FYS3150 Computational Physics - Project 3

Nicholas Karlsen

This is an abstract

INTRODUCTION

Lastly, the source code for any code discussed in this report can be found on my Github at: https://github.com/nicholaskarlsen/FYS3150

THEORY, ALGORITHMS AND METHODS

Newton's law of universal gravitation

Between every body, there is a force of attraction inversly proportional to the square of the separation, or more precisely, the force acting on some body with mass m due to a mass m' is

$$\mathbf{F} = -G\frac{mm'}{|\mathbf{r}-\mathbf{r}'|^2}\hat{\mathbf{u}}_{\mathbf{r}-\mathbf{r}'}, \quad \hat{\mathbf{u}}_{\mathbf{r}-\mathbf{r}'} = \frac{\mathbf{r}-\mathbf{r}'}{|\mathbf{r}-\mathbf{r}'|} \tag{1}$$

Where G is the gravitational constant and $\mathbf{r}, \mathbf{r}'$ denote the position vectors of bodies with mass m, m' respectively.

Chosing the the 2D cartesian coordinate system, let $\mathbf{r}-\mathbf{r}' = (x_r, y_r)$, where the r suffix denote that these coordinates are to be understood as the relative coordinates between bodies m, m'. Further, $|\mathbf{r}-\mathbf{r}'| = \sqrt{x_r^2 + y_r^2} = r$ is the distance between the two bodies.

By Newtons second law, the acceleration on body 1 due to the gravitational pull of body 2 can then be written as

$$\mathbf{a} = \frac{1}{m}\mathbf{F} = -G\frac{m'}{r^2}\frac{(\mathbf{r}-\mathbf{r}')}{r} = -G\frac{m'}{r^2}\frac{(x_r, y_r)}{r} \tag{2}$$

Written out component-wise in terms of the positions, we get the set of coupled differential equations

$$\frac{\partial^2 x}{\partial t^2} = -G\frac{m'}{r^2}\frac{x_r}{r}, \quad \frac{\partial^2 y}{\partial t^2} = -G\frac{m'}{r^2}\frac{y_r}{r} \tag{3}$$

Similar, for 3 dimensions where $\mathbf{r}-\mathbf{r}' = (x_r, y_r, z_r), r = \sqrt{x_r^2 + y_r^2 + z_r^2}$.

Relativistic Correction

The aforementioned model of gravity fails to predict the perihelion (see fig. 1) precession of mercury, which is observed to be 43" per century [2].

That is, the closed, uniform elliptical orbits predicted by the Newtonian model for gravity does not match with observations in Astronomy, where the perihelion of Mercury seems to shift over time. In fact, the perihelion

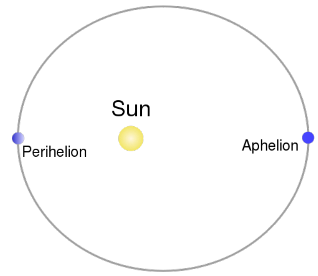


FIG. 1: In an eliptic orbit, the closest and farthest points in the orbit is defined as the Perhelion and Aphelion respectively [Image source]

precession of mercury was the first experimental confirmations of General relativity, which accurately predicts this phenomena.

And so, a correcting factor accounting for the relativistic effects is added to Newtons model, and the magnitude of the gravitational force becomes [2]

$$|\mathbf{F}| = G\frac{mm'}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right] \tag{4}$$

Where l denotes the magnitude of the angular momentum of the orbiting body and c the speed of light.

Now, in order to find the perihelion precession we define a coordinate system such that

Solving ODE numerically

Forward Euler

Consider a function $y(t)$, which derivative, $y'(t, y)$ has a known form.

