$9.1166943 \cdot 10^{27}$
Potential energy	$-1.6901182 \cdot 10^{28}$	$-1.8005763 \cdot 10^{28}$
Mechanical energy	$-8.8827804 \cdot 10^{27}$	$-8.8890687 \cdot 10^{27}$

Table 4.3: Kinetic, potential and mechanical energy of the solar system (all planets + Pluto) at the initial and final time (after 5000 years) when using the velocity Verlet algorithm and time step $h = 2$ days.

4.3 Angular momentum conservation

We would also like to do a test on the angular momentum of the circular orbit Sun-Earth system (again implying initial conditions (41) and (42)). Angular momentum describes the rotational momentum of a system and, in correspondence with Newtons first law of motion, if there's is no external torque (force) acting on the rotation, the system will continue to rotate in the same way and thus conserving its angular momentum [1]. For the Sun-Earth system with a fixed Sun, we expect the angular momentum of the Earth, rotating around an axis centered on the Cun, to be conserved. As in section 4.2 for the conservation of energy, we check the value of the angular momentum vector

$$\mathbf{L}_E = m_E(\mathbf{r}_E \times \mathbf{v}_E) \tag{45}$$

before and after the simulation for, say 500 years where we initialized the Earth in $(1\text{AU}, 0, 0)$ and with velocity given by (42). And indeed when running the code for this system we observe conservation of angular momentum over the simulation as seen in table 4.4.

$\mathbf{L}$	Initial value [kg AU^2/yr]	Final value [kg AU^2/yr]
L_x	0	0
L_y	0	0
L_z	$3.7699112 \cdot 10^{25}$	$3.7699112 \cdot 10^{25}$

Table 4.4: Angular momentum $\mathbf{L} = (L_x, L_y, L_z)$ of the Sun-Earth system (fixed Sun) at the initial and final time (after 500 years) when using the velocity Verlet algorithm and time step $h = 1$ day.

Also worth noting is that since we initialized the earth purely in the x, y-plane it is expected to only see non-zero angular momentum in the z component due to the mathematical nature of the

cross product in (45).

As with the energy conservation we would like to add some words about the evolution of angular momentum for the case of the entire Solar System. In this case we would be dealing with $N = 9$ bodies all having their own angular momentum, and so the total angular momentum of the system would be the sum of all individual angular momentums, i.e.

$$\mathbf{L}_{tot} = \sum_{i=1}^{N} \mathbf{L}_i = \sum_{i=1}^{N} m_i(\mathbf{r}_i \times \mathbf{v}_i) \tag{46}$$

In table 4.5 we see that the angular momentum of the system is conserved in this case also, providing another indication that the implementation of the algorithm seems reasonable.

$\mathbf{L}_{tot}$	Initial value [kg AU^2/yr]	Final value [kg AU^2/yr]
L_x	$1.1473267 \cdot 10^{27}$	$1.1473267 \cdot 10^{27}$
L_y	$3.6237769 \cdot 10^{26}$	$3.6237769 \cdot 10^{26}$
L_z	$4.3873165 \cdot 10^{28}$	$4.3873165 \cdot 10^{28}$

Table 4.5: Total angular momentum $\mathbf{L} = (L_x, L_y, L_z)$ of the solar system (all planets + Pluto) at the initial and final time (after 500 years) when using the velocity Verlet algorithm and time step $h = 1$ day.

4.4 Timing

Doing timing tests of the code is not essential for checking that the algorithms produce the expected results, but is interesting in order to see how they perform against each other, how fast they are in general as well as making connections to the FLOP calculations talked about above.

Time step h [yr]	Euler [s]	Velocity Verlet [s]
0.1	0.00104	0.00092
0.01	0.0128	0.0110
0.001	0.1050	0.0956
0.0001	0.996	0.980
0.00001	8.922	8.84
0.000001	84.53	87.86

Table 4.6: Algorithm performance of the Euler versus the Velocity Verlet algorithm. Timing was done for the two-body Sun-Earth system with the Sun fixed at the origin and with an upper time limit of 100 years. Timing results are average of 5 runs for each h and only the execution of the algorithms is timed, i.e. initial setup, allocation etc. is not included in the timing.

