Building a model for the solar system using ordinary differential equations

Johannes Sørby Heines

October 26, 2020

Abstract

We developed a program to simulate a system of objects interacting via classical gravity, using different differential equation solver algorithms. The program was applied to the Earth-Sun system, showing a relative increase of 67% in energy and 70% in negative angular momentum for the Forward Euler algorithm, and conservation with the Euler-Cromer and velocity Verlet algorithms. The latter was found to be the most accurate. The program was also used to show the effect of Jupiter on the Earth's orbit, and to simulate all planets in the solar system, and Pluto.

Introduction

The motion of celestial objects is usually determined only by the force of gravity. Since gravity has infinite range, every body in a system interacts with every other body in that system. This makes analytical calculations impossible in all but a very few highly specific cases. The goal of this project was to implement a program to simulate the motion of objects in space numerically, and apply it to the planets in the Solar system.

The next section describes our model of the solar system, outlines the numerical algorithms for solving it and their implementation, and summarizes the structure of the program. We then compare the algorithms for a two-body problem, study a three-body problem and apply the program to the solar system. Finally we discuss shortcomings of the program and suggest improvements.

Theoretical models and implementation

All code can be found, along with unit tests and benchmarks at `https://github.com/johashei/ComputationalPhysics/tree/master/doc/Projects/2020/Project3/code`.

The solar system

The bodies in the solar system can be modeled as point particles interacting gravitationally. The newtonian gravitational force between two bodies of mass m_1 and m_2 is

$$F = G\frac{m_1 m_2}{r^2},$$

where G is the gravitational constant and r is the distance between the bodies.

In reality the gravitational interaction is described by the general theory of relativity. However, for the solar system, the newtonian model is a good approximation for all planets except Mercury. Indeed, observed perihelion precession of Mercury is $43''$ per century larger than can be explained by classical effects, such as the gravitational force from the other planets [1].

The effect of general relativity can be accounted for by adding a correction term to the force, giving [1]

$$F = G\frac{m_\odot m_{\text{Mercury}}}{r^2}\left[1 + \frac{3l^3}{r^2c^2}\right],$$

where $m_\odot$ is the mass of the Sun, m_{Mercury} is the mass of Mercury, $l = |\vec{r} \times \vec{v}|$ is Mercury's orbital angular momentum and c is the speed of light in vacuum.

We began by creating a program for simulating the solar solar system classically. And then attempted to study the effect of the general relativity correction above on Mercury.

Measuring the effect of the general relativity correction requires the a time resolution such that the perihelion precession for the classical case is more than one order of magnitude smaller than what is expected with the relativistic correction. In other words, it must be of the order of $10^{-2}\,'' \cdot \text{yr}^{-1}$ or smaller.

Numerical integration methods

The movement of a body of mass m under the influence of a conservative force $\vec{F}$ is given by the second order differential equation

$$\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} = \frac{\vec{F}}{m}.$$

In the case of gravity, the force depends on the position and mass of all bodies in the system. There are multiple ways of solving such a problem numerically. We studied Forward Euler, Euler-Cromer and velocity Verlet. They all separate the equation into two first order differential equations, and approximate these by Taylor expansions.

$$x(t \pm h) = x(t) \pm h\dot{x}(t) + \frac{h^2}{2}\ddot{x}(t) \pm \mathcal{O}\big(h^3\big)$$
$$\dot{x}(t \pm h) = \dot{x}(t) \pm h\ddot{x}(t) + \frac{h^2}{2}x^{(3)}(t) \pm \mathcal{O}\big(h^3\big)$$

The Forward Euler method uses only a first order Taylor expansion. For M objects and N time steps, the algorithm is

```
for i = 0:N do
    for j = 0:M do
        $a_{i,j} = a(\vec{x}_i, j)$
        $v_{i+1,j} = v_{i,j} + h \cdot a_{i,j}$
        $x_{i+1,j} = x_{i,j} + h \cdot v_{i,j}$
    end for
end for
```

Here $v = \dot{x}$ is the velocity, $a = \ddot{x}$ is acceleration, and $\vec{x}_i$ contains the positions of all objects at time point i.

