Project 3

Evan Markel

My C++ program is located here in GitHub `https://github.com/evanmarkel/Project3`

Introduction

In this project, I have used numerical methods to solve second order differential equations to simulate the solar system. Using relative units for distance, velocity, and mass, I was able to discretize the equations in unitless quantities similar to previous projects. The Runge-Kutta4 method and Verlet algorithm have proven very accurate in modeling celestial dynamics according to Newton's and Kepler's laws. We begin by separating Newton's second law of motion into into its two-dimensional Cartesian components for a circular orbit here:

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{\text{Earth}}},$$

and

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{\text{Earth}}},$$

We then can use these equations along with the equation for force below to derive four coupled first order differential equations for celestial motion in two-dimensions.

$$F_G = \frac{M_{\text{body1}} v_{body1}^2}{r} = \frac{G M_{body2} M_{\text{body1}}}{r^2},$$

We then express the motion of body 1 around body 2 with four component time derivatives:

$$\frac{dv_{x1}}{dt} = \frac{-G * M_{body2}}{r^3} * x_{body1}, \qquad \frac{dx_{body1}}{dt} = vx_{body1}$$

and

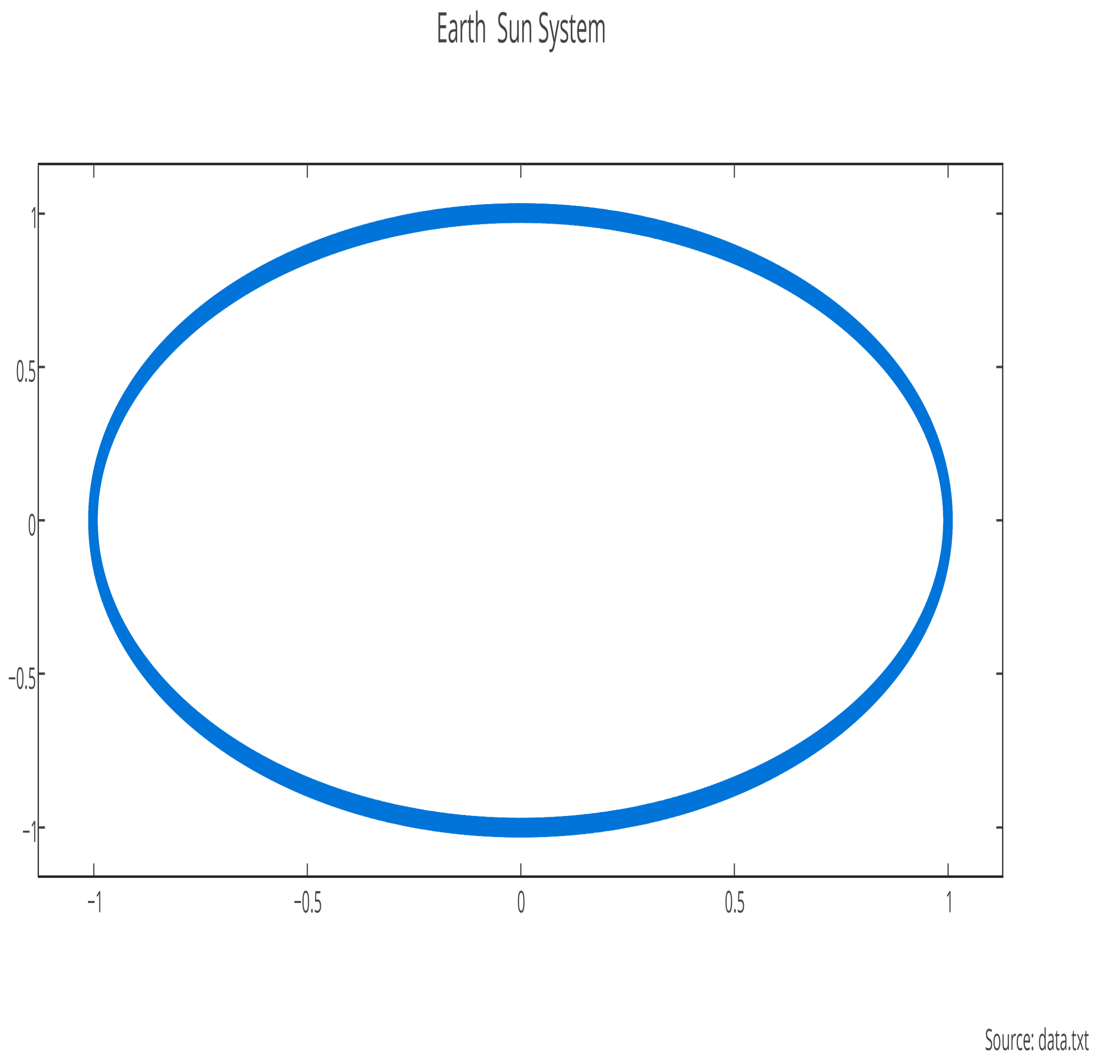
$$\frac{dv_{y1}}{dt} = \frac{-G * M_{body2}}{r^3} * y_{body1}, \qquad \frac{dy_{body1}}{dt} = vy_{body1}$$

