Project 3: A model of the solar system: Solving coupled differential equations

Adam Powers

8 May 2016

Abstract

Two different algorithms were used to solve coupled differential equations in order to track the planets and create a model of the solar system. We looked at the Verlet and fourth-order Runge-Kutta algorithms. While both work quite well in the smaller two- and three-body systems, the Runge-Kutta does much better when all eight planets (and Pluto) are included.

1 Introduction

The solar system is a great representation of a classical system. Using simple Newtonian gravity, the planets are able to stay in orbit due to the inverse square law force

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

where the constant G = $6.67\text{x}10^{-11}$ $\text{m}^3\text{kg}^{-1}\text{s}^{-2}$ = $2.959 \text{ x } 10^{-4}$ $\text{au}^3\text{m}_\text{r}^{-1}\text{d}^{-2}$.[1] Here the vector **r** is a radial line from one body to another. For N bodies in a system, this generalizes to give the following force on body *k*:

$$\boldsymbol{F_k} = \sum_{i\neq k}\frac{Gm_im_k}{r_{ik}^3}\boldsymbol{r}_{ik}$$

In Cartesian coordinates, this can be written as 3N coupled differential equations:

$$\frac{d^2x_k}{dt^2} = \sum_{i\neq k}\frac{Gm_i}{r_{ik}^3}(x_i - x_k)$$

$$\frac{d^2y_k}{dt^2} = \sum_{i\neq k}\frac{Gm_i}{r_{ik}^3}(y_i - y_k)$$

$$\frac{d^2z_k}{dt^2} = \sum_{i\neq k}\frac{Gm_i}{r_{ik}^3}(z_i - z_k)$$

where

$$r_{ik} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + (z_i - z_k)}$$

Given appropriate initial conditions, the motion of the planets can easily be simulated with an algorithm for solving the equations.

2 Algorithms

2.1 Velocity Verlet algorithm

To start the velocity Verlet algorithm, we first need the Taylor expansions for the position and velocity functions (1) and (2):[2]

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

To second order, the derivative of the second equation is

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

which can be rearranged to find

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

using this, the relationship between force and acceleration, and discretizing we end up with (5) and (6):

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

These can be solved position, then velocity, for each iteration in order to find the new position and velocity of each body in the system.[2]

The following pseudocode shows the general outline of the expected code:

```
o Find acceleration a with r using Force equation
o Find new position r' with r + hv + ½*h$^2$*a
o Find new acceleration a' with r' and Force equation
o Find new velocity v' with v + ½*h(a+a')
```

2.1 Fourth-order Runge-Kutta (RK4)

The second method we looked at is similar to Simpson's Rule:[3]

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

In our case, the value of the function is known at the first point, but not at the middle or the end of the step. At these locations, we used Euler's method to approximate the function. This leads to the following equations:[2]

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

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

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

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

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

One advantage to this method over the Verlet method is that the error scales as h^5 making this a much more precise algorithm than Verlet.

The following pseudocode shows the general outline of the expected code:

```
o  Find $k_{1v}$ with r using Force equation
o  Find $k_{1r}$ with v
o  Find $k_{2v}$ with (r + ½ h $k_{1r}$) using Force equation
o  Find $k_{2r}$ with v + ½ h $k_{1v}$
o  Find $k_{3v}$ with (r + ½ h $k_{2r}$) using Force equation
o  Find $k_{3r}$ with v + ½ h $k_{2v}$
o  Find $k_{4v}$ with (r + ½ h $k_{3r}$) using Force equation
o  Find $k_{4r}$ with v + ½ h $k_{3v}$
o  Find v' using v + ⅙ h ($k_{1v}$+2 $k_{2v}$+2 $k_{3v}$+$k_{4v}$)
o  Find r' using r + ⅙ h ($k_{1r}$+2 $k_{2r}$+2 $k_{3r}$+$k_{4r}$)
```

3 Results

The two algorithms were run through a series of tests to determine which may be better in certain situations over the other. All data was collected from the JPL HORIZONS tool.[4]

3.1 Earth-Sun system

We took it easy on our first test run and only included the Earth and the Sun in our system. We kept it in a two-dimensional system since our solar system is basically in a disk to the first approximation.