While this algorithm is very simple to implement, the error in each time step is $\mathcal{O}\big(h^2\big)$, which over $N \sim 1/h$ time steps gives a global error $\mathcal{O}(h)$. In addition, it can be shown that this algorithm does not conserve the energy of the system [2].

This last point is addressed by the Euler-Cromer algorithm. By changing the second expression to $x_{i+1,j} = x_{i,j} + h \cdot v_{i+1,j}$, the algorithm becomes energy conserving [2]. The global error, however, remains the same.

The velocity Verlet method uses second order Taylor approximations of x and v, with a first order Taylor approximation of $\ddot{v}$. The algorithm is [2]

```
for i=0:N do
    for j=0:M do
        $x_{i+1,j} = x_{i,j} + h \cdot v_{i,j} + \frac{h^2}{2}a_{i,j}$
    end for
    for j=0:M do
        $a_{i+1,j} = a(\vec{x}_{i+1}, j)$
    end for
    for j=0:M do
        $v_{i+1,j} = v_{i,j} + \frac{h}{2}(a_{i+1,j} + a_{i,j})$
    end for
end for
```

Note here that the position of every body must be calculated before calculating the velocities. In this method, the local error in each iteration is $\mathcal{O}\big(h^3\big)$, giving a global error of $\mathcal{O}\big(h^2\big)$. It can also be shown that this algorithm conserves energy [2].

Since no external force is acting on the solar system, the total energy E and angular momentum $\vec{L}$ are conserved. Calculating the change of these quantities over the course of the simulation is therefore a good way of comparing the accuracy of different algorithms.

The differences between the three algorithms were studied in a two-body system, with the Earth in a circular orbit around the sun. Here we chose a small number of time steps (10^4 over 10 years) in order to see the differences clearly. We then added Jupiter, also in a circular orbit, and observed the effects of changing Jupiter's mass from $1.9 \times 10^{27}\,\mathrm{kg} \approx M_\odot/1000$ to $1.9 \times 10^{30}\,\mathrm{kg} \approx M_\odot$.

Object orientation

Our goal was to write a general program which could simulate any three dimensional system of point particles. The first step was to create a class for defining a point particle. An instance of the PhysicsObject class contains position and velocity vectors and mass. The vectors are defined with the vec3 class, which adds vector operations to the C++ vector class. The PhysicsObject class is completely inlined, allowing it to be accessed within vectorized loops.

The simulation of the system is done by the PhysicsSimulator class. Since the numerical integration methods studied here, that is Euler, Euler-Cromer and velocity Verlet, require few lines to implements, but access to multiple class variables, we chose against creating a dedicated solver class.

The calculations of energy an angular momentum are done in a separate Tests class. This class is initialized with a reference to a PhysicsSimulator instance. It then calculates and compare E and $\vec{L}$ of the first and final time.

Results

Most of the effect of the algorithms are best illustrated using only one planet. Figure 1 shows the result of simulating the orbit of the Earth over 10 years with the three algorithms. The Earth has been given a circular orbit, so the total energy of the system is

$$\begin{aligned} E &= \frac{1}{2} M_\oplus v^2 - G\frac{M_\odot M_\oplus}{d^2} \\ &= 5.92 \times 10^{-5}\,\mathrm{M_\odot AU^2 yr^{-2}}, \end{aligned} \tag{1}$$

and the total angular momentum

$$\begin{aligned} \vec{L} &= M_\oplus \vec{r} \times \vec{v} \\ &= [0, 0, -1.88 \times 10^{-5}]\,\mathrm{M_\odot AU^2 yr^{-1}}. \end{aligned} \tag{2}$$

Here $\vec{r}$ and $\vec{v}$ are the position and velocity of the Earth relative to the Sun, respectively.

Figure 2 shows the effect of introducing a third object to the system.

Figure 3 shows plots of the planet orbits calculated over 250 years using 10^7 time steps with the velocity Verlet method.

