Lab1

Augustine Tai and Habiba Zaghloul

Foreword: the codes are also all annotated.

Q1 a) Nothing to submit

Q1 b) Pseudocode

- Import the needed libraries, numpy and matplotlib.pyplot
- Defining given constants, initial conditions, and time step parameters
    - Define Ms (mass of the sun) as = 1 = 2*10**30kg
    - Define gravitation constant G as = 39.5 (AU^3)(Ms^-1)(yr^-2)
    - In the initial conditions, velocity is measured in AU/year, position in AU, and time in years.
    - Time step for 1 year with step size of 0.0001 year
- Make empty arrays using np.empty for store values during the loop
- Place initial conditions into empty arrays created
- Use a For loop to timestep to calculate the radial length, x and y position, x and y velocities and angular momentum using the Euler-Cromer method
- Plot the x-velocity vs time, y-velocity vs time, x-position vs time, y-position vs time, and angular momentum vs time

Q1 c)*see comments on the code

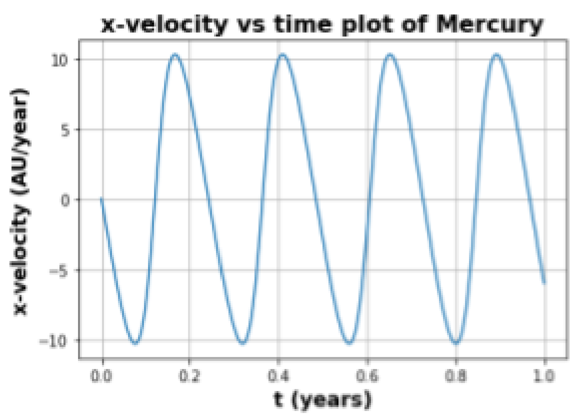

Figure 1.

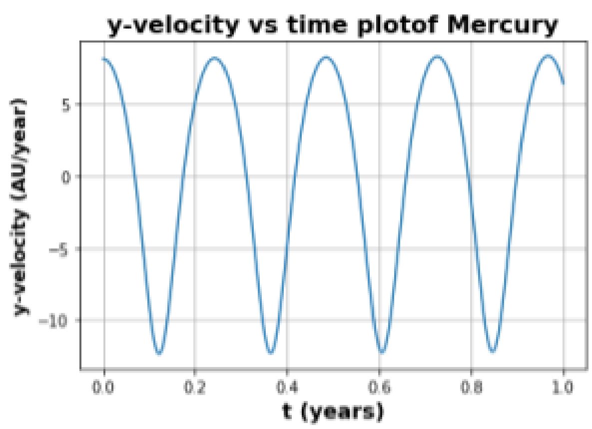

Figure 2.

First, using the initial conditions given by the land handout, the x and y components of Mercury's position and velocity were calculated for one year with a timestep interval of 0.0001 years. By observing Figures 1 and 2, the x and y velocities of the planet oscillate with time as the planet orbits around the sun, exhibiting the expected planetary motion. Furthermore, by observing Figure 3, the x vs y positional plot of Mercury clearly shows the planet's stable elliptical orbit path with little change in its orbit, unlike in Q1 d).

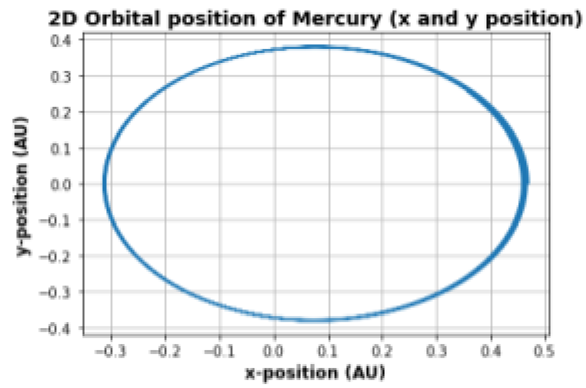

Figure 3.

