PHY480 Project 3: Solar System Model

David Butts, Daniel Coulter, and Liam Clink
Michigan State University
(Dated: April 3, 2018)

Abstract: Many problems in Physics can be written in terms of systems of coupled linear ordinary differential equations. In this project we are going to focus on Newton's Second Law, $\ddot{x} = F/m$. This equation can be written as two linear differential equations for position and velocity. The numerical solution of this type of problem amounts to time integration. We derived two integration methods, Forward Euler and Velocity Verlet, and compared their speed and accuracy. We found the increase in accuracy with Velocity Verlet was more than enough to make up for the increased computational cost. Using Velocity Verlet, we were able to make a stable model of our solar system.

I. INTRODUCTION

In this project we studied the n-body gravity problem. We compared Euler's method and Velocity Verlet at solving the differential equation that govern the solar system's motion. Gravity is a very long range force, where the potential decays as r^{-1}. With any potential dropping slower than r^{-5}, large inaccuracies and unphysical behavior can appear if a nearest neighbor scheme is used. Since a small number of particles will be used, it is reasonable to perform a straightforward integration procedure summing the forces contributed by all of the particles. We start by deriving the two integration methods we are using from Newton's Second Law, which are described in [3] and [2].

Planets in the solar system interact with a gravitational force which can be written as:

$$\mathbf{F} = \frac{GM_1M_2}{r^2}\hat{\mathbf{r}} \tag{1}$$

In this equation, G is the gravitational constant, M_1 and M_2 are the masses of the two planets interacting, and r is the distance between them. It is then trivial to calculate the acceleration using

$$\ddot{\mathbf{x}} = \frac{\mathbf{F}}{m} \tag{2}$$

Using this acceleration and the current velocity of the planet, the position can be calculated using the Forward Euler or Velocity Verlet method.

II. ALGORITHMS

We aimed to solve a second order differential equation that can be written as two coupled first order differential equations.

$$\mathbf{F}(x,y) = m\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} \tag{3}$$

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}(x,t) \tag{4}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \frac{\mathbf{F}(x,t)}{m} = \mathbf{a}(x,t) \tag{5}$$

We derived two algorithms to solve this system of equations, following Newman [3].

A. Forward Euler

The first, and easiest algorithm we will use is Euler's method. This algorithm can be derived by Taylor expanding Equations 4 and 5, and truncating at terms $\mathcal{O}\big(h^2\big)$ where h is the time step.

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\dot{\mathbf{x}}(t) + \mathcal{O}\big(h^2\big) \tag{6}$$

$$\mathbf{v}(t+h) = \mathbf{v}(t) + h\dot{\mathbf{v}}(t) + \mathcal{O}\big(h^2\big) \tag{7}$$

Henceforth, we will write $x_i = x(t), x_{i+1} = x(t+h)$, and similarly for v. Given we know what the first derivative of x and v are, we can rewrite the above equations and discretize them to find Euler's method.

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i \tag{8}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{m}\mathbf{F}(\mathbf{x}_i, t_i) \tag{9}$$

B. Velocity Verlet

The algorithm we will use in this section is the velocity verlet method. We can start by taking a Taylor expansion of Equation 4.

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\dot{\mathbf{x}}(t) + \frac{h^2}{2}\ddot{\mathbf{x}}(t) + \mathcal{O}\big(h^3\big) \tag{10}$$

Using the index formalism, this changes to:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\dot{\mathbf{x}}_i + \frac{h^2}{2}\ddot{\mathbf{x}}_i + \mathcal{O}(h^3) \tag{11}$$

The Taylor expansion of the velocity can be found and discretized in a similar way.

$$\mathbf{v}_i = \mathbf{v}_i + h\dot{\mathbf{v}}_i + \frac{h^2}{2}\ddot{\mathbf{v}}_i + \mathcal{O}(h^3) \tag{12}$$

We can write the second derivative of the velocity as follows:

$$\ddot{\mathbf{v}}_i = \frac{\dot{\mathbf{v}}_{i+1} - \dot{\mathbf{v}}_i}{h} \tag{13}$$

Using Equation 13 with Equation 12 we find an equation to update velocity.