The above equations now become the and utilizing the numerical Verlet, Euler-Cromer, and Runge-Kutta algorithms to solve the differential equations pertaining to the time derivatives in the physical laws.

Two-Body Problem These equations nicely predict the motion of the earth around the sun in my simulation. My code creates a celestial object which generates a vector in the SolarSystem class that contains the initial position and velocity for the object in two dimensions. The system is then updated using the RK4 method and the new position and velocity are logged. Proceeding in this manner, the plot of these data points produces the orbit. This graph runs for 100 years with a calculation step length of .01 years. The broad blue band shows the oscillation in the orbit. For the Earth, we can easily find the needed initial velocity for circular motion:

$$v_{earth}^2 r = GM_\odot = 4\pi^2 \mathrm{AU}^3/\mathrm{yr}^2.$$

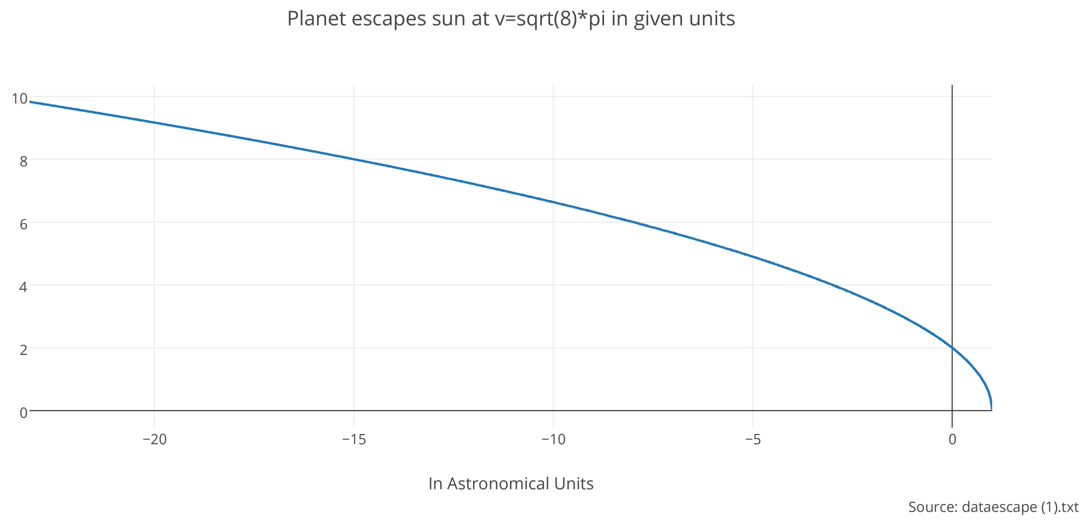
For $r = 1AU$ we see that in these units $v_{earth} = 2 * \pi$. Using this velocity and initial position of (1AU,0) we get a stable orbit.



