FYS3150 - Project 3

Daniel Heinesen, Gunnar Lange

October 25, 2016

Abstract

We present an investigation into two different numerical algorithms for solving systems of coupled ODE's by simulating a many-body problem - our solar system, with varying levels of complexity. We present the forward Euler algorithm and the Velocity-Verlet algorithm, and show that, when computation time is not a vital concern, the Velocity-Verlet algorithm outperforms the Forward Euler algorithm. We also investigate the appropriate time step to model our system over long time scales, and find that 10^5 steps per year gives us the desired combination of numerical precision and speed. Finally, we show that our program largely reproduces some known analytic results, namely conservation of energy/angular momentum, the escape velocity of earth from the sun and the perihelion precession of mercury predicted by general relativity. All data, scripts, benchmarks and plots can be found on our Git-page[1]

[1] https://github.com/dulte/Comp-Phys/tree/master/Project3Final

Contents

1 Introduction

The n-body problem has been a recurring theme in many physics publications, up to this day (see for example [Yang, 1967] or [Diacu et al., 2012]), due to its complexity and its applicability in as varied fields as statistical dynamics and astrophysics. Analytic solutions of these problems are notoriously hard to come by and are, in most situations, unattainable. Therefore, these problems are almost always solved by use of numerical techniques.

We present two different numerical methods for solving the coupled ODE's resulting from Newton's law of gravitation - Euler's method and the velocity-Verlet method. We begin with a theoretical introduction to both algorithms, as well as to the equations which we wish to solve. We then discuss some checks which our simulations should pass, if they are correctly implemented. Specifically, we look at conservation of energy and angular momentum. We continue by looking at some results which have well-known analytic solutions - specifically the escape velocity of earth from the solar system and the perihelion precession of mercury predicted by general relativity. We compare our model to these results, as another check for consistency. We also investigate what parameters to use in our algorithms to get consistent numerical results.

We then present the methods we used to implement these algorithms, as well as presenting the various models we investigated. Subsequently, we present our results and discuss these. Finally, we give a brief look ahead at possible further work that could be done in this area, based on the results we obtain.

2 Theoretical model

In this section, we introduce the quantities and equations required to model the solar system. We begin by introducing the relevant equations, and subsequently discuss how to solve these numerically. We continue with a brief discussion of the units used, followed by a discussion of tests that we can implement to test our numerical results. We conclude with a brief diversion to relativistic effects in the solar system.

2.1 Newtonian Gravity

Newton's general law of gravitation, which describes the force of gravity between two objects, is given by:

$$\vec{F}_G = \frac{Gm_1m_2}{r^3}\vec{r} \tag{1}$$

Here F_G is the force of gravity on the first mass, G is the universal gravitational constant, m_i are the masses of the two objects, and $\vec{r}$ is the vector pointing from m_1 to m_2. This law can be combined with Newton's second law of motion, to give a set of coupled differential equations for the coordinates:

$$\begin{aligned} \frac{d^2x}{dt^2} &= \frac{F_{G,x}}{m_1} \\ \frac{d^2y}{dt^2} &= \frac{F_{G,y}}{m_1} \\ \frac{d^2z}{dt^2} &= \frac{F_{G,z}}{m_1} \end{aligned} \tag{2}$$

Where $F_{G,x}, F_{G,y}$ and $F_{G,z}$ are the components of the gravitational force in the x, y and z directions respectively. These equations can be reduced to a set of first-order equations by introducing the velocity components, v, such that:

$$\frac{dx}{dt} = v_x, \qquad \frac{dv_x}{dt} = \frac{F_{G,x}}{m_1} = a_x(x, y, z) \tag{3}$$

And equivalently for the y and z direction. This gives six coupled, first-order, linear, differential equations.

2.2 Implementing the initial conditions

To find a unique solution of the set of equations in section 2.1, we require some initial conditions, i.e. some $x(t = 0)$, $v_x(t = 0)$, and equivalently for the y and z direction. For our first model (described in section 3.1.1) we will, for simplicity, let the earth be 1 AU away from the sun, with an initial velocity that ensures a circular orbit. We show in appendix A that this results in the requirement that the initial velocity be given by:

$$v_0 = \sqrt{\frac{GM_\odot}{r}} \tag{4}$$

Where $M_\odot$ is the mass of the sun and r is the distance to the sun.

For the subsequent models, with multiple planets, we will use the initial positions and velocity provided on NASA's webpages[2], letting $t = 0$ correspond to the 18th of October 1977 (right before the launch of Voyager 2, as an arbitrary start point).

We want to ensure that the center of mass of our solar system is stable at the origin. On NASA's webpage, we can choose the coordinate origin, however, it is not entirely clear if the velocities have been adjusted so that the center of mass is at rest. Therefore we adjusted the position and velocity of the sun to accommodate this. The position vector of the center of mass, $\vec{R}_{COM}$, is found from:

$$\vec{R}_{COM} = \sum_i m_i \vec{r}_i$$

The sum is over all bodies. $\vec{r}_i$ is the position vector of body i, and m_i is its mass. As the sun is by far the heaviest mass in the system, we can ensure that $\vec{R} = \vec{0}$, whilst still keeping the sun close to the origin, by letting the position vector of the sun, $\vec{r}_\odot$ be given by:

$$\vec{r}_\odot = -\frac{\sum_j m_j \vec{r}_j}{M_\odot}$$

Where the sum is over all bodies *except the sun*. We can do the same thing for the velocity of the center of mass, $\vec{V}_{COM}$, which is given by:

$$\vec{V}_{COM} = \sum_i m_i \vec{v}_i$$

Where $\vec{v}_i$ is the velocity of body i. We can then let the velocity of the sun, $\vec{v}_\odot$ be:

$$\vec{v}_\odot = -\frac{\sum_j m_j \vec{v}_j}{M_\odot}$$

Where the sum is again over all bodies *except* the sun. This ensure that the center of mass of our system is stationary at the origin.

2.3 Discretizing the equations

We will solve the equations in 2 numerically. Therefore we require discretized versions of the these equations. We want to simulate from $t = t_0$ to $t = t_f$, and choose N evenly spaced points in this interval. This gives us a timestep, h, of:

$$h = \frac{t_f - t_0}{N}$$

[2] http://ssd.jpl.nasa.gov/horizons.cgi#top

Let now $t_i = a + ih$, where i goes from 0 to N. Furthermore, let $x_i = x(t_i)$ and $v_{x,i} = v_x(t_i)$, and equivalently for the y and z direction. Finally, discretize the acceleration as $a_{x,i} = a_x(x_i, y_i, z_i)$. This allows us to discretize the derivatives in equation 3. We will do this in two separate ways; with the forward Euler algorithm and with the Velocity-Verlet algorithm.