The initial position of the earth was at a position 1 in one Cartesian coordinate and 0 in the other (units in au), with a velocity of 2π au/yr, since 1 au is considered the average distance between the Earth and the Sun. you can see the simple example of running with these parameters in Figure 1.

Due to its very small error of O(h^5), RK4 gives very good results for even moderately sized steps.

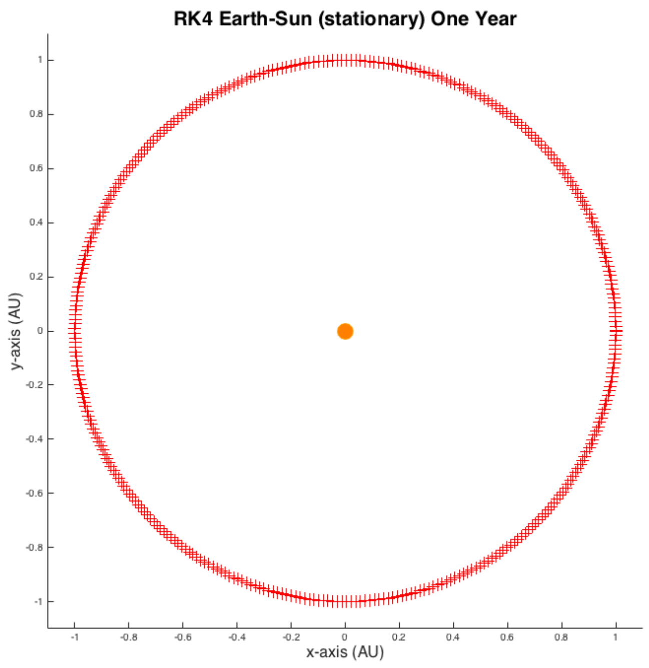


Figure 1: Earth's position over one year of travel in a circular orbit with a time interval between steps of one day using the RK4 solver.

3.3 Escape Velocity

A good test of the behavior of the Earth is to calculate its escape velocity and see if it leaves orbit at that velocity. The escape velocity is the velocity when the planet's total energy is zero[6], or where its kinetic and potential energies are equal. This happens for the earth when

$$v = \sqrt{\frac{Gm_r}{r}} = 8.8794\ au/yr$$

in the tangential direction. The results are seen in Figure 2.

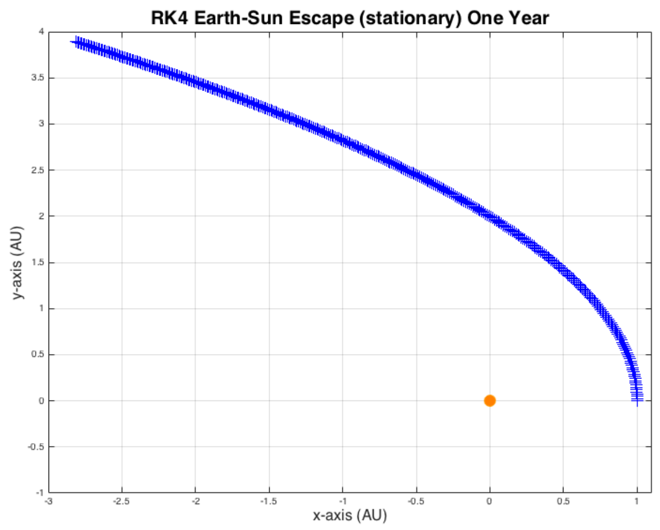


Figure 2: Trajectory of the Earth over one year with one day step size with an initial velocity equal to its escape velocity.

3.3 Earth-Sun-Jupiter system

To look at how the stability of the Earth-Sun system can be changed, we introduced Jupiter into the system. We added one more body so three more differential equations. The system was very stable as was to be expected so we increased the mass of Jupiter to 1000x its actual mass to see how that would affect Earth. The result is shown in Figure 3.

