**PHYS 514 – Solution Set # 6**

**Tevfik Can Yüce**

**Problem 19:**

**Part (a):** Let’s start with the first equation given below.

Now use the scaled quantities that we want to implement.

If we use these changes in the equation, we will get following equation.

Now use the following identities;

Then we can have the final form of the scaled equation.

Before calculating the values of the scaled constants, let’s investigate why we have used given . Kepler’s third law states that period of the motion of smaller mas around big mass is as given below.

Therefore, we can ensure that period of the small masses movement will be periodic w.r.t big mass with period in scaled unit of time. Now let’s calculate the scaled constants in the final form of equation.

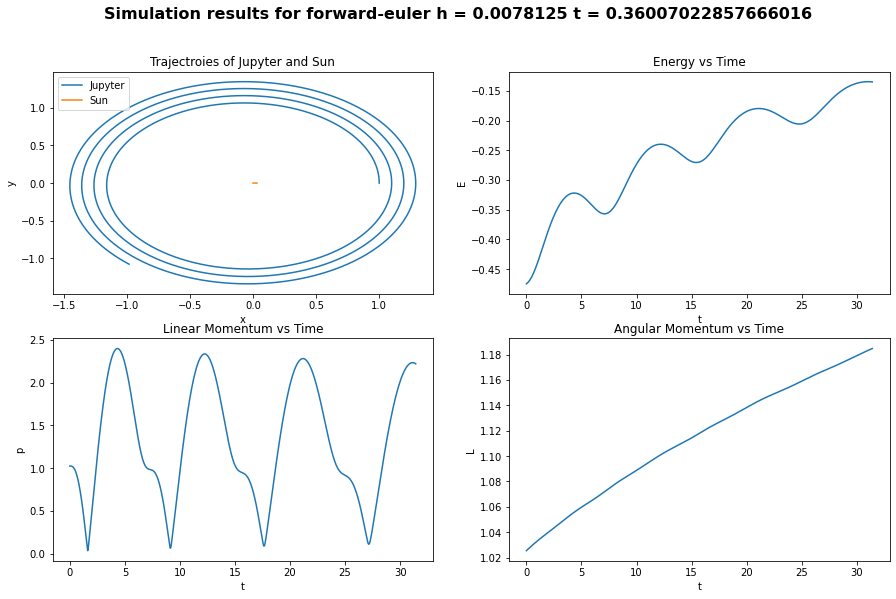
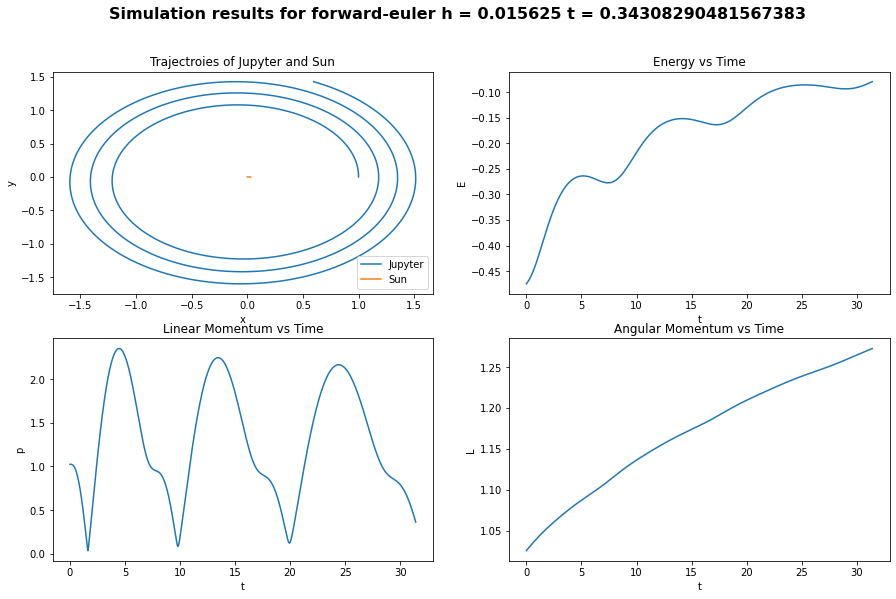
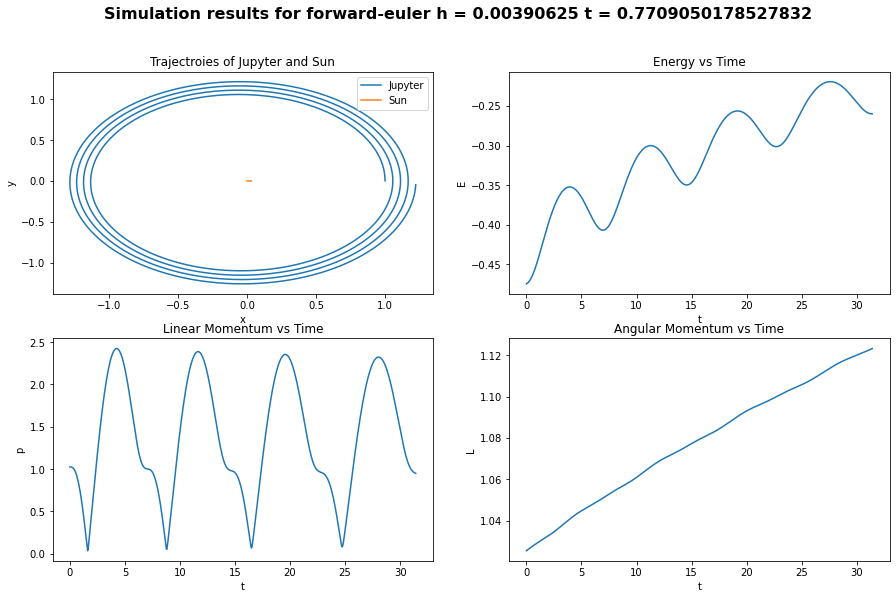
Let’s start with calculating , which is as given below.

