Numerical Models of the Solar System

Modelling the interactions and relationships
of our solar system using
Ordinary Differential Equations

University of Oslo

Simen Løken

October 2020

1 Abstract

In this report, we model a multi-body system and show how it evolves in space in relation to other bodies, and how we can use these interactions to make numerical calculations. We find that for both elliptical and circular orbits both energy and angular momentum is conserved, and that the relativistic precession of Mercury over the span of a century can be shown to be 41".

2 Introduction

The space-faring age is upon us! - well, not yet, but surely soon! And in order to acclimate ourselves to the age of space-faring mankind, we need to properly be able to model planets: their positions, velocities, angles, trajectories, interactions. It takes time getting from planet to planet, so knowing ahead of time where a planet is going to be when you finally get there is quite beneficial. Such a system, with many interacting bodies is what we're going to be modelling here in this report, in preparation for the (hopefully soon) space age.
In this report we're going to be modelling the Solar System with increasingly accurate parameters and examining it's relationships and interactions. We're going to be examining both a two-body and a multi-body system for modelling parts of our solar system. We're additionally going to be examining whether energy and angular momentum is conserved for circular and elliptical orbits. Lastly we're going to be looking at the unusual precession of Mercury, which can only be explained by general relativity, and calculating said precession.

3 Theory and Method

3.1 General Theory

For this paper we're going to be using primarily the Verlet Velocity Method (henceforth referred to as the Verlet Method) to model our solar system, but in order to add some context, let's also talk about the, perhaps more common, Forward Euler Method (henceforth referred to as the Euler Method) of solving Ordinary Differential Equations (henceforth referred to as ODEs).
In a general sense, the Euler Method is pretty simple. Assume we have an acting force F onto an undefined body with mass m and a velocity v_n:

$$F = ma, a = \frac{F}{m}$$

We know then that, using the Equations of Motion for some sufficiently small timestep Δt:

$$v_{n+1} = v_n + a\Delta t$$

It then follows that we can make a similar prediction for the position using the same equations:

$$p_{n+1} = p_n + v_{n+1}\Delta t$$

This can be repeated ad infinitum so long as you have a driving force F. However, we're going to mostly be using the Verlet Method.
Assume again that we have an acting force F onto a body with mass m with a velocity v_n:

$$F = ma, a = \frac{F}{m}$$

$$p_{n+1} = p_n + v_n\Delta t + \frac{1}{2}a\Delta t^2$$

Using our new position to retrieve a new acting force:

$$F_2 = ma_2, a_2 = \frac{F_2}{m}$$

Using this, we can then find our new velocity to be

$$v_{n+1} = v_n + \frac{a + a_2}{2}\Delta t$$

Much like the Euler Method, this process may also be repeated so long as you have a force acting onto the body we're inspecting.
As we're going to be modeling these systems, our eventual code for a two-dimensional Solar System would look like:

```
for i in range(N-1):
        r = np.sqrt(x[i,0]**2+x[i,1]**2)
        F = G*M_sun*M_earth/(r**2)
        theta = np.arctan2(x[i,1],x[i,0])
        fx = -FG*np.cos(theta)
        fy = -FG*np.sin(theta)
        ax = fx/M_earth; ay = fy/M_earth
        a = np.array((ax,ay))
        v[i+1] = v[i] + a*dt
        x[i+1] = x[i] + v[i+1]*dt
```

for the Euler Method, and:

```
r = np.linalg.norm(x2[0])
FG = G*M_sun*M_earth/(r**2)
theta = np.arctan2(x2[0,1],x2[0,0])
fx = -FG*np.cos(theta)
fy = -FG*np.sin(theta)
ax, ay = fx/M_earth, fy/M_earth
a = np.array((ax,ay))
for i in range(N-1):
    x2[i+1] = x2[i] + v2[i]*dt+0.5*a*dt*dt
    r = np.linalg.norm(x2[i+1])
    FG = G*M_sun*M_earth/(r**2)
    theta = np.arctan2(x2[i+1,1],x2[i+1,0])
    fx = -FG*np.cos(theta)
    fy = -FG*np.sin(theta)
    ax = fx/M_earth; ay = fy/M_earth
    a2 = a #old a
    a = np.array((ax,ay)) #forward a
    v2[i+1] = v2[i] + dt*((a+a2)/2)
```

for our Verlet Solution
We can then employ these methods onto our initial position and velocities.
As we're working with great distances, gravitational force will be calculated as:

$$F_G = \gamma \frac{M_{sun} M_{earth}}{r^2} \tag{1}$$

where γ is the gravitational constant and r is the distance between our two masses, M_{sun} and M_{earth}, given as $r = \sqrt{x^2 + y^2}$ for a given position x and y.
Knowing this, we can then use data gathered from [1] to intitialize our system and calculate from there. Since we're naturally working with a multi-body system, we're going to have to space out our code so that we're updating all positions and velocities continuously.

