PHYM004 Homework 2

Tom Joshi-Cale

14th Dec 2020

Abstract

Two methods of simulating the N-Body problem were implemented for the system of celestial bodies orbiting around the Solar System Barycentre - the Velocity-Verlet method and the GNU Scientific Library Fourth Order Runge-Kutta implementation. A comparison of accuracy and execution time is carried out, and it is concluded that the Velocity-Verlet method strikes the best balance between accuracy and execution time.

1 Introduction

The N-Body problem has long been a subject of discussion amongst physicists, and in 1889 King Oscar II of Sweden announced a prize of 1,500 kroner and a gold medal for anyone who could solve it. He defined the problem as "*given a system of arbitrarily many mass points that attract each according to Newton's law, under the assumption that no two points ever collide*" try to represent the coordinates of each point as a function of time, for all time. To this day, only the $n = 2$ case is completely solved; and study into greater values of N laid the ground for the field of Chaotic Dynamics, and is of particular interest to astronomers who wish to model the motions of celestial bodies under the affects of each-others gravity.

2 Background Theory

For a system of celestial bodies, the only forces acting on each body are the gravitational attraction of all the other bodies:

$$\mathbf{F_i} = \sum_{j=0, j\neq i}^{N} \frac{GM_iM_j}{||\mathbf{r_j} - \mathbf{r_i}||^2} \cdot \frac{\mathbf{r_j} - \mathbf{r_i}}{||\mathbf{r_j} - \mathbf{r_i}||}, \tag{1}$$

where $\mathbf{F_i}$ is the force vector acting on body i, G is the Gravitational Constant, M_i is the mass of body i, and r_i is the distance between body i and the centre of mass of the system. Since $\mathbf{F} = m\mathbf{a}$, the acceleration of each body is given by:

$$\mathbf{a_i} = \sum_{j=0, j\neq i}^{N} \frac{GM_j\left(\mathbf{r_j} - \mathbf{r_i}\right)}{||\mathbf{r_j} - \mathbf{r_i}||^3}. \tag{2}$$

Since acceleration is the double derivative of position, this system can be described by a second order differential equation:

$$\mathbf{r}'' - \frac{GM_j}{||r||^3}\mathbf{r} = 0, \tag{3}$$

or in Cartesian co-ordinates:

$$(\mathbf{x}, \mathbf{y}, \mathbf{z})'' - \frac{GM_j}{||r||^3}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0. \tag{4}$$

2.1 Velocity-Verlet

One of the two methods used to integrate the equations of motion for this system is the *Velocity-Verlet* algorithm, named after French physicist Loup Verlet. This method can be performed in one of two ways. First, the 4-Step method:

$$\text{1. Calculate half-step velocity:} \quad v_{0.5} = v_0 + a_0\left(\frac{\Delta t}{2}\right), \tag{5}$$

$$\text{2. Calculate position after timestep:} \quad x_1 = x_0 + v_{0.5} \cdot \Delta t, \tag{6}$$

$$\text{3. Calculate acceleration using new positions:} \quad a_1 = \sum_j \frac{F_i}{M_j}, \tag{7}$$

$$\text{4. Calculate the velocity after timestep:} \quad v_1 = v_{0.5} + a_1\left(\frac{\Delta t}{2}\right), \tag{8}$$

where each step is iterated over every body before continuing to the next step. After completing these steps, the bodies will be ready to be looped over again for the next timestep.

2.2 Runge-Kutta

The 4-step implicit Runge-Kutta algorithm is implemented using the GNU Scientific Library (GSL) ODE solvers (Galassi 2009). These routines are created to solve the n-dimensional first-order system

$$\frac{dy_i(t)}{dt} = f_i(t, y_1(t), ..., y_n(t)),$$

for $i = 1, ..., n$. However, as shown in Equations (3) and (4) this system is a second-order system. We can convert it into two first-order equations, using the fact that $x' = V_x$ and $x'' = V_x'$, leaving us with the first-order equations:

$$x' = V_x, \tag{9}$$

$$V_x{}' = -\sum_j \frac{GM_j}{r^3}\left(x - x_j\right), \tag{10}$$

which can be both repeated in y and z, giving 6 first-order differential equations to be solved for each body being modelled. Since the Runge-Kutta method is not explicity tackled in the code, the method of solving it has been omitted from this report, however for further reading an interested party would be directed towards Cheever (2020).

