Project 3

Sebastian Amundsen, Marcus Berget and Andreas Wetzel

November 1, 2020

1 Introduction

Though simulating a solar system is interesting and fun enough on it's own, it is naturally also quite useful for the study of astrodynamics. In addition, being able to simulate a solar system provides a good set of tools applicable to many other scientific areas. These tools include having a good understanding of different numerical integration methods, and being able to write a structured and fast code. In this project we wish to explore a model of our own solar system, beginning with simulating the simple two-body system including the Earth and the Sun. We will use this system to compare two different methods of numerical integration, the forward Euler method and the velocity Verlet method. This simple system also makes a good testing ground for exploring whether our model is consistent with known physical laws such as energy conservation and Kepler's laws of planetary motion. We will also test the stability of the velocity Verlet method by including Jupiter and playing around with it's mass. From there we will include the rest of the planets in our solar system, and blah blah blag general relativity.

2 Theory

2.1 Forward Euler

The forward Euler method is an algorithm to estimate the solution of a differential equation. The Forward Euler method finds the next positions $\mathbf{r}_{i+1}$ by using the position we are at $\mathbf{r}_n$, a small time step dt and the derivative of its position. The forward Euler algorithm updates the position before the velocity, hence the energy is not confirmed. The forward Euler algorithm will look like:

$$\begin{aligned}\mathbf{r}_{n+1} &= \mathbf{r}_n + \mathbf{v}_n dt \\ \mathbf{v}_{n+1} &= \mathbf{v}_n + \mathbf{a}_n dt \\ t_{n+1} &= t_n + dt\end{aligned}$$

The number of FLOPS in the Forward Euler method is 4 for each time step, when we look at the addition and multiplication of this algorithm. The number of total FLOPS is $4N$, since we are looping over a time period N. It is also important to notice that the forward Euler method does not conserve energy. This is due to the position being updated before the velocity.
For more details see Appendix B.

2.2 Verlet integration

The Velocity Verlet method is based on the kinematic equations for a moving object, which in our case is a planets orbit around the sun. For the Velocity Verlet algorithm will the velocity be updated before the position, this leads to conservation of energy.
The general algorithm for the Velocity Verlet:

$$\begin{aligned}\mathbf{r}_{n+1} &= \mathbf{r}_n + \mathbf{v}_n dt + \frac{\mathbf{a}_n}{2} dt^2 \\ \mathbf{v}_{n+1} &= \mathbf{v}_n + \frac{\mathbf{a}_{n+1} - \mathbf{a}_n}{2} dt\end{aligned}$$

NUMBER OF FLOPS
For more details see Appendix B.

3 Method

3.1 Program flow

We will here discuss the program flow of the code we have used to model the solar system. All code can be found at the GitHub repository
`https://github.com/Sebamun/FYS3150_Projekter/tree/Sebastian/Project3`
The code is mostly based on two classes, a planet class and a solver class. Initializing the planet class creates an instance which contains all the relevant information about a celestial object. When the desired celestial objects are initialized, they can then be added to an instance of the solver class. The solver class contains many different functions, mostly aimed toward aiding the velocity Verlet or Euler function. These two functions are based on using their respective algorithms to print out the relevant information of each time step to output-files. The output files can then be read by python-programs which analyze/plot the data.

3.2 Comparison of the algorithms

The timing and counting of number of FLOPS of the algorithms is done by writing two separate programs with the sole purpose of doing only this. Comparing the stability of the algorithms is done by plotting the earth-sun system for different time-steps, with the Sun as the center of mass. For the case of a circular orbit we expect that both the potential and kinetic energies are constant and conserved. This is naturally also something we check that the algorithms uphold. We test both algorithms with integration points $N = 10^5$, $N = 10^6$ and $N = 10^7$. This will tell us how fast our two integration methods are.

3.3 Earth-Sun system

We have used the Earth-Sun system to compare the two algorithms, but we will now proceed by ditching the forward Euler method, using only the velocity Verlet method instead. Kepler's second law of planetary motion states that the imaginary line joining a planet and the sons sweeps equal areas of space during equal time intervals as the planet orbits. As shown in the appendix, this is equivalent to conservation of angular momentum. [noe mer om dette].

3.4 Testing forms of the force

We know that the gravitational force is an inverse square force. We wish to know what would occur if we changed this force to a higher order inverse force. This is explained in appendix A.