In pseudo-code written out, we're looking to:

```
for i in range(N-1):
    for body in bodies:
        p[i+1] = p[i] + v[i]*dt+0.5*a*dt*dt
        force = body.Force(bodies)
        a2 = a
        a = force/body.m
        v[i+1] = v[i] + dt(*(a+a2)/2)
```

We make sure that if body = bodies, force will not be calculated.
The force() function will calculate all forces acting upon body, and then add them together to find the total force acting upon body for our time-step i.

3.2 Energy and Angular Momentum

We'll also need to calculate the energy of our system:
The total energy on a celestial body at any time is given as:

$$E = U + K \tag{2}$$

where U and K are the potential and kinetic energy respectively, given as:

$$U = -\gamma \frac{M_{sun} M_{earth}}{r}$$

$$K = \frac{1}{2} m \vec{v}^2$$

We'll also find the angular momentum at any given position using

$$L = \vec{r} \times \vec{p} \tag{3}$$

where p is the momentum vector given as $p = m\vec{v}$
We can arrive at the fact that these values should be constant for a circular orbit, given that $\vec{v}$ is unchanging and $\vec{v}$ is constant, thus leading to a constant L. This is also true for elliptical orbits, where as $\vec{r}$ decreases, $\vec{v}$ increases proportionally so as to maintain a constant L. We can also see that (as discussed), since $\vec{r}$ and $\vec{v}$ are constant, both kinetic energy K and potential energy U should also remain constant. Using the same logic as applied to the angular momentum, if $\vec{r}$ decreases, $\vec{v}$ increases proportionally, thus leading to the conclusion that energy is also retained for an elliptical orbit.
The escape velocity for any given distance r orbiting the sun can be calculated as:

$$v_{escape} = \sqrt{\frac{2\gamma M_{sun}}{r}} \tag{4}$$

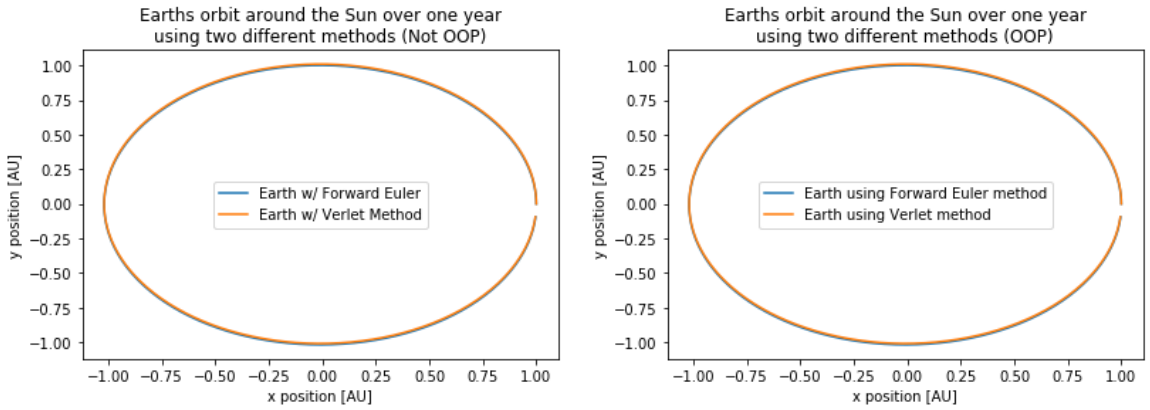
3.3 Mercury at Perihelion - A test for general relativity

Mercury has a very elliptical orbit, which means it's precession changes a relatively large amount from year to year. However, in this case, reality deviated from theory, and scientist were stumped for several hundred years as to where this deviation of 43 arcseconds came from. It took Einstein's General Theory of Relativity to finally understand what was happening.
Einstein's theory explained that the unexplained precession was explained by the gravitational curvature of spacetime, since Mercury is so close to the sun.
In order to numerically solve this properly, we're going to have to make a relativistic correction to our force, which becomes:

$$F_{rel} = \gamma \frac{M_{sun} M_{mercury}}{r^2} \left[1 + \frac{3l^2}{r^2 c^2} \right] \tag{5}$$

4 Results

Firstly we run a simple two body system where forces only act on Earth as to demonstrate/verify that our algorithm is working:



(a) Non-OOP solution

(b) OOP Solution

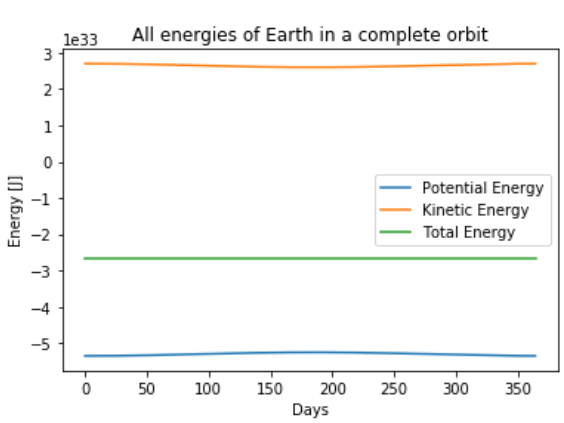
Figure 1: Using both the Euler and Verlet method with and without objects to plot the orbit of Earth for one year using the initial conditions $x_0 = (1\text{AU}\ ,\ 0)$, $v_0 = (0\ ,\ 30 \times 10^3 m/s)$

We additionally compare the run-times and find that (on average over 5 runs):

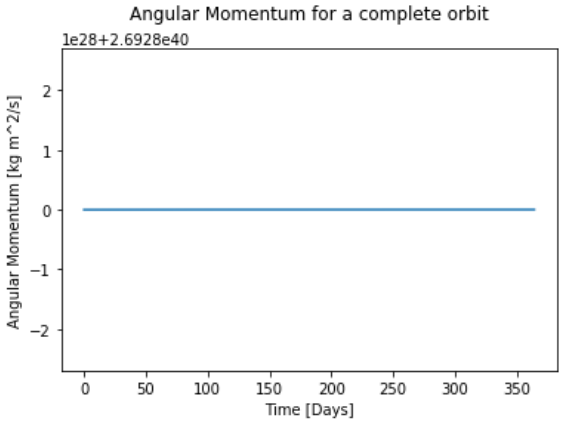
	Euler [s]	Verlet [s]
Mean	8.2575×10^{-3}	9.1917×10^{-3}
STD	3.6258×10^{-5}	1.4581×10^{-4}

Table 1: Mean runtimes of the Euler and Verlet solution respectively over 5 runs with standard deviations

We can now calculate the energy and angular momentum for a circular/elliptical orbit and validate that these values (according to theory) are retained:

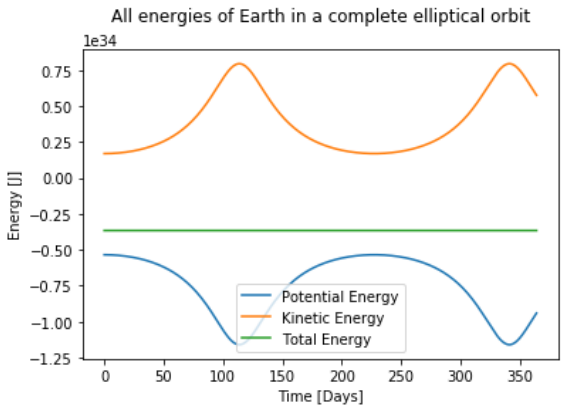


(a) All Energies over a year

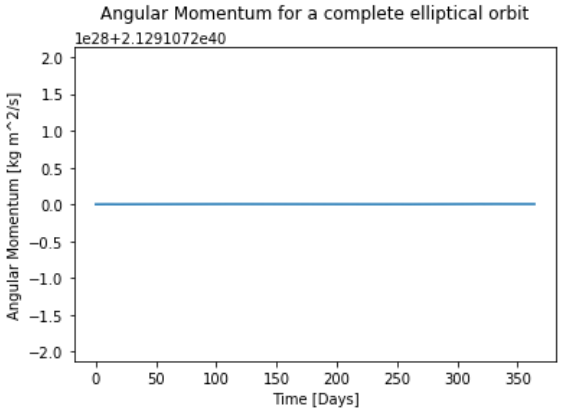


(b) Angular momentum over a year

Figure 2: Using the Verlet method to verify that energy and angular momentum is retained during Earths orbit for initial conditions $x_0 = (1\text{AU}\ ,\ 0)$, $v_0 = (0\ ,\ 30 \times 10^3 m/s)$ over a year



(a) Energy of elliptical orbit



(b) Angular momentum of elliptical orbit

Figure 3: Using the Verlet method to verify that energy is retained during Earths elliptical orbit for initial conditions $x_0 = (1\text{AU}\ ,\ 0)$, $v_0 = (0\ ,\ 23.72 \times 10^3 m/s)$

We let β in $F = \frac{1}{r^\beta}$ approach a breakpoint where we no longer orbit and the body escapes the gravitational pull:

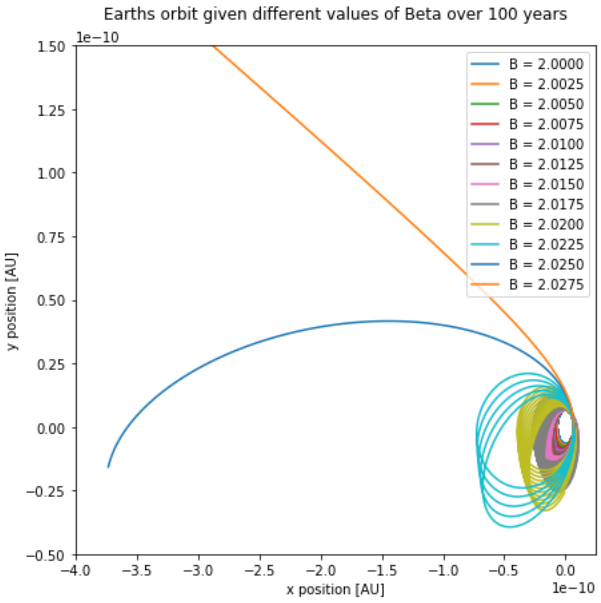
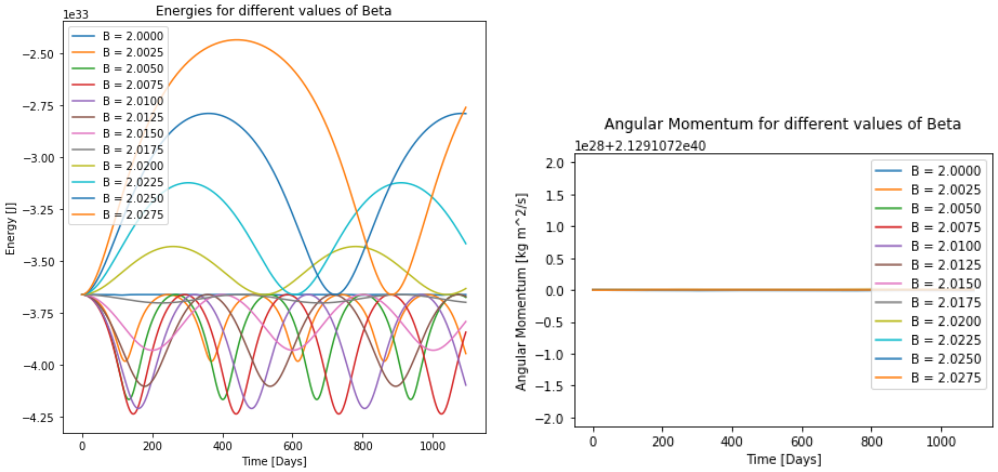


Figure 4: Orbits for an increasing β in r^β over a century.

Let's now examine the energy and angular momentum of such a system for different values of β:



(a) Energy of elliptical orbit

(b) Angular momentum of elliptical orbit

Figure 5: Using the Verlet method with differing βs to examine that energy is retained during Earths elliptical orbit for initial conditions $x_0 = (1\text{AU}\ ,\ 0)$, $v_0 = (0\ ,\ 30 \times 10^3 m/s)$ over 3 years

Using Equation [4] we can calculate the escape velocity in Earths orbit to be:

$$v_{velocity} = \sqrt{\frac{2\gamma 2 \times 10^3 0}{1AU}} = 42230 \frac{m}{s}$$

Knowing this, we can examine this numerically by letting our initial velocity go from $35 \times 10^3 \frac{m}{s}$ to $45 \times 10^3 \frac{m}{s}$:

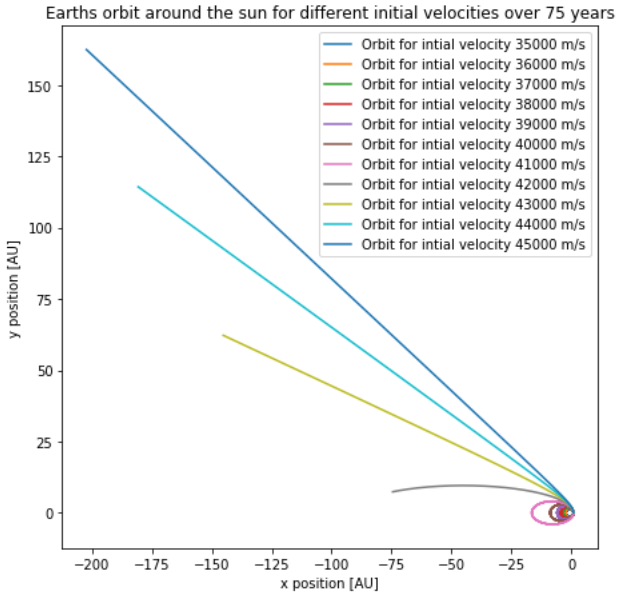
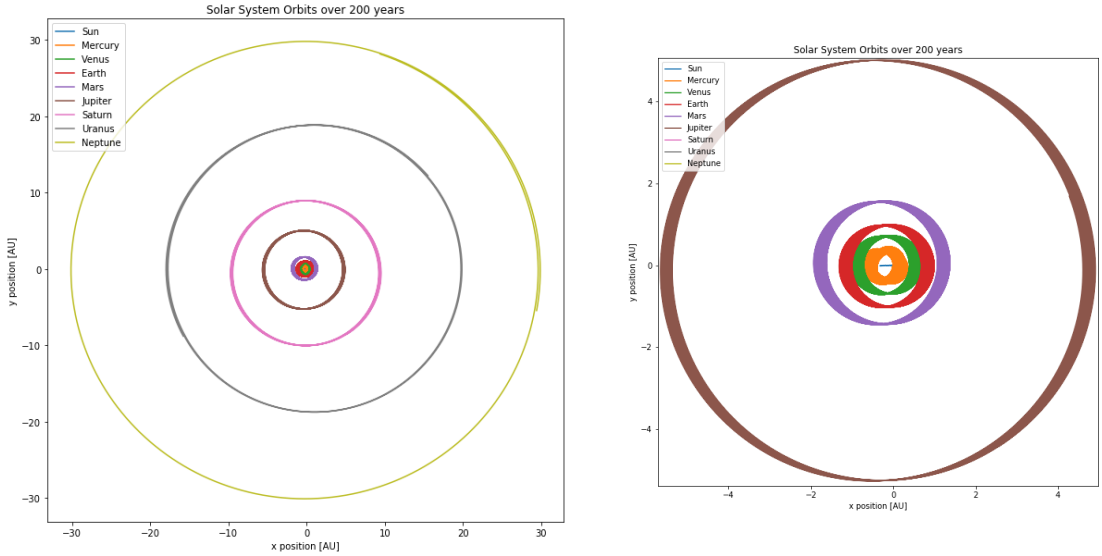


Figure 6: Orbits given different initial velocities.

Let's now examine our system as a whole, we add all 8 planets (and the sun) and plot:



(a) Orbits of all planets in our entire solar system over 200 years

(b) Orbits of the five innermost planets (and the sun)

Figure 7: Orbits in our solar system using initial conditions from [1]

Using 5, we let Mercury orbit the sun for 100 years, before stopping it at it's Perihelion:

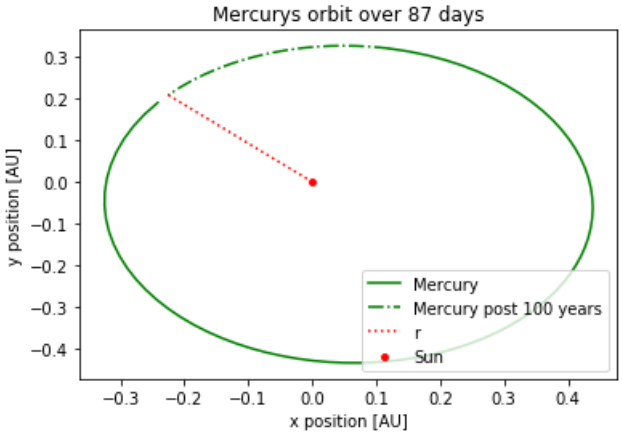


Figure 8: The final 87 days of orbit before we stop to examine it's angle and coordinates.

We find that this angle is $\theta = 2.393194381674042$ rad, and that the annual change in angle is $\Delta\theta_{annual} = -0.024518933385077806$ rad. This means that for one century + the additional days to hit Perihelion, our total theoretical θ_{total} change is:

$$\theta_{total} = \Delta\theta_{annual} * \frac{36500days + \frac{275hours}{24hours}}{88days} = -3.889792132218641rad = 2.393393174960945rad$$

The deviation from the theoretical is then given as:

$$\theta_{relativity} = |\theta - \theta_{total}| = 0.0001987932869029762 6rad = 41"$$

which is fairly close to the real answer of 42.98"

5 Discussion

Discussing results

β-values

You might've noticed that I instead of letting β go from 2 to 3. This is simply because even for a value as low as $\beta = 2.1$, the planet completely broke off from the orbit. As you can see in Figure [4], for a β as small as 2.0275 it breaks off. I obviously haven't been able to figure out why this happens, but I think it might have to do with me using SI units instead of scaling everything accordingly to Astronomical Units, which results in increasingly large r denominators, causing the force to become close to zero.

