Gravitational N-body Problem

by

Karolina Suszek
Andrea Vilcacundo
Graeme Ko

PHYS 349

University of Waterloo

April 23, 2023

Statement of contributions

- Karolina Suszek (20849304)

I mostly worked on animations and the 3 Body Scattering experiment. I also troubleshooted some of the code.

- Andrea Vilcacundo (20756368)

I worked on the general report writing, troubleshooting the Leapfrog method and the Lagrange points study case

- Graeme Ko (20739728)

I did the galaxy collision simulations and animations/troubleshooting for the Lagrange point study.

Introduction

We will be using integration methods for ordinary differential equations to make N-body simulations. These simulations will examine the behavior of N-body systems under the influence of Newtonian gravitational forces.

The magnitude of the Newtonian gravitational force will be given by the equation $|F| = \frac{Gm_i m_j}{r^2}$; where m_i, m_j are the corresponding masses of the particles i, j, G is the gravitational constant, and r is the separation between particles i, j. This force corresponds to an acceleration of $\overrightarrow{a_{ij}} = -\frac{Gm_j\overrightarrow{r_{ji}}}{|\overrightarrow{r_{ji}}|^3}$ where $\overrightarrow{a_{ij}}$ is the acceleration vector of the i^{th} body due to the j^{th} body and $\overrightarrow{r_{ji}}$ is the separation vector from the j^{th} body to the i^{th} body. For a single body i, the net acceleration, due to other j bodies, is therefore given by $\overrightarrow{a_i} = \sum_j^N -\frac{Gm_j\overrightarrow{r_{ji}}}{|\overrightarrow{r_{ji}}|^3}$. This acceleration will be used to update the phase space vectors as we integrate through the simulation.

Force softening

Force softening is considered to accurately represent the forces in the system. This modification to the Newtonian gravity by introducing a softening parameter (ϵ) prevents close encounters between bodies in the simulation by smoothing the function, and it improves code efficiency in numerical integration and fluctuations due to the N being finite [1]. This is important since we aim to generate a simulation with an optimized processing time.

For collisionless simulations, the softening length is introduced in the equations of motion (1) as seen in equation (2) to avoid the singularity from the potential $\frac{1}{r}$ such that it becomes $\frac{1}{\sqrt{r^2+\epsilon^2}}$, similar to the potential for a Plumer sphere [1]. From Newton's Law of Gravitation, the interaction between N-bodies with masses m is given as:

$$\frac{d\overrightarrow{v_i}}{dt} = -G\sum_{j\neq 1}^{N} m_j \frac{\overrightarrow{r_i}-\overrightarrow{r_j}}{|\overrightarrow{r_i}-\overrightarrow{r_j}|^3} \quad (1)$$

$$\frac{d\overrightarrow{v_i}}{dt} = G\sum_{j\neq 1}^{N} m_j \frac{\overrightarrow{r_j}-\overrightarrow{r_i}}{\left(|\overrightarrow{r_j}-\overrightarrow{r_i}|^2+\epsilon^2\right)^{3/2}} \quad (2)$$

For our simulations, considering the softening parameter depends on the distribution of mass and particles on the system, and it is usually set to the average distance between bodies [2, 3], an optimal softening parameter was determined to be 0.1 pc for its later use in the application case of galaxy collisions. In the Earth-Sun test case, the softening parameter is set to 0.1 as this is a collisionless two-body problem and does not require the force softening.

Description and comparison of algorithms

The algorithms implemented include four numerical methods to solve ordinary differential equations and a pre-built ODE solver to access the accuracy of the results. A final version of the code is presented in *Appendix A* and some secondary versions of the code are included in *Appendix B*. For the methods described below, we wrote a function that calculates one step and the 'odeint' function is included to compare the results.