Velocity-Verlet Algorithm

for $i = 0, \dots, N-1$
 $\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{v}_i\Delta t + \frac{1}{2}\mathbf{a}_i(\Delta t)^2$
 $\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{1}{2}(\mathbf{a}_{i+1} + \mathbf{a}_i)\Delta t$

Forward Euler Algorithm

for $i = 0, \dots, N-1$
 $\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}_i\Delta t$
 $\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{v}_i\Delta t$

Euler-Cromer Algorithm

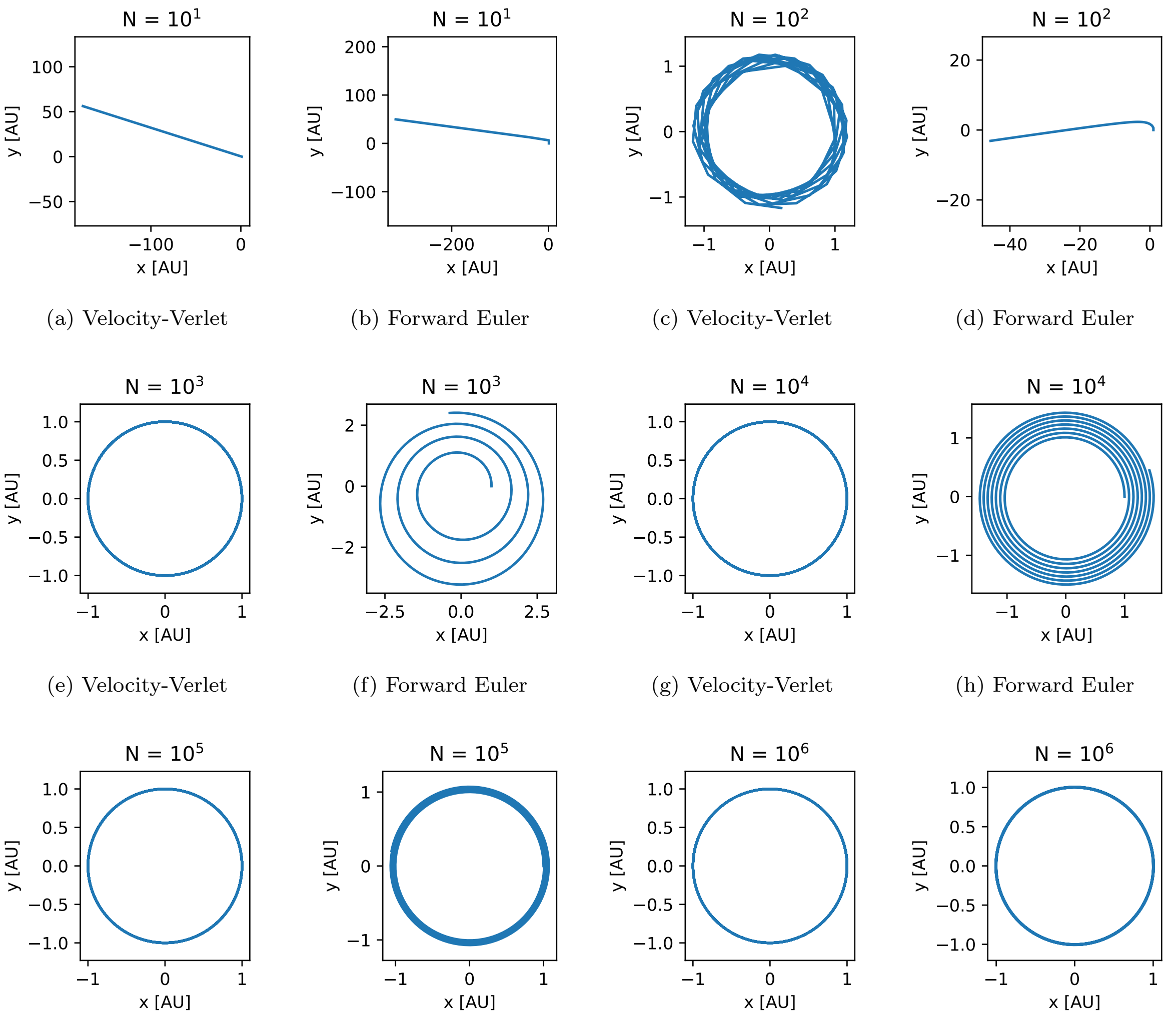
for $i = 0, \ldots, N-1$
$\quad \mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}_i \Delta t$
$\quad \mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{v}_{i+1} \Delta t$