3 Method of Solution

3.1 Setting up the problem

When modelling an N-Body problem, the first thing that is required is to store the relevant properties of the orbital bodies. This was done by defining a `struct` for each body, which stores the name of the body, its mass, and a series of arrays for its position, velocity and acceleration in the x-, y- and z-directions. The X, Y and Z co-ordinates were also defined in an `enum` to allow looping over each direction. So that the code would be valid for any number of inputted bodies, so long as they were inputted correctly in a text file, the number of lines in the inputted text file should first be counted, so that it is known how much memory to allocate to the `struct` for each body. Then, the time-step and maximum time are read in from the command line; and initial conditions for each body from a text file.

The text file containing the initial conditions of each body was generated using the NASA HORIZONS Web-Interface (NASA Jet Propulsion Laboratory 2020), which can generate information about the orbital properties of each body for a given day in a Solar System Barycentric (centre of mass) frame of reference.

3.2 Velocity-Verlet

The Velocity-Verlet algorithm was implemented over two functions, the first of which was created to calculate the acceleration of all the bodies by implementing Equation 2 in a double `for` loop over all bodies and all Cartesian directions. It is important at the beginning of this function to set the acceleration for each body to zero, so when summing the contributions of every other body to the acceleration of one body the acceleration is not compounded.

The second function is where the *Velocity-Verlet* algorithm is implemented. It first performs Steps 1 and 2 of the 4-step method (Equations 5 & 6) for all bodies in all directions, and then since all bodies are now in their new position, it calculates the acceleration for all bodies. Then it calculates the new velocity of all the bodies, ready to be repeated in the next timestep.

3.3 Runge-Kutta

When describing the system for the GSL routines, the user can only pass one array of parameters into the function, and so an array was created which contained: in position 0 the number of bodies being modelled; and in position $N > 0$ was the mass of body $N - 1$. This is because the only parameters needed to set up the first order differential equations outlined in Section 2.2 are G, which is defined globally, and the masses of each of the bodies. The reason the number of bodies is stored in in the array is to allow for loops over all bodies within `func`.

The initial conditions also need to be fed into `func` through a `double` array. To create one array which could hold the x-, y- and z-components of position and velocity for all bodies,

the array was created with a size $6 * N$, and the the data saved into the array using a loop over all bodies with `array[i * DIM + X]` where `X` varies from 0 to 5 and holds the x-, y-, z-component of position from 0 to 2, and the x-, y-, z-component of velocity from 3-5 for body `i`.

When performing the actual solving of the differential equations, GSL allows the user to choose between a fixed step size, or a variable step size. A variable step size was chosen as it allows the function to select a step size to maximise the accuracy of the simulation.

3.4 Plotting

After completion of the modelling, the code executes a Gnuplot script to plot the output file with the positions of all the bodies at each timestep. The script used can be varied by changing the value of the relevant `#define` before compilation.

4 Results

The code displays the paths taken for each body over the timescale inputted in a Gnuplot graph that can be scaled and zoomed by the user. Examples of outputs are shown in Figure 1.

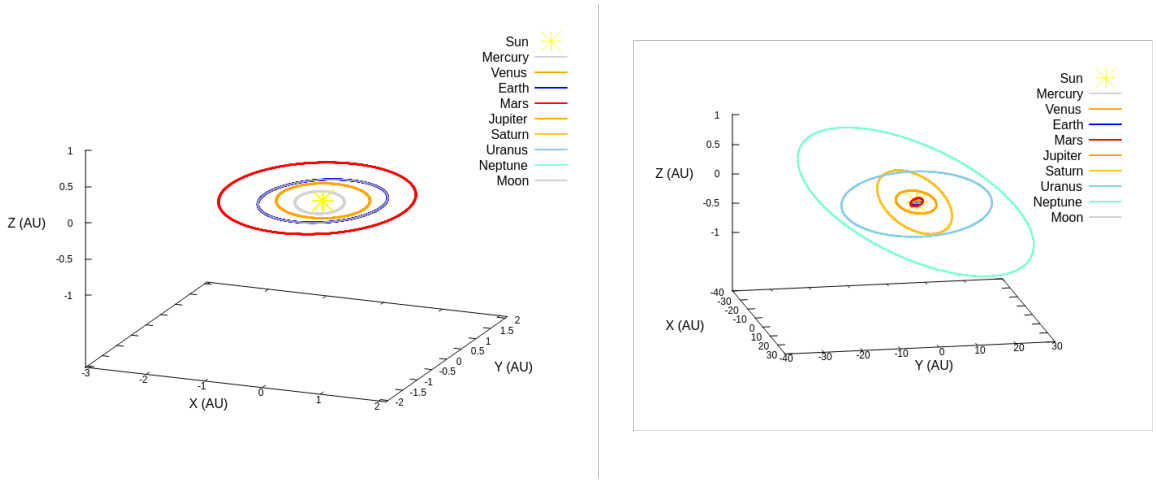

Figure 1: Left: The output of running the Nbody code to perform the Runge Kutta method with a timestep of 1 day, for a total time of 30 years, zoomed in to show only the inner planets. Right: The output of the Runge-Kutta method with a timestep of 1 day, for a total time of 170 years, on a scale to show primarily the orbital paths of the outer planets.