3.5 Many-body system

Before we include all the planets of the solar system, we start by including only Jupiter. But instead of using our own manufactured initial conditions, we extract the initial conditions from the NASA website `https://ssd.jpl.nasa.gov/horizons.cgi#top`. We write the initial conditions to a text file, before we convert the velocities to AU/Y. We also have to take into account, that the velocities extracted from NASA is relative to the center of mass. We therefore have to subtract the velocity of the center of mass to all the velocities. The velocity of the center of mass of the system is found by:

$$v_{cm} = \frac{\sum v_i m_i}{M} \tag{1}$$

where M is the total mass of all the objects in the system, and v_i and m_i are the respective velocities and masses of the objects. We will test our three body problem with different masses for Jupiter. We want to see how this impacts the orbit of the objects. The mass is increased by a factor of ten each time (for Jupiter).

Finally we are going to use the Verlet algorithm to simulate all the planets in the solar-system. The initial values are extracted from NASA. We study a time period of 170 years.

3.6 Perihelion precession of mercury

Just like all the other planets in orbit around the sun, mercury's orbit has an perihelion precession. However, unlike the other planets, mercury's precession can't be explained only by the gravitional tug from other solar bodies. Mercury's precession deviates from the predicted value by 43 arcseconds per century. The deviation can be explained by general relativity by adding a correction to the Newtonian gravitational force, so that the force becomes

$$\mathbf{F} = -G\frac{M_\odot M_{Mercury}}{r^2}\left[1 + \left(\frac{3l^2}{r^2c^2}\right)\right]\frac{\mathbf{r}}{r} \tag{2}$$

where $M_\odot$ is the mass of the Sun, $M_{Mercury}$ is the mass of Mercury, r is the distance between Mercury and the Sun, $l = |\vec{r} \times \vec{v}|$ is the magnitude of Mercury's orbital angular momentum per unit mass, and c is the speed of light in vacuum. We want to see if our code produces the same observations, so we start by setting the simulation time to 100 years. Since the we only want to observe the change in precession due to the correction in the gravitational force, we only simulate the Sun and Mercury. Since 43 arc seconds is a very small change in precession, we use as many integration points as possible. To avoid creating too large text files with information about every timestep, we set up the code such that if we use more than 10^7 integration points, the code only prints out the final position, and velocity to the text file. We then use those positions and velocities as initial conditions for an additional run where the simulation time is about 0.3 years (a little more than a full orbit of mercury around the sun). We then find which time step the minimum distance between the Sun and Mercury took place. We then use this time step to find how much the y-value of the perihelion position has changed. Finally, we can use this value to find the perihelion precession using

$$\tan(\theta_P) = \frac{y_P}{x_P} \tag{3}$$

4 Results

4.1 Comparison of the algorithms

Algorithm	$N = 10^5$	$N = 10^6$	$N = 10^7$
Euler	0.30419 s	2.89849 s	38.3016 s
Verlet	0.37503 s	3.16336 s	33.3088 s

Table 1: The speed of the different algorithms.

4.2 The Earth-Sun system

From Keeplers third law we have

$$P^2 = \frac{4\pi^2}{GM_\odot} a^3 \tag{4}$$

$$GM_\odot = 4\pi^2 \frac{AU^3}{yr^2} \tag{5}$$

4.3 Conservation of angular momentum

If we substitute equation 15 into equation 14

$$\frac{dA}{dt} = \frac{1}{2}\frac{L}{m} \tag{6}$$

Sine $\frac{dA}{dt}$=constant and the mass m is constant means that the angular momentum also is constant and therefore also conserved.

4.4 Escape velocity

In the figure below we can see the behavior for the Sun-Earth system with different initial velocities.

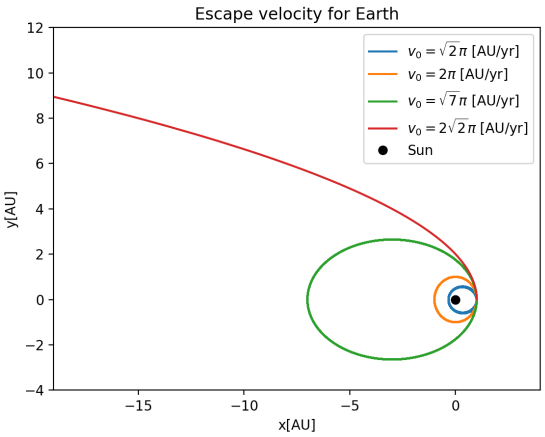

Figure 1: Escape velocity for the Sun-Earth system for different initial velocity

4.5 Testing forms of the force

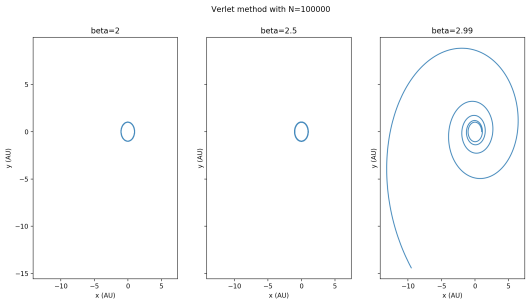

Figure 2: The planetary motion of earth at different β.