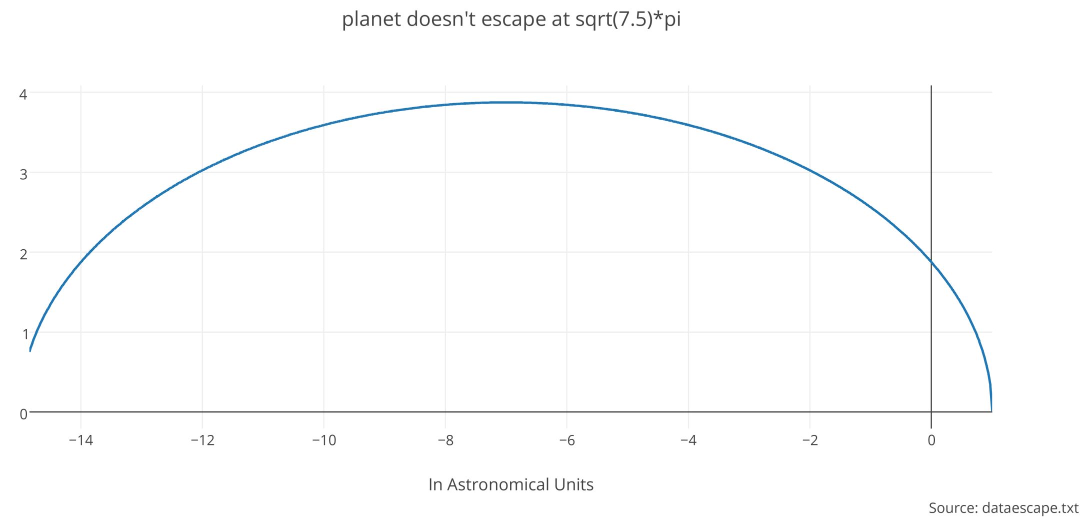
This orbit is stable as we can see from the graph. Earth orbits the sun (invisible at center). The mass of earth is given in the relative units of the sun's mass (3e-6) and the distance is 1AU. Also, the kinetic energy, potential energy, and angular momentum are all constant in this simple scenario. We see that $KE = \frac{1}{2} * m * v^2$, $PE = \frac{M_\odot * M_{earth}}{distance=1AU}$, and $L = distance x velocity$ are constant with constant velocity, distance. In this scenario I found the kinetic energy to be 5.9217e-5 and the potential energy to be the mass of the earth in the given units ($M_\odot$ and $r = 1$). Changing the velocity of Earth can disrupt this stable orbit. I found that by increasing the velocity by about 40 percent will allow the earth to escape the sun's gravity. There is an analytical solution as the escape velocity is given by

$$v_{escape} = \sqrt{\frac{2 * G * M_\odot}{r}}$$

We can see that at $v_{earth} = \sqrt{8} * \pi$ earth does escape. In more physical units, the escape velocity of Earth is 42.1 km/s.