RESULTS AND DISCUSSIONS

Object orientation

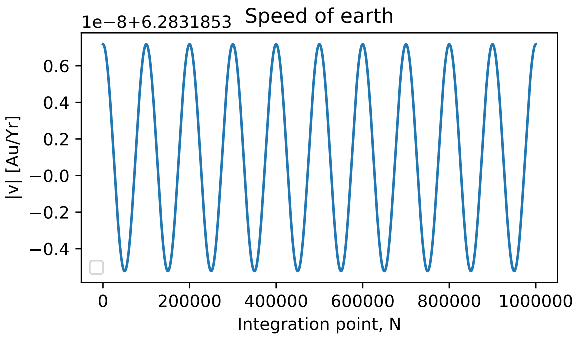
When designing the class solarsystem, i wanted to strike a balance between utilizing the benefits, and expandability you get from object orientation whilst also not abstracting the data too much. As such, the class simply manipulates arrays of a particular format in a particular way. Not abstracting the process by creating objects for each planet (or something else of that nature), which seems to be a pitfall of object orientation as i percieve it.

A particular benefit in the way i wrote my code, is that there is no difference if the input data is in 2 Dimensions or 3. The code will work just the same either way by making full use of the way Numpy arrays work.

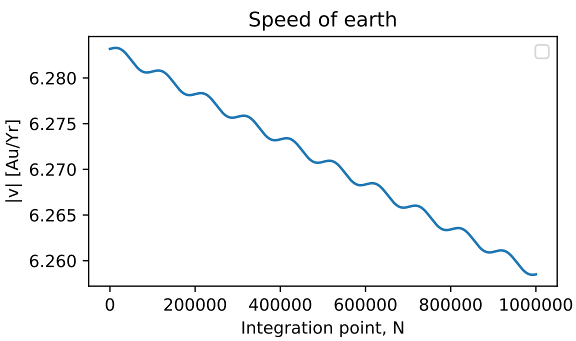


(a) Velocity-Verlet (b) Forward Euler (c) Velocity-Verlet (d) Forward Euler
(e) Velocity-Verlet (f) Forward Euler (g) Velocity-Verlet (h) Forward Euler
(i) Velocity-Verlet (j) Forward Euler (k) Velocity-Verlet (l) Forward Euler

FIG. 2: The orbit of earth around a stationary sun for a timeperiod of 10 Years with simulated with a varying number of integration points N (see fig titles) using the Velocity-Verlet and Forward Euler algorithms



(a) Velocity-Verlet



(b) Forward Euler

FIG. 3: The fluctuation of the total energy of the Earth in the Earth-Sun system for solutions using the Velocity-Verlet algorithm (a) and the Forward Euler algorithm (b) in a 10 Year simulation

CONCLUSIONS

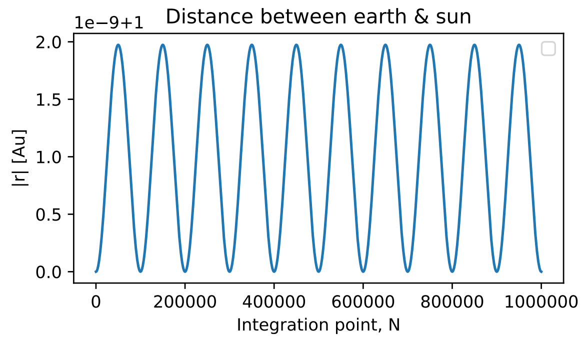
In trying to generalize my code through the use of object orientation, i had to strike a balance between generality and complexity

[1] M. Hjorth-Jensen, Computational Physics - Lecture Notes 2015, (2015).

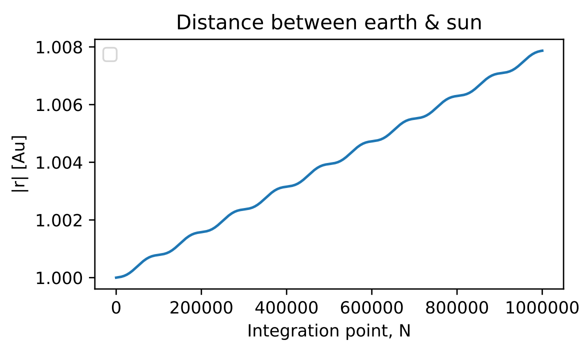
[2] M. Hjorth-Jensen, Building a model for the solar system using ordinary differ- ential equations - Project 3 (2018)

Fetching data from Horizons using horizons.py

In order to streamline the process of fetching data from the Horizons system i created a small script, `horizons.py` that utilizes the Astroquery python package and returns only the select data that i am interested in. The function, `fetch_data` takes input as a dictionary, rather than a list of id numbers. Whilst this may not be as extensible or practical, it makes the code easier to read and understand, by making the instant connection between the planet name and id. For the purposes of this project, i find that much more valuable since i am only dealing with a limited amount of planets anyway.

