Project 3: Ordinary Differential Equations and the Solar System

Michael Saybolt
Bjon Charlery

5/2/2017

Contents

Abstract

This project explored a model of the solar system created by solving Ordinary Differential Equations (ODE). Numerical integration was done using the Euler method, as well as a Verlet method. In addition, emphasis was placed on object orienting the code for modularity, and continuously testing as the code is developed to stop bugs before they grow or are harder to find.

1 Introduction

The objective of this project was to solve coupled differential equations responsible for the trajectory of planets around the solar system. Initial conditions for the solver were extracted from the following NASA website http://ssd.jpl.nasa.gov/horizons.cgi#top .

For ease of debugging, a simulation of a simple binary system is created with just the Sun and the Earth. The only force in the problem is gravity. Newton's law of gravitation is given by a force F_G

$$F_G = \frac{GM_{\odot}M_{\text{Earth}}}{r^2}, \tag{1}$$

where $M_{\odot}$ is the mass of the Sun and M_{Earth} is the mass of the Earth. The gravitational constant is G and r is the distance between the Earth and the Sun. Eventually the other planets will be added, but for now just this system will be used to test the layout of the algorithms.

2 Description

This section will provide details about the general equations describing the sytem, and the general approach of the problem.

2.1 Block tied to wall

To refresh understanding of differential equations and how to code them, the block tied to a wall example was copied and analyzed.
Newton's equation of motion for system:

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

aka:

$$\frac{d^2x}{dt^2} = -\frac{k}{m}x = -\omega_0^2 x,$$

with the angular frequency $\omega_0^2 = k/m$. This DE has solutions that can be obtained analytically of the form

$$x(t) = Acos(\omega_0 t + \nu),$$

where A is the amplitude and ν the phase constant.

see line 1116 of ode.tex @Author: mhjensen for more equation

2.2 Euler Method

The Euler method is good for a quick and dirty way to see approximately what a solution looks like, at the expense of accuracy and numerical stability. There is also an enhanced Euler method, which offers better numerical stability that was not officially covered in this unit, but was discovered accidentally while making code. This is discussed later in the results setcion.

2.3 Verlet Method: Velocity Verlet

The Verlet method is another method of numerical integration, and offers good numeric stability thus is usually accurate. The Velocity Verlet method specifically is covered in this report, as that is the implementation chosen to solve this problem.

3 Implementation

This section will transform the equations described in the previous section into usable code, and also describe difficulties encountered, and implemented solutions. For simplification in understanding, a 2D simplification of applying Newton's law of gravitation is studied. Recalling that Newton's law of gravitation is (1), the position and velocity can be described by 4 coupled DEs:

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x, \tag{2}$$

$$\frac{dx}{dt} = v_x, \tag{3}$$

$$\frac{dv_y}{dt} = -\frac{GM_\odot}{r^3}y,$$

$$\frac{dy}{dt} = v_y$$

This would be the same for 1 or 3 dimensions. But,

$$GM_\odot = v^2 r,$$

and assuming Earth's velocity is: $v = 2\pi r/\text{yr} = 2\pi\text{AU/yr}$

Thus can equate the following for the acceleration in some (x) direction:

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x = 4\pi^2\frac{(\text{AU})^3}{\text{yr}^2} \tag{4}$$

replacing equation (2).

3.1 Euler

Descretizing the domain into timesteps h, the Euler method describes the next step's position and velocity as:

$$x_{i+1} = x_i + hv_i \tag{5}$$

$$v_{i+1} = v_i - \frac{d(v_i)}{dt} \tag{6}$$

where x signifies position, and v signifies velocity in some direction at time step index i.

The result of equation (4) can be substituted into (6), resulting in

$$v_{i+1} = v_i - h(4\pi^2)\frac{x_i}{r_i^3} \tag{7}$$

insert code

3.1.1 Enhanced Euler

The task for the Euler method was to use the regular Euler method. According to the notes, and equations (5) and (6), which originate from the notes, or (7), the position and velocity are updated simultaneously. Because code is sequential, the position must be stored before it is updated, so that it can be used in the velocity. But, discovered accidentally and observed in provided C++ implementation of the Euler method, if the value is not stored and the velocity is updated using the next iterations position, numeric stability is improved giving a more accurate solution. The C++ code should probably mention this distinction in my humble opinion.

This difference would make equation (7) look like:

$$v_{i+1} = v_i - h(4\pi^2)\frac{x_{i+1}}{r_{i+1}^3} \tag{8}$$

because position is updated before velocity when programmed, so without saving the 'old' position, the velocity actually contains the next iteration's position values.