We can see the different planetary motion of earth as we adjust β in Figure 2.

4.6 Many-body system

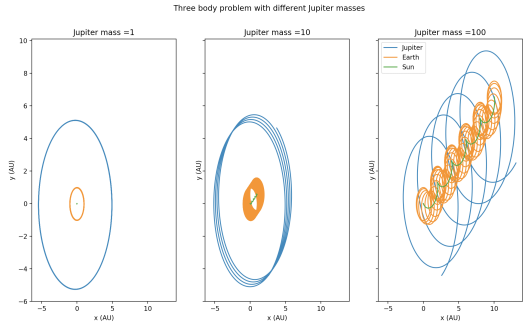

Figure 3: Three body problem with varying Jupiter masses.

We can see the different orbits of the three body problem with varying masses for Jupiter in Figure 3.

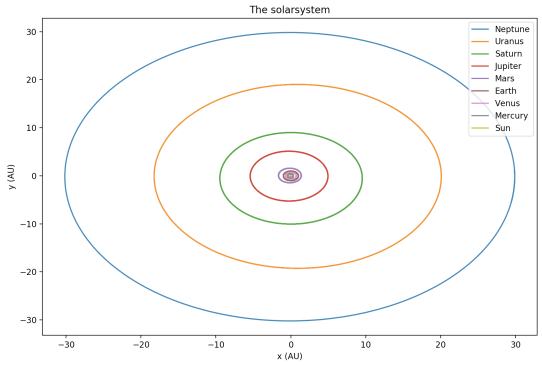

Figure 4: The solar system.

We solved the many body problem with 8 planets and the sun and the plot is given in Figure 4.

5 Discussion

5.1 Testing forms of the force

As we increase β in equation 13 we see that the orbit becomes more unstable (Figure 2). This comes from the fact that an increased β leads to a smaller gravitational force. In other words, we keep the initial distance form the sun and decrease the force that works on the planet. When this force is lower than a certain threshold we will lose our planet from orbit. In our case this limit was at $\beta = 2.99$.

5.2 Escape velocity

If we increase the initial velocity we see that the orbit of the earth changes from circular orbit to an elliptical orbit, and when the elliptical orbit increase will the position between the two object increase. Hence, will the force of gravity decrease. And when the initial velocity to the planet make it to the escape velocity it will be tear loose from the gravitational field.
It is also important to add that we say the center of mass does not move, which means that the center of mass will be constant and it will be very close to the Sun.

A Appendix A

A.1 The Earth-Sun system

We assume that the earths orbit around the sun is circular. Circular motion and Newtons gravitational force gives us:

$$F_G = G\frac{M_E M_\odot}{r^2} = \frac{M_E v^2}{r} \tag{7}$$

Where G is the gravitational constant, $M_\odot$ is the mass of the sun, M_E is the mass of the earth, r is the distance between the earth and the sun, and v is the velocity of the moving object (earth).

We want to show that the velocity v can be written as:

$$v^2 r = GM_\odot = 4\pi^2 \frac{AU^3}{yr^2} \tag{8}$$

We can use the centripetal force to rewrite equation 7, since we are assuming circular orbits:

$$M_E \omega^2 r = G\frac{M_E M_\odot}{r^2}$$

Where ω^2 is the angular velocity of the earth. Which is given by $\omega^2 = 2\pi/P$:

$$M_E \left(\frac{2\pi}{P}\right)^2 r = G\frac{M_E M_\odot}{r^2}$$

Where $P = 1$ year is the period of earth's orbit around the sun. Now we can use Kepler's third law, which states that the square of an orbital period P^2 equals the semi-major axis as it orbits the sun cubed a^3. This gives us:

$$P^2 = \frac{4\pi^2}{GM_\odot} a^3 \tag{9}$$

Where the circular radius r is substituted with the semi-major axis a. In our case $a = 1AU$.