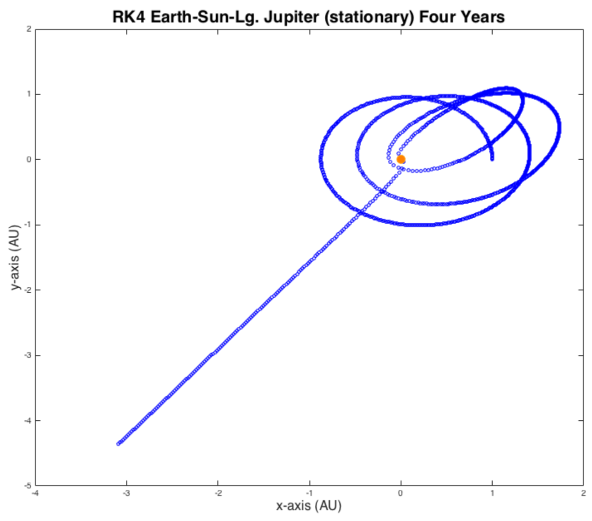


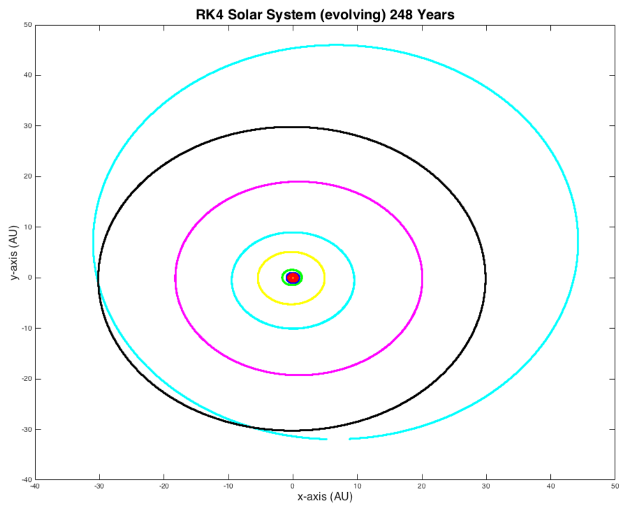
Figure 3: Trajectory of the Earth over four years with one day step size in a solar system with a massive Jupiter.

The Earth would not last long with a planet that size in the solar system, but as time has shown, our solar system is stable the way it is.

3.4 Entire Sol solar system

We added the rest of the planets (and Pluto for posterity's sake) to test which solver is more stable or gives us the least amount of error. We also allowed for the Earth to move, but in our figures, you will not be able to notice this too much.

Both the RK4 and Verlet mappings of the entire solar system showed us results we were looking for as seen in Figure 4.



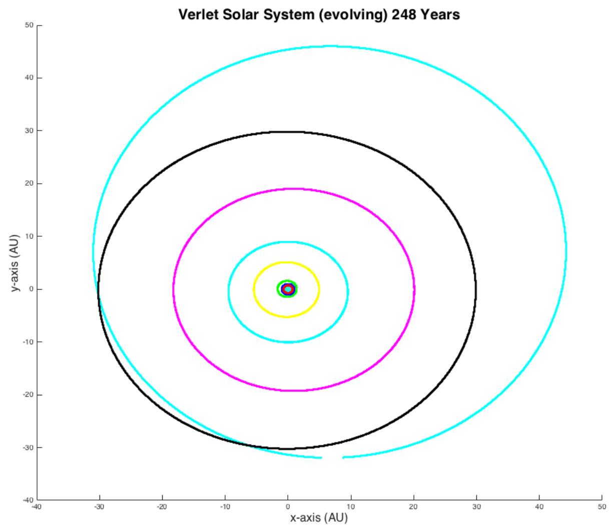
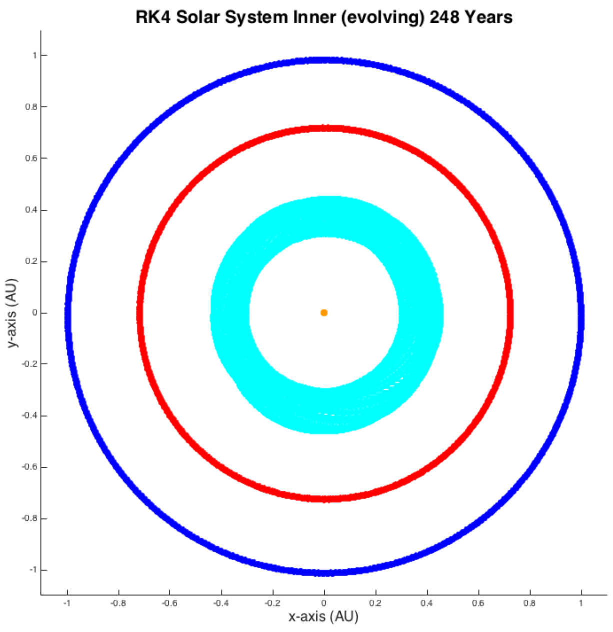