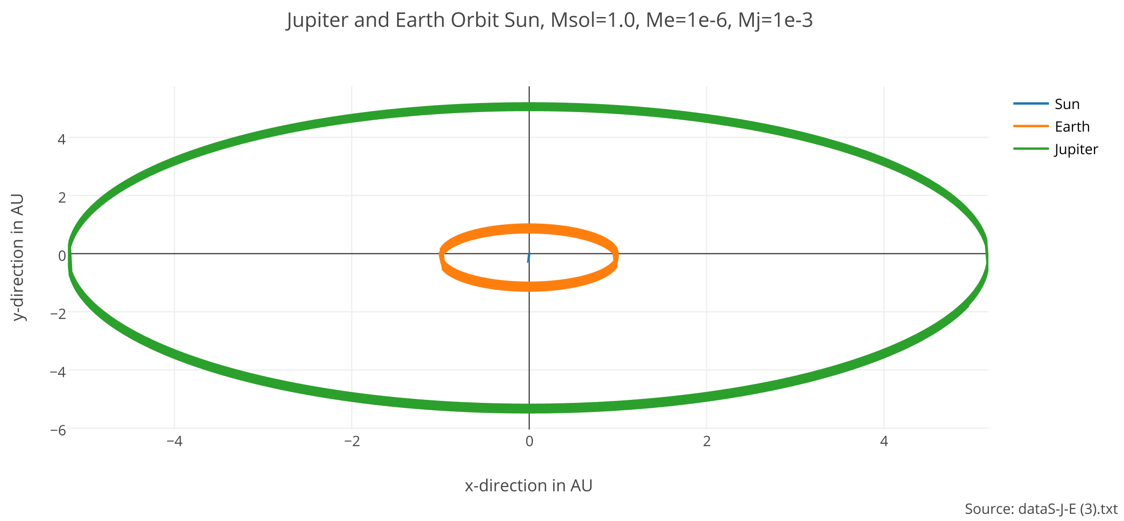
However, at $v_{earth} = \sqrt{7.5} * \pi$, the earth does not escape but instead takes a much longer orbit.



hree-Body Problem Now I add Jupiter to the solar system. The introduction of a third body means that the force interaction between Earth and Jupiter must be accounted for in my code as well as the major force of each body to the sun:

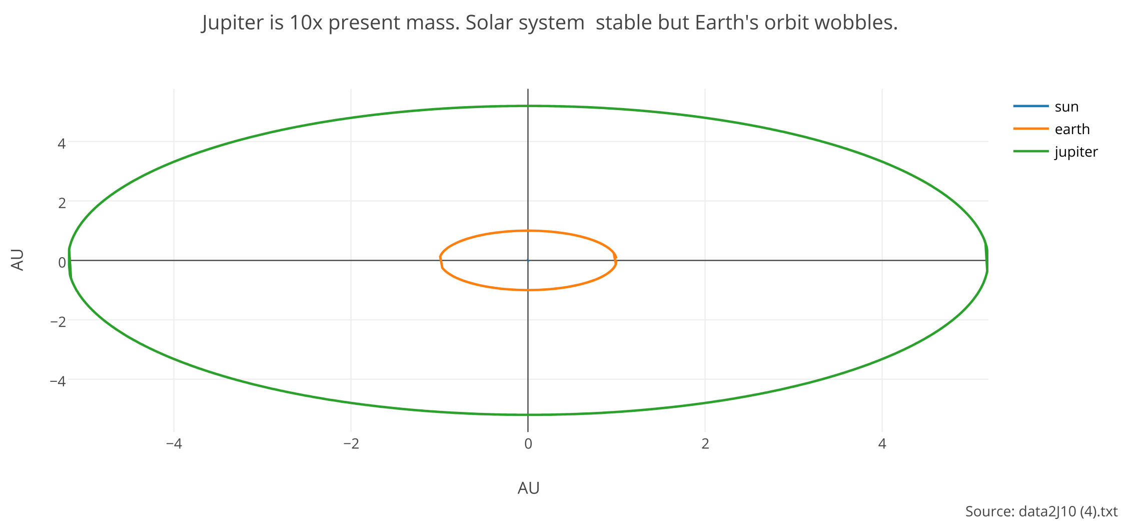
$$F_{\text{Earth}-\text{Jupiter}} = \frac{GM_{\text{Jupiter}}M_{\text{Earth}}}{r^2_{\text{Earth}-\text{Jupiter}}},$$

To do this, my code contains a double loop that calculates the forces and potential energy between all bodies in
the solar system. This is useful for easily adding new celestial objects to the solar system.