We did not observe the expected effect of the relativistic correction on Mercury's orbit. When performing the simulation outside the class and noting the perihelion at each rotation, we observed an uneven oscillation, as shown in table 1.

Table 1: The simulated perihelion angle in arc seconds with 10^8 time steps over 1 year

time	no correction	correction
0.240732	−0.0216691	−0.0411645
0.481463	0.0399781	0.000987479
0.722195	0.0180526	−0.0404334
0.962927	−0.00400282	0.0014609

Discussion

We observed in figure 1, as expected, that the Forward Euler method causes the energy to increase and the angular momentum to decrease as the planets spiral outwards. The relative change is 67% in energy and 70% in angular momentum For the Euler-Cromer and velocity Verlet methods, the changes in energy and angular momentum are over ten orders of magnitude smaller then the values. This is small enough to be consistent with the expected conservation. The changes are still considerably smaller with the velocity Verlet method, showing that it is indeed more accurate.

In figure 2 we see that while both the Earth and Jupiter were initialized to form circular orbits on their own, neither of them form perfect circular orbits in the three body system. As the mass of Jupiter increases, the radius of the Sun's orbit increases, and the Earth's orbit becomes elliptical with a perihelion precession. This is clearly visible in figure 2c. When the mass of Jupiter approximately equals that of the Sun (figure 2d), they form a binary system of rotation around the center of mass, while the Earth seems to move chaotically before it is ejected. In this case the program doesn't conserve energy. This likely happens as the Earth gets very close to the Sun at $(-0.5, 1.5)$. The acceleration would be too large compared to the time step, resulting in the observed increase in energy.

Over one year, we expect, for Mercury, to observe a perihelion precession of $0.45''$ when the relativistic correction is applied. Instead we observe, with or without the correction, an oscillation with amplitude of the order $10^{-2}\,''$. The relativistic correction seems to increase the amplitude slightly, but there is no indication of a consistent perihelion precession. We obtained similar behaviour when using a longer time or an increased correction, as well as when switching to the Euler-Cromer algorithm. We have not been able to identify the error.

Simulating complete orbits of the outermost planets, and Pluto, requires a simulation time of multiple centuries. Since we use the same time steps for all planets, this results in inaccurate orbits for the inner planets with the number of time steps we are capable of simulating. In order to get accurate orbits for the inner planets, the simulation time must be reduced below ten years. This is a major weakness of the program, and potential solutions are discussed below.

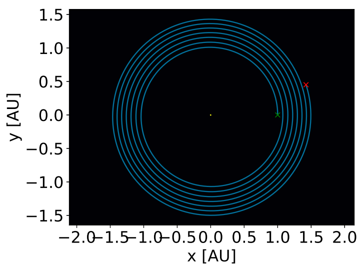
The accuracy of the program is severely limited in the number of time steps which can be simulated. We present here some possible improvements. The current program stores all positions and velocities within the PhysicsSimulator class, to be extracted after the simulation has been completed. In order to save memory and allow for simulations with more time steps, it would have been better to store only the values used by the class, and extract the values required by the main program at each step. This would remove memory as a limiting factor on the number of time steps. Another improvement would be to employ adaptive methods. These adapt the time step to motion of the objects, resulting in consistent accuracy over the entire solar system.

Conclusion

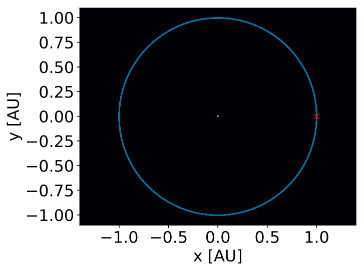
The PhysicsSimulator class can simulate a system of multiple bodies interacting via classical gravity using the Euler, Euler-Cromer and velocity Verlet algorithms. Applying the program to a two body system showed that using Euler's method resulted in the bodies spiraling outwards, thus increasing energy and decreasing the angular momentum, with a relative change of 67% and 70%, respectively. The two other algorithms both conserve energy and angular momentum, and we confirmed that velocity Verlet is more accurate than Euler-Cromer.