In addition, initial distance of Jupiter to sun is . Now let’s look at the initial velocity of Jupiter which is calculated as . Now calculate the reduced masses.

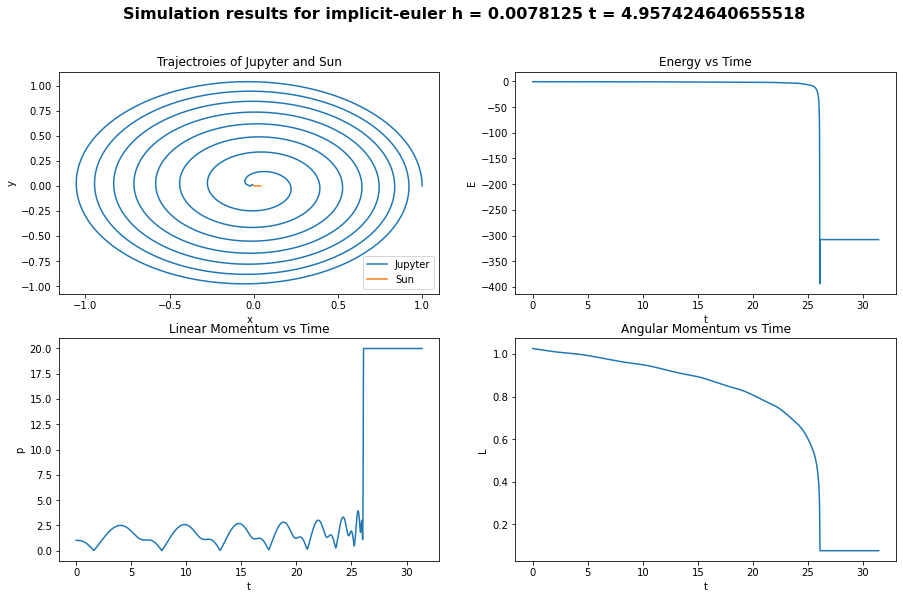
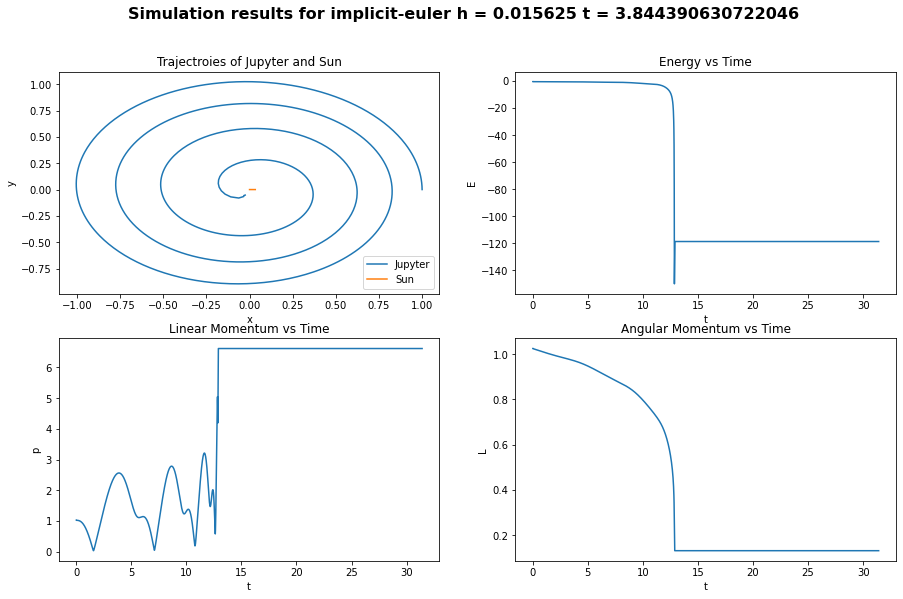
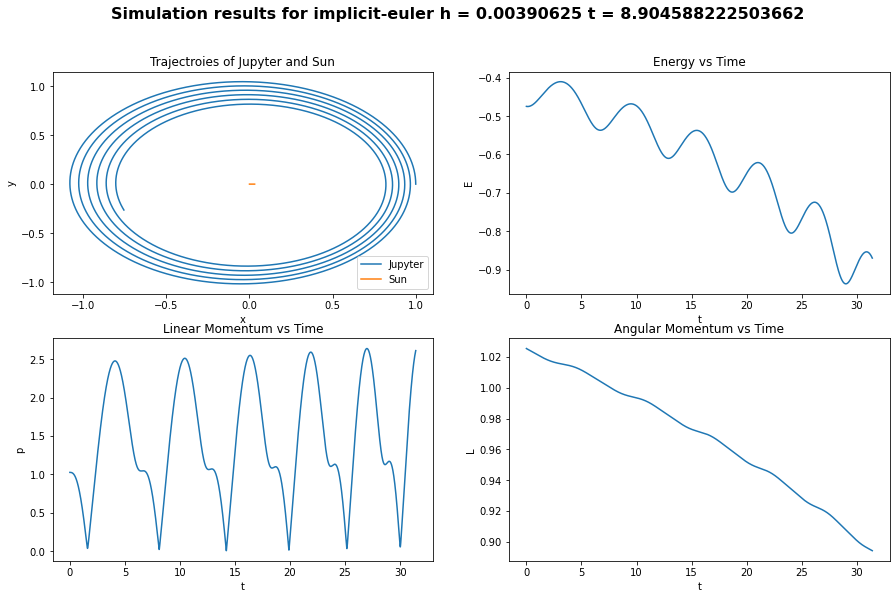
**Part (b):** We can express second order ODE with two variables as first order ODE of four variables. Let’s start with defining our vectors and write time evolution equations again.

