Modelling the Solar System

Marco Santia

Abstract

In this work we develop and compare computational methods of tracking orbits of planetary bodies in the solar system. The methods are investigated for two, three, and N-body systems and consist of the Euler, Verlet, and Runge-Kutta algorithms. While the Euler approach is determined to be the most simplistic, the error is found to be large while Runge-Kutta was the most accurate in the simple cases. For many-body tests including the full solar system, the Verlet method is determined to be the most computationally feasible due to its wider range of stability compared to Runge-Kutta. In all tests the orbits agree well with known orbits.

I. INTRODUCTION

Many branches of physics, chemistry, engineering, etc. often involve the same fundamental computational problems to be solved, differing only by physical application. Perhaps the most universal is the N-body problem in which a physical system contains many interacting objects that each individually may be desired to be tracked and then statistically analyzed as an ensemble. Molecular, electronic, and even gravitational systems find common structure in how they are mathematically decomposed into a computational problem. Therefore, it is not just instructive but practical for us to explore various solution strategies for the N-body problem. Additionally, the polymorphous nature of these physical systems allows for full utilization of object-oriented capabilities of modern programming languages. Through the mechanism of object-oriented programming the N-body problems universality and generality manifests in code, not just theory.

This work specifically focuses on the problem solved through the lens of gravitational physics i.e dynamics of solar bodies. In which we will model the orbits of various celestial bodies interacting via Newton's laws. Modelling the time evolution will be done with the well-known Runge-Kutta integration scheme as well as the more classical Euler scheme and finally the velocity Verlet algorithm. The applicability of each integration scheme will be compared for the two-body (Earth-Sun), three-body (Earth-Sun-Jupiter), and many-body (full solar-system) examples.

II. THEORY

Using non-relativistic theory, the force of interaction from Newton's laws for two bodies is:

$$\mathbf{F} = \frac{Gm_1m_2}{r^3}\mathbf{r}$$

where the standard notation is used for gravitational constant G . Analgously for the N-body system the generalized interaction is represented with the summation giving rise to N coupled equations to solve:

$$\mathbf{F}_k = \sum_{i \neq k} \frac{Gm_im_k}{r_{ik}^3}\mathbf{r}_{ik}$$

Once the force interactions for each body with respect to each other body is established, the integration of the dynamical system depends solely on the integration scheme chosen. In subsequent sections we outline the methods used here.

A. Euler Method

One of the oldest and most simplistic models of integrating a differential equation itertively is Euler's method which is linear in nature. It is mathematically the basis to many modern integration schemes that achieve higher-order accuracy[?] . In the method, position $\mathbf{r}_i$ and velocity $\mathbf{v}_i$ are advanced to a the next iteration in time as given in[?] and repeated here,

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{\mathbf{F}(\mathbf{r}_i, t_i)}{m_i}\Delta t \tag{1}$$

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{v}_i \Delta t. \tag{2}$$

Algorithmically it can be written as given in Alg. 1.

Algorithm 1 Euler method to update position and velocity

function UPDATEEULER(r,v)

$\mathbf{a} \leftarrow$ acceleration($\mathbf{r}$)

$\mathbf{v}' \leftarrow \mathbf{v} + h\mathbf{a}$

$\mathbf{r}' \leftarrow \mathbf{r} + h\mathbf{v}$

end function

The disadvantage to this method is that it is only linear in accuracy so the error should grow quite rapidly. This is especially the case when long timesteps are used and thus the extrapolation is more inaccruate. Shorter timesteps can minimize error however for most realistic problems an exceedingly short time-step will be computationally infeasible.