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{2}(\dot{\mathbf{v}}_i + \dot{\mathbf{v}}_{i+1}) + \mathcal{O}(h^3) \tag{14}$$

Ignoring terms $\mathcal{O}(h^3)$ we find the Velocity Verlet method

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\dot{\mathbf{v}}_i \tag{15}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{2}(\dot{\mathbf{v}}_i + \dot{\mathbf{v}}_{i+1}) \tag{16}$$

III. RESULTS

A. Earth and Sun

We first started with a system of just the Earth and the Sun. Since gravity is a central force, this means that the simulation may be done in 2D without any loss of generality. To find the initial velocity of the Earth needed for circular motion, we started the Earth 1 AU in the x-direction from the Sun. Doing this constrains the velocity in the y-direction. From our introductory physics courses we know the relation between linear and rotational velocity for circular motion is:

$$v = \omega r \tag{17}$$

We know the Earth orbits approximately 1 AU from the Sun and orbits it with a period of one year. Therefore the initial velocity needed for circular motion is 2π AU/yr.

We tested stability by varying the length of the time steps. We found at a time step of about .05 our Velocity Verlet started to exhibit erratic nonphysical behavior, and

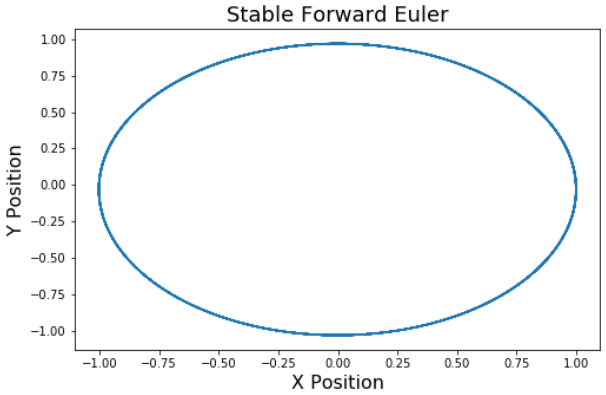

FIG. 1. Results of our Forward Euler calculating the position of the Earth with a time step of .01

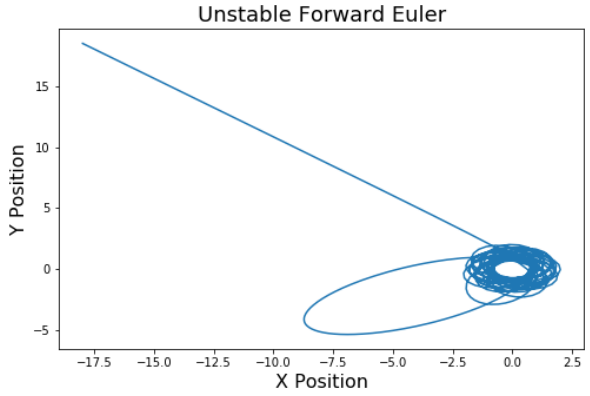

FIG. 2. Results of our Forward Euler calculating the position of the Earth with a time step of .13

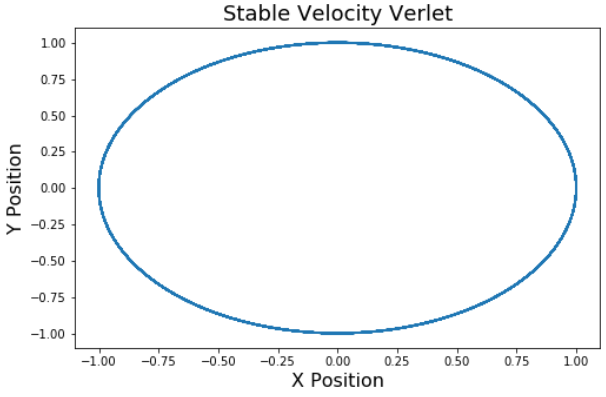

FIG. 3. Results of our Velocity Verlet calculating the position of the Earth with a time step of .01

at a time step of .03 for Forward Euler. The Velocity Verlet algorithm ejects the earth at a time step of .17, while the Forward Euler algorithm ejects earth at a time step of .13. Figures (1-5) show stable runs of the algorithms with