In 4.6 there are results from timing of solving the Sun-Earth system for various values of the time step h. In particular h is decreased by one order of magnitude for each run, meaning the largest time step is 36 days and the smallest 32 seconds. The results are averages of doing the timing five times for each value of h. What is immediately surprising is that the Verlet algorithm matches the speed (and is even slightly faster) of the Euler method for all time steps except the last one. For $h = 10^{-6}$yr, Euler is faster, although not by much. One possible reason for Verlet matching the speed of Euler at large values of h could be due to overhead and setup costs overshadowing the actual time spent computing the solution. In general we should expect to see the Verlet method being slightly slower due to a few extra floating point operations at each time step. Even though the Verlet method involves more computations, it seems to hold up very well against the simpler Euler method for the parameters tested here.

Decreasing h by a factor of 10, as done in the table, implies an increase in number of time steps needed to reach the same upper time limit. This further implies increasing the number of total FLOP in the simulation by a factor of 10. Following the analysis in section 3.5 we would then expect the time usage to also increase by around a factor of 10 since the complexity of both the Euler and Verlet algorithms were found to be $\mathcal{O}(n)$, i.e. a linear dependence on the number of FLOP. And indeed this is what we observe in table 4.6. For every decrease of factor 10 in h (increase in factor 10 of n) we observe roughly an increase of factor 10 in the time usage. This applies to both algorithms.

4.5 Stability

Both the Euler and Verlet algorithm are expected to perform better as h decreases due to the error discussed at the end of sections 3.2 and 3.3. However, the mathematical error isn't the only reason we should not make the time step too large; we also need to consider numerical stability. A theoretical stability analysis is outside the scope of this project, but we may still test our implementation for unphysical behaviors as a function of the time step h.

Let's first consider the Euler method (and still for the circular orbit sun-earth system). As discussed above in section 4.2 there are inherent violations of energy conservation in the Euler method. Based on visual inspection, this seems to manifest itself in the form of orbits with constantly increasing radius, but while still remaining fairly circular. This seems to be the case irrespective of the time step, although smaller time steps decreases the magnitude of the effect. However, when using $h > 1.0$ day it seems as if the orbits begin tendencies of more elliptic shapes. Keep increasing h to above 10 days and the velocity of the Earth grows so big it even leaves the gravitational field of the Sun within 500 years, which clearly shouldn't happen. Increasing h even more makes the Earth leave the Sun earlier. Due to this "vague" instability behavior, we found it hard to conclude with an precise value of h making the solution go unstable. That being said, it seems like instabilities of the Euler method start appearing at approximately $h = 1$ day.

It's a different story for the Verlet algorithm. After experimenting with various time steps a quite conclusive answer was found. Almost exactly at time step $h = 60.95$ days, the calculations goes unstable and the solution blows up with the Earth getting ridiculous speeds and floating several thousands astronomical units away within 500 years. However below this value, the method seem to produce stable and good results.

For the Verlet algorithm we also tried to check the stability for the Sun-Earth-Jupiter system (Sun fixed) both for the realistic mass for Jupiter and for the case where Jupiter's mass is a factor of 1000 larger than in reality. For the former case the solution seems fairly stable up until about a time step of $h = 60$ days. For time steps above this we get similar phenomena as described in the paragraph above. However for the case with the massive Jupiter, the stability criteria seems much harder to satisfy. Here we needed a time step of around 0.01 days before the solution became satisfactory stable within the time frame (100 years) we were observing. The reason for this is not so clear, but one might speculate it is connected to the chaotic evolution of such a system. For the case where Jupiter mass is realistic, the system produces almost circular orbits and not much more is going on. For the massive Jupiter case the situation is entirely different and the trajectories are truly chaotic.

5 Results

5.1 The Earth-Sun System

As a starting step in building a complete numerical model of the evolution of our solar system's orbital motion, we want to investigate how our model functions for a system with only two bodies, the Sun and the Earth. We have fixed the Sun in the origin, as the motion of the Sun due to gravitational pull from the Earth is negligible. Figure 5.1 shows the resulting plot of the Earth's orbit when we utilize the velocity Verlet algorithm with a time step of one hour and initial position $\mathbf{r} = [1, 0, 0]$ (AU) and velocity $\mathbf{v} = [0, 2\pi, 0]$ (AU/Year) for the Earth.

We want to find the escape velocity, v_e, the Earth needs to escape the gravitational forces from the Sun both analytically and numerically in our two body system. To find v_e analytically we utilize equation (17) and plug in the mass of the Sun and the distance between the Earth and the Sun. This gives us that v_e must be equal to or larger than $\pm 8.9112 \frac{AU}{year}$.