B. Verlet Method

One method popularized in the field of molecular dynamics simulations is the Verlet algorithm. The method utilizes Taylor expansions of solution variables. Expanding position

and velocity gives,

$$\mathbf{r}(t+h) = \mathbf{r}(t) + h\dot{\mathbf{r}}(t) + \frac{h^2}{2}\ddot{\mathbf{r}}(t) + O(h^3) \tag{3}$$

$$\mathbf{v}(t+h) = \mathbf{v}(t) + h\dot{\mathbf{v}}(t) + \frac{h^2}{2}\ddot{\mathbf{v}}(t) + O(h^3). \tag{4}$$

truncating at second order simplifies this to,

$$\dot{\mathbf{v}}(t+h) = \dot{\mathbf{v}}(t) + h\ddot{\mathbf{v}}(t) + O(h^2)$$

or alternatively,

$$\frac{h^2}{2}\ddot{\mathbf{v}}(t) = \frac{h}{2}[\dot{\mathbf{v}}(t+h) - \dot{\mathbf{v}}(t)] + O(h^3).$$

When combined with the velocity expansion we get,

$$\mathbf{v}(t+h) = \mathbf{v}(t) + \frac{h}{2}[\dot{\mathbf{v}}(t+h) + \dot{\mathbf{v}}(t)] + O(h^3).$$

Rewriting this using in a discrete form with the definitions for force and velocity,

$$\mathbf{r}_{i+1} = \mathbf{r}_i + h\mathbf{v}_i + \frac{h^2}{2}\frac{\mathbf{F}(\mathbf{r}_i, t_i)}{m} + O(h^3) \tag{5}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{2}\left(\frac{\mathbf{F}(\mathbf{r}_{i+1}, t_{i+1})}{m} + \frac{\mathbf{F}(\mathbf{r}_i, t_i)}{m}\right) + O(h^3) \tag{6}$$

The iterative procedure can be outlined as follows,

Algorithm 2 Verlet algorithm iterative process

function UPDATEVERLET(r,v)

$\mathbf{a} \leftarrow$ acceleration($\mathbf{r}$)

$\mathbf{r}' \leftarrow \mathbf{r} + h\mathbf{v} + \frac{1}{2}h^2\mathbf{a}$

$\mathbf{a}' \leftarrow$ acceleration($\mathbf{r}'$)

$\mathbf{v}' \leftarrow \mathbf{v} + \frac{1}{2}h(\mathbf{a} + \mathbf{a}')$

end function

C. Runge-Kutta Method

Lastly, the arbitrary-order Runge-Kutta method can be used which potentially gives the highest accuracy although more laborious. The premise is simple and it is essentially an extension of the famous Simpson rule of integration from fundamental calculus[?] which

utilizes midpoints and endpoint values for a given step length in addition to the starting value. However of course in the case of temporal evolution the mid- and end- points are unknown, thus Euler's method is used to approximate these points for the sake of integration. If we use a fourth-order method, we use 4 intermediate points and the iterative process can be written out as[?] ,

$$k_1 = f(x_i, t_i) \tag{7}$$

$$k_2 = f(x_i + \tfrac{h}{2}k_1, t_i + \tfrac{h}{2}) \tag{8}$$

$$k_3 = f(x_i + \tfrac{h}{2}k_2, t_i + \tfrac{h}{2}) \tag{9}$$

$$k_4 = f(x_i + hk_3, t_i + h) \tag{10}$$

$$x_{i+1} = x_i + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5). \tag{11}$$

or as an algorithm,

Algorithm 3 Fourth-order Runge-Kutta update sequence

function UPDATERK4(r,v)

$\mathbf{k}_1^v \leftarrow$ acceleration($\mathbf{r}$)

$\mathbf{k}_1^r \leftarrow \mathbf{v}$

$\mathbf{k}_2^v \leftarrow$ acceleration($\mathbf{r} + \frac{h}{2}\mathbf{k}_1^r$)

$\mathbf{k}_2^r \leftarrow \mathbf{v} + \frac{h}{2}\mathbf{k}_1^v$

$\mathbf{k}_3^v \leftarrow$ acceleration($\mathbf{r} + \frac{h}{2}\mathbf{k}_2^r$)