2.3.1 Euler's forward method

Euler's forward algorithm amounts to discretizing the derivatives from section 2.1 as:

$$\frac{dg}{dt} \approx \frac{g_{i+1} - g_i}{h} \tag{5}$$

This approximation is based on Taylor-expanding the function $g(t)$ to the second order, as explained by [Zeltkevic, 1998]. Inserting this approximation for the derivatives in equation 5, and rearranging some of the terms, gives:

$$\begin{aligned} v_{x,i+1} &= v_{x,i} + ha_{x,i} \\ x_{i+1} &= x_i + hv_{x,i} \end{aligned} \tag{6}$$

And again equivalently for the x and y direction. This is Euler's forward algorithm. Combining this with the initial conditions, $\vec{v}(t_0) = \vec{v}_0$ and $\vec{x}(t_0) = \vec{x}_0$, makes it possible to solve the equations of Newtonian gravity. Closer inspection (through Taylor expansion, see for example [Ledder, 2004]), reveals that the local error is proportional to h^2, whereas the global error is proportional to h. Thus, whilst this is a straightforward algorithm, the errors quickly accumulate. It is therefore of interest to investigate a slightly improved algorithm:

2.3.2 Velocity-Verlet Algorithm

The Velocity-Verlet algorithm is based on the idea of Taylor-expanding the acceleration, in addition to the velocity and position. The details can be found in appendix B, but the method produces the following equations:

$$\begin{aligned} x_{i+1} &= x_i + hv_{x,i} + \frac{h^2}{2}a_{x,i} \\ v_{x,i+1} &= v_{x,i} + \frac{h}{2}\left(a_{x,i+1} + a_{x,i}\right) \end{aligned} \tag{7}$$

And similarly for the x and the y equations. Note that a_{i+1} depends on x_i, y_i and z_i, and therefore all three position coordinates must be computed before we can compute the velocity. As shown in our analysis in appendix B, the local error is proportional to h^3, and in turn the global error is proportional to h^2, as discussed by [Mazur, 1997].

Notice that $a_{x,i+1}$ can be reused in the next step, so this method, when compared to the Forward Euler algorithm, does not incur any additional calls to the acceleration function in the main loop. The method only costs a few more FLOPs;

- We must perform two additional multiplications for each dimension, specifically $(h^2/2) \cdot a_{x,i}$ and $(h/2) \cdot a_{x,i+1}$
- We must perform two further additions per dimension, namely adding $(h^2/2)a_{x,i}$ and $(h/2)a_{x,i+1}$ to respectively x_{i+1} and $v_{x,i+1}$.

This gives four additional FLOPs per timestep and dimension, i.e. 12 FLOPs per timestep. Thus if we have N points in the main-loop, this gives an additional $12N$ FLOPs. Note that $h^2/2$ and $h/2$ can be precomputed, and therefore do not contribute to the number of FLOPs in the main loop.

2.4 Choice of units

As we will be modelling our solar system, a natural unit of length is the average distance between the earth and the sun, the astronomical unit, AU. In SI-units, $1\text{AU} \approx 1.496 \cdot 10^{11}\text{m}$. As the most massive object in the solar system is the sun, we choose one solar mass, $M_\odot$ as our mass unit. In SI units, $M_\odot \approx 1.99 \cdot 10^{30}$. We will use the mean tropical year, which is about 365.24 days of 864000 seconds ([Weisstein, 2007]) as our unit of time. This choice of units has one considerable advantage - it simplifies the expression for the gravitational constant, G. As shown in appendix A, with this choice of units the gravitational constant becomes:

$$G = 4\pi^2 \text{ AU}^3\text{yr}^{-2}\text{M}_\odot^{-1}$$

As discussed in section 2.7, we will also need the speed of light. With our units, this is given by:

$$c = 63\ 198 \text{ AU/yr}$$

2.5 Conserved quantities in the system

To check the consistency of our results, we will investigate whether certain quantities, which we expect to be conserved, are indeed conserved.

2.5.1 Conservation of mechanical energy

As there are no external forces in the system, we expect the mechanical energy to be conserved. There are two kinds of energy to consider in our system: kinetic and potential energy. The kinetic energy of a single body is simply given by:

$$E_k = \frac{1}{2}mv^2$$

Where m is the mass of our celestial body and v is the absolute value of its velocity. This is the standard formula for the kinetic energy. The potential energy can be found by integrating the force, choosing infinity as a reference point, i.e. $U(\infty) = 0$:

$$U = -\int_r^\infty \frac{GMm}{r'^2}dr' = -\frac{GMm}{r}$$

This is the potential energy between the bodies. Thus the total energy (which should be conserved) is given by:

$$E_{tot} = \sum_i \frac{1}{2}m_i v_i^2 - \sum_{i<j}\sum_j \frac{Gm_i m_j}{r_{ij}} \tag{8}$$

Where i runs over all the bodies and r_{ij} is the norm of the distance between body i and j. Note that the second sum of the potential energy only goes to $i < j$, to avoid double-counting the energy.

2.5.2 Conservation of angular momentum per unit mass

As there is no external force, there is also no external torque. Therefore, the net angular momentum per unit mass should be conserved in our system. The angular momentum per unit mass of a body is generally given by:

$$\vec{l} = \vec{r} \times \vec{v}$$

Where $\vec{v}$ is the velocity of the body, and $\vec{r}$ is the distance to the center of mass. The total angular momentum per mass is then is then:

$$\vec{l}_{tot} = \sum_i \vec{r}_i \times \vec{v}_i \tag{9}$$

Where the sum is over all bodies.

Equations 8 and 9 both describe quantities which should be conserved. Thus, an important consistency check for our code, is to test if these quantities are indeed (approximately) constant throughout our simulation.

2.6 Escape velocity

Another way we can check the consistency of our results is to investigate the escape velocities of the planets. This can be done by equating the potential energy to the kinetic energy. This is a difficult problem for a many-body problem. Therefore, we will only implement this in the two-body case, with the sun and the earth, considering the escape velocity of the sun. Equating potential and kinetic energy of the earth gives:

$$\frac{1}{2}v_{esc}^2 = \frac{GM_\odot}{r}$$

Solving gives:

$$v_{esc} = \sqrt{\frac{2GM_\odot}{r}} \tag{10}$$

Which is the escape velocity of the earth, if we ignore all other planets. In our units, $v_{esc} = 2\sqrt{2}\pi$ AU/yr ≈ 8.89 AU/yr. We can experiment with the initial velocity of earth, and check if the velocity at which earth manages to escape indeed equals this quantity. This gives us another way to check the consistency of our simulations.