3.2 Velocity Verlet

Descretizing the domain into timesteps h, the Velocity Verlet method describes the next step's position and velocity as:

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}\frac{dv_i}{dt} \tag{9}$$

$$v_{i+1} = v_i + \frac{h}{2}\left(\frac{d(v_{i+1})}{dt} + \frac{d(v_i)}{dt}\right) \tag{10}$$

As with the creation of the discretized Euler equations, equation (4) can be substituted into (9) and (10), resulting in

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}\frac{-4\pi^2}{r_i^3}x_i \tag{11}$$

and

$$v_{i+1} = v_i + \frac{h}{2}\left(\frac{-4\pi^2}{r_{i+1}^3}x_{i+1} + \frac{-4\pi^2}{r_i^3}x_i\right) \tag{12}$$

Note that generating the next iteration's velocity requires the next iteration's position already, which complicates things, especially when introducing additional forces from other planets, not just the force of gravity from the Sun acting on earth.

3.3 Adding Planets

The previous equations describe the position and velocity of a planet in the next timestep for a binary system containing only the Earth and the Sun. Recall that the only force the orbiting planet experiences is from the gravity of the Sun, described by equation (1), thus giving equation (4).

Adding additional planets changes the acceleration. Instead of the acceleration in some direction being described by equation (4), it contains additional components from the gravity of the other planets acting upon it, and the other planets contain components of that planet's gravity, therefore each

planet's acceleration contains a component of the force of gravity from the sun **and** a summation of the force of gravity from all other planets in the system, yielding:

$$\frac{dv}{dt} = \frac{-4\pi^2}{r^3}x + SUM(-4\pi^2\frac{\frac{mo}{m_\odot}}{rd^3}xd) \tag{13}$$

where xd and rd are the difference between the current planet and some other planet's position, and md is the other planet's mass used in this summation of gravitational forces from all different planets, denoted as 'SUM'. To better match the code, for this summation, the quantity md (mass difference, or really a difference ratio) is assigned $\frac{mo}{m_\odot}$. Thus the acceleration of a planet due to the force of gravity from the sun and all other planets can be written as

$$\frac{dv}{dt} = -4\pi^2(\frac{x}{r^3} + SUM(xd\frac{md}{rd^3}) \tag{14}$$

This modification to the previous substitution is integrated into the equation for the next time step's velocity in both the Euler and Verlet methods.

3.3.1 Euler

When doing the transformation shown by turning equation (6) into (7) instead using equation (14) containing gravity components from the other planets, the discretized velocity next iteration looks like

$$v_{i+1} = v_i - h(4\pi^2)(\frac{x_i}{r_i^3} + SUM(xd_i\frac{md_i}{rd_i^3})) \tag{15}$$

instead of

$$v_{i+1} = v_i - h(4\pi^2)\frac{x_i}{r_i^3} \tag{16}$$

(equation (7))

because the acceleration now contains the forces of gravity from all other planets.

3.3.2 Verlet

Unlike the Euler method, the Verlet method contains an acceleration component in it's position update, not just the velocity. So both position and

velocity updating must be modified to contain the forces from other planets, however if the position is left out, not much change is noticed and this bug survived until writing all this out and seeing all the equations next to each other.

The Verlet position update was:

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}\frac{-4\pi^2}{r_i^3}x_i$$

but to model the forces from gravity of other planets, becomes:

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}(-4\pi^2)(\frac{x_i}{r_i^3} + SUM(xd_i\frac{md_i}{rd_i^3})) \tag{17}$$

Likewise, the next iteration's velocity was:

$$v_{i+1} = v_i + \frac{h}{2}(-4\pi^2)\left(\frac{x_{i+1}}{r_{i+1}^3} + \frac{x_i}{r_i^3}\right)$$

(equation (12) but with $-4\pi^2$ factored), but becomes:

$$v_{i+1} = v_i - 4\pi^2\frac{h}{2}\left((\frac{x_{i+1}}{r_{i+1}^3} + SUM(xd_{i+1}\frac{md_{i+1}}{rd_{i+1}^3})) + (\frac{x_i}{r_i^3} + SUM(xd_i\frac{md_i}{rd_i^3}))\right) \tag{18}$$

4 Results

This section contains results of the final versions of the implementations of the Euler, enhanced Euler, and Velocity Verlet methods.

4.1 Euler Method

Shown in figure 1 is the result of the Euler method for the binary system. Due to numerical instability, the Earth's trajectory does not form a complete circle. Figure 2 shows when the planets Mercury, Earth, Mars and Jupiter are modeled together, the numerical instability becoming even more apparent as something strange happens to Mercury. It is interesting to note the asymmetry of Mercury's death spiral.