To put this in other words, we have to multiply the velocity of 2π with a factor 1.41826 to arrive at the lower limit of the escape velocity for the Earth in the Sun-Earth system. Here we have assumed that the distance between the Earth and the Sun is exactly equal to 1.0 A.U.

Numerically we have found that by initializing the Earth with position $\mathbf{r}$ = [1AU, 0, 0] and velocity $\mathbf{v}$ [0, 2*pi, 0], the Earth will escape the Sun's gravitational influence if we multiply the initial velocity in the y-direction by a factor ~ 1.412. Here we have taken great effort in extending the simulation interval (sufficient increase in t_{final}) to be certain that the Earth does not reenter the Suns gravitational influence through a very large elliptical orbit. We have used $t_{final} = 40000$ years and a time step $h = 10$ days.

Our numerical results differs slightly from the analytical result. This is most likely due Taylor expansion truncation error or some loss of numerical precision, i.e. round-off errors. As we remember from table 4.2, the kinetic energy does not remain completely constant throughout the computation. As $E_k \propto v^2$, we would expect some imprecision in calculations of v after a considerable amount of time steps. The relative error is however of order, 10^{-3} which we find quite acceptable considering our rather large time step.

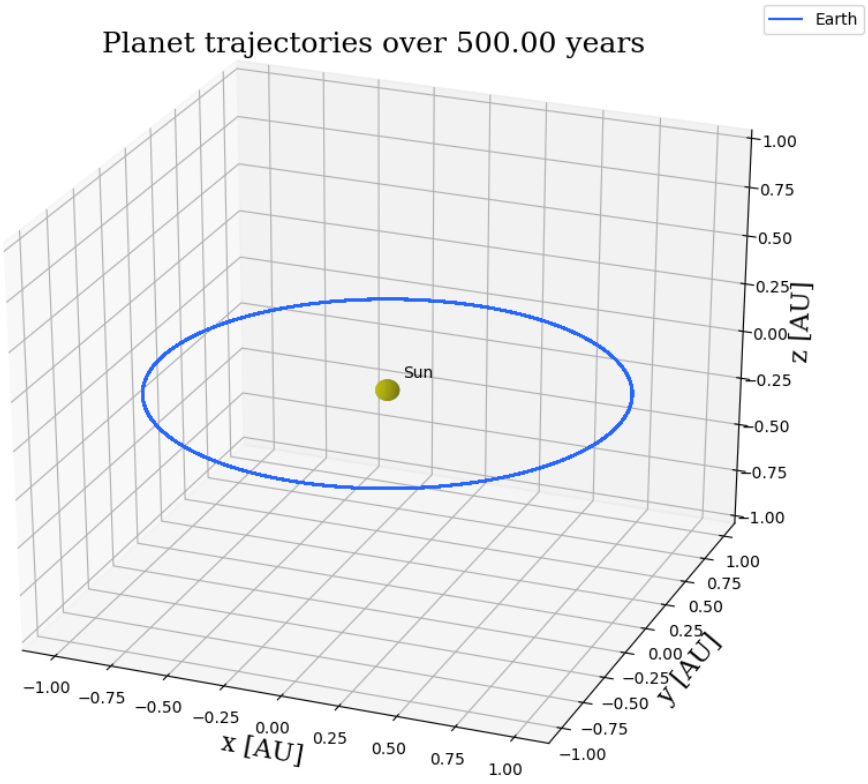

Figure 5.1: Figure shows the Earths orbit around the Sun over a time period of 500 years using the Verlet algoritm with a timestep $h = 1$ hour.

Next we want to investigate what would happen if we change the gravitational force given in equation (5) to;

$$F = \frac{GM_S M_E}{r^\beta} \tag{47}$$

where we let β vary in the interval [2,3]. Figure 5.2 shows the Earth's trajectory for $\beta = 2.5$ and for $\beta = 3$.

For $\beta = 2.5$ the Earth starts spiraling slowly outwards, but is still orbiting the Sun after 2000 years. As the gravitational force is proportional to $1/r^\beta$ and the distance between the Sun and the Earth is quite large, the gravitational force will decrease as β increases. This accounts for the outwards spiraling of the Earth in the left part of figure 5.2. Increasing β to 3 makes the Earth start spiraling quickly, and after 2000 years the Earth even seems to have escaped the gravitational pull of the Sun. As we remember from (17), the escape velocity of the Earth is given by

$$v_e \geq \sqrt{\frac{2GM}{r^{\beta-1}}} \tag{48}$$

Increasing β thus leads to a smaller escape velocity, eventually making it possible for the Earth to escape the Sun's gravitational pull for an initial speed of $v = 2\pi$ AU/year.