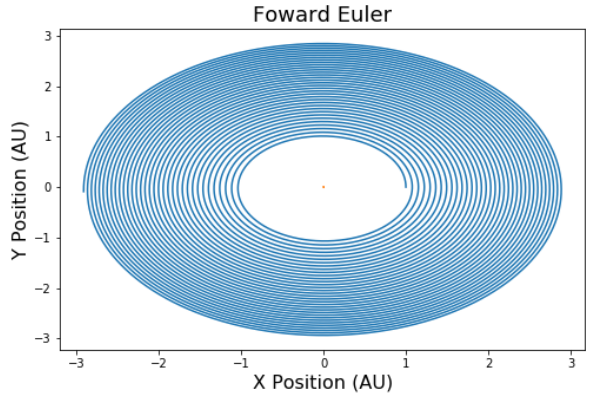

FIG. 4. This plot shows the outward spiraling behaviour that we expect from the Forward Euler algorithm. It is not a viable solution for use in scientific computation.

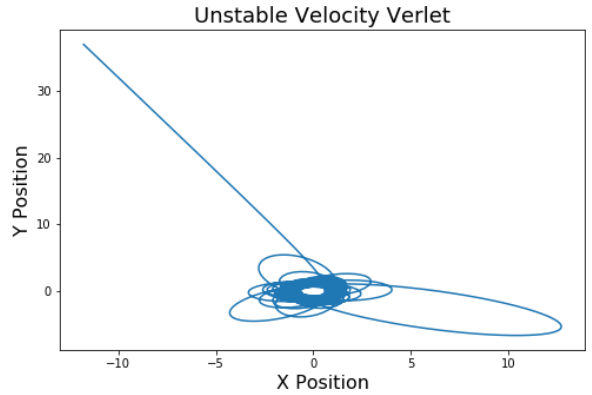

FIG. 5. Results of our Velocity Verlet calculating the position of the Earth with a time step of .17

a time step of .01, and both algorithms ejecting earth from orbit.

In the circular motion case, we checked to see if we were conserving the total energy of the system. We found that the total energy oscillates around a value for the Velocity Verlet algorithm. The Forward Euler algorithm has such a large loss of energy due to the outward spiraling that you can not see the oscillation in the Velocity Verlet. Forward Euler has this behavior because in circular motion, the tangential vector always points outside of the circle, so following it always leads to an outward spiral. The energy of the system for 10000 time steps with a time step of .01 can be seen in Figure 6.

As you can see the total energy is much better conserved using the Velocity Verlet algorithm. This is not surprising since Velocity Verlet is a second order method, so it should be more accurate.

The final test on our algorithms we performed was a timing study. We ran each algorithm for 10^1 - 10^5 iterations, with a time step of .01. The results of our

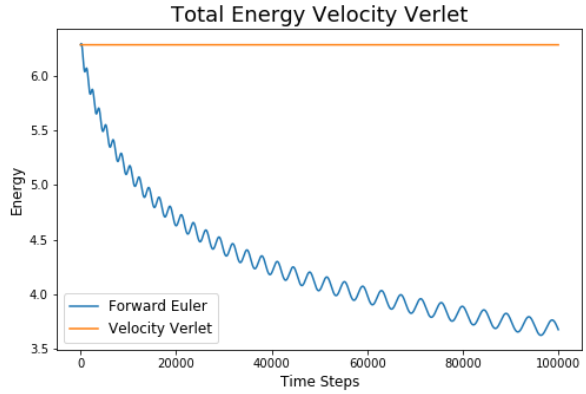

FIG. 6. The total energy of our system each time step. The Forward Euler algorithm is shown in blue, and the Velocity Verlet is shown in orange.

timing study can be seen in Figure 7.

