Project 3 FYS4150
Ordinary differential equations

Audun Tahina Reitan and Marius Holm

October 23, 2018

1 Abstract

We give a brief introduction to differential equations and how we can discretize such equations in order to solve them using computer algorithms.

More of our results can be found in the GitHub repository linked in the implementation section.

2 Introduction

In this project we'll develop code for simulating the solar system, using the Verlet algorithm for solving coupled ordinary differential equations. For this project we will focus on the use of classes in our code, which makes it a lot easier to reuse larger parts of our code for problems of similar character. In our case of the solar system we can create a class describing a planet, a moon, or some other astronomical body which we want to include in our solar system.

In order to test that our algorithm works, we start by looking at a simple hypothetical system consisting of only the Earth orbiting the Sun. We then investigate the stability of our algorithm and plot the position of the Earth orbiting the Sun. Furthermore we consider the initial velocity needed for the planet to escape the gravity of the Sun. Then we'll consider a three-body system consisting of the Earth, the Sun, and Jupiter. We then expand our three-body model to the entire solar system including Pluto. In the final part of our project we look at the perihelion precession of Mercury.

3 Methods

3.1 Differential equations

The order of an ordinary differential equation (ODE) is given by the highest order of derivative found in the left-hand side of the equation. A first order differential equation is typically of the form

$$\frac{dy}{dt} = f\big(t, \frac{dy}{dt}, y\big) \tag{1}$$

where f is an arbitrary function. A well known second order differential equation is Newton's second law

$$m\frac{d^2x}{dt^2} = -kx \tag{2}$$

where k is the force constant. All ODE's depend on one variable only, compared to partial differential equations which can depend on several variables. PDE's are widely used, but won't be of any concern in this project.

3.2 Newton's law of gravitation

The first part of our project we take a look at a simplified solar system consisting of the Sun and the Earth. Newton's law of gravitation then takes the following form:

$$F_G = \frac{GM_\odot M_{Earth}}{r^2}, \tag{3}$$

where $M_\odot$ is the mass of the Sun and M_{Earth} is the mass of the Earth. G is the gravitational constant and r is the distance between the Earth and the Sun.

We assume that the orbit of the Earth around the sun is co-planar, and we let this be the xy-plane. We can then use Newton's second law of motion which gives us two differential equations, one for each coordinate.

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{Earth}}, \tag{4}$$

and

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{Earth}}, \tag{5}$$

where $F_{G,x}$ and $F_{G,y}$ are the x and y components of the gravitational force.

3.3 Useful constants

During our project we will use astronomical units (AU) as a measure of length. 1 AU is defined as the average distance between the Sun and the Earth, that is 1 AU $= 1.5 \times 10^{11}$ m. The mass of the Sun is $M_{\text{sun}} = M_\odot = 2 \times 10^{30}$ kg. We will also need the mass of the planets we include in our code, which are given in table 1. In our code we have defined the unit mass to be equal to $M_\odot$, i.e. $M_\odot = 1$. We therefore also present the planetary masses given in the unit $M_\odot$.

Table 1: Mass and distance from the Sun for each planet in our solar system.

Planet	Mass in kg	Mass in unit M_{Sun}	Distance to sun in AU
Earth	$M_{\text{Earth}} = 6 \times 10^{24}$ kg	3.04×10^{-6}	1 AU
Jupiter	$M_{\text{Jupiter}} = 1.9 \times 10^{27}$ kg	9.54×10^{-4}	5.20 AU
Mars	$M_{\text{Mars}} = 6.6 \times 10^{23}$ kg	3.23×10^{-7}	1.52 AU
Venus	$M_{\text{Venus}} = 4.9 \times 10^{24}$ kg	2.45×10^{-6}	0.72 AU
Saturn	$M_{\text{Saturn}} = 5.5 \times 10^{26}$ kg	2.86×10^{-4}	9.54 AU
Mercury	$M_{\text{Mercury}} = 3.3 \times 10^{23}$ kg	1.66×10^{-7}	0.39 AU
Uranus	$M_{\text{Uranus}} = 8.8 \times 10^{25}$ kg	4.37×10^{-5}	19.19 AU
Neptune	$M_{\text{Neptune}} = 1.03 \times 10^{26}$ kg	5.15×10^{-5}	30.06 AU
Pluto	$M_{\text{Pluto}} = 1.31 \times 10^{22}$ kg	7.40×10^{-9}	39.53 AU

We include Pluto, even though it is technically not regarded as a planet anymore.