Euler Method

The Euler method is simple to implement, however, it is only first order and does not conserve energy. It can be derived by keeping the first two terms of a Taylor expansion around some time, t.

$$x(t+h) = x(t) + hx'(t) \tag{3}$$

RK2

The second order Runge-Kutta method can be thought of as an improved version of the Euler method. The error is improved by incorporating a Euler half-step, thereby also calculating the derivative in the middle of each time step. The algorithm works as follows:

$$k_1 = hf(x,t) \tag{4}$$

$$k_2 = hf(x + k_1/2, t + h/2) \tag{5}$$

$$x(t+h) = x(t) + k_2 \tag{6}$$

RK4

RK4 is the fourth order Runge-Kutta method. It has the highest order out of all the algorithms tested in this experiment. It works as follows:

$$k_1 = hf(x,t) \tag{7}$$

$$k_2 = hf(x + k_1/2, t + h/2) \tag{8}$$

$$k_3 = hf(x + k_2/2, t + h/2) \tag{9}$$

$$k_4 = hf(x + k_3, t + h) \tag{10}$$

$$x(t+h) = x(t) + (k_1 + 2k_2 + 2k_3 + k_4)/6 \quad (11)$$

Leapfrog method

This method is implemented to conserve energy as it is reversible in time and symmetric. It is implemented as follows:

$$x\left(t + \tfrac{1}{2}h\right) = x(t) + \tfrac{1}{2}hf(x, t + \tfrac{1}{2}h) \quad (13)$$

$$x(t + \tfrac{3}{2}h) = x(t) + hf(x\left(t + \tfrac{1}{2}h\right), t + h) \quad (14)$$

This method works for problems where the acceleration depends only on the position, not the velocity. Gravitation is an example of this. A major advantage of the leapfrog method is that it is time reversible and conserves energy in the long term, making it a good choice for problems involving periodic motion or simulations that run for a long time. The time reversibility can be demonstrated by plugging in $-h$ into the algorithm, showing that the steps are the same both forwards and backwards in time.

ODEint

The built-in function ‘odeint’ from scipy.integrate was used to test the accuracy of results since this function is expected to generate better results than the RK4 method and similar to the Leapfrog method.

Analysis and application to specific test cases

Our test case was the Earth's orbit around the sun, with the following initial conditions:

- The sun starts at rest.
- The Earth starts at $x = 1AU = 1.5 \times 10^{11} m$ with an initial velocity $v_y = 29784.8\ \mathrm{m/s}$, the known average velocity of the Earth [4].

We ran the simulation for a time period of 3.154×10^7 seconds (equivalent to 1 year), with a step size of 86000 (equivalent to one Earth day). The analytic solution to this scenario was that the Earth would complete exactly one circular orbit in one year.

The Euler method solution was not consistent with this prediction (Fig. 1). Increasing the time period (to five years, for example) showed that the Earth would drift away from the sun in a spiral trajectory for long periods of time (Fig. 2), indicating that the Euler method would not be reliable for long N-body simulations.

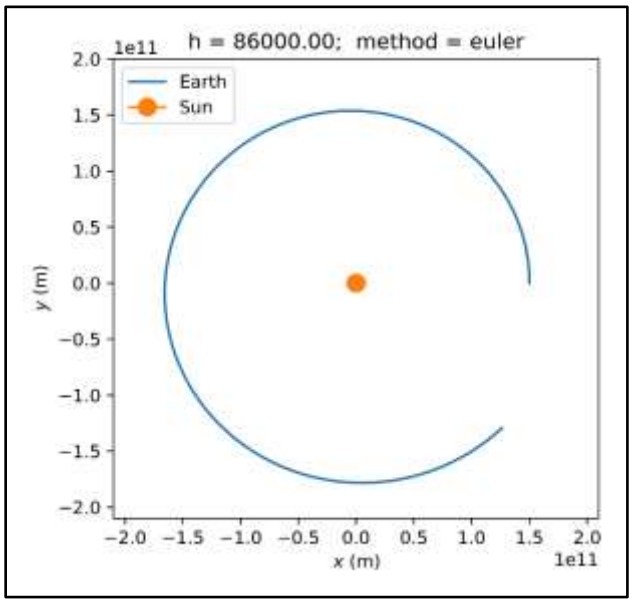

Figure 1. Earth-Sun test case using the Euler method for 1 year.