Figure 4: Entire Solar system over 248 years with the expected crossing of Pluto and Neptune's orbits.

What gets interesting is when we zoom in to see what is happening with Mercury, Venus, and Earth in Figure 5.



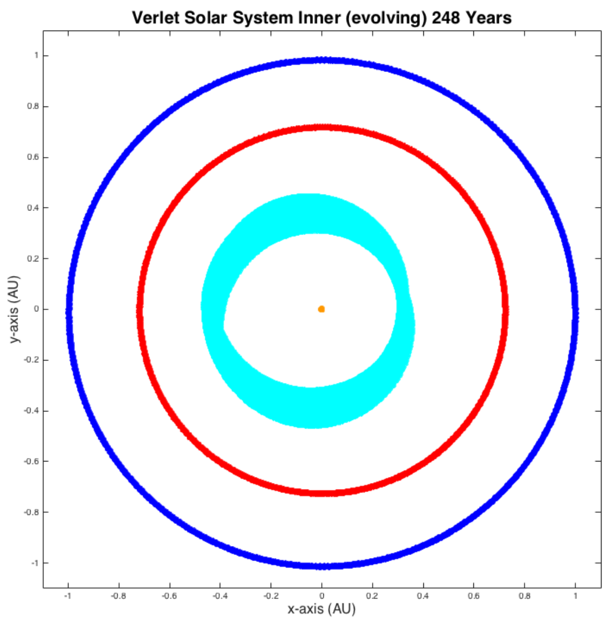


Figure 5: Inner solar system within the entire solar system model. It is obvious that the Verlet orbits of Mercury are more deformed than the RK4 orbits. This is with steps twice every day.

Even with steps twice as often as the previous systems, Mercury still does not have as tight an orbit as all the rest of the planets. The difference in errors can be seen easily in Figure 5, showing that the RK4 method is much better for precision than the Verlet method.

4 Conclusion

RK4 consistently produced the most accurate results proving to be more helpful when looking for accuracy than the Verlet method. However, it did prove to be computationally denser and therefore ran slower than the Verlet algorithm. If accuracy can be forfeited in order to decrease time, Verlet is the way to go. But if accuracy is needed, RK4 has an error two orders of magnitude smalled than the Verlet algorithm.

[1] Wolfram-Alpha, "Wolfram Alpha," http://www.wolframalpha.com.
[2] M Hjorth-Jensen, "Computational Physics MSU," https://github.com/CompPhysics/ComputationalPhysicsMSU (2016).
[3] J. Stewart, *Calculus: Early Transcendentals*, 6th ed. (Cengage Learning, Belmont, CA, 2008).
[4] "NASA JPL HORIZONS System," https://ssd.jpl.nasa.gov.
[5] H. Goldstein, C. Poole, and J. Safko, *Classical Mechanics,* 3rd ed. (Addison Wesley, San Fransisco, CA, 2002).