3.4 Discretizing differential equations

We assume that Earth's orbit is circular around the Sun. For circular motion we have

$$F_G = \frac{M_{\text{Earth}} v^2}{r} = \frac{G M_\odot M_{\text{Earth}}}{r^2}, \tag{6}$$

where v is Earth's velocity. By simple algebraic manipulation we find the relationship

$$v^2 r = G M_\odot = 4\pi^2 \text{ AU}^3/\text{yr}^2 \tag{7}$$

By defining $M_\odot = 1$ we have $G = 4\pi^2 \text{ AU}^3/\text{yr}^2$.

3.4.1 Euler's forward algorithm

Suppose we have an initial function value $y(t_0) = y(t = t_0)$ for a function $y(t)$. The function is defined in the well-behaved domain $[a, b]$. By splitting the domain into N sub intervals we find the step size h as

$$h = \frac{b-a}{N}. \tag{8}$$

Using the step size, h, and the derivative of y we can find an expression for the function at the next time step as $y_1 = y(t_1 = t_0 + h)$. We can generalize this method which gives us a general procedure for finding the step y_{i+1} in terms of the previous time step.

$$y_{i+1} = y(t = t_i + h) = y(t_i) + h\Delta\big(t_i, y_i(t_i)\big) + O(h^{p+1}), \tag{9}$$

where $O(h^{p+1})$ represents the truncation error. Using Taylor expansion we find

$$\Delta(t_i, y_i(t_i)) = (y'(t_i) + \cdots + y^{(p)}(t_i)\frac{h^{p-1}}{p!} \tag{10}$$

By defining $y'(t_i) = f(t_i, y_i)$ and truncating Δ at the first derivative, we have

$$y_{i+1} = y_i + hf(t_i, y_i) + O(h^2) \tag{11}$$

which combined with $t_{i+1} = t_i + h$ gives us Euler's forward algorithm.

In our case we want to apply Euler's forward algorithm to Newton's

3.4.2 Velocity Verlet method

The final equations for position and velocity are then given by

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}v_i^{(1)} + O(h^3), \tag{12}$$

and

$$v_{i+1} = v_i + \frac{h}{2}\left(v_{i+1}^{(1)} + v_i^{(1)}\right) + O(h^3) \tag{13}$$

3.5 Implementation

.[Hjort-Jensen, 2015]

All our code, calculations, and plots used can be found in Auduns GitHub repository.