The only reason those planets were chosen is because of geographical proximity, and to see the effect of a massive planet such as Jupiter.

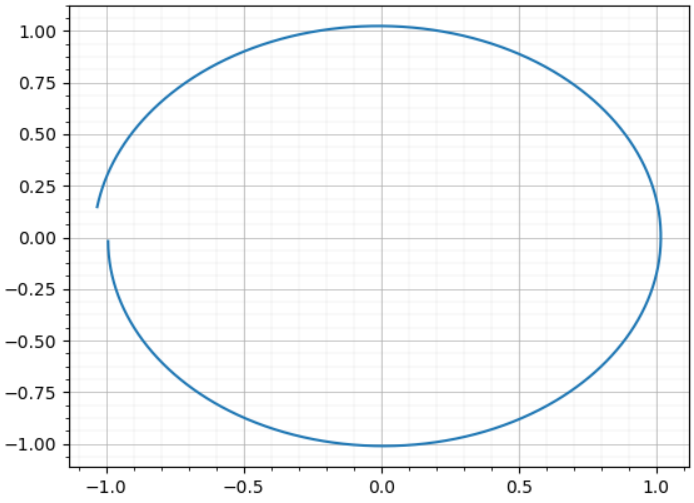

Figure 1: Normal Euler method: binary system

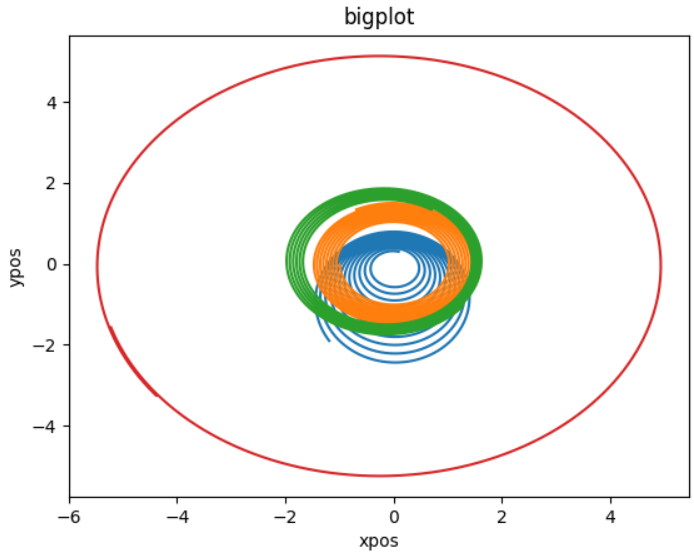

Figure 2: Euler Method: four planet system

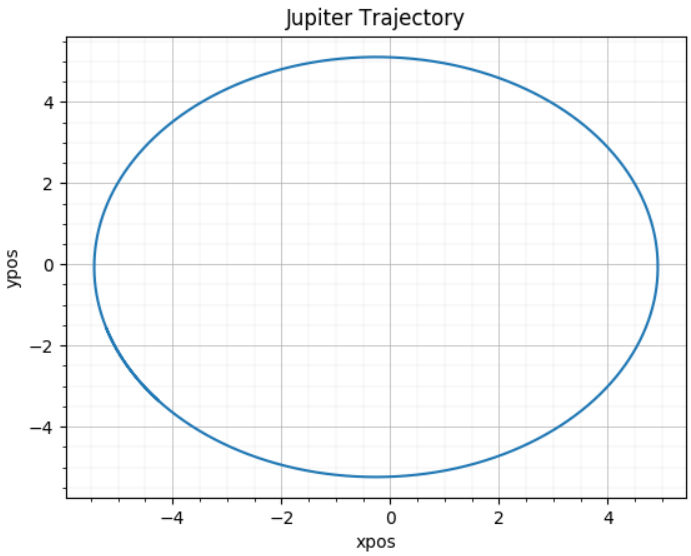

Figure 3: Enhanced Euler Method: Jupiter alone

4.2 Enhanced Euler

The enhanced Euler method offers better numical stability leading to greater accuracy in the solution compared to the 'normal' Euler method.

There is no visible gap here from numerical instability, however figure 4 does show slight instability compared to the Verlet Method shown in figure 6.

4.3 Velocity Verlet

The Velocity Verlet solver shows the best numerical stability, leading to the best accuracy. While the enhanced Euler shows slight 'blurring' of the trajectories in figure 4, the Velocity Verlet solution in figure 6 appears to contain perfect elipses.