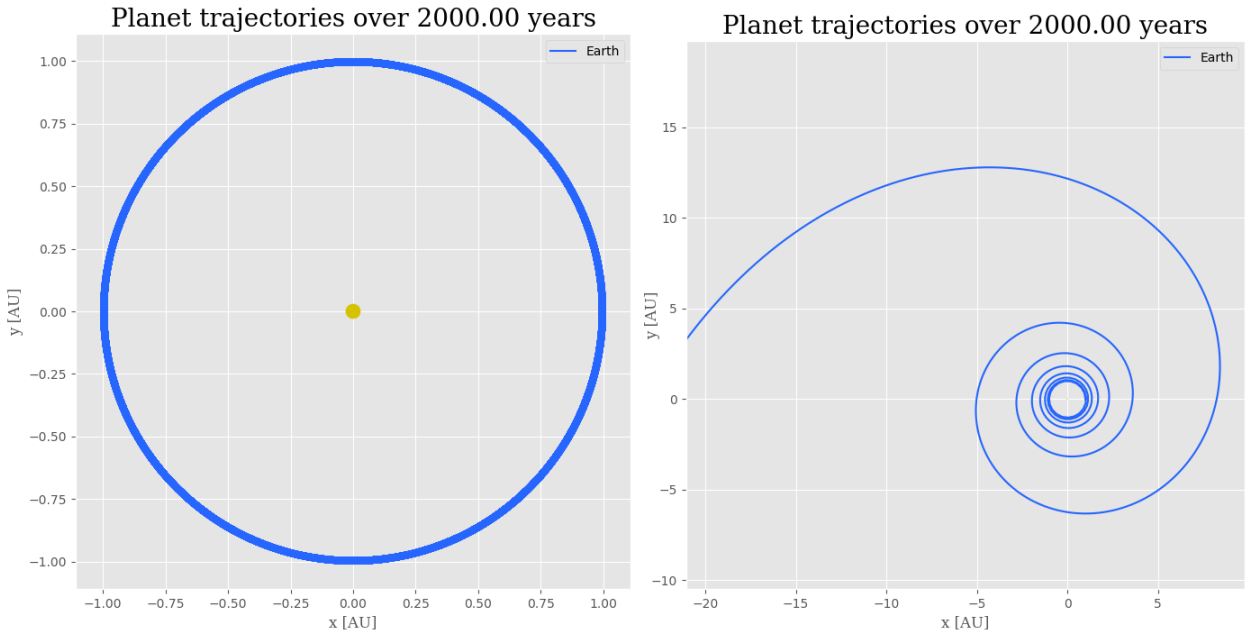

Figure 5.2: Time evolution over 2000 years for the sun-earth system using $\beta = 2.5$ (left panel) and $\beta = 3$ (right panel). The simulation was run with a time step of 5 days.

5.2 The Three Body System

So far we have seen that when utilizing the Verlet velocity solver, the Earth's motion is relatively stable and constant in time as we can see in figure 5.1.
Now we want to extend our Solar System to include a third body, namely Jupiter. We will continue to keep the Sun fixed in the center of our system and are interested in taking a closer look at how Jupiter, the largest planet in our solar system, influences the motion of the Earth. Plot 5.3 shows the orbits of Earth and Jupiter. Here we have used initial conditions for the planets velocity and position collected from Nasa's website: `https://ssd.jpl.nasa.gov/horizons.cgi?CGISESSID=173099c92a870c8502c247317aa137a6&s_type=1#top` from October 5.

Figure 5.3 shows that the Solar System remains stable when adding Jupiter, which is as we expected. The question now is: what would happen if we increased the mass of Jupiter by a factor of 10 or even a factor of 1000? The results are plotted in figure 5.4.

We see from figure 5.4 that changing Jupiter's mass by a factor of ten has little influence on the orbits, but it will of course influence the acceleration of Jupiter as we see from equation 8 because the factor $\frac{M_J}{M_S}$ will yield a larger value. This results in a slight increase in Jupiter's orbital velocity thereby decreasing Jupiter's turnaround. This effect will increase proportionally when we increase Jupiter's mass with a factor of one thousand.