2.7 The perihelion precession of Mercury

One of the tests for Einstein's theory of relativity is the perihelion precession of Mercury. Over time, the perihelion (the point where Mercury is closest to the sun) changes. This was initially thought to be due to the gravitational attraction of other planets, but the effect was too large.It turns out that this effect is due to general relativity, which predicts that gravity warps spacetime. This boils down to modifying the gravitational force to:

$$\vec{F}_G = \frac{GM_\odot M_{\text{Mercury}}}{r^3}\left[1 + \frac{3l^2}{r^2c^2}\right]\vec{r} \tag{11}$$

Where $l = |\vec{l}|$ is the absolute value of the angular momentum per mass of the body the force acts on, and c is the speed of light. This assumes that all other effects, such as gravitational attraction from other planets, have been ignored. In this case, experimental data shows that the perihelion angle precesses $43'' = 0.2085$ mrad per 100 earth years. We will investigate if this relativistic correction can correctly predict the precession, by comparing the motion of Mercury (removing all other planets) with Newtonian gravity and with the relativistic expression for gravity.

3 Methods

In this section we give an overview of the methods that we employed to solve the problems outlined in the previous sections. We discuss the models we have implemented, the numerical implementation of these models and the equations from the previous section and finally the implementation of initial conditions.

3.1 Models

We will implement four different models, to test our simulations under multiple circumstances.

3.1.1 First model - The earth-sun system

We begin with a simplified model of our solar system, containing only the earth and the sun. We simulate this using both the Euler forward algorithm and the Velocity-Verlet algorithm. This enables us to compare these methods.

3.1.2 Second model - The three body problem

We slightly expand our system by also including the most massive planet - Jupiter. We can then investigate how large the effect of Jupiter is on the orbit of the earth. We also make this slightly more extreme, by increasing the mass of Jupiter by a factor 10 and a factor 1000. In the latter case, we will be approaching a binary star system, as the mass of Jupiter is about 1000 times less than the mass of the sun. Thus we expect a fairly chaotic behavior, which will allow us to study the stability of our Verlet solver.

3.1.3 Third model - Entire solar system

We then include all eight planets in the solar system, as well as Pluto for historical reasons. We also include all interactions between the planets,and simulate for 300 years (to ensure one complete orbit of Plute). We use the Velocity-Verlet method to solve this numerically. This gives us the opportunity to study the stability of the Verlet-method on larger scales, both temporally and spatially.

3.1.4 Final model - The mercury-sun system

Finally, we model only mercury and the sun, to investigate the perihelion precessions of mercury described in section 2.7. Here we use the Verlet-solver, and implement both the Newtonian and the relativistic expressions for the gravitational force.

3.2 Some interesting features of our numeric scripts

3.2.1 Data flow

To ease the flow of data, we implement a class structure to our code. We have three interacting classes:

1. Particle class - a class which stores all variables related to a single planet.
2. System class - A class which stores all and computes all species related to the entire system.
3. ODESolver class - A class which computes acceleration based on the algorithms detailed in section 2.3

Finally we also implement a tailored vector class, which implements some key features of vector algebra needed to solve the equations in section 2.1, such as addition, multiplication, norms, inner products and the cross product.

3.2.2 Retrieving the initial conditions

As described in section 2.2, the initial positions and velocities of all planets can be retrieved from NASA's webpages. It is possible to establish a telnet connection to these websites, and retrieve the data automatically. We took inspiration from [Rob, 2013] when implementing this.

With the class structure and the initial conditions in place, we can then simply initialize the solar system by a code akin to the one shown below:

```
infile=open('Planet_initial_data.txt')
solar_system=System()
for line in infile: //Read in planet data
        particle_name, initial_pos, initial_vel, mass=line.split()
        if particle_name is not "Sun": //So that COM is at x0=0, v0=0
                solar_system.create_particle(initial_pos, initial_vel, mass)

total_position=sum(particle_mass*particle_position)
total_momentum=sum(particle_mass*particle_velocity)
sun_pos=-total_position
sun_vel=-total_momentum/mass_of_the_Sun

solar_system.create_particle(sun_pos, sun_vel, mass_of_the_Sun)
```

And then we can simply solve the differential equations by calling:

```
steps_per_year=1e7
years=10
timestep=1/steps_per_year
solver=ODESolver(timesteps)
N=steps_per_year*years

for i in range(0, years, N):
        solver.Velocity_verlet_one_step //update parameters in solar system
        solar_syststem.dump_positions_to_file
```

3.3 Checking the consistency of our results

3.3.1 Checking conservation of energy and angular momentum per unit mass

We check conservation of energy for every 100^{th} timestep in all of the models described in section 3.1, except for the case with relativistic gravity and the case where Jupiter is 1000 times heavier in reality. We neglect the first one, because our background in general relativity is relatively limited, we are unsure whether or not energy and angular momentum per mass are conserved in this case. We neglect conservation of energy for the case where Jupiter is 1000 times heavier because, as we will show, Jupiter actually crashes into the earth in this case. This causes a breakdown in our numeric algorithm, as there are massive forces at work.

Because magnitudes of deviations are notoriously difficult to interpret, we will instead record and analyze the relative error at a time, t, defined as:

$$\epsilon_{rel} = \frac{E(t) - E_{\text{initial}}}{E_{\text{initial}}} \tag{12}$$

Where E is the energy. We employ the same expression to check conservation of angular momentum per unit mass.

3.3.2 Checking escape velocity

We found an expression for the escape velocity in section 2.6. To show that this indeed gives the required behavior, we place the earth at $x = 1$ AU and give it an initial velocity in the y-direction. We let this initial velocity be just above and just below the escape velocity, and check whether or not we see the earth escape.

3.3.3 Checking the precision of the Velocity-Verlet method

As we mentioned in section 2.3, the Velocity-Verlet method is significantly more precise than the forward Euler algorithm. We therefore investigate the stability of the Velocity-Verlet method for a variety of timesteps, and simulate for two years, to investigate when the two graphs of the two years are acceptably close. This will enable us to choose a sensible time step for our simulations.