4 Results

4.1 Preservation of dot product

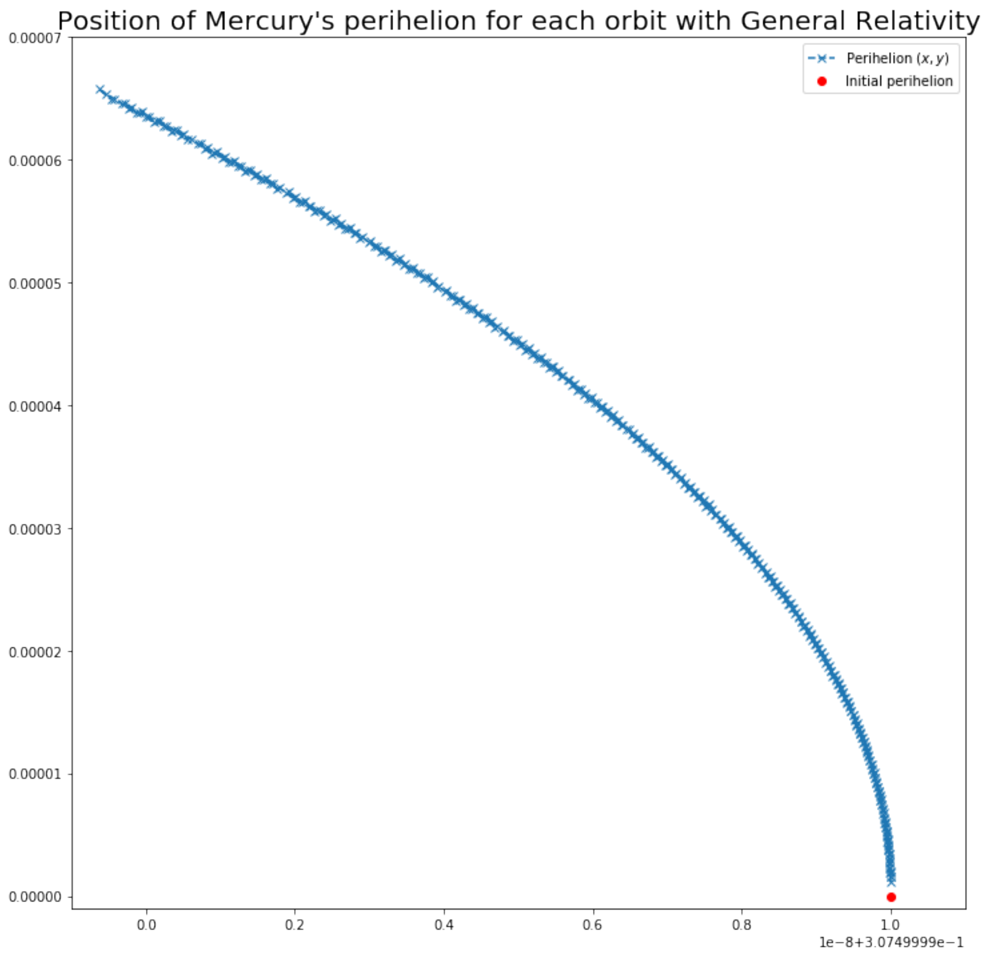

Figure 1: General relativity Mercury.

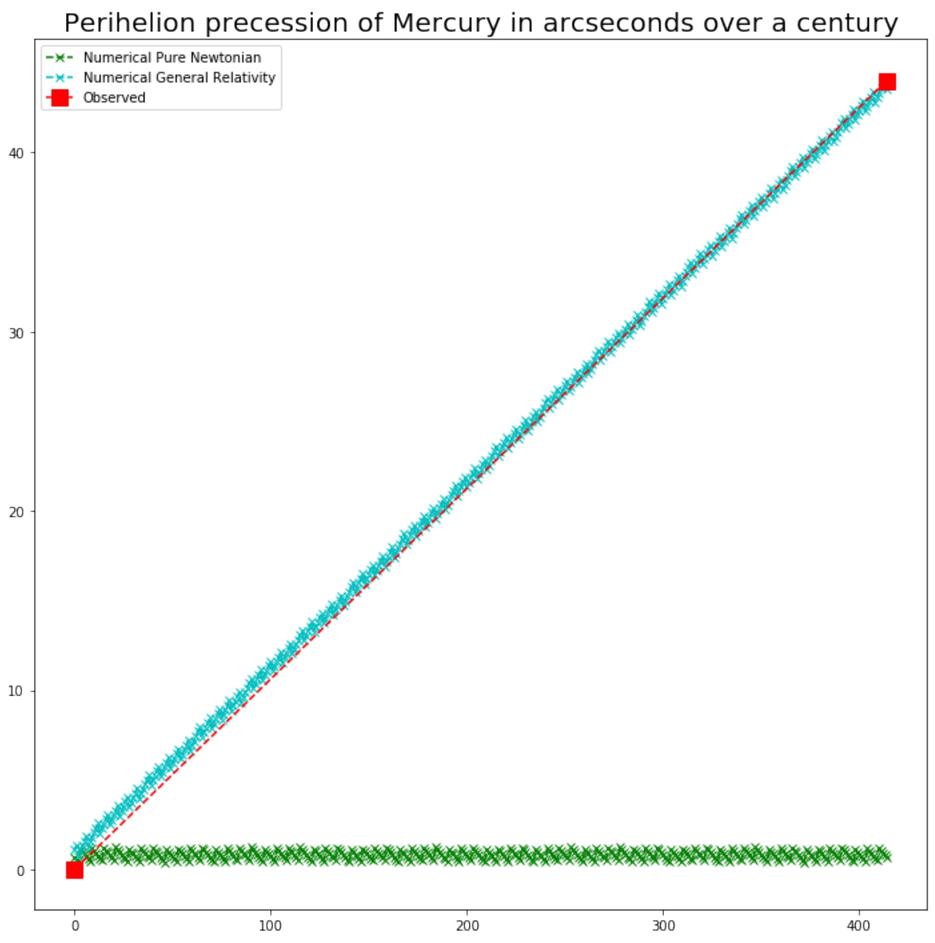

Figure 2: Perihelion precession of Mercury.