From the right plot in figure 5.4 we see that changing Jupiter's mass by a factor 1000 has dramatic effects on the dynamics of our three body system. And this is no surprise, Jupiter's mass is about 1/1000 of the Sun's mass, so increasing Jupiter's mass by a factor 1000 leaves the mass of Jupiter at about the same size as the mass of the Sun. And as we see the Earth starts alternating between orbiting the Sun and Jupiter, but seemingly mainly becoming a moon of Jupiter. Think about

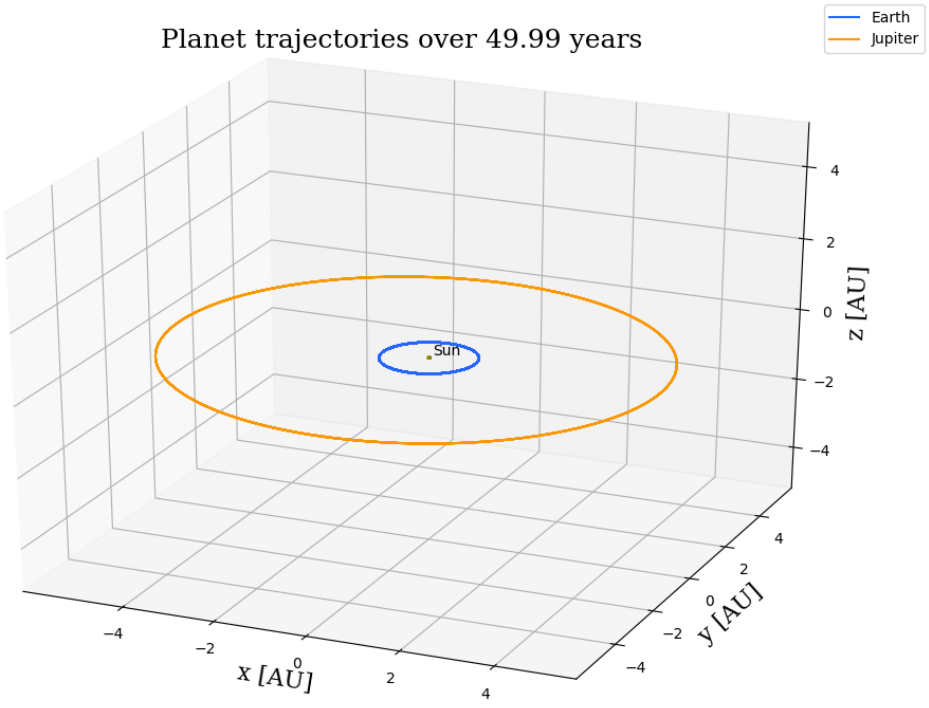

Figure 5.3: Figure showing the Earth's and Jupiter's orbits around the Sun, which is kept fixed at origin, over a time period of almost 50 years.

the consequences this would have for life here on Earth. An animation of the dramatic change is uploaded to YouTube at `https://youtu.be/sQuR-t25Zhw`.

5.3 The Solar System

To create a model of the whole Solar System, we have expanded the equation in (8) to include all planets. The Sun will also be treated as a planet in our further calculations. Here we have used initial conditions given by Nasa's website for 5. October 2017. As a start we wish to keep the Sun as the mass center and fixed in origin. Figure 5.5 shows the resulting orbits over a time period of 500 years where we have used a time step, h, of one day. Here it is possible to see that the bodies classified as planets (not Pluto though) in our solar system mainly orbit the Sun in the equatorial plane. This indicates that performing our simulation for only two spatial directions, the xy-plane, will result in a fairly good representation of our solar system and this will of course save the number of FLOPs needed, thereby reducing CPU time.

To make the calculations as realistic as possible within the limits of our approximation, we want to give the Sun an initial velocity so that the total momentum of our system is exactly equal to zero as given by (15). Doing this will prevent the solar system from getting a net velocity in some direction. We start our calculations with the Sun in the origin, but now with the initial velocities that reduces the total momentum of the system to zero. Figure 5.6 shows the resulting motion of the Sun when it is given an initial velocity and is influenced by gravitational forces. We see that the Sun is orbiting in close proximity to the origin and well within (roughly one fifth of) the