3.4 Investigating the perihelion processions of mercury

To find the perihelion processions of mercury, we must develop an algorithm which checks for a minimum of the distance between the sun and mercury. We do this by storing the absolute value of the position vector for three subsequent time steps, to then check if the value in the middle of these three is lower than the two others. If this is the case, we save it as a perihelion passing, and compute the angle. This is illustrated in the pseudo-code below:

```
previous_r =0
previous_previous_r=0

for k in range(0, time, N):
    r_vec = mercury_position - sun_position
    r = r_vec.length();
    if previous_r < r and previous_r < previous_previous_r:
        theta = arctan2(r_vec.y,r_vec.x);   //Save as perihelion passing
```

3.4.1 Unit test

We test the implementation of our code by including a unit test which cancels the program once the relative error of the energy in larger than 10^{-6}. This ensures that we do not get an utterly non-physical system.

4 Results

In this section we present the orbits of the different models, as well as quantities such as conservation of energy/momentum and escape velocity/perihelion precession. Our calculations were all done in 3D, but we include only 2D plots, as these are easier to interpret. The orbits of all planets, except Pluto, are almost two-dimensional anyway, as can be seen from the solar_system_3D.gif file on our GIT-page[3]. We postpone an extended discussion of our results until section 5.

4.1 Results from the first model - Earth-Sun

4.1.1 Position

The position of the earth and the sun, as well as a zoomed-in version of the plot, are shown in the figures below:

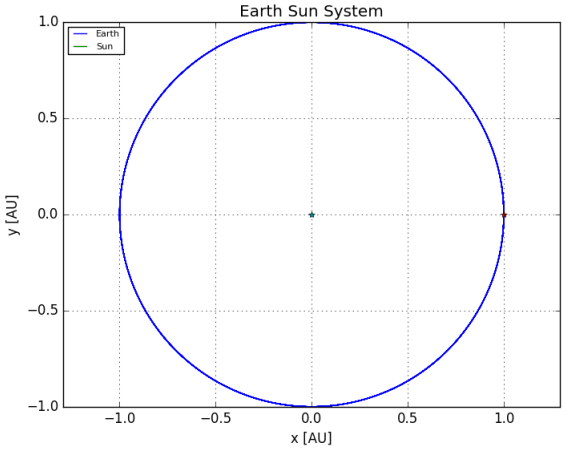
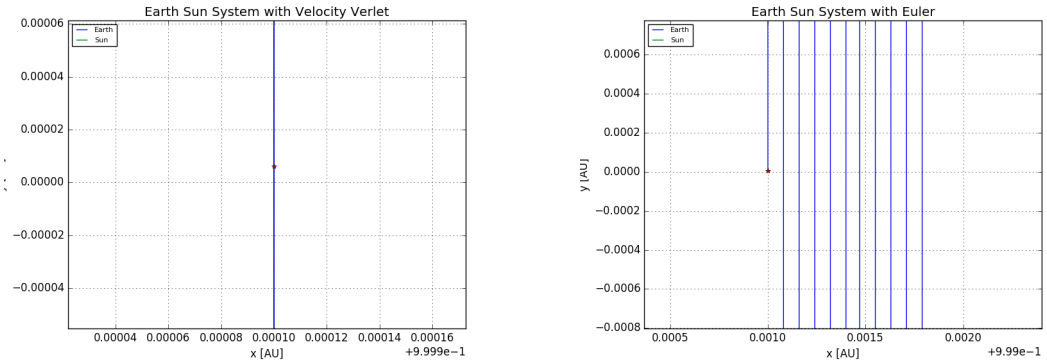


Figure 1: The earth-sun system, simulated for 10 years, with 10^5 timesteps per year. We are using the Velocity-Verlet method, though there is no visible difference between the Velocity-Verlet and the Euler method at this scale.



(a) Velocity-Verlet algorithm

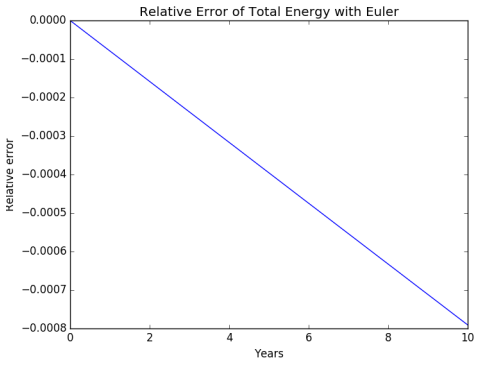
(b) Euler's forward algorithm

Figure 2: Zoomed-in version of the figure above with two different solution methods. Note that the scale on subplot *a* is ten time smaller than the scale of subplot *b*.

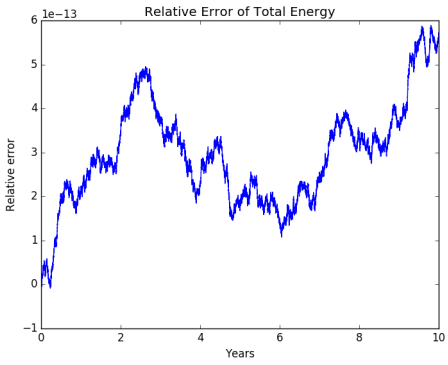
[3] https://github.com/dulte/Comp-Phys/tree/master/Project3Final

4.1.2 Conservation of energy and angular momentum per unit mass

A plot of the relative error in the total energy and the total angular momentum per unit mass for the Velocity-Verlet and the Euler algorithm is shown below:

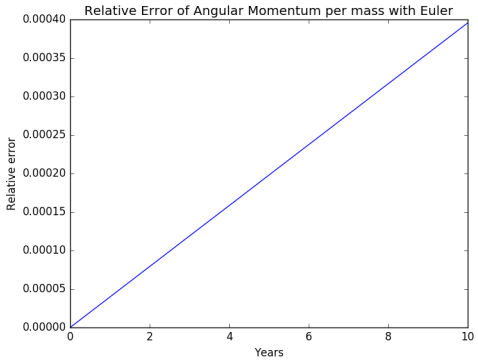


(a) Forwad Euler algorithm

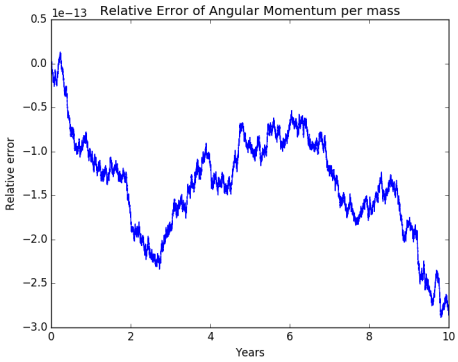


(b) Velocity-Verlet algorithm

Figure 3: Variations in the error of the total energy as a function of time for 10 years of orbit of the earth-sun system. Note the scale on the axis.