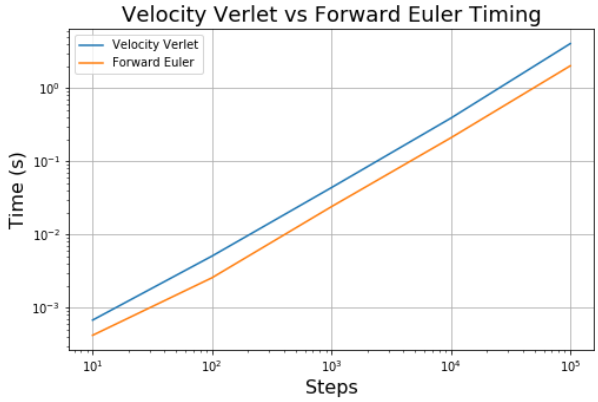

FIG. 7. Results of our timing study for the Forward Euler and Velocity Verlet algorithms. The Forward Euler time can be seen in blue, and the Velocity Verlet can be seen in Orange.

Unsurprisingly, the Velocity Verlet algorithm took more time than the Forward Euler, since it has more calculations. Forward Euler is about twice as fast, but much less accurate. The trade off in time is worth the extra accuracy of the Velocity Verlet method.

B. Escape Velocity

We can calculate an escape velocity given the equation

$$V_e = \sqrt{\frac{2GM}{r}} \tag{18}$$

where G is the gravitational constant, M is the mass of the object you are trying to escape, and r is the distance from that object you are starting. To find the escape

velocity from the sun, for the earth we can use the gravitational constant in units of $AU^3yr^{-2}M_s$and $r = 1AU$. Substituting those values into our equation allows us to find the escape velocity of the sun for earth to be 8.88 AU/yr.

To compare our Sun and Earth system to the calculated result, we started the Earth one AU from the Sun in the x direction. We decided to do this so we could constrain the velocity in the y-direction. By slowly raising the y-velocity, we were able to find a value at which the Earth's orbit was no longer bound. Through this process, we found Earth had an escape velocity of approximately 8.88 AU/year in our model. Our numerical result just about perfectly matched our calculated result.

C. Three Body Problem

Next, we extended our model to the three-body problem by introducing the most massive planet in the solar system, Jupiter, which has a mass of approximately 1/1000 that of the sun. Much like the two-body problem, Earth is ejected from the solar system with time step of 0.17 years (see Figure 8) and larger using the Velocity Verlet method (within approximately 1000 steps). We can therefore conclude that the affect of Jupiter on the Earth is negligible.

Since the Sun is so much larger than both Earth and Jupiter, the system was quite stable. Therefore, we also ran the simulation increasing the mass of Jupiter by factors of 10 and 1000 (Figures 9 and 10). When the mass of Jupiter was increased by a factor of 10, the time step at which the Earth was ejected was still 0.17 years, however the system was somewhat less stable due to the increase in momentum. On the other hand, when Jupiter was 1000 times as massive, the system was extremely unstable and devolved into chaos in less than two years at a relatively small step size.

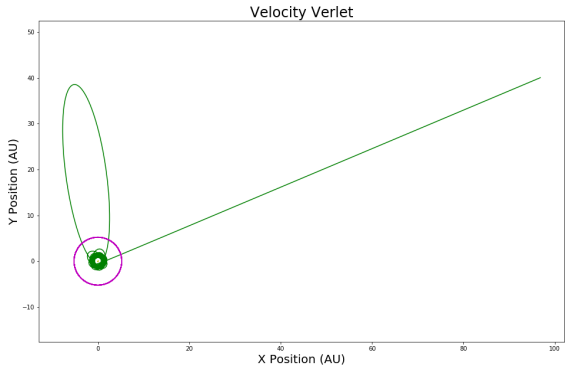

FIG. 8. Trajectories of the Sun (yellow), Earth (green), and Jupiter (magenta) for the three-body problem showing Earth kicked out at a step size of 0.17.

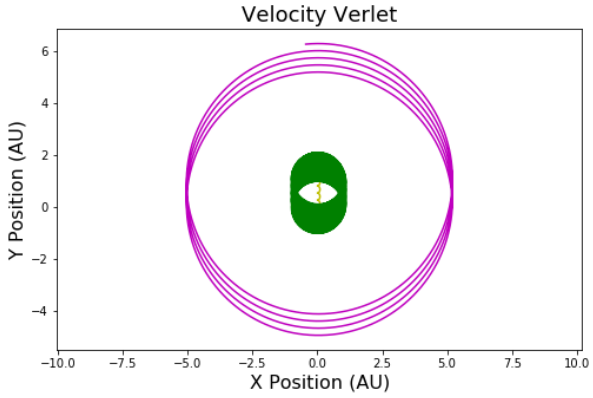

FIG. 9. Trajectories of the Sun (yellow), Earth (green), and Jupiter (magenta) if Jupiter had 10 times its normal mass for 5000 time steps with a time step of .01.