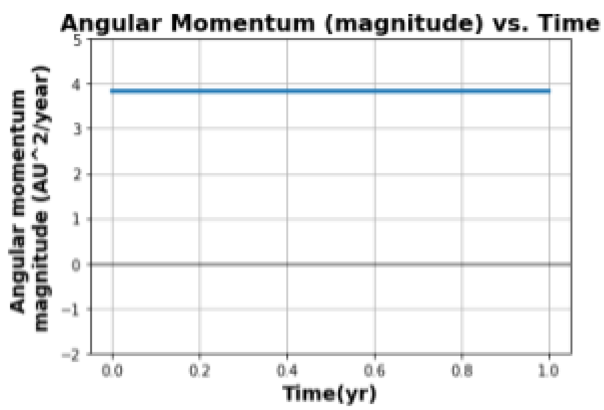

Figure 4.

By observing Figure 4, the angular momentum remains constant throughout the year, indicating that momentum is fully conserved. The angular momentum was calculated by the cross product of the planet's linear momentum (p = mass*velocity) and its position vector (r): L = r x p. Because we are only considering the movement of the planet in the x and y plane the Linear momentum of the planet is = x-position*mass*y-velocity - y-position*mass*x-velocity. Even though the planet's mass (Mercury) was not given, by observing the angular momentum's magnitude, angular momentum remains constant, as the planet's mass would only be a scaling factor (scalar multiple).

Q 1 d) In this part, we altered the gravitational force and used the constant a=0.01 AU^2 to be able to see the effect the sun has on Mercury's orbit. This exaggeration helps clearly see the changing aphelion of Mercury's elliptical orbit. By looking at figure 5, the aphelion can be seen to get as far as (0.5,0.0). Note that these are the cartesian coordinates of Mercury, given in AU.

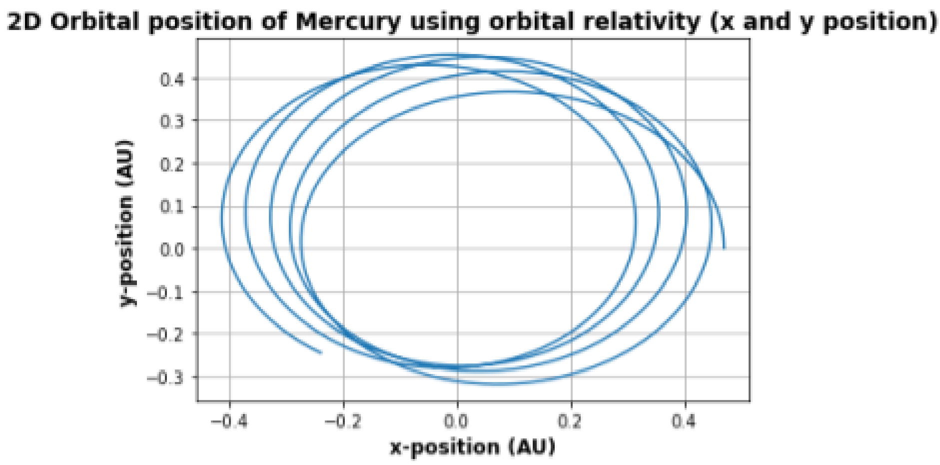

Figure 5.

2a)

- Define given constants for earth and jupiter
- Make empty arrays (for both earth and jupiter) to be used in the loop
- Place initial conditions into the arrays made
- Set time to 10 earth years and create a new time array
- Calculate the distance between earth and jupiter
- Use a for loop to calculate
    - 1. Radial length for earth, jupiter and distance between earth and jupiter
    - 2. X and y velocity vectors for earth and jupiter
    - 3. Use the x and y velocity vectors to find the x and y vectors of both planets
- Plot x vs. y positions of both planets for 10 years and for 3 years (q2b)
- Change jupiter's mass to equal the mass of the sun in q2b
- Q2c: set new conditions for the asteroid and change time to 20 earth years
- Create empty arrays and add the initial values
- Repeat the for loop and replace earth's values with the asteroids'
- Plot the x vs y positions of the asteroid + jupiter