(a) Forwad Euler algorithm

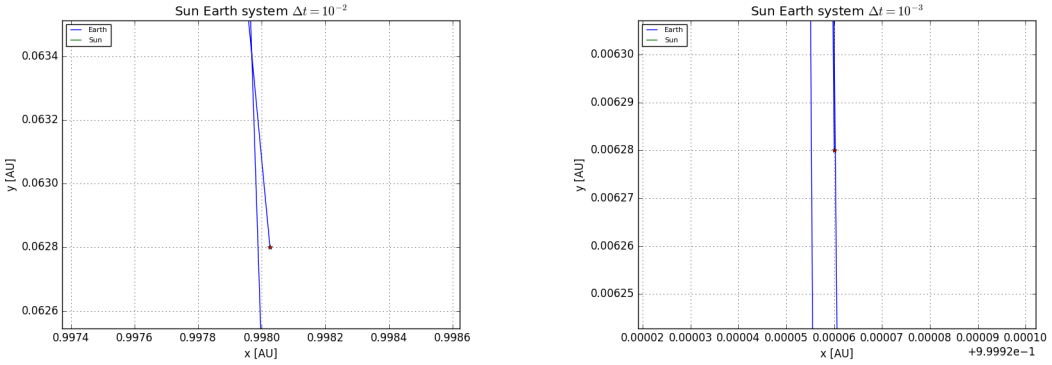


(b) Velocity-Verlet algorithm

Figure 4: Variations in the error of the total angular momentum per unit mass as a function of time for 10 years of orbit of the earth-sun system.

4.2 Investigating the time step

Plots for a variety of a timesteps are shown below. Δt is the timestep. Note that a similar plot was already shown for $\Delta t = 10^{-5}$ in figure 2 above.



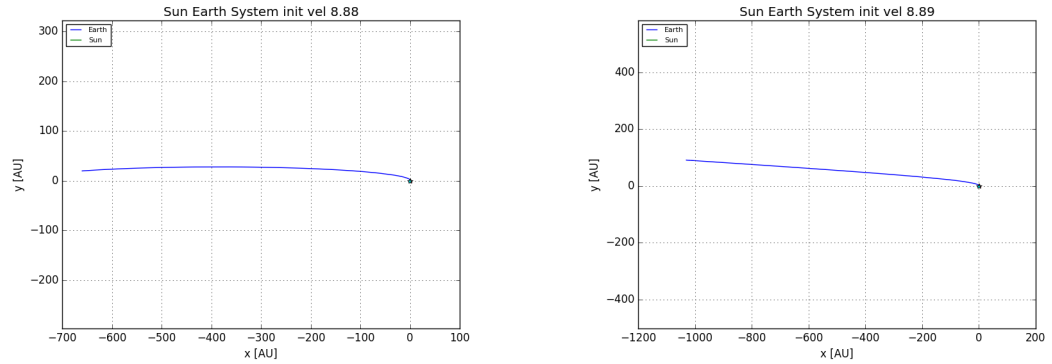
(a) Close-up of timestep of 10^{-2}

(b) Close-up of timestep of 10^{-3}

Figure 5: Plot of the earth-sun system for different time steps of the simulation. These simulations are run over 2 years.

4.2.1 Investigating the required escape velocity

The plot for the earth-sun system with the initial velocity of the earth slightly varied is shown below.



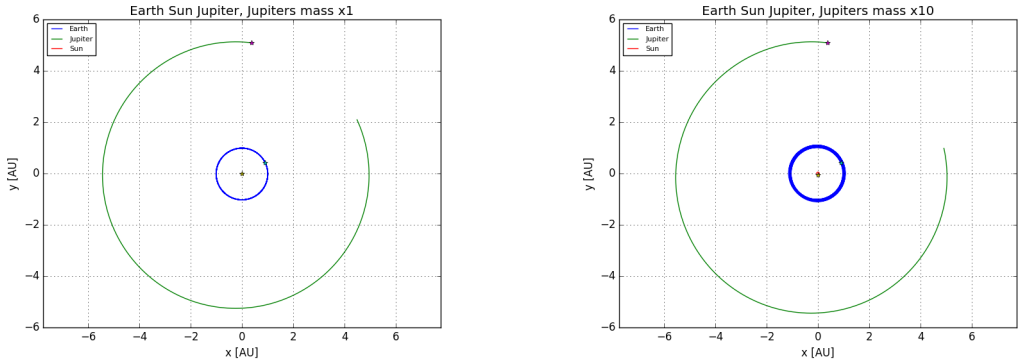
(a) Initial velocity of 8.88 AU/yr

(b) Initial velocity of 8.89 AU/yr

Figure 6: Plot of the earth-sun system for different initial velocities. The simulation runs for 2000 years, with 10^5 timesteps per year.

4.3 Results from the second model-Earth, Jupiter and Sun

The orbit of the three-body system consisting of the earth, Jupiter and the sun is shown below. We have also included a zoomed in version of the same plots.

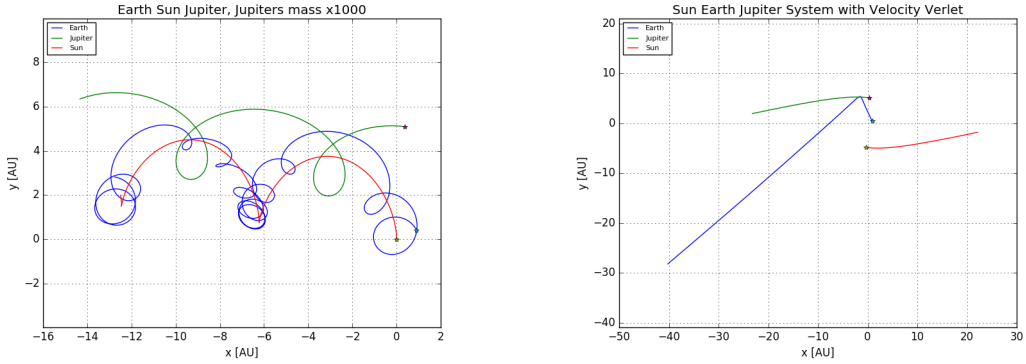


(a) Jupiter with original mass

(b) Jupiter with mass ten times its original mass

Figure 7: Sun, Earth and Jupiter system for different masses of Jupiter. Notice the larger deviations in Earth's orbit in plot b) (thicker line).

We wanted to do a similar thing for the mass of Jupiter being 1000 times larger than it is in reality, but here we ran into a problem. If we let the center of mass be at the origin, with zero velocity, the earth crashes into Jupiter. When we let the sun be at the origin, however, we get a chaotic system, but without crashes. We have therefore included both of these possibilities.