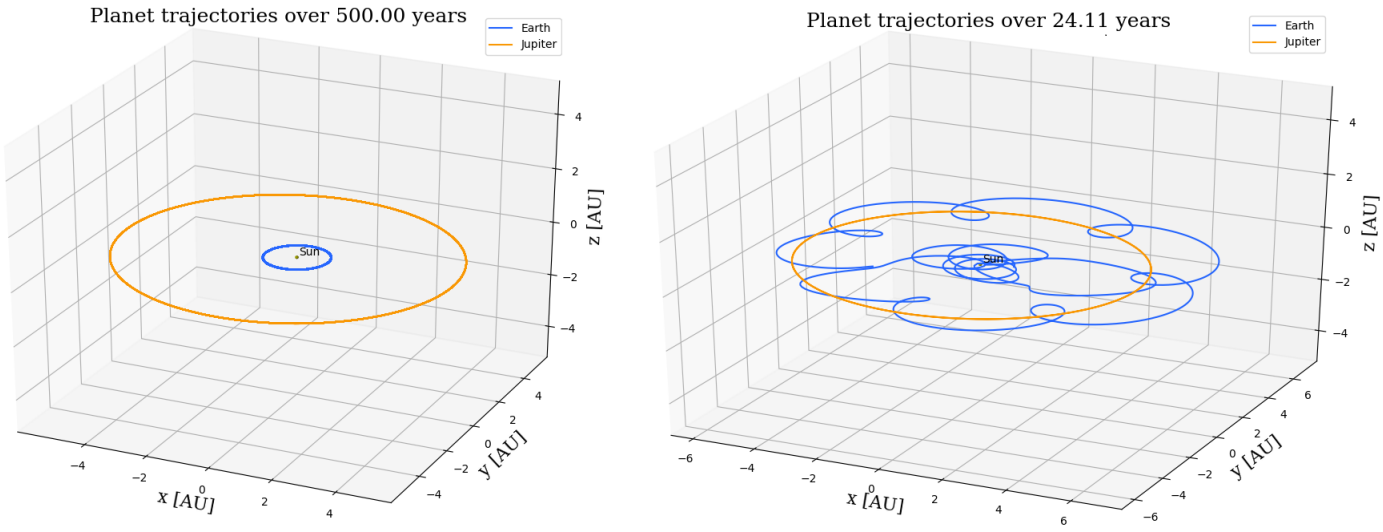

Figure 5.4: Figure showing the resulting orbits of Earth and Jupiter when we change Jupiter's mass by a factor 10 (left) and 1000 (right). Here we have used a timestep, h, equal 3 min. We see clearly from the right plot that changing Jupiter's mass by a factor 1000 has dramatic consequences for the dynamics of our three body system. The reason for 24 years in right panel is for readability and to avoid too much clutter.

radius of the Sun (about 0.05 AU). Studying the motion of the Sun, it would thus not seem to have circular orbits, but rather to 'wobble' around about its center of mass.

Our observation that the Sun only orbits well within its real radius, indicates that putting the Sun at rest in origin is a good approximation when we are only interested in the dynamics of the orbiting bodies in our solar system. Especially since we have already made the quite crude simplification where we treat all the bodies as point particles. A further argument is that when we find the mass centers position, the returning coordinates are quite close to the origin and within the radius of the Sun, with $r_{MC} \sim [-0.0023, -0.0055, 0.00013]$. We also know that the Sun contains 99.86% of our solars systems total mass [3]. So keeping the Sun fixed in origin looks like an attractive option as this lets us see the planets orbits (here also Pluto is included) clearly and saves us FLOPs and CPU time.

An animation of the full solar system (sun not fixed) simulation over 250 years can be found at `https://youtu.be/Gi8v0cHA5IM`. We also wanted to explore the case of a massive celestial body (perhaps a large asteroid of some sort) with mass of 1/20 of that of the sun entering our solar system with some velocity. We observed that the solar system responded quite dramatically and the gravitational pull towards the asteroid disturbed the entire solar system. This gives an indication of what might happen if such an event were to occur. An animation can be seen at `https://youtu.be/PtQ8x0xHPpQ`.

5.4 The Perihelion of Mercury

So far, our simulations has not included any corrections from general relativity.