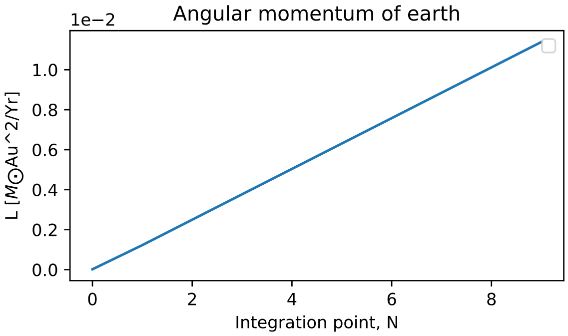


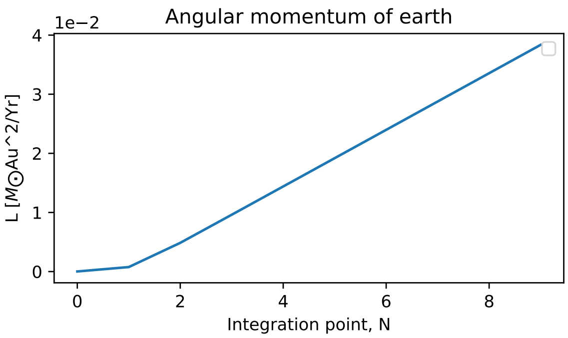
(a) Velocity-Verlet

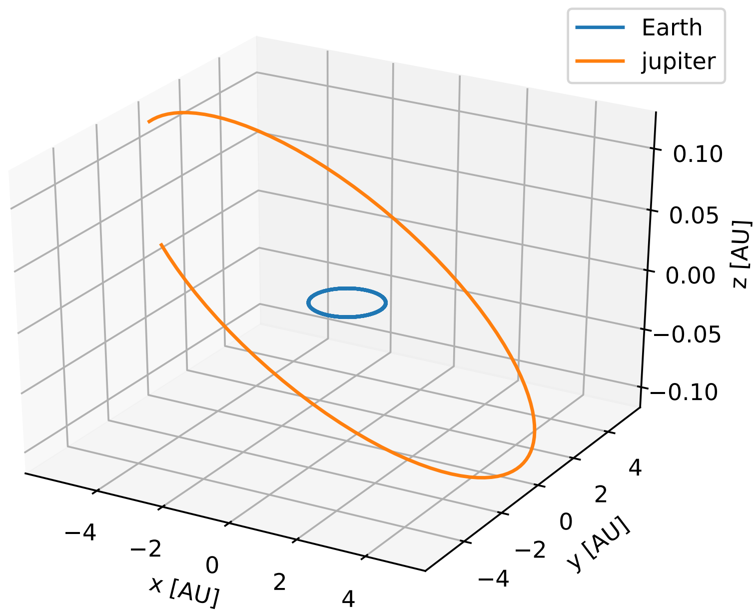


(b) Forward Euler

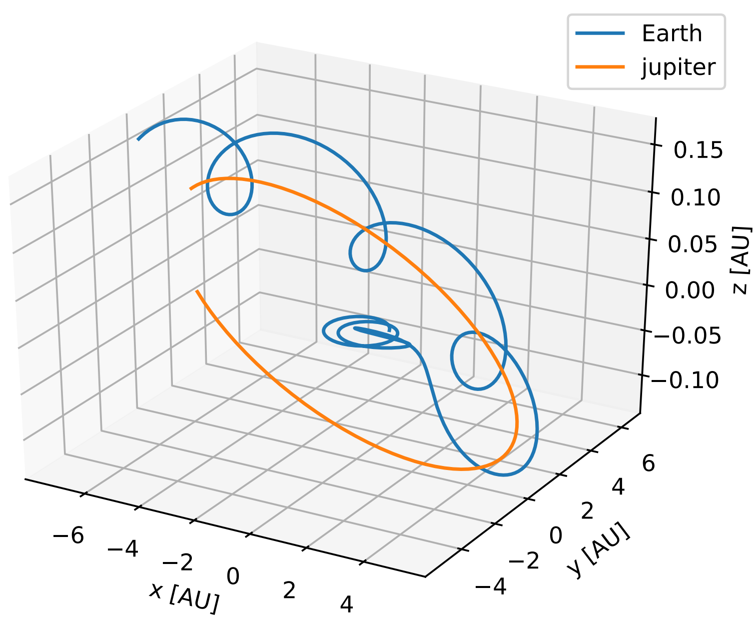
FIG. 4: The fluctuation of the total energy of the Earth in the Earth-Sun system for solutions using the Velocity-Verlet algorithm (a) and the Forward Euler algorithm (b) in a 10 Year simulation


Angular momentum of earth
1e−2
1.0
0.8
0.6
0.4
0.2
0.0
L [$M_\odot$Au^2/Yr]
0
2
4
6
8