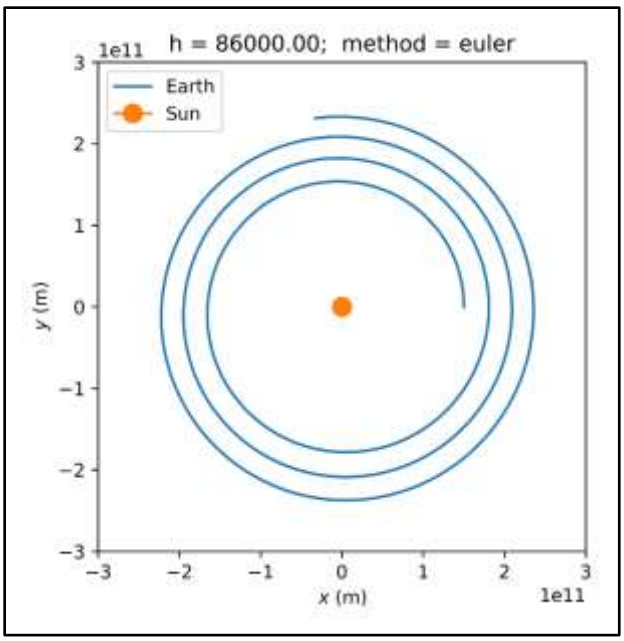

Figure 2. Earth-Sun test case using the Euler method for 5 years, demonstrating that the Earth drifts away.

Meanwhile, the 2nd and 4th order Runge-Kutta methods did result in circular orbits around the sun (Fig. 3 & 4). When the time period was increased to 10 years (not pictured), these methods continued to give the expected results.

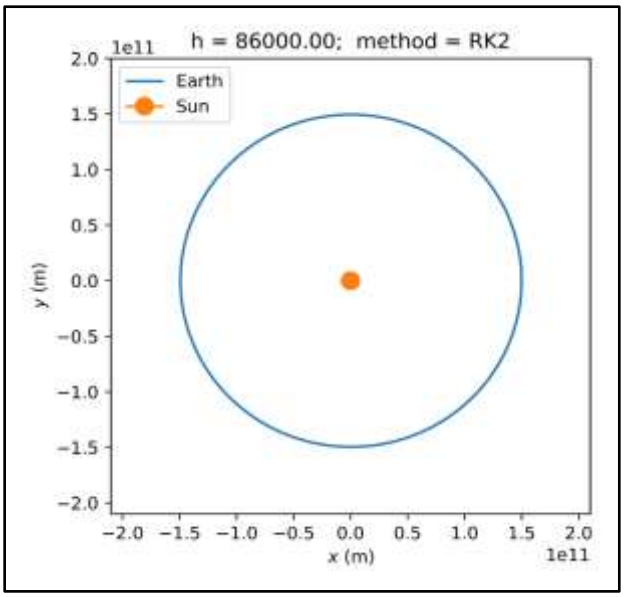

Figure 3. Earth-Sun test case using the RK2 method for 1 year.

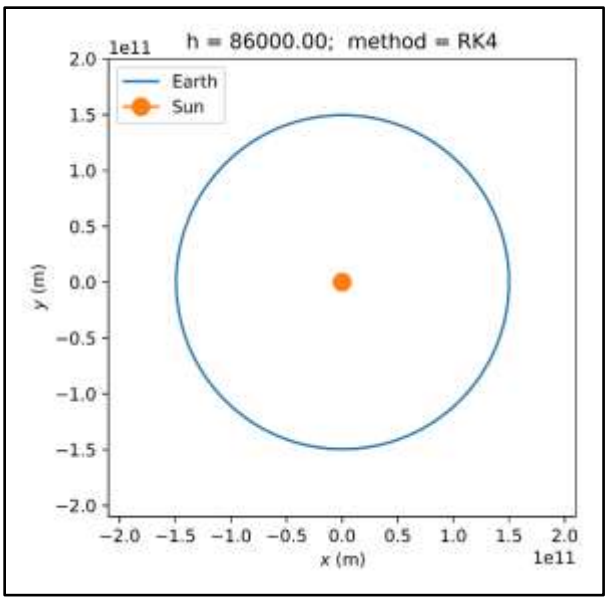

Figure 4. Earth-Sun test case using the RK4 method for 1 year.