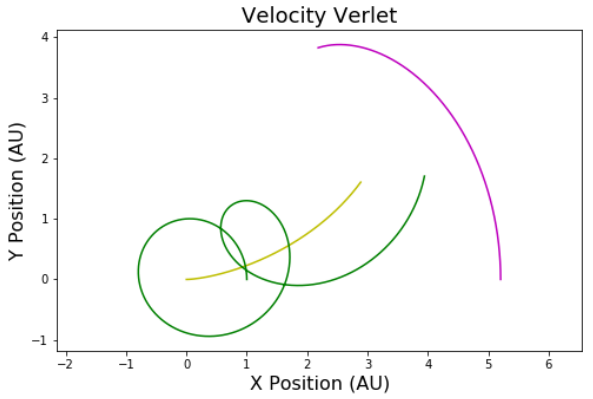

FIG. 10. This plot show the trajectories of the Sun (yellow), Earth (green), and Jupiter (magenta) if Jupiter had 1000 times its normal mass for 2000 time steps with a time step of .001.

D. Full Solar System Model

Finally, we extended our simulation to the whole solar system (including poor little Pluto because we still love him). We assumed circular orbits, which is somewhat accurate for all of the planets except for Pluto. We also used 2D, enforcing that the planetary orbits are coplanar. The final major simplification that we made was that the initial position of the planets were all along the x axis. This made initializing the velocity simpler; we only needed to set the y component to the magnitude calculated using $v = \frac{2\pi R}{T}$, where the period T was taken from [1].

IV. CONCLUSION

In this project we compared two integration algorithms for coupled differential equations, the Forward Euler method and the Velocity Verlet. The Forward Euler

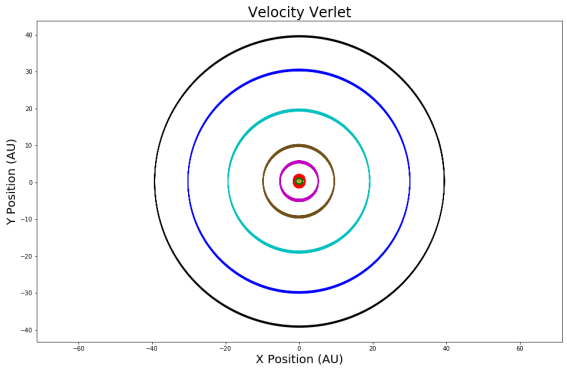

FIG. 11. Full solar system for 1000 years assuming circular orbits. We gave the Sun an initial velocity so that the total momentum is zero.

method is first order integration method and the Velocity Verlet is a second order integration method. As expected, the Velocity Verlet method provided much more accurate results in terms of stability of the systems. This was at the cost of a larger computation time since the Velocity Verlet method involves more intermediate calculations. However, both had time complexity $\mathcal{O}(n^2)$, so Velocity Verlet is the preferred method of integration for this type of problem.

Using a object oriented programming paradigm can make solving problems consisting of many similar data structures that are associated with each other in an easy to read and use way. This is important since if code is not readable, it is very difficult to maintain and improve. Most importantly, using object oriented programming allows this problem to be easily generalized to any number of bodies.

Lastly, our entire group found this project to be both educational and enjoyable. Specifically, it gave us experience with common integration techniques, which we were then able to easily analyze for stability and time complexity, as well as see the difference a symplectic integrator makes. Also, this was the perfect type of project to introduce object oriented programming, a very useful skill. Finally, the problem we are solving is interesting and it is very rewarding when your solar system model stays nice and stable for thousands of years.

[1] Horizons web-interface. `https://ssd.jpl.nasa.gov/horizons.cgi#top`.

[2] Patrick Chai and Evan Anzalone. Numerical integration techniques in orbital mechanics applications, 08 2015.

[3] Mark Newman. *Computational Physics*. CreateSpace Independent Publishing Platform, 2012.