We now want to compute the perihelion angle of Mercury with and without the general relativity correction to the law of gravity (18). The result is presented in table 5.1 , where we used a

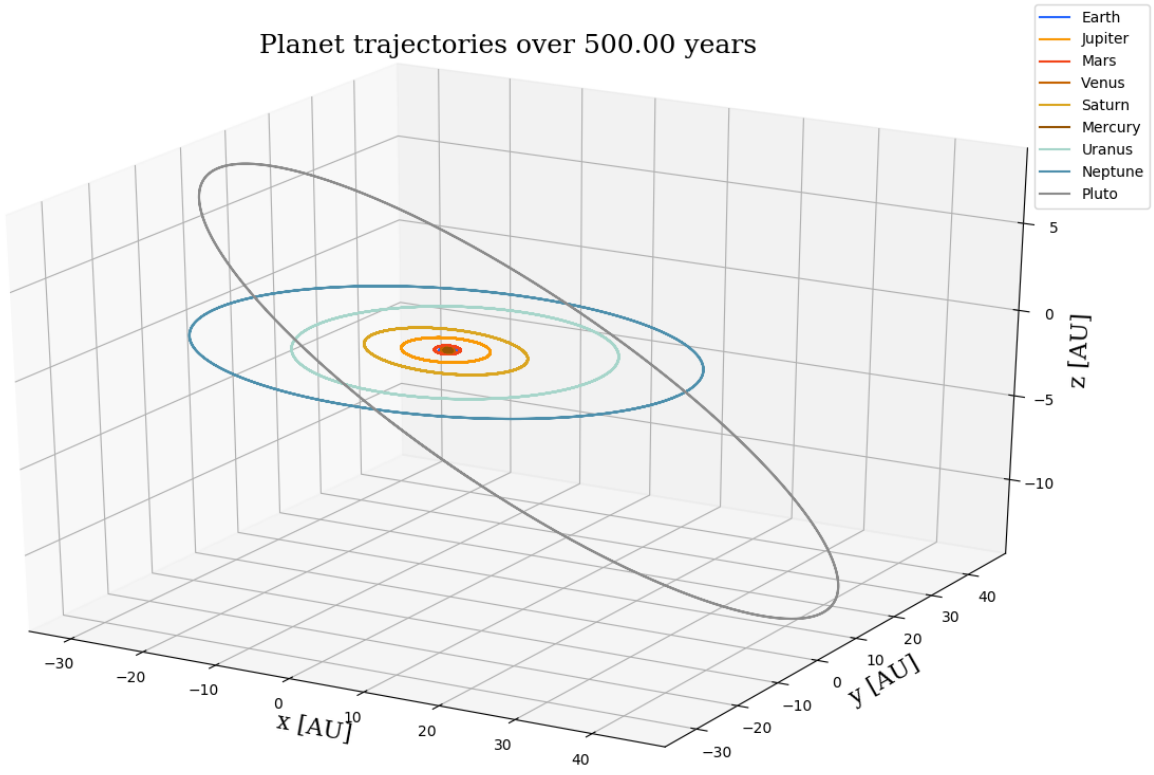

Figure 5.5: Figure shows the evolution of our solar system, included Pluto, when we use the Sun fixed in origin throughout our simulation (moving sun gives almost exact same figure).

With general relativity	Newtonian
43.025	0.6032

Table 5.1: The perihelion angle (in arc seconds) of Mercury after 100 years with and without general relativity correction to the law of gravitation. We have used a time step $h = 8.64$ seconds

time step of 8.64 seconds. As can be seen in the table, the computed perihelion angle when general relativity corrections are taken into account, is quite close to the observed angle of 43". The perihelion angle when using only Newton's law of gravitation, is about two orders of magnitude smaller than the observed one. It seems that the perihelion angle of Mercury cannot be explained using only Newton's law of gravitation, but by also including general relativity.

6 Concluding remarks

Science has made many groundbreaking discoveries through the ages. Among these, the discovery of the Sun as the center of the Solar System is perhaps the most famous. One can only imagine how difficult and extremely time consuming it must have been to find an approximate model of the motion of the planets back then. Fortunately, technology has moved a lot forwards since then, and it is now possible to find approximate solutions to differential equations in a matter of minutes or