But note that, each of them is a vector, then we have eight variables to be evaluated. But since x and y equations are independent of each other, we can use vector notation instead of that.

Simulations are done with four different methods and three different time step values. Firstly, results of forward Euler method is given below. From these figures, we can see that this method adds extra energy to the system where total energy and angular momentum in the system increases. Which is visible from the extension in the path of the Jupyter around the Sun. Even qualitative behavior doesn’t change with decreasing time step, the amount of the numerically added energy and diversion in the path decreases as time step decreases.



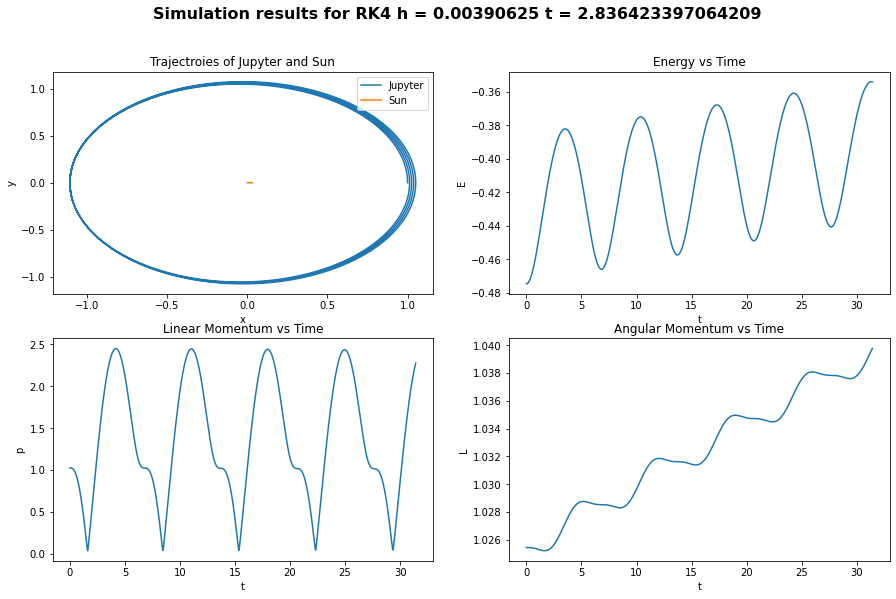
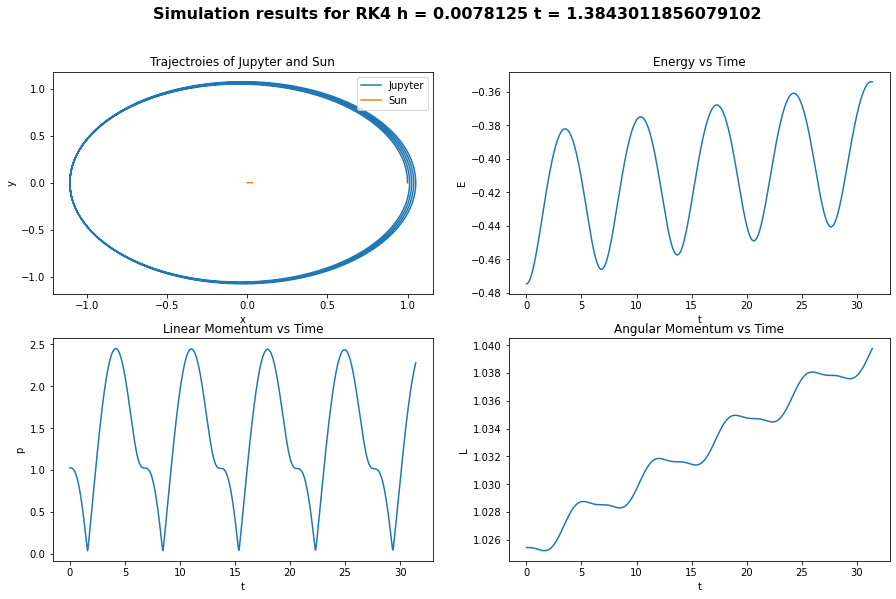
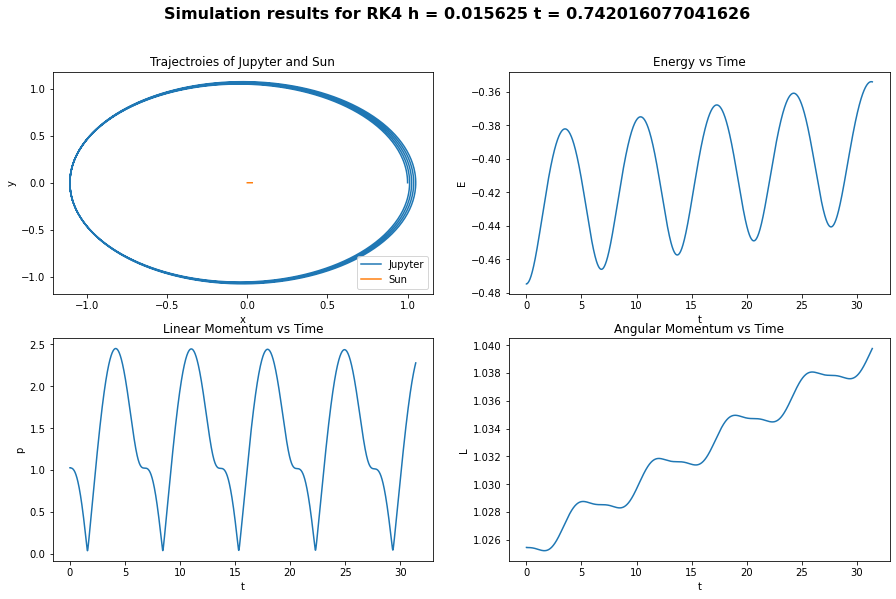
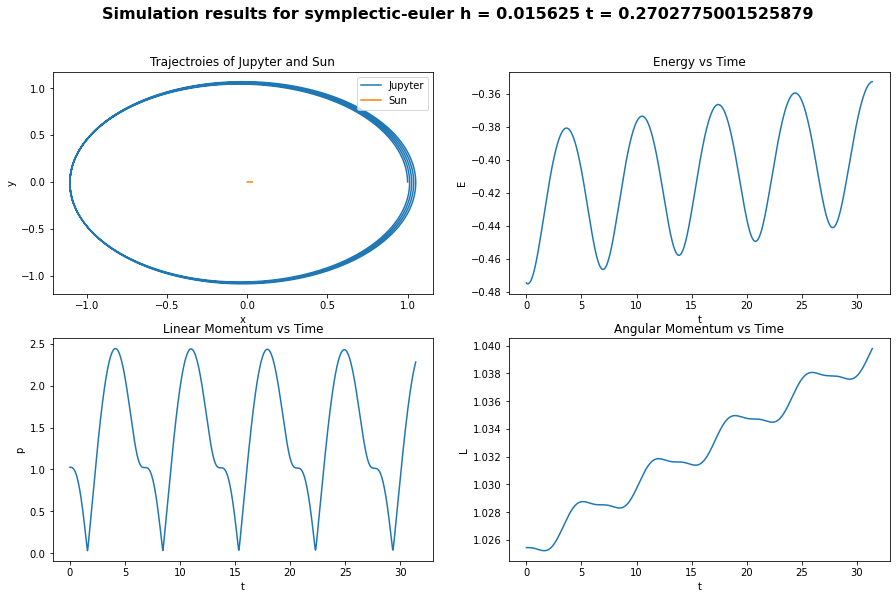
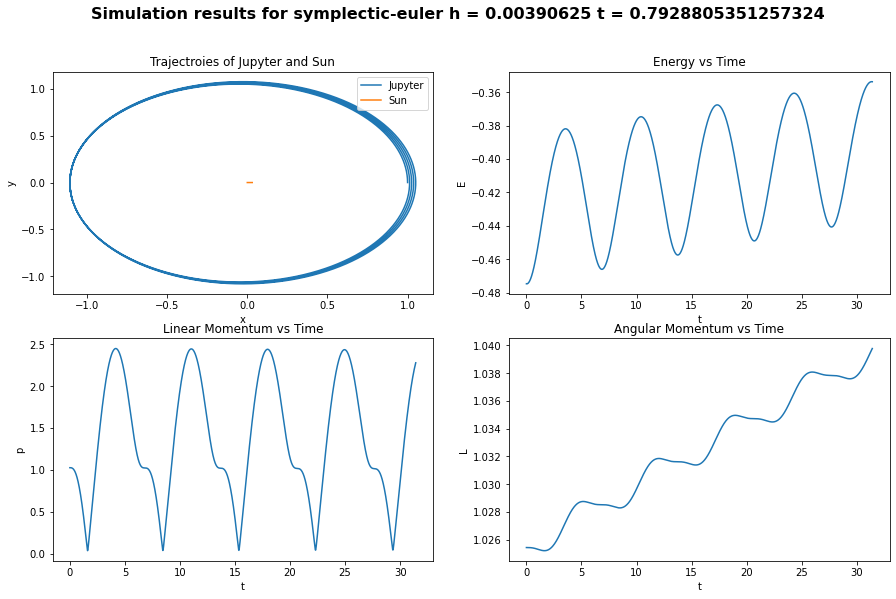
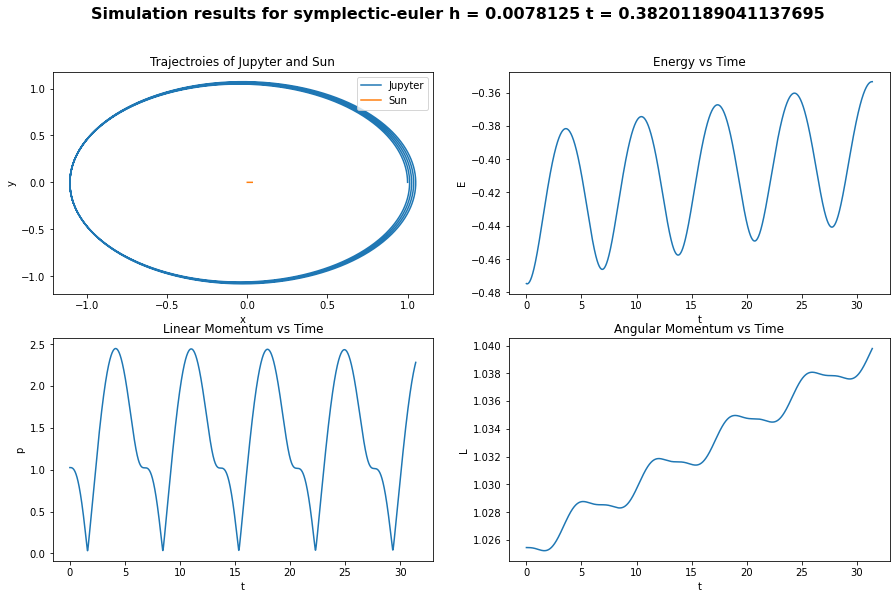
Next method to be plotted and analyzed is implicit Euler.