(a) Jupiter at 1000 times its original mass, sun at the origin with zero initial velocity

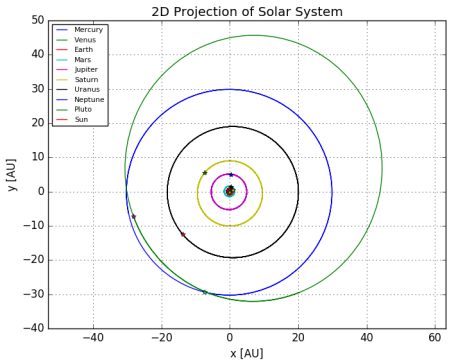
(b) Jupiter at 1000 times its original mass, center of mass at origin with zero velocity.

Figure 8: Sun, Earth and Jupiter system with Jupiter's mass increase by a factor 1000. The stars indicate the initial positions of the objects. We simulate for 10 years, with 10^5 timesteps per year. Notice how earth crashes into Jupiter in subplot b).

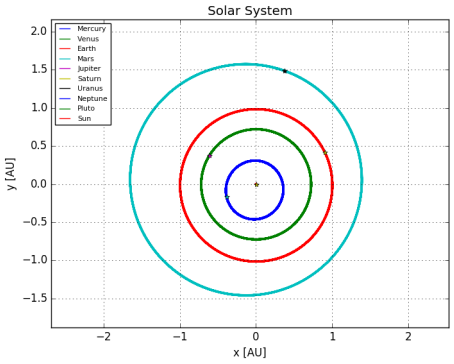
4.4 Results from the third model - Entire Solar System

4.4.1 Position

A plot of the entire solar system is shown in the figures below. We have also included a zoomed in version, to illustrate the orbit of the closest planets.



(a) Position of all planets

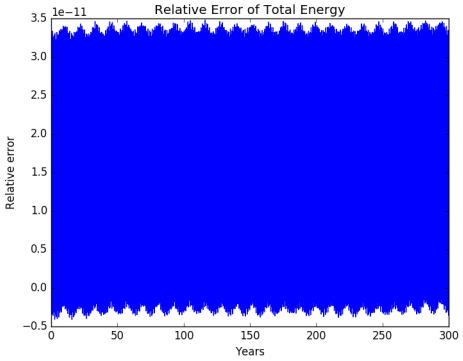


(b) Zoomed in plot, showing only the innermost planets

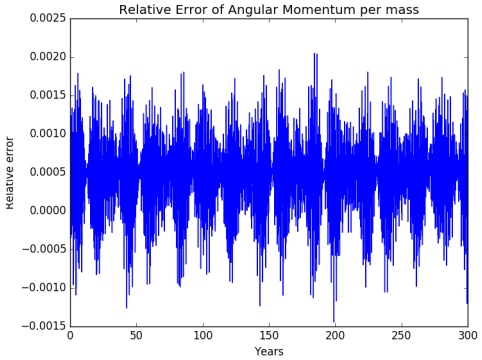
Figure 9: Plot of the entire solar system, as well as a zoom on the innermost planets. This was produced with 10^5 steps per year, running the simulation for a total of 300 years.

4.4.2 Conservation of energy and angular momentum per unit mass

A plot of the relative error in the total energy and the total angular momentum per unit mass for the entire solar system is shown below:



(a) Relative error in the total energy



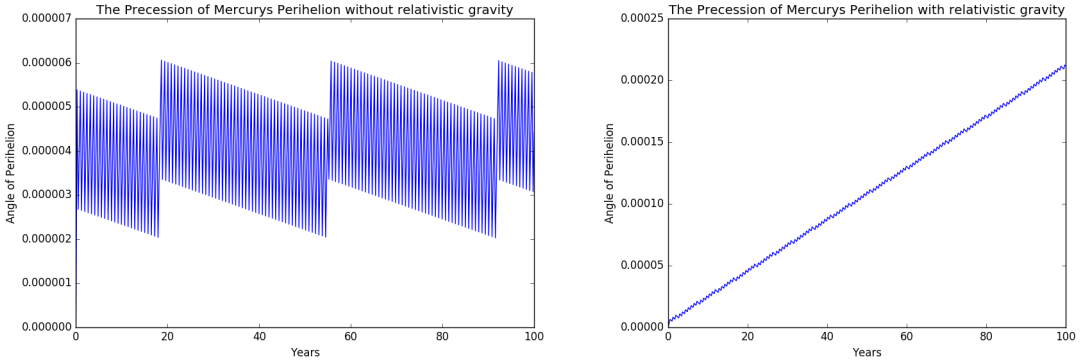
(b) Relative error in the total angular momentum per unit masss

Figure 10: Relative error of quantities that should be conserved for the entire solar system.

4.5 Results from the fourth model - Mercury-Sun System

4.5.1 Position

For this system, we plot the variation in the perihelion angle with and without Newtonian gravity below:



(a) Newtonian gravity

(b) Relativistic gravity

Figure 11: Plot of the variation of the perihelion angle over a century, using both the Newtonian and the relativistic expressions for gravity. Here we employed a higher precision of 10^7 steps per year, because the variations in angle are so small.

5 Discussion

In this section we discuss the results from the previous section for the various models.

5.1 Interpreting the results from the Earth-Sun model

5.1.1 Choice of algorithm

Position graphs: The earth-sun model, whilst simple, provides us with important insight into the details of our solvers. Note how both the Euler forward algorithm and the Velocity-Verlet algorithm produce acceptable result without too much computational power (only about 0.8 seconds for the Euler forward algorithm and just over 1 second for the Velocity-Verlet algorithm). It is clear, however, that the Velocity-Verlet is significantly better. As can be seen in figure 1, the different transits (which differ due to numerical errors), are easily distinguishable with the Forward Euler algorithm. When the Velocity-Verlet is zoomed in 10 times as much, however, it is still not possible to differentiate the transits. This shows that the Velocity-Verlet method significantly boost accuracy, without increasing computation time by much.

Conservation of energy Another important argument comes from observing the energy. Notice how the error in relative energy and angular momentum per unit mass varies systematically for the Forward Euler algorithm. The relative error of the energy is always negative and decreasing. Notice that, from section 2.5, the energy must always remain negative if potential energy is to be greater than kinetic energy, and the earth is to remain gravitationally bound. Thus, when calculating the relative error, in equation 12, we are subtracting two negative numbers and then dividing by another negative number. If this is negative, this means that initial energy is less than the energy at a time t. Thus the energy is increasing with time. This also explains the trend of the relative error in angular momentum per unit mass. Notice how this error is increasing as time, which means that angular momentum per unit mass is increasing with time. Thus, it seems that velocity, and therefore kinetic energy and angular momentum per unit mass, is increasing with time. It is not entirely obvious why the forward Euler algorithm would do this. The forward Euler is biased, as it computes the position based only on the forward velocity estimate, ignoring the velocity of the last time step. This does not entirely explain the observed trend however, as the velocity should be oscillating in time.