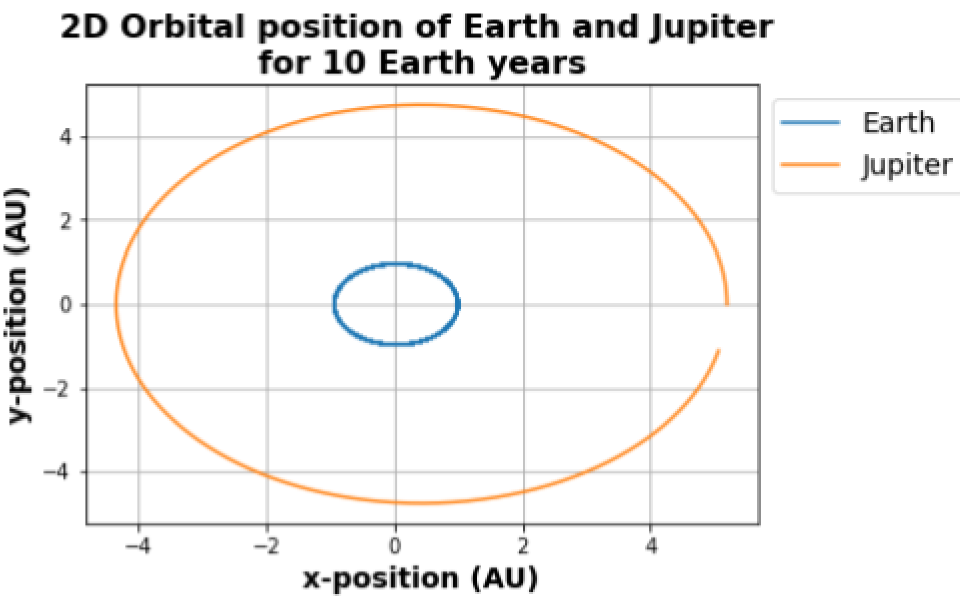

Figure 6.

Observing Figure 6, even with the addition of Jupiter to the Earth and Sun system, no significant effect on Earth's planetary motion can be seen. This makes sense as the gravitational force from the Sun would be significantly larger than Jupiter's, as well as being closer to Earth. As a result, Jupiter's effect on Earth with the given initial conditions should be minimal. Note that because Jupiter's orbit period is around 12 years, it hasn't completed a full orbit of the sun, explaining the gap in its orbit.

Q2b) In this part, Jupiter's mass was increased to 1000x its original mass making it the same mass as the sun to see the effects of a sun-like Jupiter on Earth. This change in Jupiter's mass helps visualize the effects of another planet on Earth. Observing Figures 7a and 7b shows that the orbital path of Earth during 3 years under the influence of a supersized Jupiter shows only slight variations in its orbital motion.

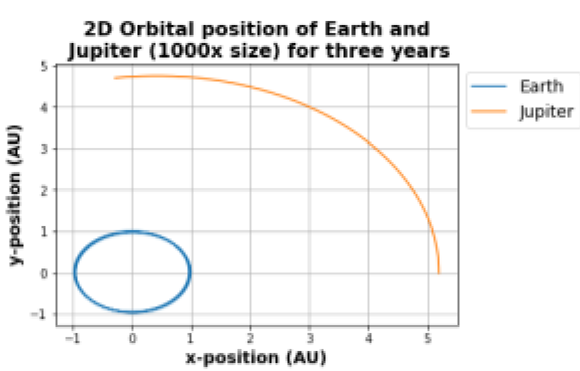

Figure 7a.