5 Conclusions and Future Work

While the final presentation of all planets was not shown, due to the modularity of the code and the `Planet` class contained in `solarsystem.py`, the

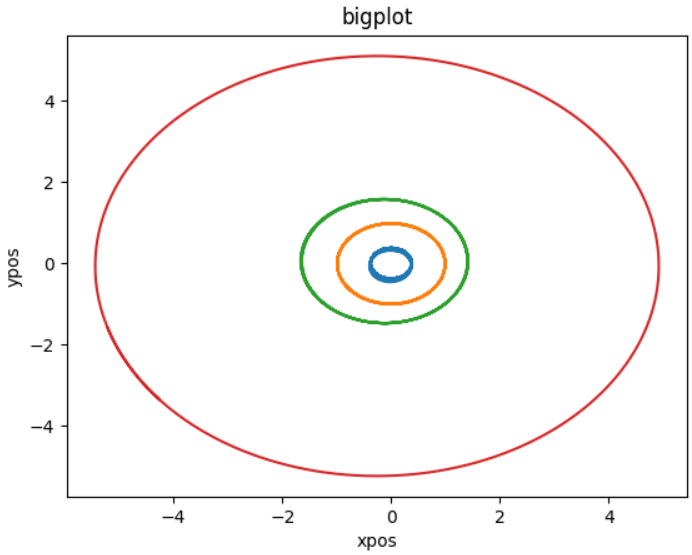

Figure 4: Enhanced Euler Method: four planet system

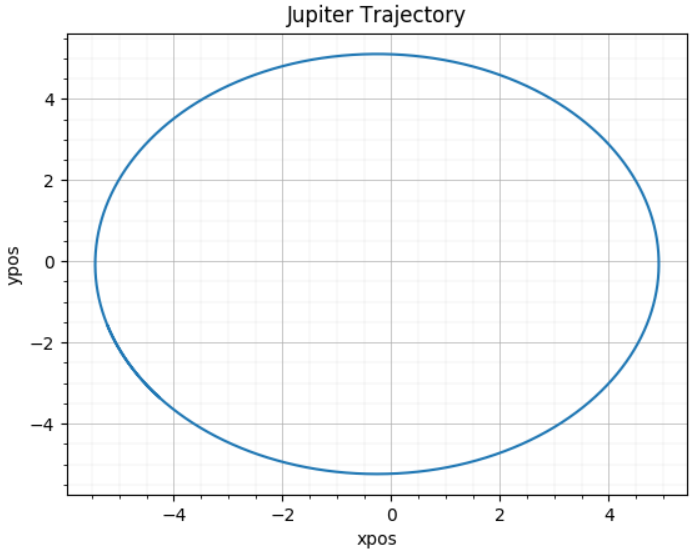

Figure 5: Velocity Verlet: Jupiter alone

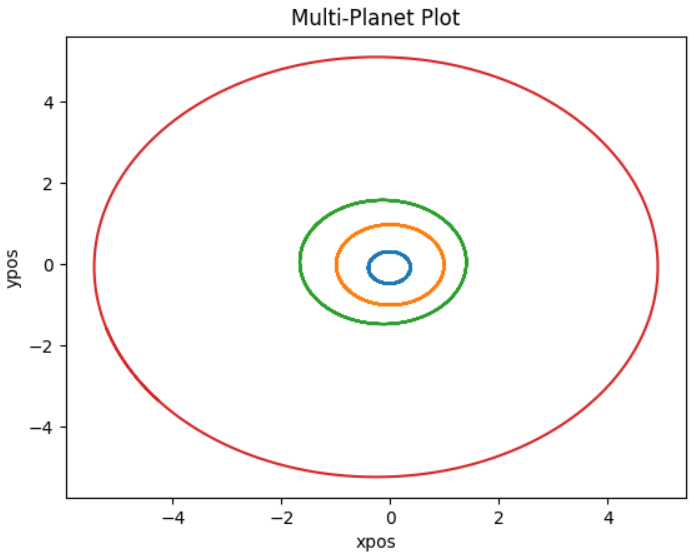

Figure 6: Velocity Verlet: four planet system

rest of the planets work just as well, but if all are plotted, then the middle ones are not clear.

Additionally, this code could be made to solve this in 3D space, and was structured such that the coordinates come last in a function (such as being returned by the planet difference function that computes distances between planets), to make adding the z axis as non-invasive as possible. Dare I might say trivial. Overall, this was an interesting project that really feels like something was 'built from scratch'. The process covered and skills learned were important, and it was nice to get experience with object orienting Python as 90% of the use for it, especially as a replacement for Matlab in coursework seems to not require or benefit from using classes.