Integration point, N

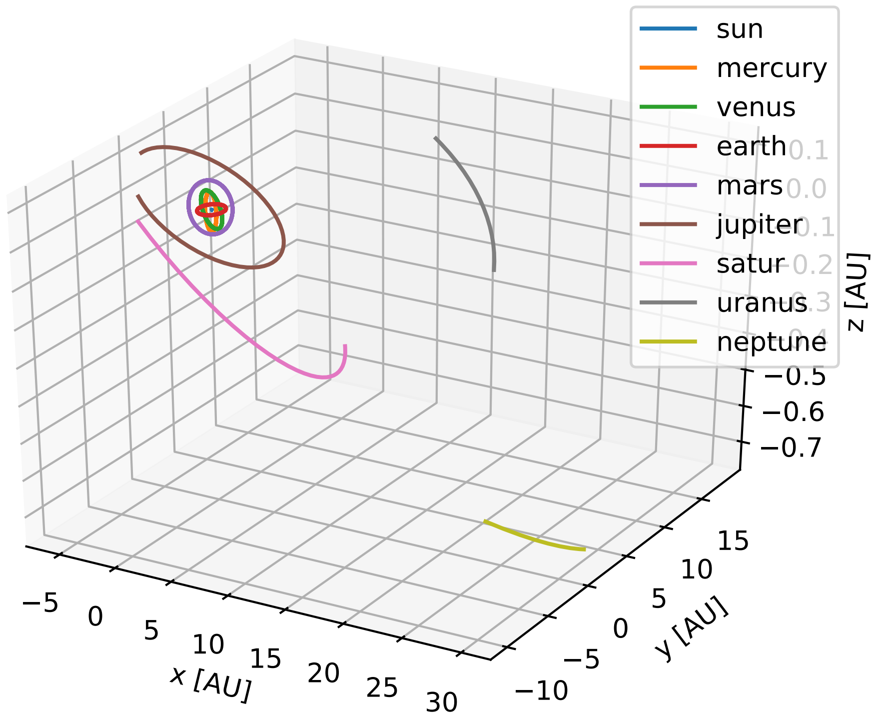

Angular momentum of earth
1e−2
4
3
2
1
0
L [$M_\odot$Au^2/Yr]
0
2
4
6
8
Integration point, N


Earth
jupiter
0.10
0.05
0.00
−0.05
−0.10
z [AU]
−4
−2
0
2
4
x [AU]
−4
−2
0
2
4
y [AU]

(a)


Earth
jupiter
0.15
0.10
0.05
0.00
−0.05
−0.10
z [AU]
−6
−4
−2
0
2
4
x [AU]
−4
−2
0
2
4
6
y [AU]

(b)


sun
mercury
venus
earth
mars
jupiter
satur
uranus
neptune
x [AU]
y [AU]
z [AU]
−5
0
5
10
15
20
25
30
−10
−5
0
5
10
15
0.1
0.0
−0.5
−0.6
−0.7