In implicit Euler we observe the, opposite effect observed in forward Euler method. This time total energy of the system decreases by time and radius of its orbit decreases. Where, eventually Jupyter almost collides with Sun for two of the three different time steps. Again qualitative behavior doesn’t change by time step values, but numerical error decreases as time step gets smaller.

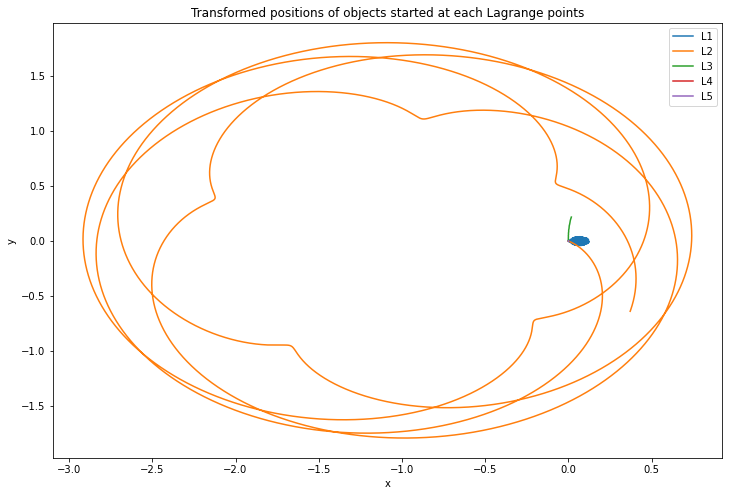
Next method to evaluate is the symplectic Euler whose simulation results are given below. If we look at the how energy changes in symplectic Euler, we can see that even it oscillates, magnitude and mean of the oscillations is the almost same. Therefore, we can say that this method is successful for conserving the total energy in the system. We can also see that from the trajectory of the Jupyter, where radius of its orbit doesn’t change significantly. In addition, changing time stepping also didn’t change much thing for this method.

Final method to evaluate is RK4 whose simulation results are given below. Simulation results of RK4 is also similar to symplectic Euler method. Even it is an explicit method and expected to increase total energy in the system it doesn’t do that significantly.