Notice how total energy and the total angular momentum per unit mass follow the same pattern for the Velocity-Verlet method (a negative relative error in the energy corresponds to a positive error of the angular momentum), but there is no clear bias - there are oscillations and seemingly random movements. Furthermore, the magnitudes of the errors are *significantly* lower than the errors for the Forwad Euler algorithm.

All these factors together justify our choice to use the Velocity-Verlet method for the rest of the investigation.

5.1.2 Choice of time step

As can be seen from figure 5, for $\Delta t = 10^{-2}$ and $\Delta t = 10^{-3}$, the results are rather poor - the deviations are apparent at a scale of 10^{-3} AU. For $\Delta t = 10^{-4}$ this was significantly better for the simple earth-sun system,but this broke down for the more complex system. For $\Delta t = 10^{-5}$, matplotlib would not let us zoom in close enough to see any deviations. Whilst these deviations are undoubtedly there (as can be seen from the fact that we used significantly larger time steps when searching for the perihelion angle), we decided to keep $\Delta t = 10^{-5}$, i.e. 10^5 timesteps per year, for most of the simulations, as it appeared to strike an acceptable balance between CPU time and precision.

5.1.3 Escape velocity

As can be seen in figure 6, our simulation seems to reproduce the expected escape velocity. Careful inspection of the plot reveals a curving for the lower velocity. This clearly indicates an elliptical orbit. For the higher velocity, this curving is not present, which indicates that this is a parabolic (or mayhap hyperbolic) escape. This is a reassuring check for the consistency of our system, and it confirms that our choice of Δt is justified

5.2 Interpreting the results from the Earth, Sun and Jupiter model

It is reassuring to see, as shown in figure 7 a), that our system still appears to be stable with the most massive planet in the solar system added. On the scale shown, there are no visible deviations from the earlier plot, and even when zooming these deviations are hard to see. This shows that the sun utterly dominates in the solar system.

For the case when Jupiter is 10 times heavier, we can seem from figure 7 b) that Earth's orbit becomes thicker - the earth wobbles. A zoomed in version of the same image is shown below:

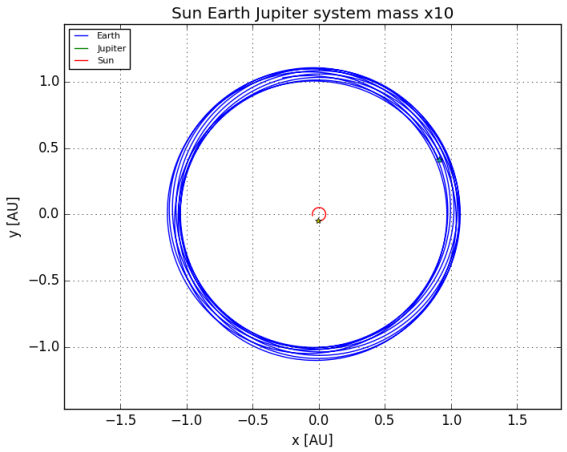


Figure 12: A zoomed in version of figure 7 b).

This shows that earth's orbit is moving slightly back and forth, due to the tidal forces from Jupiter. Nonetheless, it is clear that the sun is still the dominating body in this case.

For the case where Jupiter is 1000 times heavier than in reality, things break down. Here, the mass of Jupiter is comparable to the mass of the sun, and consequently this turns into a chaotic three-body problem. It is interesting to observe that the system keeps together in an oscillating state when the sun starts at the origin, whilst it flies apart if the center of mass starts at the origin. This illustrates the complex and frequently unstable behavior of three-body systems.

5.3 Interpreting the results from the full solar system

5.3.1 The orbit in the solar system

The orbits are shown in figure figure 9. Note that these are two-dimensional projections of the orbit. An animated version of both the 2D and 3D orbits can be found on our GIT page[4]. As can be seen there, the orbits are almost entirely coplanar, except for the orbit of Pluto. Notice how the orbits are stable over 300 years. Whilst the innermost orbits show some variations (thick line),

[4] https://github.com/dulte/Comp-Phys/tree/master/Project3Final

both due to numerical errors and due to the effects of the more massive planets, they nonetheless stay in their orbit with the given time step. Note that these simulations only took about 20 seconds to run. This indicates that the Verlet algorithm is fast and precise at our chosen time step.

5.3.2 Conservation of energy angular momentum per unit mass

As shown in figure 10, the oscillations of the relative error in the total energy is minuscule but, interestingly, not with a mean of zero. There seems to be a bias towards positive errors of the total energy. A similar pattern holds true for the total angular momentum per unit mass - it also seems to have a positive bias. Furthermore, there seems to be an oscillating behavior in the error of the total angular momentum per unit mass.

The reason for this is not entirely obvious. We tested our model, and found that this behavior only appears if we include the two heaviest planets (Jupiter and Saturn). This may indicate an error in our model. One possible explanation is that the center of mass - against our will, is moving. We defined angular momentum with respect to the origin. Therefore, a moving center of mass would result in a systematic error for the angular momentum. This does, however, not explain the periodic behavior, nor the non-zero mean. It may be possible that the deviations of the energy is simply a numerical error. as they are so small (though the non-zero mean is an odd behavior). This seems improbably for the angular momentum per unit mass. This is clearly a potential source for future investigation.

5.4 Discussing the perihelion precession of mercury

We show the perihelion angle of Mercury with Newtonian gravity in figure 11 a). This angle is not exactly zero, contrary to what one may expect. Because of the discretization of the position of Mercury, its closed distance to the sun will rarely be at the exact perihelion. Instead, our algorithm will tend to over- or undershoot the perihelion. We therefore expect the angle to oscillate around zero. Looking at the amplitude of the oscillation, we see that the numerical error of the angle appears to be of the order 10^{-6}. If we factor out this error we expect to get a constant angle of 0. But doing this with the data in figure 11 a) gives a constant angle of $\sim 4 \cdot 10^{-6}$. We are not sure of the reason for this discrepancy, but below we shall argue that this error is so small that it can be ignored. Even with the discrepancy of $\sim 4 \cdot 10^{-6}$, the angle is more or less constant over a century, meaning that Newtonian gravity fails to give the 43 arcsec or 0.0002085 rad precession of the angle that we expect from observations.