For the leapfrog method, as seeing in Fig.5, it shows the conservation of energy for the period run. Although it looks similar to the plot obtained with RK4, this method will provide more accurate results when working with more bodies in the system in which conservation of energy is fundamental.

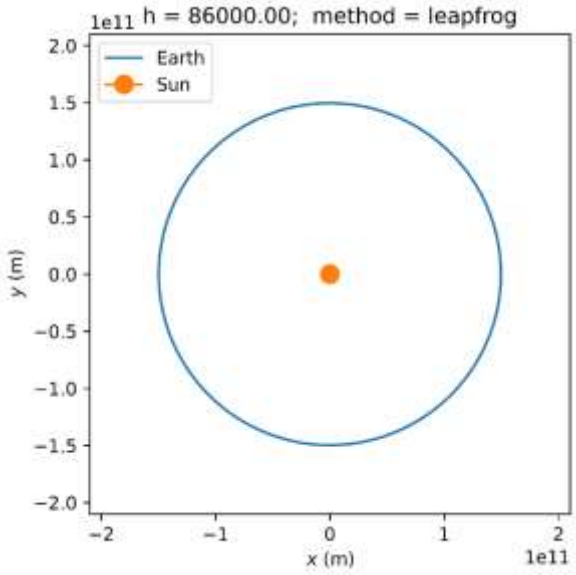

Figure 5. Earth-Sun test case using the leapfrog method for 1 year.

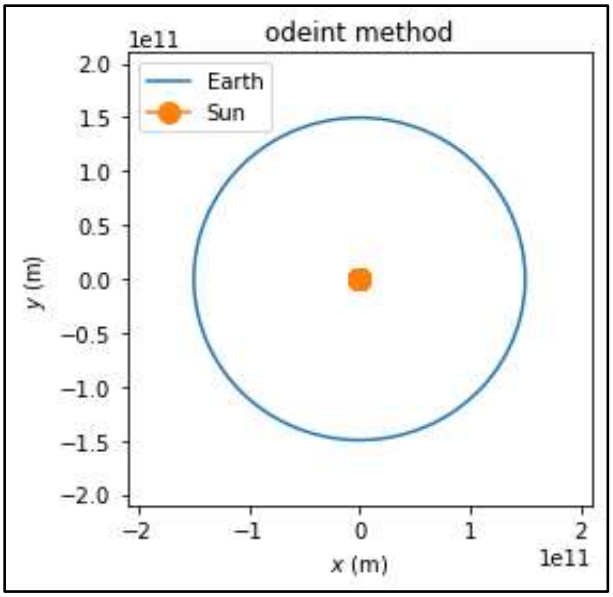

Figure 6. Earth-Sun test case using the ODEint method for 1 year.

Therefore, the preferred method is Leapfrog followed by RK4 and ODEint. These can be implemented in the study cases below.

Studies

This section includes a description of applicable cases in which the core code has been tested including figures and animations.

Galaxy Collisions

For the galaxy collision case study, two galaxies of interest were simulated. One galaxy which we will refer to as the 'main' galaxy has a mass of $10^{11}M_{\odot}$ and starts at rest in the simulation reference frame. The other galaxy referred to as the 'perturber' has a mass of $10^{10}M_{\odot}$ and will start at a distance of approximately 100kpc away from the main galaxy. Each galaxy was modelled as having a ring of stars in circular orbits at radial distances of 5kpc, 10kpc, and 15kpc from the galactic centers. The position of each star is generated randomly. To achieve a stable circular orbit, the speed of each star is calculated using $v = \frac{\sqrt{GM}}{r}$ where G is the gravitational constant, M is the mass of the galaxy, and r is the orbital radius of the star. To mimic a galaxy, all stars were chosen to rotate with the galaxy in the same direction, either clockwise or counter-clockwise in the x-y plane. To accomplish this, initial conditions were set

such that the cartesian components of the velocity vector would orient where the velocity is perpendicular to the radial vector. Then, based on what section of the orbit the star is initialized in, the signs of the velocity components were changed to reflect the desired direction of the galactic spin.