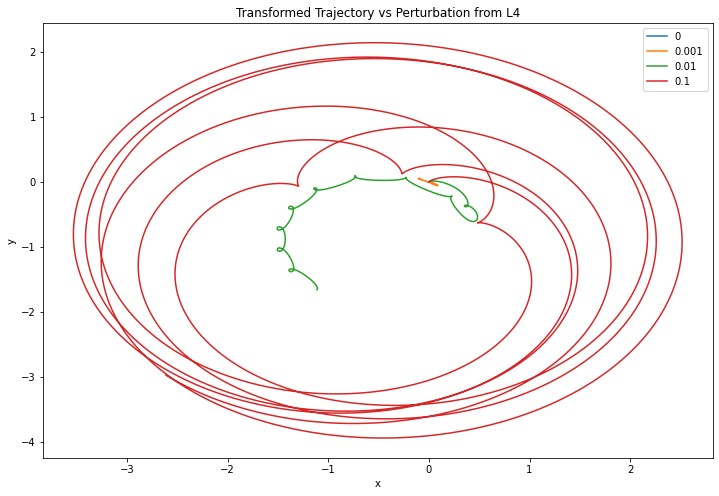
**Part (b):**

In this part, firstly we are asked to find Lagrange points of the Jupyter, Sun system which is assumed to follow circular paths for both of the objects. Lagrange points are calculated with help of the Wikipedia page. After calculating Lagrange points, objects are started to simulation for restricted three body system in those points. Simulation results are given below for different Lagrange points.



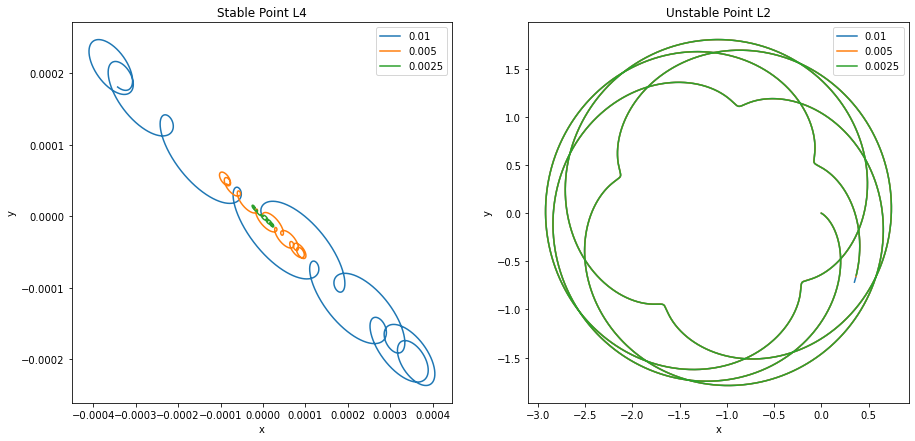
From these plot, we can see that three of the Lagrange points are unstable. Which are L1, L2 and L3. We can see that L2 diverges from the circular trajectory most. L1 and L3 still diverges but their divergence is small compared to L2. For the stable points L4 and L5 diversion from the trajectory is in the numerical error levels, that is the reason we don’t see their paths in this plot.

As a next step, we added perturbations to L4 and re-run simulation for each perturbation. How objects diverged from their path is given below.



**Part (d):**

In this part, we have selected one stable point (L4) and one unstable point (L2), and calculated how their trajectories change by decreasing time stepping. Simulations are conducted repeatedly for different time steps for these two points and results are given below.



For the plot at the left, we can see that diversion from circular path decreases for stable point as time step decreases. Which shows us that this point is physically stable and diversion from the trajectory comes from numerical errors. But for unstable point, diversion from trajectory doesn’t change even if we change the time stepping. Therefore, we can say that this diversion doesn’t come from numerical errors.

**Problem 20:**

**Part (a):**

Let’s start with writing down the initial differential equation and then continue with scaling of the equation.

Now use the scaled variables.

Now we can determine the constants and as following, then obtain desired equation.