To check the accuracy of the simulation; gravitational potential energy and kinetic energy were calculated at each timestep, and summed to find the total energy. It would be expected that each body has a constant total energy, and it can be seen in Figure 2 that this appears to be the case regardless of which method is used. If the total energy of a body is observed on a much smalled y-axis scale, as in Figure 3, it can be seen that the total energy is displaying a small

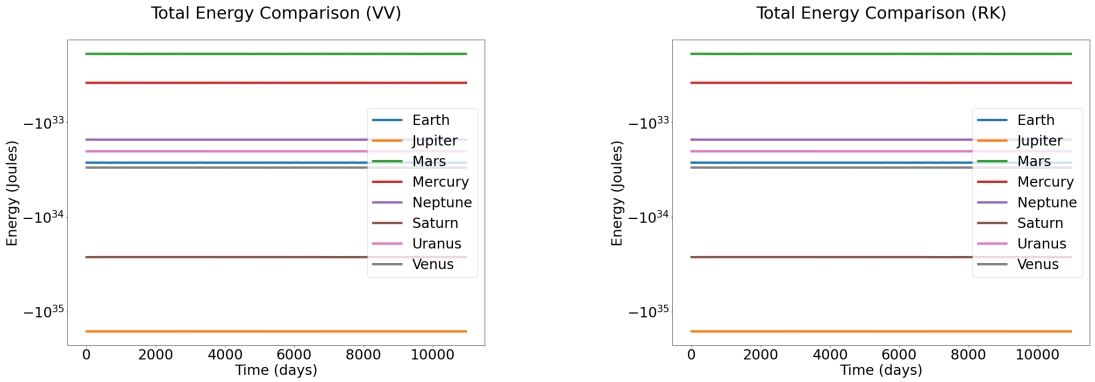

Figure 2: The comparison between total energies for all bodies, using energies calculated by the Velocity-Verlet method (left) and the Runge-Kutta method (right). The energies are equal for both bodies.

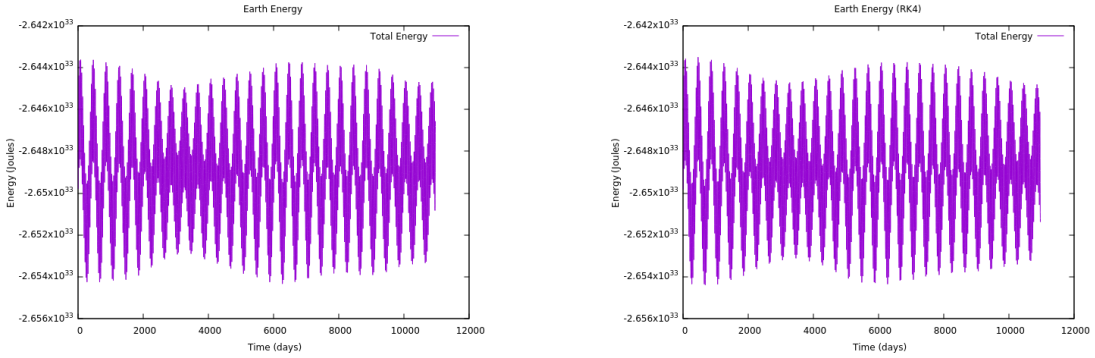

Figure 3: The total energy of Earth, calculated using the Velocity-Verlet method (left) and the Runge-Kutta method.The plot has been zoomed in on the y-axis to show the small periodic variability it displays. It can be seen that the variations are very similar for both methods.

amount of periodic variation, however taking estimates of the maximum y-value of the energy compared to the value around which the energy is varying we get $\frac{E_0 - E_{max}}{E_0} \times 100 = \pm 0.188\%$ for both methods (where E_0 is the value around which the energy is oscillating, and E_{max} is the maximum value the energy reaches), which is a very small variation to the extent where we can say that the energy of the body is effectively constant.