Simulations of the three body system consisting og the Sun, Earth and Jupiter showed that Jupiter's influence on the Earth's orbit results noticeable perihelion precession.

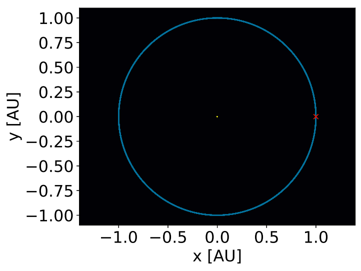
When simulating the entire solar system, the time required to get full orbits of the outermost planets meant our program lost the time resolution necessary to accurately simulate the innermost planets. We've outlined potential ways to fix this problem. The limit on the number of time steps can be removed by restructuring the data flow, and the time resolution for the inner planets can be improved by implementing adaptive methods.



(a) Forward Euler method
$\Delta L = -4.23\times10^{-6}\,\mathrm{M}_\odot\mathrm{AU}^2\mathrm{yr}^{-1}$
$\Delta E = 1.97\times10^{-5}\,\mathrm{M}_\odot\mathrm{AU}^2\mathrm{yr}^{-2}$

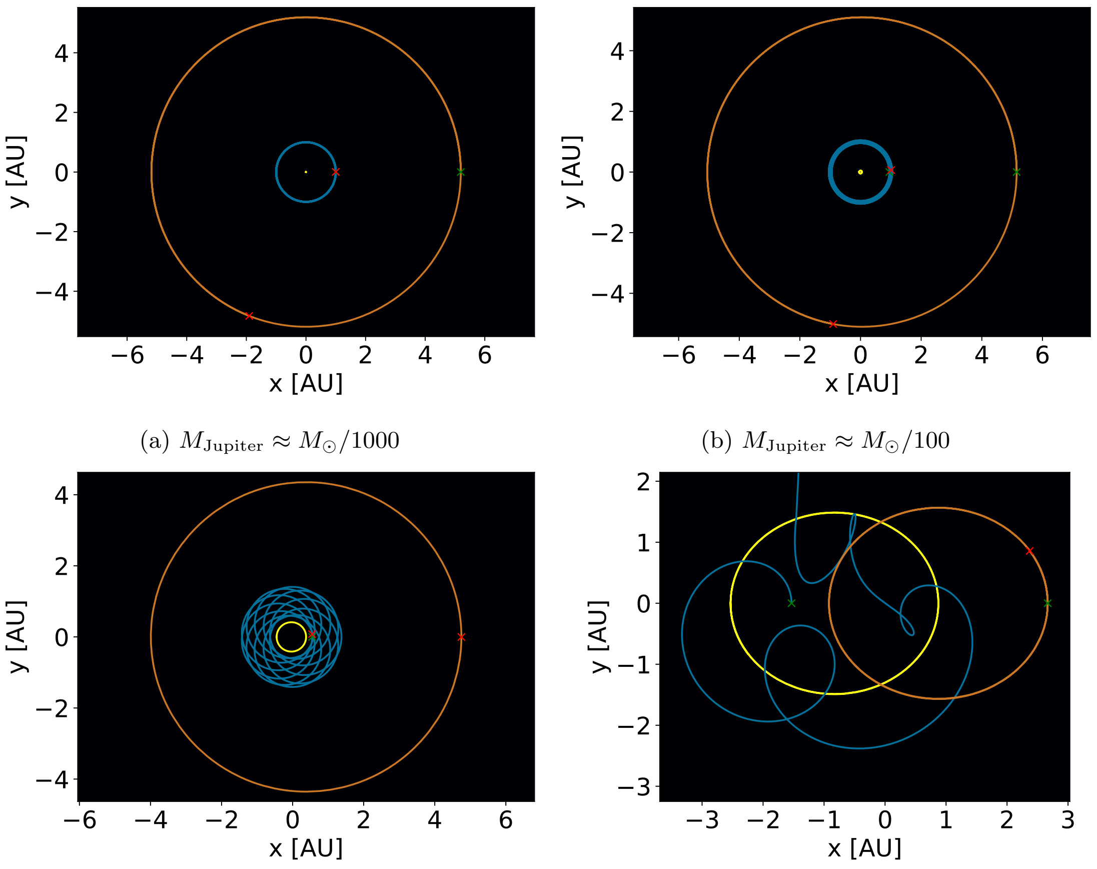


(b) Euler-Cromer method
$\Delta L = 1.12\times10^{-19}\,\mathrm{M}_\odot\mathrm{AU}^2\mathrm{yr}^{-1}$
$\Delta E = 1.19\times10^{-15}\,\mathrm{M}_\odot\mathrm{AU}^2\mathrm{yr}^{-2}$



(c) Velocity Verlet method
$\Delta L = -7.12\times10^{-20}\,\mathrm{M}_\odot\mathrm{AU}^2\mathrm{yr}^{-1}$
$\Delta E = 4.13\times10^{-19}\,\mathrm{M}_\odot\mathrm{AU}^2\mathrm{yr}^{-2}$