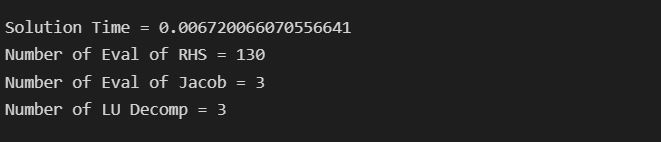
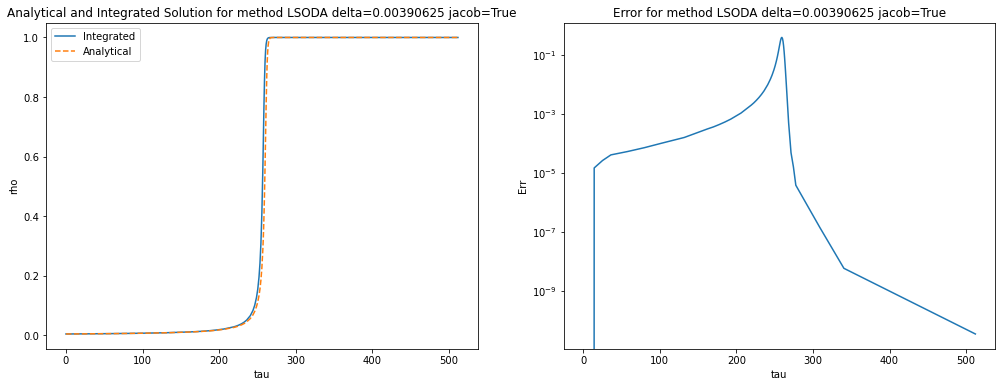
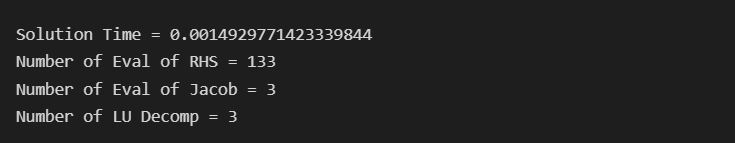
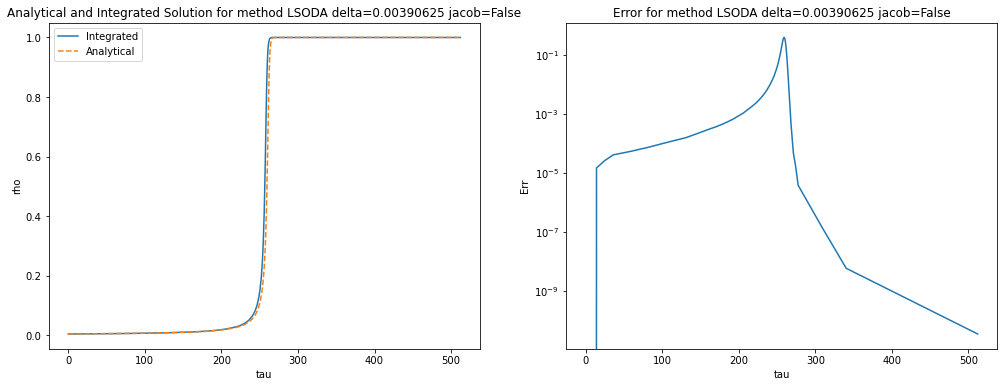
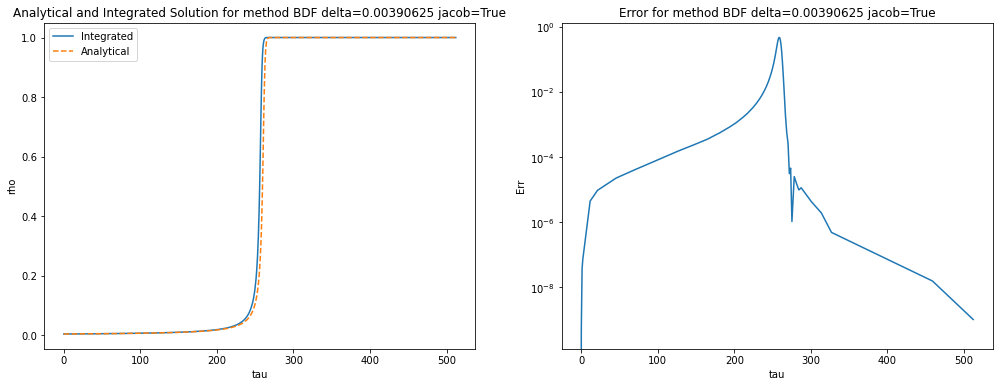
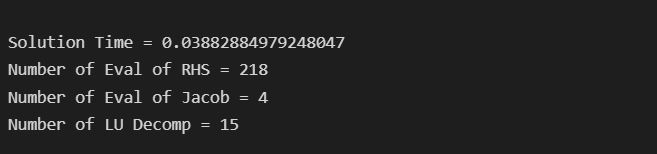
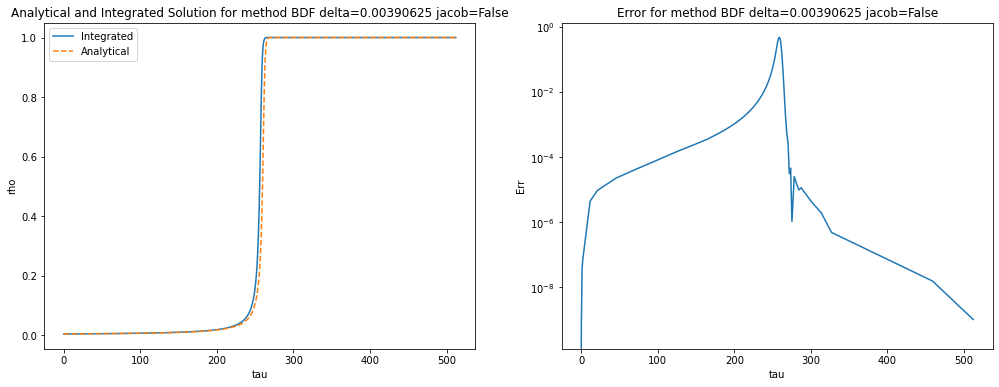
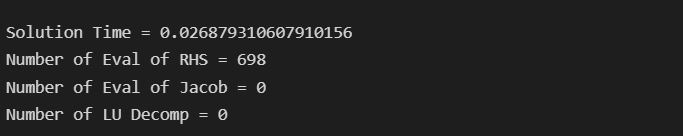
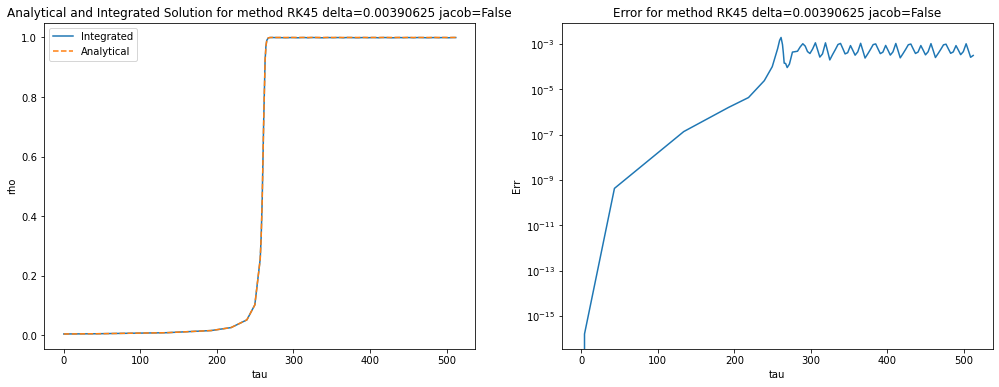
Now test that whether given function is a solution to this equation or not.