As shown in figure 11 b), relativistic gravity no longer gives us a constant angle of the perihelion, but instead a constant precession. To try to eliminate the discrepancy in the angle of $\sim 4 \cdot 10^{-6}$, we don't look at the value of the angle after a century, but take the difference of the endpoints, giving us a precession of the angle of 0.00020841 or 42.9876", which gives 0.03% relative error from the analytic result.

The difference between the Newtonian and relativistic angle is of magnitude 10^{-4}, meaning that it is significantly larger than the error due to the discretization of position. Therefore, we can conclude that our algorithm reproduces evidence that general relativity gives a better description of gravity than Newtonian gravity.

6 Conclusion and outlook

6.1 Conclusion

We have shown that the Verlet solver works well for our chosen parameters,as it reproduces a stable solar system. We have also shown that the Verlet solver is far superior to the forward Euler algorithm. This illustrates that, when solving complex coupled differential equations numerically, the choice of methods and parameters is crucial. We have also investigated how well our model reproduces the expected escape velocity for earth, and the perihelion precessions of mercury predicted by general relativity. We found that our model reproduces both of these results to an acceptable precision.

We did have some rather odd behavior for the angular momentum per unit mass and the conservation of energy for some of our models. The fluctuations were small, however, and did not impact our results significantly. Nonetheless, this seems to be an indication of an error somewhere, as the periodic change in the relative error goes against physical intuition and theoretical predictions.

6.2 Outlook

There is much further work which could be put into this project. Most interestingly, perhaps, would be to explain the periodic behavior of the angular momentum. One way to approach this may be to calculate the position of the center of mass at every time step, and subsequently taking the angular momentum relative to this, to ensure that angular momentum is always taken relative to the center of mass as it was initially. If the center of mass moves for any reason (such as numerical errors), and the calculations do not take this into account, there may be an unexpected torque on the system.

Another interesting future topic may be to investigate the behavior of the forward Euler algorithm further, in particular why it produces linear graphs for the relative error. This is not entirely obvious, and may be interesting.

Finally, it may be very rewarding to have a closer look at the three-body problem posed by Jupiter being 1000 times heavier. As can be seen from figure 8, the behavior of the system is stunning, and may well be worth investigating further.

References

[Diacu et al., 2012] Diacu, F., Pérez-Chavela, E., and Santoprete, M. (2012). The n-body problem in spaces of constant curvature. part i: Relative equilibria. *Journal of Nonlinear Science*, 22(2):247–266.

[Ledder, 2004] Ledder, G. (2004). Error in euler's method. Available at `http://www.math.unl.edu/~gledder1/Math447/EulerError`. Retrieved 23. Oct 2016.

[Mazur, 1997] Mazur, A. K. (1997). Common Molecular Dynamics Algorithms Revisited: Accuracy and Optimal Time Steps of Störmer-Leapfrog Integrators. *Journal of Computational Physics*, 136:354–365.

[Rob, 2013] Rob (2013). Accessing a telnet session in python. Available at `http://stackoverflow.com/questions/19845525/accessing-a-telnet-session-in-python`. Retrieved 23. Oct 2016.

[Weisstein, 2007] Weisstein, E. (2007). Tropical year. Available at `http://scienceworld.wolfram.com/astronomy/TropicalYear.html`. Retrieved 23. Oct 2016.

[Yang, 1967] Yang, C. N. (1967). Some exact results for the many-body problem in one dimension with repulsive delta-function interaction. *Phys. Rev. Lett.*, 19:1312–1315.

[Zeltkevic, 1998] Zeltkevic, M. (1998). Forward and backward euler methods. Available at `http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html`. Retrieved 23. Oct 2016.

Appendices

A Initial velocity for circular orbits and the value of G

If we assume a two-body system, consisting of the earth and the sun, transform to a system where the sun's velocity is zero, and assume circular orbit, we can equate the force of gravity to the centripetal force. For the earth this gives:

$$\frac{GM_\odot m_e}{r^2} = m_e \frac{v_{circ}^2}{r}$$

Where m_e is the mass of earth. Rearranging gives:

$$v_{circ} = \sqrt{\frac{GM_\odot}{r}} \tag{13}$$

Inserting $r = 1\text{AU}$ gives for the earth $v_{circ} = 29\,780$. This is the velocity we must give the earth (in a frame where the initial velocity of the sun is zero)[5], to ensure that it orbits in a circular orbit of radius 1AU.
From this equation, we can also find a value for G in the units astronomical units per year. We simply say that the earth takes one year to complete one revolution, which gives $v_{circ} = 2\pi r/T = 2\pi$ AU/yr. Solving equation 13 for G gives:

$$G = \frac{v_{circ}^2 r}{M_\odot}$$

Seeing as $r = 1$ AU, this gives:

$$G = 4\pi^2 \ \ \text{AU}^3\text{yr}^{-2}\text{M}_\odot^{-1}$$

Which also shows that $v_{circ} = 2\pi$ AU/yr.

B Deriving the Velocity-Verlet algorithm

Here will derive the Velocity-Verlet method. We will derive this for a single dimension (x), however the three-dimensional generalization is straightforward, as it simply amounts to repeating the exact same equation with x exchanged with y and z respectively.

The Velocity-Verlet method is based on Taylor-expanding the acceleration, a, around $x = x_0$. This gives:

$$a(x_0 + h) = a(x_0) + ha'(x_0) + O(h^2)$$

Or, discretizing as described in section 2.3, gives approximately:

$$a_{i+1} \approx a_i + ha_i'$$

Or, rearranging slightly:

$$ha_i' \approx a_{i+1} - a_i \tag{14}$$

Now we expand the velocity in the same way, but include another term in the Taylor expansion. This gives:

$$v_{i+1} = v_i + ha_i + \frac{h^2}{2}a_i' + O(h^3)$$

[5] We will use this value as initial velocity of the earth in the earth-sun system, even though we give the sun an additional velocity to ensure that the center of mass does not move. It is clear that these differences will be minuscule, and doing this otherwise would result in a complicated dynamic problem.

Inserting from equation 14 gives:

$$v_{i+1} = v_i + a_i h + \frac{h}{2}\left(a_{i+1} + a_i\right) + O(h^3)$$

Finally, we Taylor expand the position in the same way to get:

$$x_{i+1} = x_i + h v_i + \frac{h^2}{2} a_i + O(h^3)$$

These are the equations of the Velocity-Verlet algorithm described in equation 7. Notice how the error is proportional to h^3, as opposed to the Forward-Euler algorithm, where the error is proportional to h^2.