For the galactic collision simulation, the perturber galaxy was offset slightly from the main galaxy such that it achieved a parabolic orbit with a close approach radius of r_b. The speed necessary to achieve a parabolic orbit was calculated from the initial position and mass conditions such that the orbital energy is equal to 0. There were four test-cases of varying r_b values to test the results of different impact parameter values, and each of those test-cases different galactic spin configurations were also simulated, yielding a total of 16 collision scenarios. The r value displayed in the animations depict the total distance between the galactic centers. A gravitational softening parameter of 0.1kpc was used for these collisions.

All simulations clearly showed that the main galaxy retained the orbits of its stars better than the perturber galaxy. The lower mass perturber did not have as much gravitational potential to offset the effects of the higher mass galaxy and this can be seen in the animations as many of the stars are sent flying away from the galaxies. Some stars that were originally part of the perturber galaxy are even caught into stable orbit by the main galaxy at close approaches. At close approaches, distortion can be seen in the main galaxy as well, though not as severe. The lower radius orbits of the main galaxy are especially better at retaining shape compared to the outer orbits. The animations show that for both the main and perturber galaxy (Figures 7,8), there is a greater deformation effect when the galaxy is spinning with prograde motion where the direction of the stellar angular momentum is in the same direction as the angular momentum of the orbit. This finding was particularly apparent for larger distance approaches such as for $r_b = 60kpc$, where the prograde perturber was highly deformed compared to the retrograde perturber.

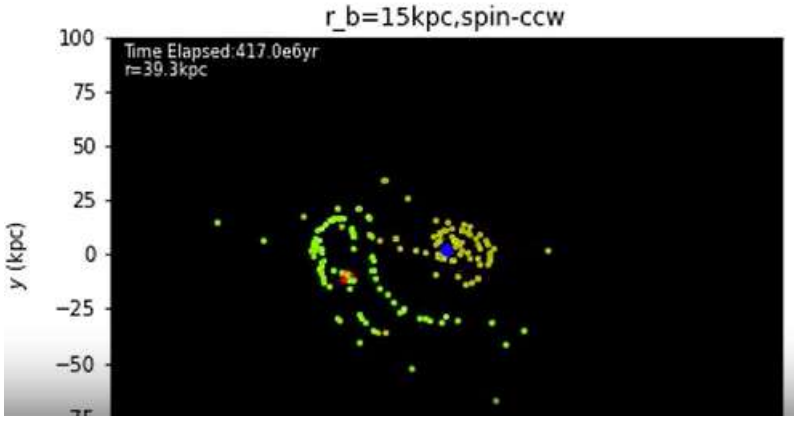

Figure 7. Collision of both galaxies with prograde motion (perturber galactic center in red with green stars, main galactic center in blue with yellow stars)

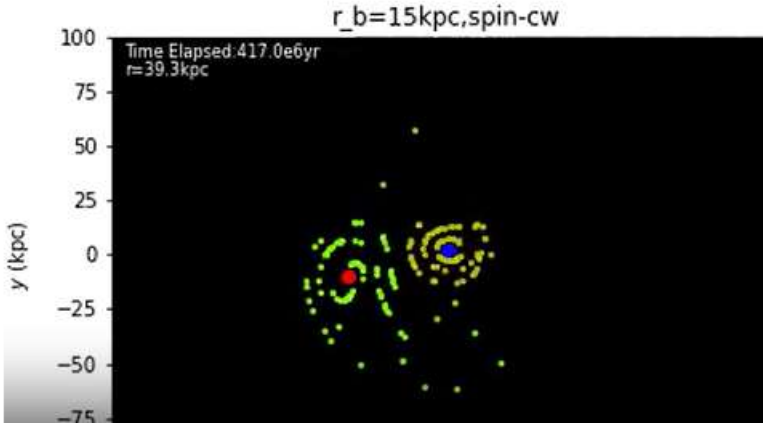

Figure 8. Collision of both galaxies with retrograde motion (perturber galactic center in red with green stars, main galactic center in blue with yellow stars)