$\mathbf{k}_3^r \leftarrow \mathbf{v} + \frac{h}{2}\mathbf{k}_2^v$

$\mathbf{k}_4^v \leftarrow$ acceleration($\mathbf{r} + h\mathbf{k}_3^r$)

$\mathbf{k}_4^r \leftarrow \mathbf{v} + h\mathbf{k}_3^v$

$\mathbf{v}' \leftarrow \mathbf{v} + \frac{h}{6}(\mathbf{k}_1^v + 2\mathbf{k}_2^v + 2\mathbf{k}_3^v + \mathbf{k}_4^v)$

$\mathbf{r}' \leftarrow \mathbf{r} + \frac{h}{6}(\mathbf{k}_1^r + 2\mathbf{k}_2^r + 2\mathbf{k}_3^r + \mathbf{k}_4^r)$

end function

This method can result in high accuracy at the expense of computational time.

III. ESCAPE VELOCITY

One practical test will be an escape velocity calculation i.e when the orbiting bodies kinetic energy exceeds its potential. The escape velocity of earth is,

$$v = \sqrt{\frac{2Gm_{\text{sun}}}{r}} = 2.4327e - 2\,au/d \tag{12}$$

which we will compare to as a means of validating the solvers.

IV. RESULTS

A. Earth-Sun

For the most basic testing we explore the orbit dynamics of the two-body sun-earth system. This can be a good validation as the sun is effectively stationary for the sake of determining properties over a single timescale. For demonstrating accuracy given a small timestep that ensures stabiliy for all methods, we show in Fig.1 how much deviation there is in earth's orbit which should be ideally circular.

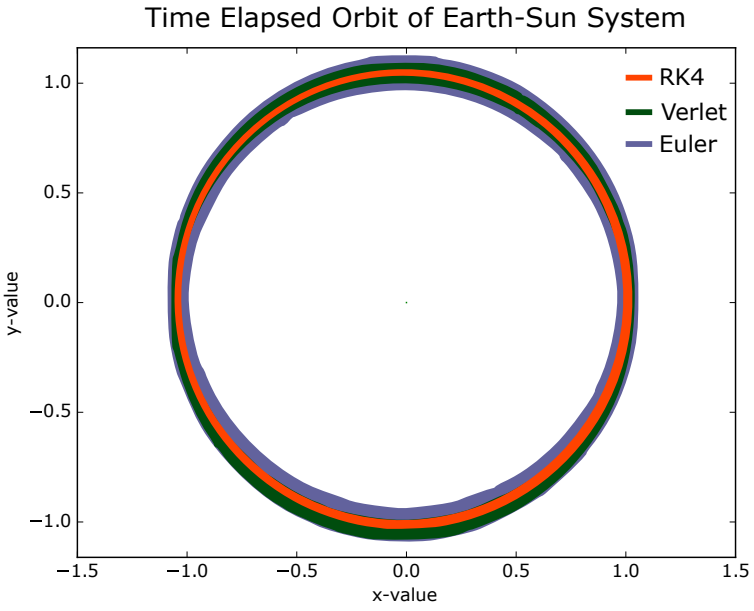

FIG. 1. Comparison of positions that are occupied for each method over several orbits. Runge-kutta is the most circular orbit and thus the most accurate given a timestep of $dt = 10^{-4}$

Euler's method while the most straightforward to implement and possessing a wide range of stability, is the least accurate with Verlet next and RK4 as the most accurate. This is

TABLE I. Timings for Earth-Sun system with $dt = 10^{-4}$

N	Euler	Verlet	RK4
100	0.00297	0.00351	0.00405
500	0.01647	0.01971	0.02565
1000	0.03888	0.04077	0.04293
5000	0.01647	0.01836	0.18657
10000	0.33237	0.33642	0.34938

consistent with[?] where it was determined that RK4 has scaling on the order of $O(h^5)$ while Verlet posses $O(h^4)$.