Now check whether it is equal or not to right hand side of the ODE.

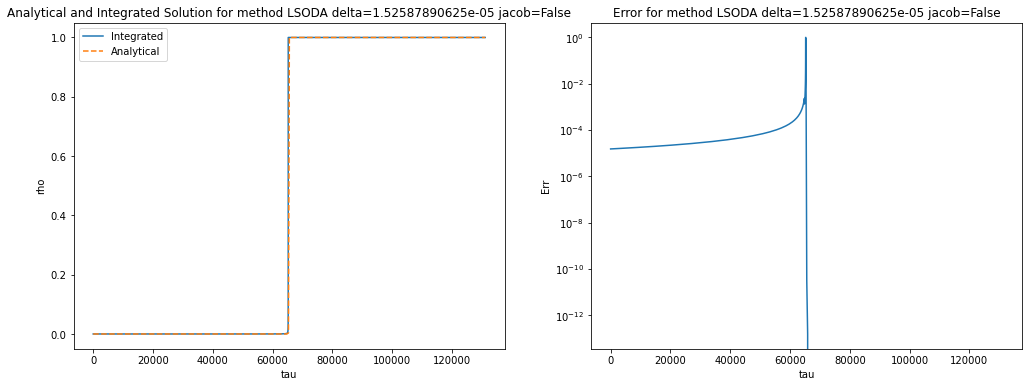
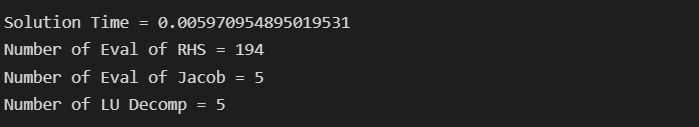
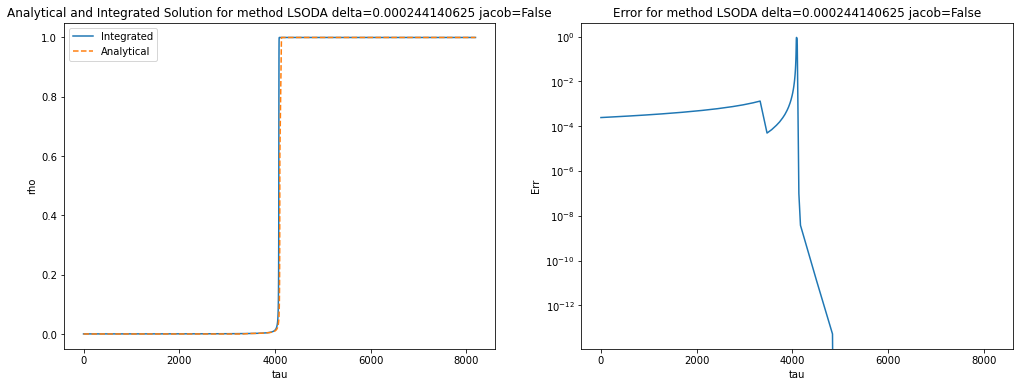
Therefore, given function is the solution of this ODE.

**Part (b):**

In this part we are asked to solve given differential equation with different integrators with built-in function “scipy.integrate.solve\_ivp”. Initial value problem is solved with different kind of integrators and their results are given below for .



From the solutions we can see that RK45 gives the best result compared to analytically known result. Even it has very low error compared to other methods, it takes too much sample points so it iterates slowly and computation time is significantly larger than other methods. For other two implicit methods BDF and LSODA, LSODA works almost ten times faster than BDF with similar error rates. Therefore, our selection for best method is LSODA. With using LSODA, and cases are solved with absolute tolerance parameters set to . Results are given below.



**Part (c):**