Figure 1: Plots of the Sun-Earth system obtained with the different algorithms. All simulations used 10^4 time steps over 10 years. The start and end points are marked by a green and a red cross ($\times$), respectively. The total change in angular momentum L and energy E are given in each case.



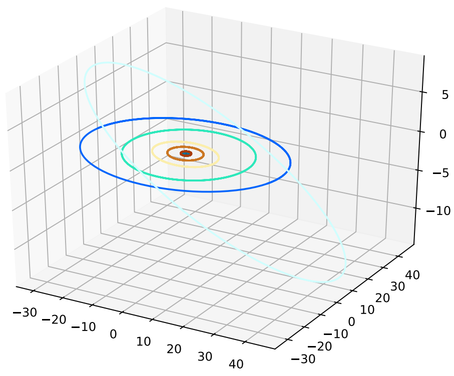
(a) $M_{\text{Jupiter}} \approx M_\odot/1000$

(b) $M_{\text{Jupiter}} \approx M_\odot/100$

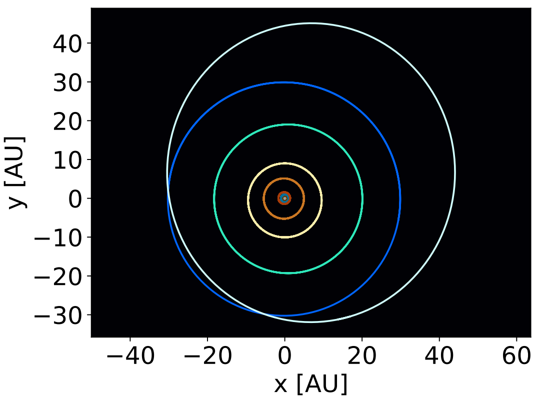
(c) $M_{\text{Jupiter}} \approx M_\odot/10$

(d) $M_{\text{Jupiter}} \approx M_\odot$

Figure 2: Plots of the Sun-Earth-Jupiter system for different masses of Jupiter, obtained with the velocity Verlet method with 10^7 time steps. In the first two cases, energy and angular momentum are conserved. In the last, angular momentum is conserved, but $\Delta E = 0.60\,\mathrm{M_\odot AU^2 yr^{-2}}$



(a) 3D



(b) 2D plot in the xy-plane.

Figure 3: Plots in 3D and 2D of the planet orbits calculated over 250 years using 10^7 time steps with the velocity Verlet method.

References

[1] Morten Hjorth-Jensen. Project 3. 2020. URL: `http://compphysics.github.io/ComputationalPhysics/doc/Projects/2020/Project3/html/Project3.html`

[2] Morten Hjorth-Jensen. Computational Physics Lectures: Ordinary differential equations. 2020. URL: `http://compphysics.github.io/ComputationalPhysics/doc/pub/ode/html/ode.html`