While RK4 may be the most accurate in this simple problem for reasonable time steps, the timings for the respective algorithms are also important to consider and thus presented in Table. I. Overall we the methods to not differ too significantly. The Euler while the fastest is also significantly lower in accuracy so we will limit further discusion on it. The Verlet algorithm and RK4 have similar speed however RK4 has higher accuracy. Generally however RK4 is more sensitive to timestep constraints and a smaller stability region, so it is most practical to begin calculations with Verlet until it the exact accuracy/speed requirements are more understood. Then RK4 can be tuned accordingly to improve accuracy and performance.

Lastly for the two-body system, we use the known escape velocity given in the methodology as an initial condition for our solver to confirm energy is being calculated correctly. The unstable orbit is seen correctly in Fig.2.

B. Earth-Sun-Jupiter

Adding another body in the physical system will give insight into how dependent earth's orbit is on interaction distances as well as mass differences. Jupiter is chosen as it has the largest mass and should perturb Earths circular orbit the most prominently. Additionally, the Verlet method is also utilized for its stability, allowing us to use a timestep of 10^{-4} s. In Fig.3 we show the computed orbit of Earth and Jupiter in this three-body system however there is no obvious change in orbit for the case of a single Jupiter mass. Hence we also give

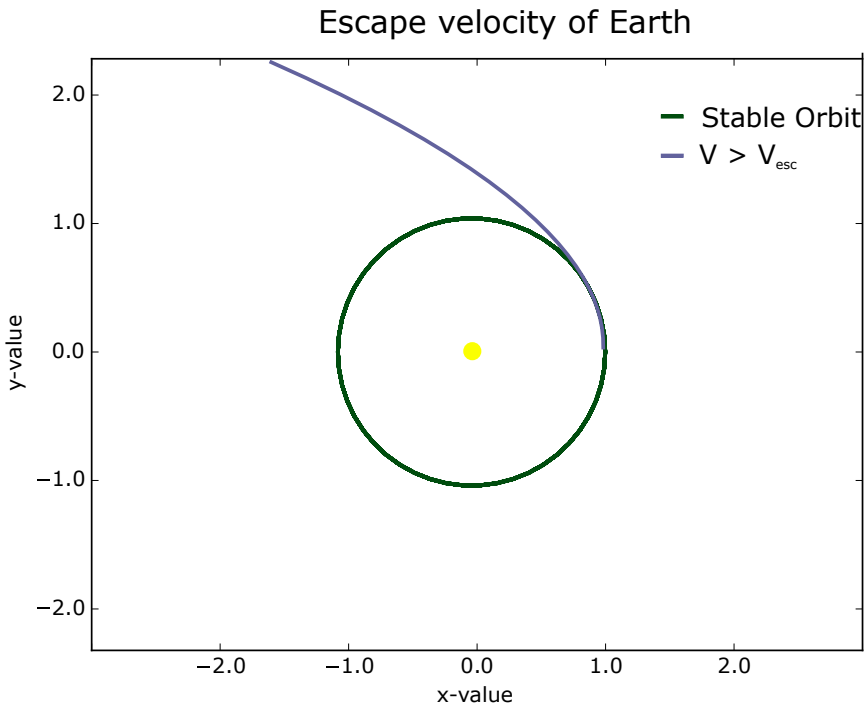

FIG. 2. Orbit trajectory for earth-sun system when the system has been driven by a velocity exceeding the escape velocity (using Verlet integration).

the case where Jupiter's mass is artificially increased to ten times the standard mass.

For the 10x mass case, we can see a slight alteration of the planets orbits over several periods. It is interesting to note however that there still exists bound orbits with respect to the sun in both planets, indicating our solar system is quite stable in this respect. If we further increase Jupiter's mass to 1000 times its original mass, we see a rapid breakdown and a completely unstable orbit as shown in Fig.4. Note that we have chosen to only show earths trajectory in this figure for the sake of illustrating the unstable orbital system.