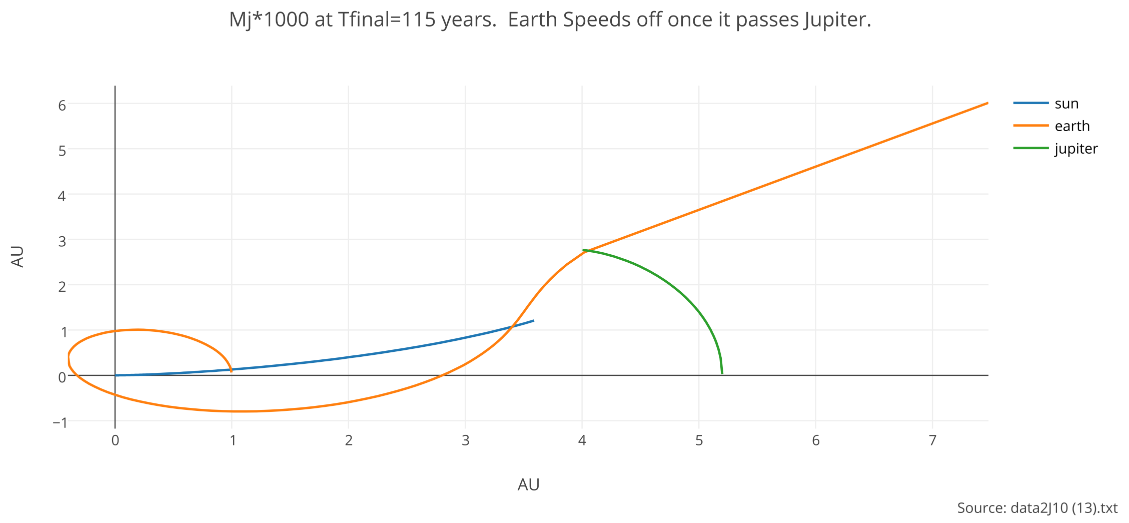


Jupiter is a large object (one thousandth the mass of the sun) and perturbs the orbit of the earth. The RK4 and Verlet algorithm is now less stable because of this. I found that when my step length became larger than .05 years, the system would become unstable. Run over long periods of time, one of the positions would diverge, meaning that the object would be ejected out into space. However, keeping the step length at .01 and running the system for 700 years gave pretty stable results. The orbital velocity of Jupiter can be found by the same means as earth's. We can just take a ratio here of the distance from the sun divided by the orbital period. In jupiter's case 5.2AU/12years gives the orbital velocity as .434 that of earth's.

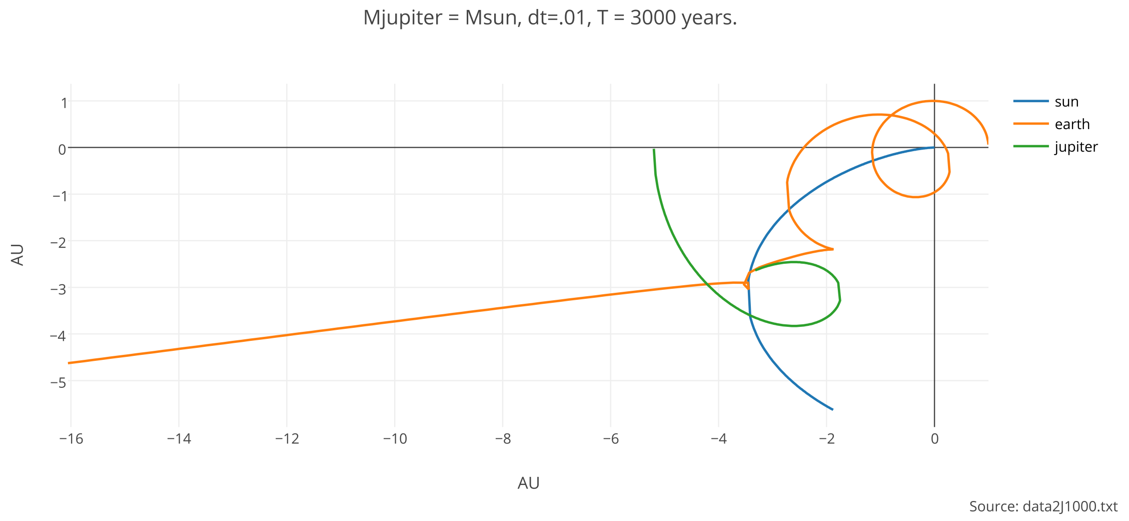
Now let's examine two cases that can further de-stabilize the solar system. When the mass of Jupiter is increased, this has a noticeable effect on earth. First, when Jupiter is ten times more massive, the solar
system is still stable but the earth begins to spiral within its larger orbit around the sun.