Mercury's Precession

In regards to the analytical vs. numerical solutions. We find that $\theta_{relativistic}$ =41", which is off by about 4.5%. This seems pretty massive, but I'm assuming it's perhaps due to some kind of rounding error in Python, as arc seconds scale linearly with radians, but very steeply. As such, all things considered, I wouldn't consider this a bad result.

Other discussions

Runtimes and FLOPs

As shown in Table [1], our Euler Method was actually faster than it's Verlet counterpart. This is to be expected however, as if we decompose our algorithms we find that:
Euler:

- Finding a_x,a_y = 2FLOPs
- Finding new v = 3FLOPs
- Finding new p = 3FLOPs

Verlet:

- Finding new p = 4 FLOPs
- Finding a_x, a_y = 2FLOPs
- Finding new v = 3FLOPs

Note that we've excluded the force function from this calculation, as it is the same across both. Therefore we find that

	Euler [s]	Verlet [s]
Mean	8.2575×10^{-3}	9.1917×10^{-3}
STD	3.6258×10^{-5}	1.4581×10^{-4}
FLOPs	8N	9N

Table 2: Mean runtimes of the Euler and Verlet solution respectively over 5 runs with standard deviations, in addition to the number of FLOPs for each algorithm

So why use Verlet when Euler is faster?
This mostly comes down two main points. Firstly, all the calculations we're doing in this project aren't very costly, meaning runtime doesn't even become an issue, even for something seemingly colossal, like calculating the orbits of the entire solar system over 200+ years.
Secondly, and most importantly, the Verlet method is more accurate than Euler and is symplectic, which means it works very well for systems where energy is conserved.

Stability and corrections

We were asked to talk about the stability of our solution, which I figured I'd do here. Let's look at how the Verlet solution behaves for different timesteps Δt:

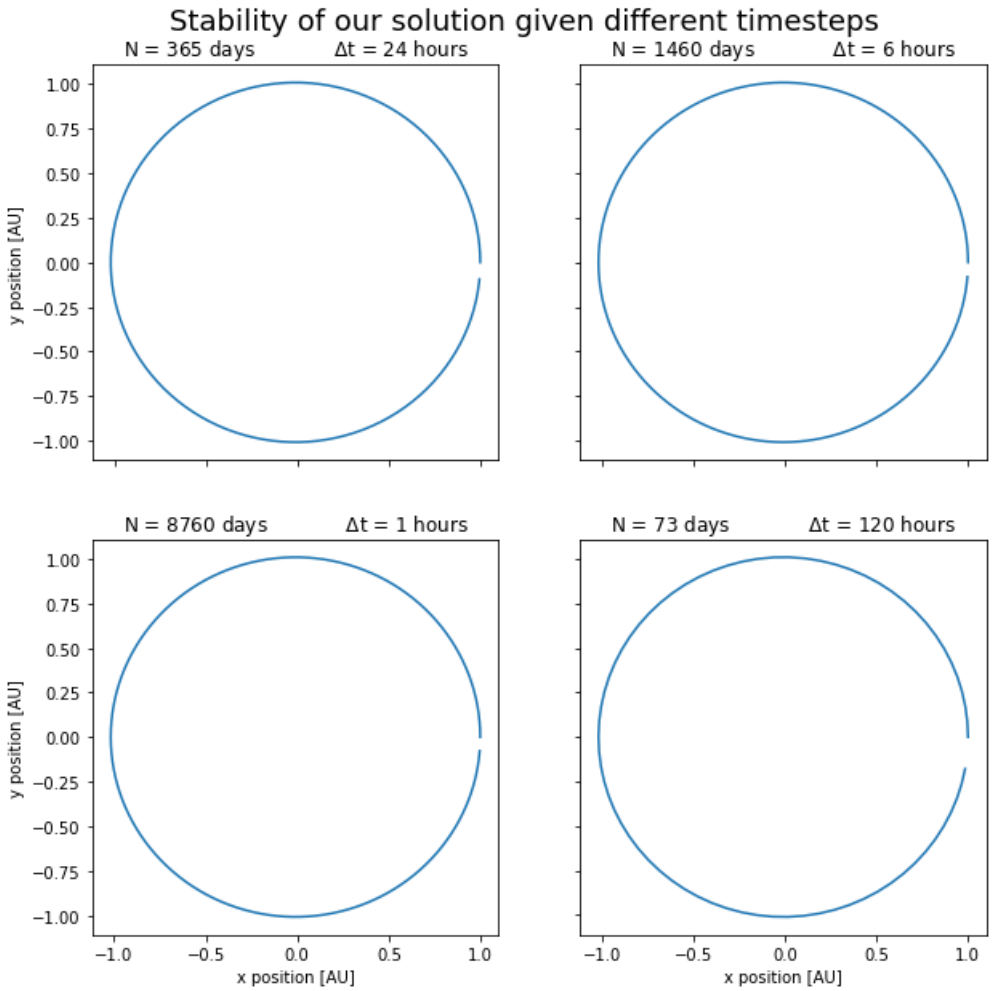


Figure 9: Stability of our solution for four different timesteps. N is scaled accordingly so as to keep the "total time" close to one year