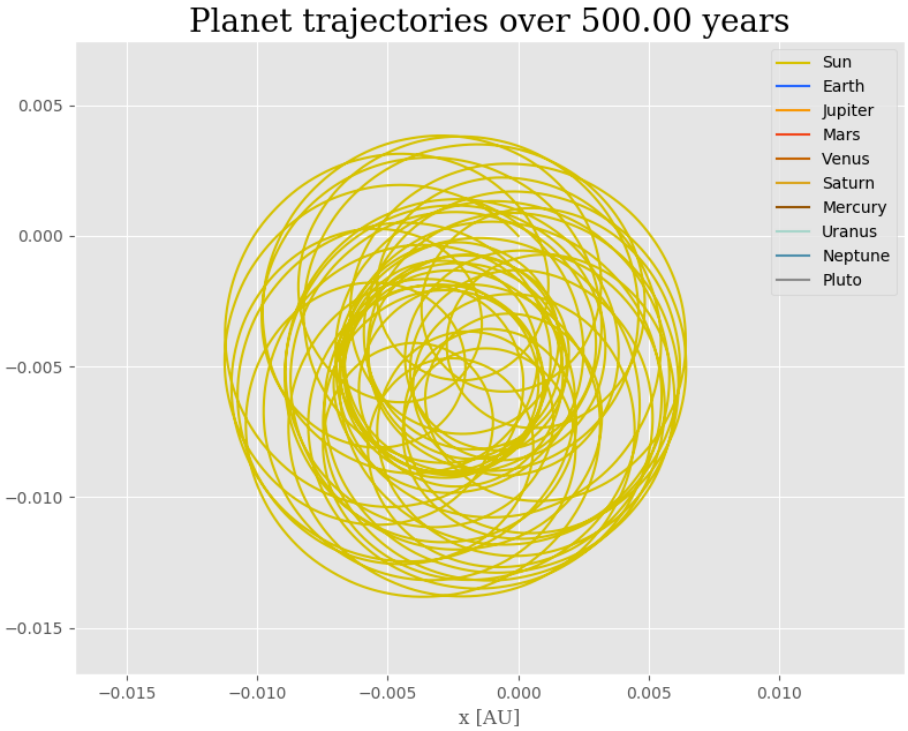

Figure 5.6: Figure shows the resulting motion of the Sun when we give the Sun an initial velocity that makes the total momentum of the system equal to zero. Here we have used a timestep, h, equal to 0.1 days and zoomed in on the central areas of our solar system to better visualize the Suns motion.

even seconds (depending of course on the chosen number of time steps) by solving the equations numerically.

In this project our goal was to build a model of the Solar system using numerical methods, namely Euler and Velocity Verlet. Using known physical properties of the Solar System allowed us to compare the exactness of the methods. In section 4 we saw that the Verlet method is superior to the Euler when it comes to conservation of energy, and that it also to a high degree conserves angular momentum. Table (4.6) also shows that the difference of time used by both methods is quite small, leading us to conclude that Velocity Verlet is by far the superior method. In addition, as discussed in section 3, the Verlet algorithm is a higher order numerical scheme.

Throughout the project we have also experimented with changing different parameters, most notably the mass of Jupiter and the dependency of the radius in the gravitational force. Figure 5.2 and figure (5.4) show how dramatic the consequences of this would be. Most of us take the stability of stability of the Solar System as granted (overlooking of course the eventual 'death' of the Sun). Our results in this project show how amazing it is that the planets orbit the Sun in such an ordered manner, as small changes to velocities, mass or the gravitational law, severely alters

the motion.

Were one to work further on this project, it would be interesting to take the diminishing mass of the Sun into consideration, and study what effect this would have on the motion of the planets after millions of years. Including the moons of the different planets would also be interesting, as this would lead to an even more realistic model. With more time we would also have liked to perform more timing tests for varying number of bodies N to confirm the N^2 cost dependency on the number of bodies, but also to observe that the difference between Euler and Verlet time usage would shrink with increasing N. And although we were pleased by the precision of the Velocity Verlet algorithm (especially considering the time efficiency of the method) it might be worthwhile to look into a numerical method of a higher order error term, e.g. Runge Kutta 4, to check if this would lead to a considerably improvement of the precision.

References

[1] Encyclopædia britannica. https://www.britannica.com/science/conservation-law. Accessed: 21.10.2017.

[2] slate.com. http://www.slate.com/blogs/bad_astronomy/2014/07/14/solar_wind_versus_fusion_how_does_the_sun_lose_mass.html. Accessed: 17.10.2017.

[3] Space facts. https://space-facts.com/the-suns-mass/. Accessed: 23.10.2017.

[4] Morten Hjorth-Jensen. Computational physics. University Lecture Notes, 2015.

[5] Morten Hjorth-Jensen. Project 3. University Project Decription, 2017.

[6] Isaac Newton. Philosophiae naturalis principia mathematica. Daniel Adeen, 1687.

[7] Juliet Brosinger Thomas Griffiths. *The Physics of Everyday Phenomena*. McGraw-Hill Education, 2015.