When Jupiter is 1000 times more massive then it's equal to the sun. The earth becomes a planet in a binary star system. I'll examine two cases, both of which are unstable for earth and the planet is ejected out from the solar system and a high velocity. In the first graph we see what happens when earth and jupiter start along the same axis although with the earth at 1AU and jupiter at 5.2AU and moving in the same direction.



The sun and jupiter begin to move toward each other and earth is slingshotted out of the picture. Note, the step length here is .01 years so the timeframe of the motion is 1.15 years.

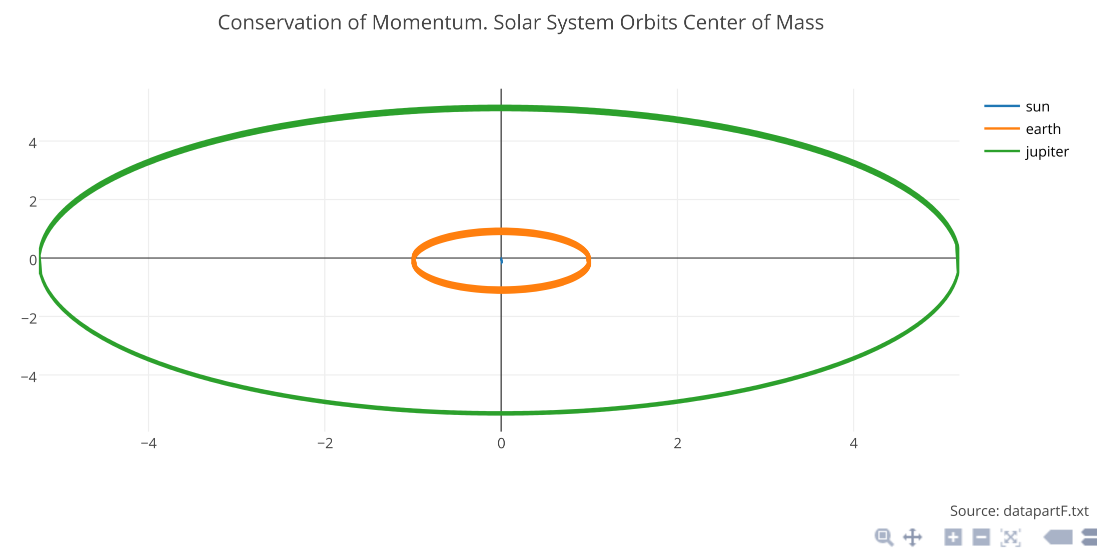


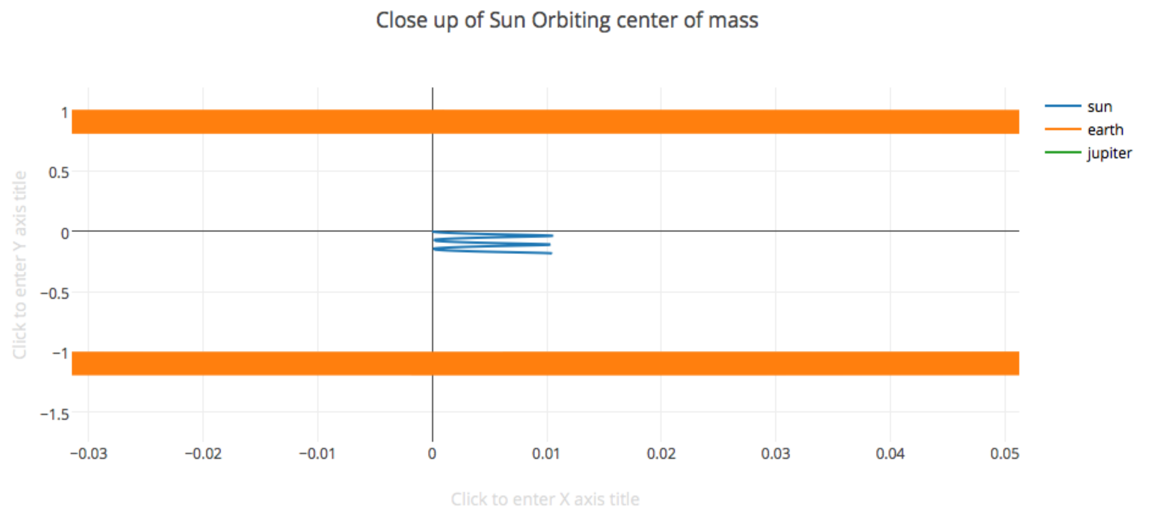
In this graph, earth and jupiter begin along the same axis but this time they move in opposite directions. The step length is again .01 so 30 years are shown. The earth remains in orbit around the binary system for about twice as long in this case but is ultimately ejected. The system is slightly more stable in this case because the momentum imbalance of the system is less severe as compared to the first graph of $M_{jupiter} = M_{\odot}$.

The total momentum of the system needs to be conserved in order for the solar system to be stable. The sun is not stationary but orbits the center of mass of the solar system as do all of the planets. To conserve momentum for the three body simulation, I set the velocity of the sun in such a way that it balances out the momentum contributions from earth and jupiter according to this equation:

$$M_{\odot} * vy_{sun} = M_{earth} * vy_{earth} + M_{jupiter} * vy_{jupiter}$$

We know the velocity of earth is $2 * \pi$ and the velocity of jupiter is $.4727 * \pi$. Solving for vy_{sun} as all other components are known, we find the sun should have an initial velocity of .00890885 in the opposite direction of the earth's and jupiter's in order to conserve momentum. The results are below.





Here we see in the closeup that the sun is not stationary but undulates with the periodicity of the two other smaller bodies.

N_{step}	λ_1	λ_2	λ_3	**No. iterations**
10	2.9338	6.6598	10.1415	139
50	2.9970	6.9850	10.963	4,039
100	2.9992	6.9962	10.991	16,474
125	2.9995	6.9975	10.994	25,826
150	2.9997	6.9983	10.996	37,443
175	2.9997	6.9987	10.997	50,991
200	2.9998	6.9990	10.998	66,827
225	2.9998	6.9992	10.998	84,775
250	2.9999	6.9994	10.999	104,290
275	2.9999	6.9996	10.999	128,507
400	3.0000	6.9998	11.000	269,020

Now we'll look at the run-time of the Jacobi method versus the the armadillo solver. The values :ϵ=1e-8 and ρ_{max} =5.

N_{step}	**Armasolve time**	**Jacobisolve time**
10	.00071311	$7.4863x10^{-5}$
50	.00955606	.021553
100	.119834	.297856
150	.292907	1.43716
200	.543296	4.43368
250	.854114	10.6005
400	.879044	73.0668

We see that the Jacobi is actually faster for N=10 but then gets much slower as N increases into the range that we need for numerical precision.