Let's now also look at a three-body system and it's stability:

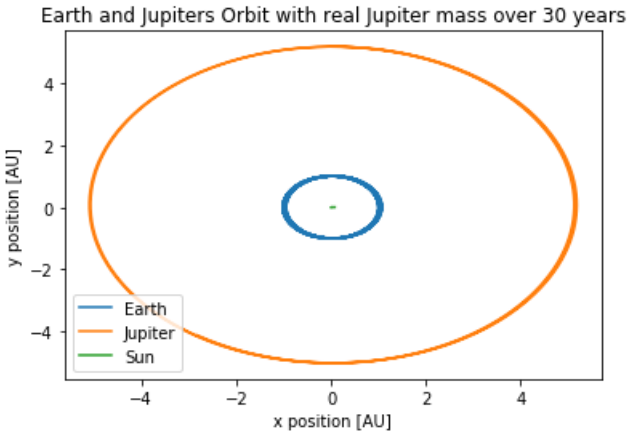
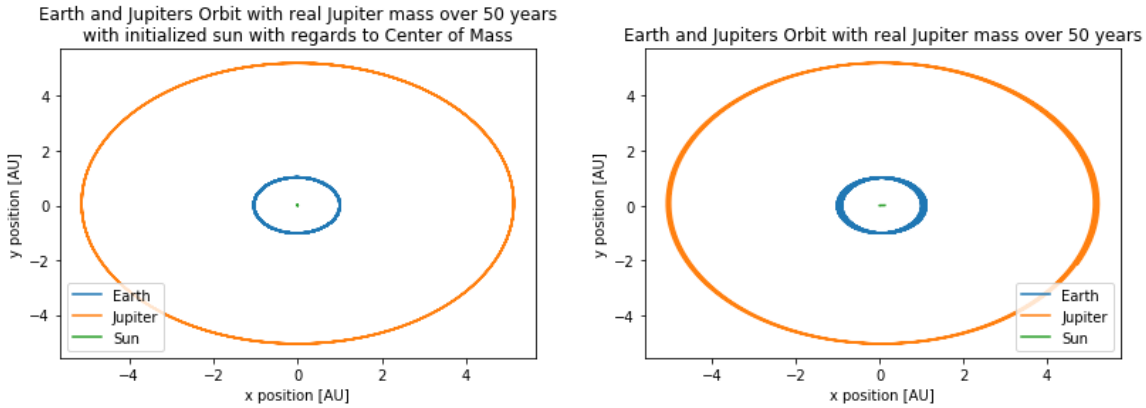


Figure 10: A three-body system using an initialized Earth and Jupiter with the initial conditions:
Earth: $x_0 = (1\text{AU}\ ,\ 0)$, $v_0 = (0\ ,\ 30 \times 10^3 m/s)$
Jupiter: $x_0 = (0\ ,\ 5.2\text{AU})$, $v_0 = (13 \times 10^3 m/s,\ 0)$

Let's now examine the same system, but with and without the sun initialized so as to make the total momentum 0, and letting the center of mass be (0,0) over 50 years:



(a) Orbit over 50 years with an initialized sun so as to increase the stability of our system

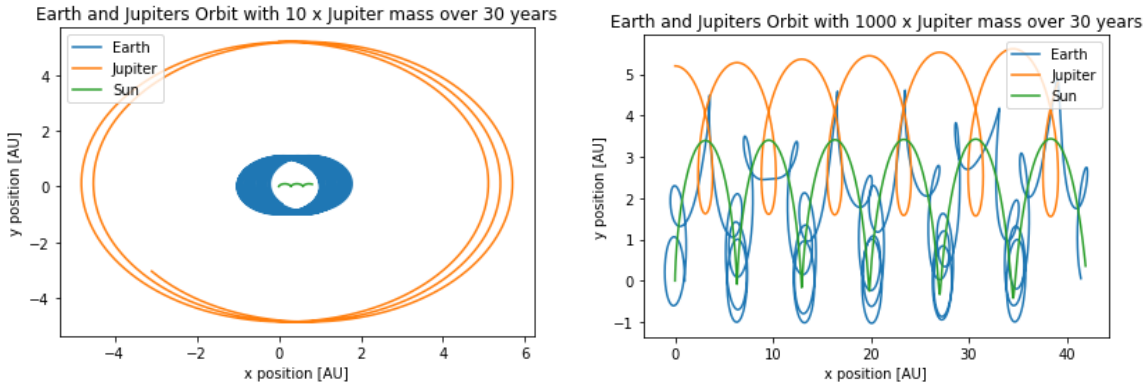
(b) Orbit over 50 years without an initialized sun