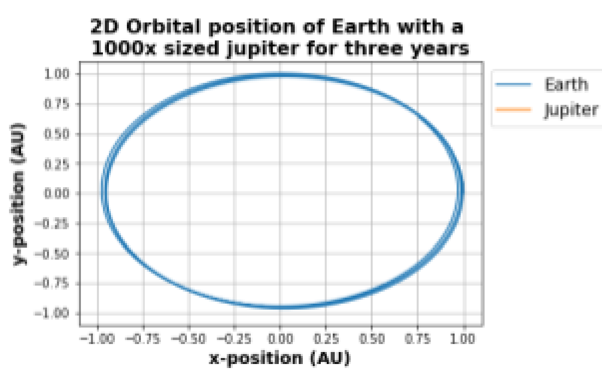

Figure 7b. (earth's orbit from figure 7a)

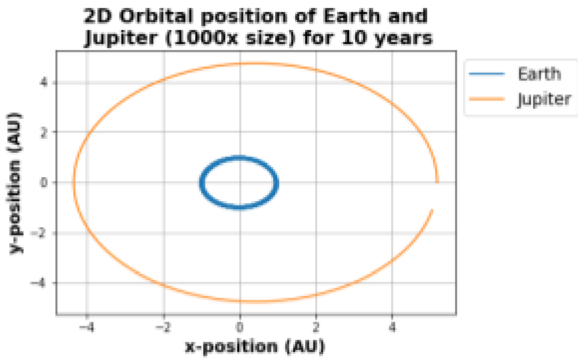

Figure 8a.

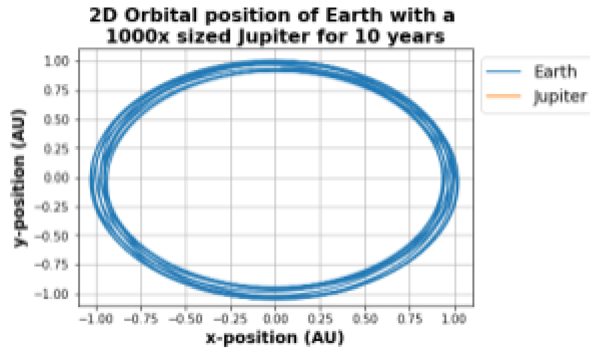

Figure 8b. (Earth's orbit from figure 8a)

But when Earth's orbital motion is plotted for 10 years under the influence of the large Jupiter, Figures 8a and 8b show the significant change in Earth's orbit. Despite this, it seems that Earth will still remain in the Sun's orbit and not be kicked out of its elliptical orbit path by Jupiter despite the significant effect of the supersized Jupiter on Earth's motion.

Q2c) In this part, Jupiter returned to its normal size, but the Earth was replaced with an asteroid orbiting farther away from the sun and slower (see lab for new initial conditions) to see the effects of Jupiter on an asteroid orbiting much closer to it. Despite the asteroid's proximity to Jupiter, the asteroid still exhibits a stable elliptical orbit, as seen in Figure 9a. By observing a zoomed-in version of the asteroid's motion in Figure 9b, some perturbations in its orbit indicate the possible effects of Jupiter on the asteroid's orbit. But overall, it seems that Jupiter does not significantly impact the asteroid's orbit.

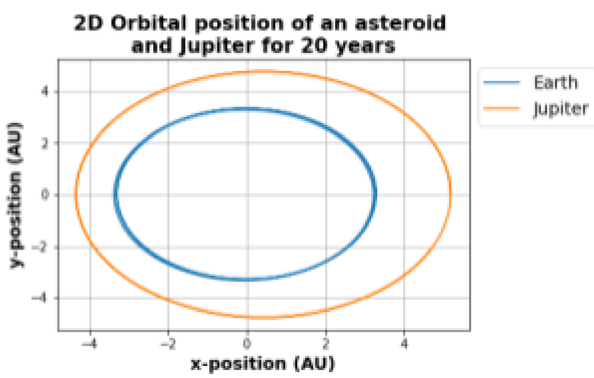

Figure 9a.