Part C We now move on to the two electron case. The main difference here is the change in potential. The Schrödinger equation simplifies as in part a as follows:

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4\rho^2\psi(\rho) + \frac{m\alpha\beta e^2}{\rho\hbar^2}\psi(\rho) = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho).$$

The change in potential from a goes as ρ^2 to $\omega_r^2\rho^2 + 1/\rho$. The ω_r term indicates the strength of the oscillator potential and will influence the values of the energy states of the two particle system. We look at the cases where $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1$, and $\omega_r = 5$. We change the potential and re-run the program for the 4 different values of ω_r. The first 3 eigenvalues are given in the table below. The results use N_{step} at 250, ρ_{max} at 5, and ϵ at 1e-8.

ω_r	λ_1	λ_2	λ_3
0.01	0.84075	2.1737	4.2372
0.5	2.2310	4.1691	6.3858
1.0	4.0577	7.9091	11.818
5.0	17.445	37.057	56.817

Here are the values for the first 3 energy states. We see the dependence on ω_r is quite large for the values of the energy levels. For $\omega_r = .01$, the electrons are not closely coupled and the energy level of the oscillator is even lower than for the single electron case. We see when ω_r grows to 1 and 5, the values are higher. Physically, the electrons are brought closer together in the oscillator well and the repulsive force means that the energy needed for the state to exist is greater. The greater potential should lead to a higher value of the eigenvalues.

Part D Finally, we want to plot the wave functions for the first 3 energy states for the 4 ω_r values. The eigenvectors contain the constituent elements of the wave equation at the various positions along $[0, \rho_{max} = 5.0]$. The tricky part is indexing the locations of the eigenvectors within the eigenmatrix from the position of the corresponding eigenvalue. Since the Jacobi algorithm does not necessarily return the eigenvalues in sorted order,

another for loop is needed to extract the first 3 eigenvectors–> corresponding to the 3 lowest energy states.
The next tricky part is determining how to normalize the eigenvectors. From quantum mechanics, we know that in an infinite potential well, the wave function should be normalized by $sqrt(2/lengthofwell)$. In our case, we have abstracted the parameters in this assignment, and therefore the normalization is a bit more abstract here. There should be an h term-or a similar term that takes into account the length of the well-included in the normalization. While this may be a concern, the probability distribution function is the square of the wave equation and must sum to 1. Therefore, any constant in front of the wave equation can be eliminated as it will not change the relative values of the wave equation when multiplied uniformly. And since we are normalizing to 1, the term can be omitted. So in my code, I've squared the eigenvectors and applied the armadillo normalise() function. I've included 4 plots for the 4 different values of ω_r. The first 3 eigenvectors are graphed on top of each other in each case to highlight the relative probability strengths.

We can see the coupling effect of the oscillator strength. The electron can spread out more freely with lower ω_r values and is more tightly bound for higher ones. The shape of the distribution functions seems correct and the strong and weak particle interactions due to the strength of the harmonic oscillator as ω_r are also apparent from the plots. The area under the curve for each of the 3 energy states sum to 1. The peaks for the energy states 2 and 3 have higher probabilities closer to zero for ω_r= .01 and this is reversed for ω_r = 0.5, 1.0, and 5.0. I have some reservations about the correctness of the eigenvector plots, but here's my hand-waving attempt: I imagine the probability of the smaller relative radius would be greater for a weaker harmonic oscillator coupling than for a stronger one. Based on these plots, I conjecture that the higher ω_r means a more tightly bound nucleus with the electrons pushing more towards the outside of the well (Coulomb repulsion) which is a result of the higher potential energy of that system.The lower ω_r strength has a higher entropy and lower energy than the higher ω_r strengths. We can also see this higher cost energy effect in the relative growth of the first 6 eigenvalues as plotted below in a stacked area chart divided by the four oscillator strengths:

Conclusion This project has interesting physical takeaways and demonstrates how some simple algorithms can aid a great deal toward solving problems where there is no easy analytical solution.