C. Solar System

Lastly, we test the entire solar system using the Verlet method and the trajectories are given in Fig.5. Unlike that of the two- and three-body systems, the full solar-system introduces difficulties computationally due to the disparate orbital periods. Especially when comparing the most inner (mercury) with the most outer(pluto) planets. This will discussed in more detail in the next section. There is a degree of accuracy that is sacrificed in the

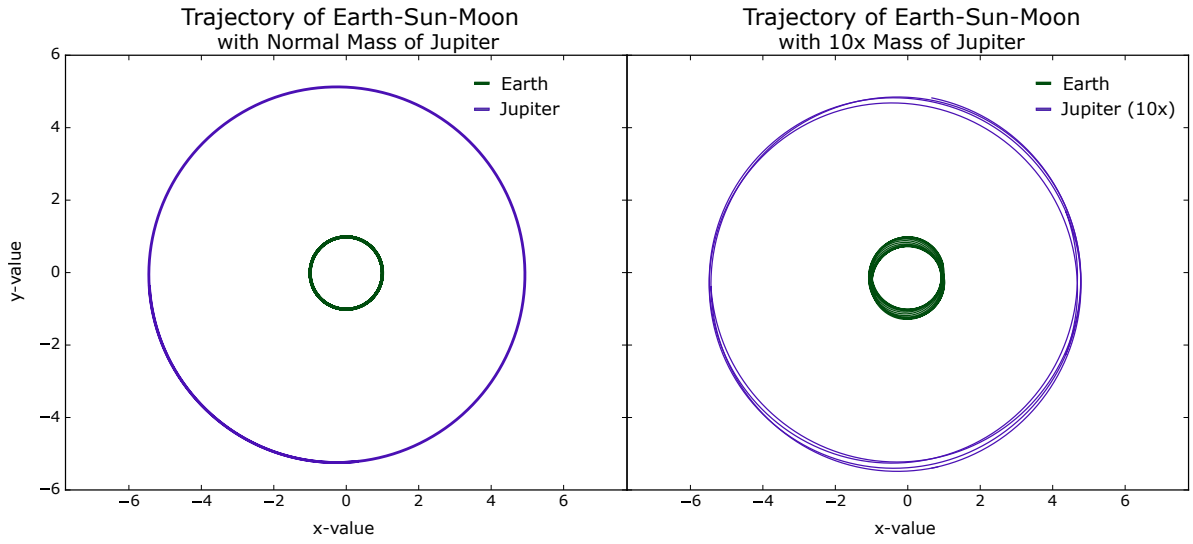

FIG. 3. Trajectories for earth-sun-jupiter system when the mass of jupiter has been set to 10 times the normal mass.

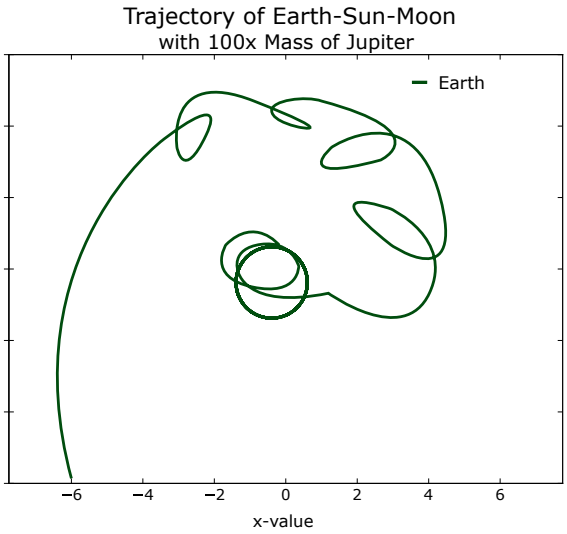

FIG. 4. Earth trajectory for earth-sun-jupiter where jupiter is given 1000 times the normal mass. Only Earth's trajectory is shown for clarity while the other bodies experience similar erratic trajectories.

solution of this problem with completeness, and that arises from the long period of the outer planets. To feasibly capture one entire revolution around the sun, the timestep had to be increased to 10^{-3}, which still falls under the stability region of the Verlet integration scheme.