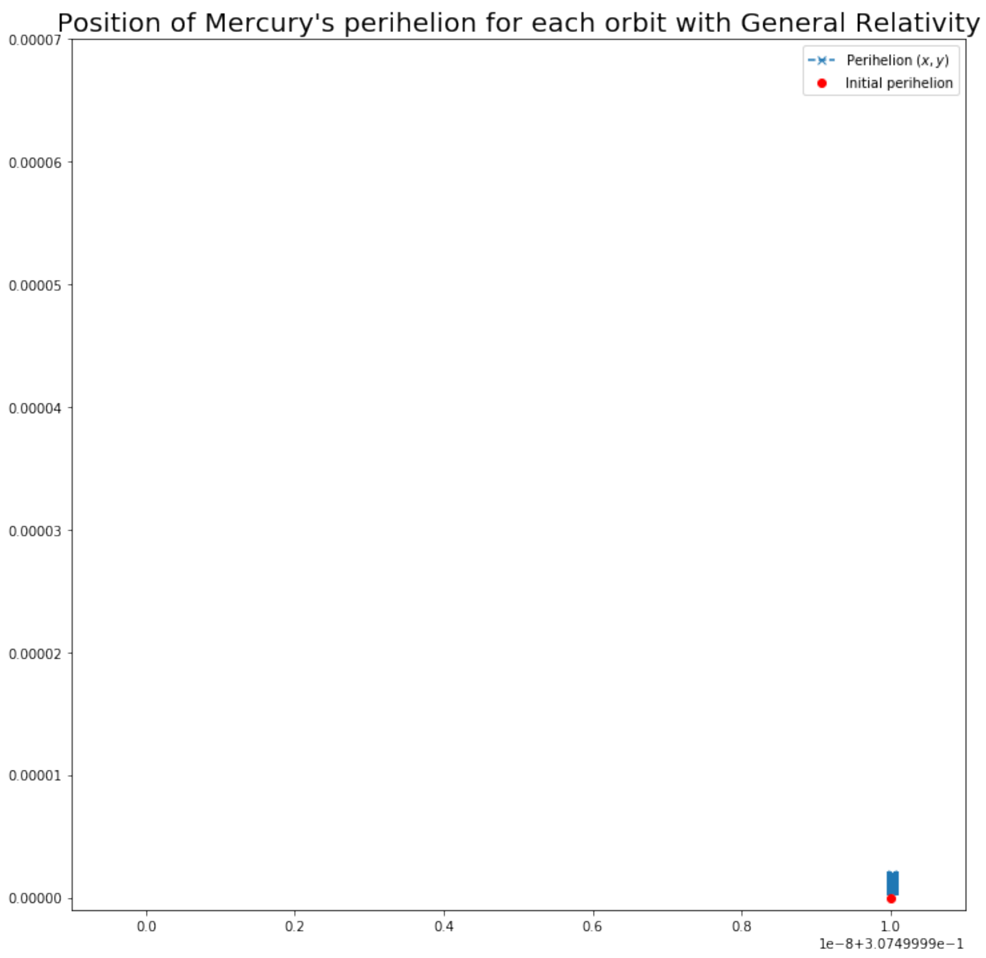

Figure 3: Newtonian Mercury.

Testing Anders' code for automatic inclusion of data to table.

Table 2: The greatest table.

n	Relative error
$1 \cdot 10^1$	$6.61 \cdot 10^{-2}$
$1 \cdot 10^2$	$8.13 \cdot 10^{-4}$
$1 \cdot 10^3$	$8.32 \cdot 10^{-6}$
$1 \cdot 10^4$	$8.32 \cdot 10^{-8}$

4.2 Eigenvalues of one electron in the harmonic oscillator

5 Discussion

5.1 Jacobi method

References

[Hjort-Jensen, 2015] Hjort-Jensen, M. (2015). Computational physics. lecture notes. Accessible at course github repository. 551 pages.