Figure 11: Two different plots showing the effect of initializing the sun so as to further stabilize our system

Notice how the lines on the leftmost plot are significantly thinner than the one on the right. This is because the system we have created for the leftmost plot has close to no drift, meaning it is pretty much a perfect circular orbit, as opposed to the right-side plot which seems to have some drift, and is likely somewhat elliptical in it's orbit.

Jupiter's mass

What happens to the stability of our system if we were to amplify the mass of Jupiter? For reference, a normal three-body system consisting of Earth, Jupiter and the sun looks like Figure [10].



(a) A 30 year orbit with Jupiter having 10× its normal mass(b) A 30 year orbit with Jupiter having 1000× its normal mass

Figure 12: Two different plots showing the effect of Jupiter's mass on Earth and our three-body system.

This results is of course to be expected. Jupiter actually almost overtakes the sun in mass when multiplying by 1000, resulting in a very unstable system. Notice how Earth starts orbiting the Sun more like a moon than a planet (except it's sort of the same thing).

Pluto - a potential problem with the model

I originally wanted to include Pluto in Figure 7a, however it deviated severely from it's analytical values, so much so that it finished it's orbit about 50 years prior to when it analytically should. For reference:

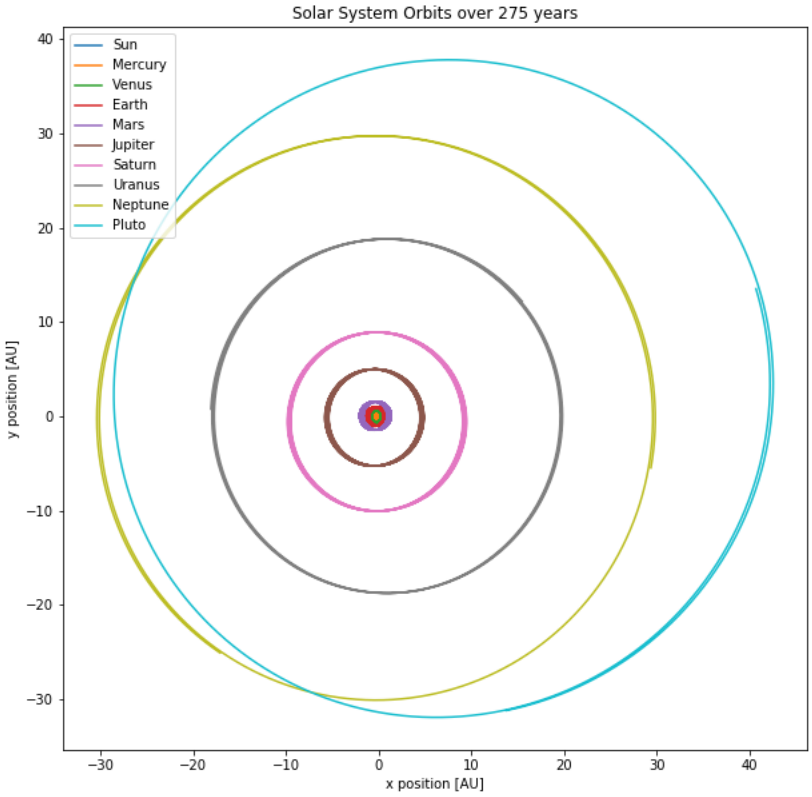


Figure 13: The same system as 7a, this time with Pluto included

which was cause for some concern. I wasn't able to deduce what the cause for this would be, but I'm guessing it could be the relative error growing as we increase the amount of time-steps we do, or alternatively that the orbital inclination of Pluto (17.14°) is big enough that it causes problems with a two dimensional model.

6 Conclusion

In conclusion, we've shown how a multi-body system in space interacts with each other. We've additionally shown that both energy and angular momentum are conserved in both circular and elliptical orbits.
Lastly we've showed that Mercury's relativistic precession can be shown to be 41" at the Perihelion over a period of 100 years.

References

[1] Horizons. URL: https://ssd.jpl.nasa.gov/horizons.cgi.

[2] Morten Hjorth-Jensen. *Project 3*. UiO, 2020.

[3] Anders Malthe-Sørenssen. *Elementary Mechanics using Python*. Springer, 2015.