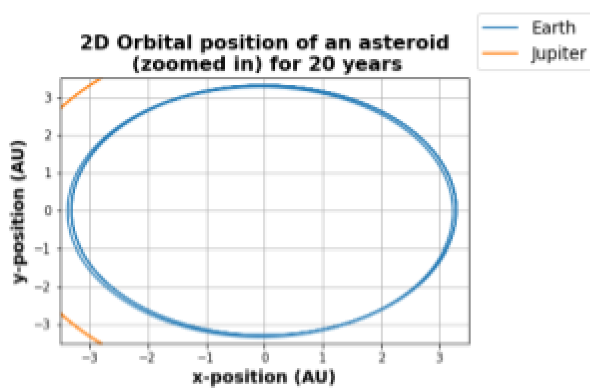

Figure 9b. (Asteroids orbit from figure 9a)

3) In this question, we took N to be 300 (range 2-300). The time() function was used to calculate the time it took for each multiplication to be executed. Using the snippet of code provided by example 4.3 in the textbooks, matrices of various sizes were multiplied and timed in several For loops (see code for details). As the matrix size grew, the elapsed time increased exponentially, as seen in Figure 10. On the other hand, when the matrices were multiplied together using the numpy.dot function, the time required to complete each multiplication remained constant as matrix size increased. As a result, based on Figure 10, numpy.dot is significantly better at multiplying matrices.

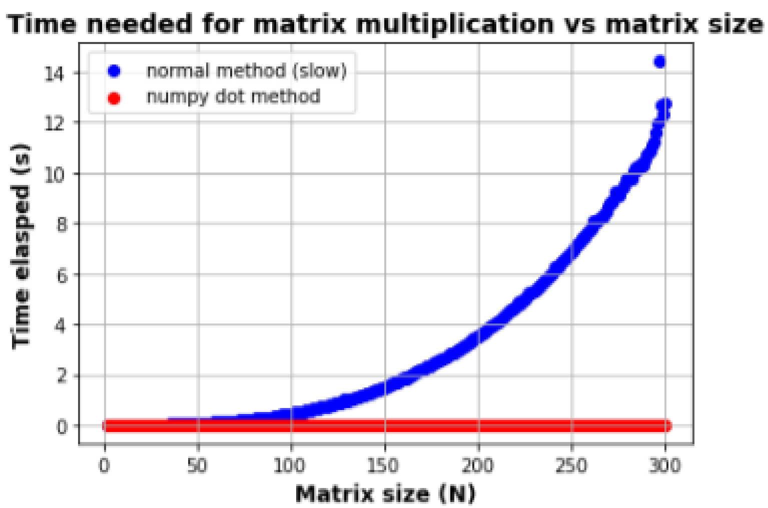

Figure 10.

Furthermore, by plotting the elapsed time as a function of N^3, Figure 11 shows that the time needed to multiply matrices using the example 4.3's method became linear with N^3. This linear trend suggests that the relationship between time elapsed and matrix sizes are cubic, with time elapsed increasing exponentially (cubic) as matrix size increases. Once again, indicating numpy.dot is significantly more efficient at performing matrix multiplication when the sizes are larger.

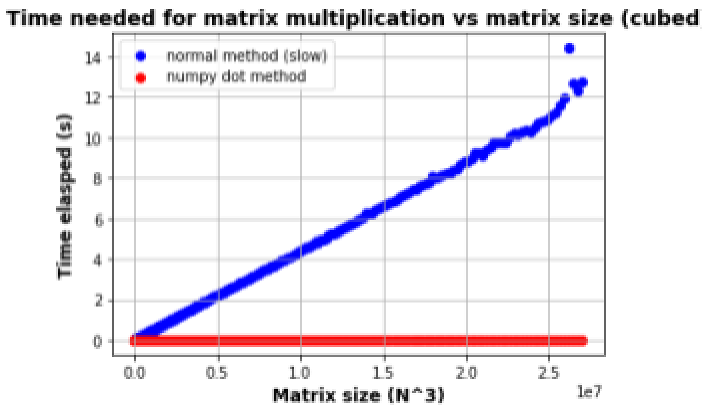
Time needed for matrix multiplication vs matrix size (cubed)
normal method (slow)
numpy dot method
Time elasped (s)
0
2
4
6
8
10
12
14
0.0
0.5
1.0
1.5
2.0
2.5
1e7
Matrix size (N^3)

Figure 11.