The trajectories for the outer planets is consistent with known trajectories, including the out-of-plane motion of the most outer bodies, such as pluto, which is negligible in all inner

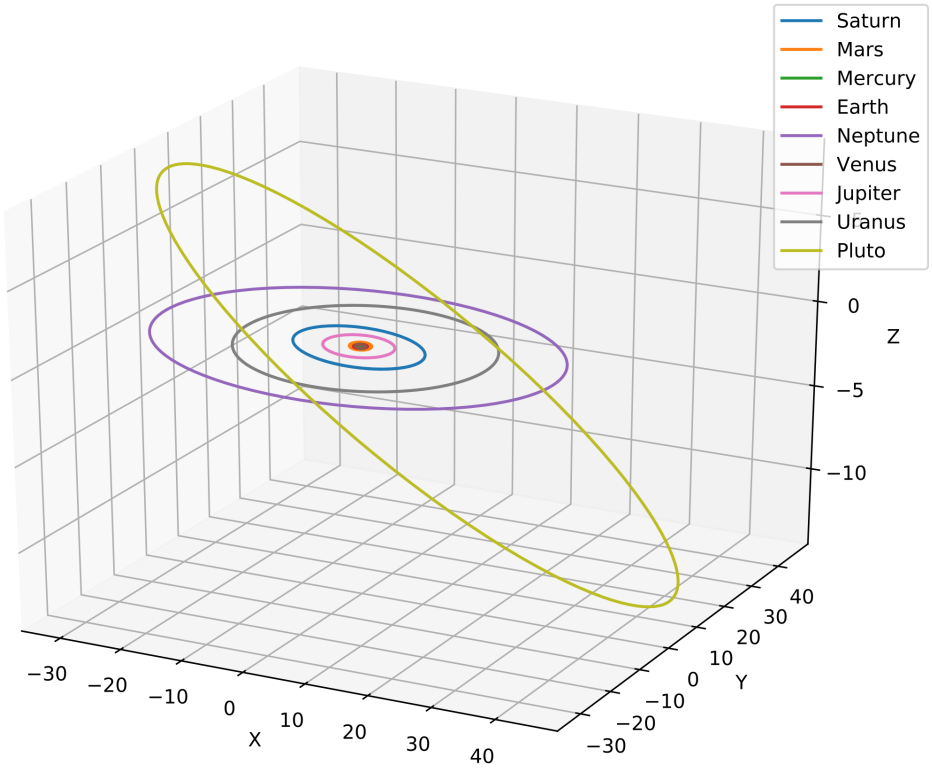

FIG. 5. Trajectories of full solar system using Verlet solver and a timestep of 10^{-3}.

planetary bodies.

V. CONCLUSIONS

The Verlet and RK4 integration schemes both prove to be accurate methods for timestepping. In this work, we found Verlet's increased stability to be more practical when simulating gravitational bodies albeit less accurate. This leads us to believe that the RK4 integration scheme will be increasingly useful for smaller scale problems such as a molecular dynamics simulation, which will be the next topic of investigation. Another important note returning to the concept of stability in the integration scheme can be seen when comparing timescales of relevant interactions. When the timescales of relevant physical phenomena are largely disparate, many integration schemes experience enormous computational bottlenecks. For example in our case, Pluto has an orbital period of 250 years while Mercury has a period of only 88 days. This much finer scale for physics to occur prohibits larger timesteps to be chosen as a means of speeding up a computation involving both Mercury and Pluto. This

concept in general proves to be an exceedingly difficult problem in almost every scientific field hence the recent focus in many research communities on collaborative development of multiscale computational methods. In future works, we hope to investigate the validity of time integration schemes in these multiscale frameworks.