A.2 Conservation of angular momentum

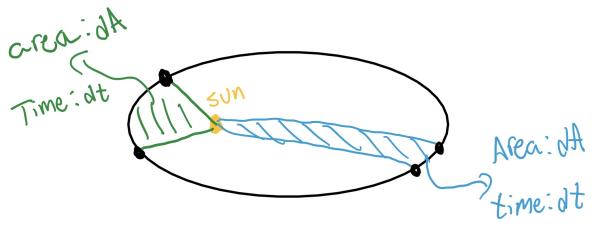

Figure 5: The area dA is the same for the two time intervals.

The definition of Keplers law is that the area A, which is formed from a connecting line between a planet and the sun is always the same for a given time interval. This is illustrated in Figure 5.

We have that the angular momentum is

The best way to show that the angular momentum is conserved by using Keplers second law is to make a drawing:

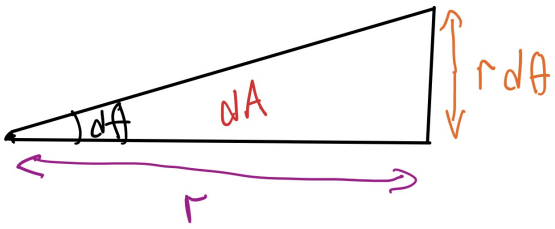

Figure 6: This square is the infinitesimal area dA that the planet has moved by a infinitesimal time interval dt

Figure 3 shows us that the area is:

$$dA = \frac{1}{2}r^2 d\theta \tag{10}$$

We now want to find an infinitesimal area where the planet has moved around the sun, over an infinitesimal time step, where we use equation 10.

$$\frac{dA}{dt} = \frac{1}{2}r^2\frac{d\theta}{dt} \tag{11}$$

$$= \frac{1}{2}rv_\theta \tag{12}$$

And the definition of angular momentum is given as

$$L = mrv_\theta$$

With equation 12 and 10 we can show that the angular momentum is conserved.

A.3 Testing forms of the force

The force F_g is on the form of an inverse square force. We can replace the r squared in 7 with a β which we can alternate $\beta \in [2,3]$. This gives us the equation:

$$\mathbf{F}_G = -\frac{GM_\odot M_{planet}}{r^\beta}\frac{\mathbf{r}}{r} \tag{13}$$

A.4 Escape velocity

We will now look at a planet which begins at a distance 1 AU from the sun, and see how fast the initial velocity must be for the planet to be able to escape the sun. We find the analytical solution to compare it to our numerical

solution.
The analytical expression is given as

$$| \mathbf{V}_{esc} | = \sqrt{\frac{2GM_{\odot}}{r}} \tag{14}$$

By plugging equation 2 into 16, and use that our initial position of the planet is $r = AU$ we get

$$| \mathbf{V}_{esc}^2 | = \frac{2 \cdot 4\pi^2 AU^3}{yr^2} \cdot \frac{1}{r} \tag{15}$$

$$= \frac{2 \cdot 4\pi^2 AU^3}{yr^2} \cdot \frac{1}{AU} \tag{16}$$

$$| \mathbf{V}_{esc} | = 2\sqrt{2}\pi \cdot \frac{AU}{yr} \tag{17}$$

A.5 Perihelion precision

LES GJENNOM DETTE MARCUS
The definition of perihelion precision is that the perihelion point of an objects, in our case Mercury, changes its position because of the gravitational field deflection from light. RIKTIG? The perihelion point for Mercury changes with 43″ arcseconds for each century.

Mercury's orbit has a small prescision for each century. This comes from that Mercury is the nearest planet to the sun, and therefore its acceleration and velocity will be large, which means that the space time will wrap and we have to obey the rules of General Relativity. Hrnce, we rewrite the expression for Newtons gravitational force to a relativistic gravitational force.

$$\mathbf{F} = -G\frac{M_{\odot} M_{Mercury}}{r^2}\left[1 + \left(\frac{3l^2}{r^2 c^2}\right)\right]\frac{\mathbf{r}}{r} \tag{18}$$

B Appendix B

B.1 Forward Euler

The forward Euler method is an algorithm to estimate the solution of a differential equation. The Forward Euler method finds the next positions $\mathbf{r}_{i+1}$ by using the position we are at $\mathbf{r}_n$, a small time step dt and the derivative of its position. Which can be expressed as:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{r}'_n \cdot dt \tag{19}$$

Where $\mathbf{r}'_n$ is the derived of the current position. This algorithm is an abbreviated version of a Taylor expansion, where we only expand the series one step at a time. By doing this we will get a local truncation error, which causes an error for each step we take:

$$\begin{aligned} &\mathbf{r}(t_n + dt) = \mathbf{r}_{n+1} \\ =&\mathbf{r}(t_n) + dt\left(\mathbf{r}'(t_n) + \frac{\mathbf{r}''(t_n)}{2!}dt + ... + \frac{\mathbf{r}^p(t_n)}{p!}dt^{p-1}\right) \\ &+ O(dt^{p+1}) \end{aligned} \tag{20}$$

Where $O(dt^2)$ is the local truncation error. Since the Forward Euler method is a first-order method, the local truncation error will be proportional to the square of the step size.