Another check of accuracy for the simulation is to calculate the orbital period of each body, and comparing it to the known values of period, and the outcome of that check is shown in Table 1. It can be seen that the Runge-Kutta method gives a more accurate value for the orbital period of the planets, however both methods give estimates that are accurate to $< 1\%$ for all bodies, with the least accurate body being the Earth's Moon in both cases. Taking an average of all the percentage accuracies of all bodies, to act as a quick way of comparing the two methods, the Velocity-Verlet gives the periods to 0.276%, and the Runge-Kutta method gives them to 0.234%.

The efficiency of the two methods were also tested, and the output can be seen in Figure 4.

Table 1: Compares the modelled average orbital periods of each celestial body to the expected orbital periods from Williams (2019). The data used here was modelled for 170 years, so the number of orbits completed by each body is also given.

(a) Velocity-Verlet Method

Planet	Expected	Observed	% Accuracy	N Orbits
Mercury	88.0	88.1207	0.137	704
Venus	224.7	224.826	0.056	275
Earth	365.2	365.521	0.088	169
Moon	27.3	27.5539	0.930	2251
Mars	687	687.223	0.034	90
Jupiter	4331	4335	0.092	14
Saturn	10747	10757.4	0.097	5
Uranus	30589	30692	0.337	2
Neptune	59800	60225	0.711	1

(b) Runge-Kutta Method

Planet	Expected	Observed	% Accuracy	N Orbits
Mercury	88.0	87.9901	-0.011	705
Venus	224.7	224.768	0.030	276
Earth	365.2	365.485	0.078	169
Moon	27.3	27.1035	0.720	2289
Mars	687	687.211	0.031	90
Jupiter	4331	4335	0.092	14
Saturn	10747	10757.4	0.097	5
Uranus	30589	30692	0.337	2
Neptune	59800	60225	0.711	1

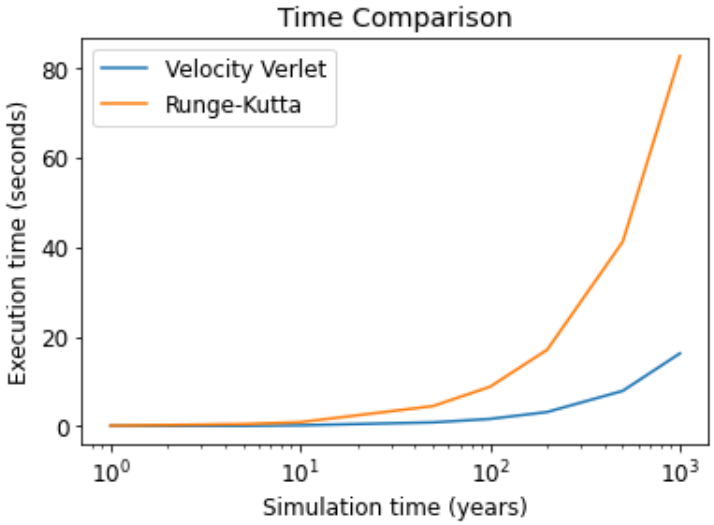

Figure 4: A comparison of the execution time for both methods, with a timestep of 1 day, and varying the simulation time in years. It can be seen that the Runge-Kutta method is less efficient for all timescales.

It can be seen that the Runge-Kutta method is less efficient at all timescales, and also increases in execution time much faster than the Velocity-Verlet method.

5 Discussion

Both methods of modelling the N-Body problem are accurate, producing percentage uncertainties in orbital periods of less than 1% at worst, and for the majority of the bodies, less than 0.1%, however the Velocity-Verlet method is much faster at compiling, completing the simulation of 1000 years, or 365,000 timesteps, five times faster than the Runge-Kutta method.

Conclusions

Between the Runge-Kutta and Velocity-Verlet methods of simulating the N-Body problem, the Runge-Kutta method calculates the orbital period to ~ 0.04 percentage points greater accuracy than the Velocity-Verlet, but takes significantly longer. If accuracy is the only consideration, then the Runge-Kutta method should be used, however for most use-cases the Velocity-Verlet method does not sacrifice accuracy to a great enough level to make its savings in computational time not significant, and as such should be used.

References

Cheever E., , 2020, Fourth Order Runge-Kutta, `https://lpsa.swarthmore.edu/NumInt/NumIntFourth.html`, [Accessed: 17 Dec. 2020]

Galassi M., 2009, GNU scientific library reference manual, 3rd edn. Network Theory Ltd.

NASA Jet Propulsion Laboratory, 2020, HORIZONS Web-Interface, https://ssd.jpl.nasa.gov/horizons.cgi, [Accessed: 08 Dec. 2020]

Williams D. R., , 2019, Planetary Fact Sheet - Metric, https://nssdc.gsfc.nasa.gov/planetary/factsheet/