In our case $\mathbf{r}(t_n) \rightarrow \mathbf{r}_n$ is the position, $\mathbf{r}'(t_n) = \mathbf{v}(t_n) \rightarrow \mathbf{r}'_n = \mathbf{v}_n$ is the velocity and $\mathbf{r}''(t_n) = \mathbf{a}(t_n, r_n) \rightarrow \mathbf{v}'_n = \mathbf{a}_n$ is the acceleration. The update of our position and velocity with a time step is then given as

$$\begin{aligned}
\mathbf{r}(t_{n+1}) &= \mathbf{r}(t_n) + \mathbf{v}(t_n)dt + O(dt^2) \\
\mathbf{r}_{n+1} &= r_n + v_n dt + O(dt^2) \\
\mathbf{v}(t_{n+1}) &= \mathbf{v}(t_n) + \mathbf{a}_n dt + O(dt^2) \\
\mathbf{v}_{n+1} &= \mathbf{v}_n + \mathbf{a}_n dt + O(dt^2)
\end{aligned}$$

Hence, the forward Euler algorithm will look like:

$$\begin{aligned}
\mathbf{r}_{n+1} &= \mathbf{r}_n + \mathbf{v}_n dt \\
\mathbf{v}_{n+1} &= \mathbf{v}_n + \mathbf{a}_n dt \\
t_{n+1} &= t_n + dt
\end{aligned}$$

Pseudocode:

```
v[0] = v0   # Boundary conditions
r[0] = r0   # Boundary conditions

for (n = 2; n < N-1; n++) {
      V[n+1] = v[n] + a[n] * dt;
      r[n+1] = r[n] + v[n] * dt;
      t[n+1] = t[n] + dt;

}
```

B.2 Velocity Verlet method

The Velocity Verlet method is based on the kinematic equations for a moving object, which in our case is a planets orbit around the sun. If we want to find the next time step for the velocity and position we do an approximation and use a Taylor-expansion:

$$\begin{aligned}
&\mathbf{r}(t \pm dt) = \mathbf{r}(t) \pm \mathbf{r}'(t)dt + \frac{\mathbf{r}''(t)}{2}dt \\
&+O(dt^3)
\end{aligned}$$

This gives us the position and velocity:

$$\mathbf{r}_{n+1} = \mathbf{r_n} + \mathbf{v}_n dt + \frac{\mathbf{a}_n}{2}dt^2 + O(dt^3) \tag{21}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n dt + \frac{\mathbf{v}''_n}{2}dt^2 + O(dt^3) \tag{22}$$

We need to define $\mathbf{v}''_n$. We do this by by defining the next step in acceleration $\mathbf{a}_{n+1}$:

$$\mathbf{a}_{n+1} = \mathbf{v}'_{n+1} = \mathbf{v}'_n + \mathbf{v}''_n dt + ...$$

We can rewrite this as:

$$\mathbf{v}''_n = \frac{\mathbf{v}'_{n+1} - \mathbf{v}'_n}{dt} = \frac{\mathbf{a}_{n+1} - \mathbf{a}_n}{dt} \tag{23}$$

By inserting equation 23 into equation 22 we get the general algorithm for the Velocity Verlet:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n dt + \frac{\mathbf{a}_n}{2} dt^2 + O(dt^2)$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{\mathbf{a}_{n+1} - \mathbf{a}_n}{2} dt + O(dt^3)$$

NUMBER OF FLOPS
Pseudocode:

```
v[0] = v0;   # Boundary conditions
r[0] = r0;   # Boundary conditions

for (i = 2; i < n-1; i++) {
    v[n+1] = v[n] + (a[n+1]-a[n]) * dt / 2
    r[n+1] = r[n] + v[n] * dt + a[n] * pow(dt,2) / 2;
}
```

C Calculations

References

[1] Skriv inn kilde her.