Lagrange Points

In this study case, a 3-body system composed of sun, Earth and Moon is created to describe the Lagrange points L_2 and L_4 in the orbit. The two primary masses M_1 and M_2 are the sun and Earth' masses, respectively. The Moon's mass is considered negligible in comparison to them. The position of the center of mass ($\overrightarrow{r_{CM}}$) is the reference frame in the Earth-sun system. Let $\overrightarrow{r_1}$ be the r_{CM}-Sun distance, $\overrightarrow{r_2}$ the r_{CM}-Earth distance, and r_{12} Earth-Sun distance [5]. Also, r_{CM}-Moon distance is $\vec{r}$.

By definition of the center of mass,

$$r_{CM} = \frac{M_1 r_1 + M_2 r_2}{M_1 + M_2} = 0 \tag{15}$$

$$M_1 r_1 + M_2 r_2 = 0 \tag{16}$$

$$r_2 = r_1 + M_2 r_{12} \tag{17}$$

$$r_2 = \frac{M_1}{M_1 + M_2} r_{12} \quad ; r_1 = -\frac{M_2}{M_1 + M_2} r_{12} \tag{18}$$

Considering a dimensionless ratio and mass of the larger M_1 and smaller body M_2 and let $\mu = \frac{M_2}{M_1 + M_2}$, and $\mu_2 = \frac{M_1}{M_1 + M_2}$ then,

$$\frac{r_1}{r_{12}} = -\mu; \frac{r_2}{r_{12}} = \mu_2 \tag{19}$$

The equations of motion are given by [6]:

$$\ddot{x} - 2\dot{y} = x - \frac{dV}{dx} = \frac{\partial \Omega}{\partial x} \tag{20}$$

$$\ddot{y} + 2\dot{x} = y - \frac{dV}{dy} = \frac{\partial \Omega}{\partial y} \tag{21}$$

Ω: pseudo-potential

$$\Omega = \frac{1}{2}(x^2 + y^2) + \frac{(1-\mu)}{r_1} + \frac{\mu}{r_2} \tag{22}$$

Considerations:

At the Lagrange points, the velocity and acceleration are zero; $\frac{\partial \Omega}{\partial x} = \frac{\partial \Omega}{\partial y} = 0$

The pseudo-potential equation can be solved to find the Lagrange points; collinear points (L_1, L_2, L_3) and triangular points (L_4, L_5)[7].

For the collinear points L_1, L_2, L_3 lying on the x axis,

$$\frac{\partial \Omega}{\partial x} = x - \frac{(1-\mu)(x+\mu)}{\left(\sqrt{(x+\mu)^2}\right)^3} - \frac{\mu(x-1+\mu)}{\left(\sqrt{(x-1+\mu)^2}\right)^3} \tag{23}$$

For L_1: $\frac{\partial \Omega}{\partial x} = -x - \frac{(1-\mu)}{(x+\mu)^2} + \frac{\mu}{(x-1+\mu)^2}$

For L_2: $\frac{\partial \Omega}{\partial x} = x - \frac{(1-\mu)}{(x+\mu)^2} - \frac{\mu}{(x-1+\mu)^2}$

For L_3: $\frac{\partial \Omega}{\partial x} = x + \frac{(1-\mu)}{(x+\mu)^2} - \frac{\mu}{(x-1+\mu)^2}$

The points L_4, L_5 have analytical solutions

$$L_4: \left(\frac{1}{2} - \mu, \sqrt{3}/2\right)$$

$$L_5: \left(\frac{1}{2} - \mu, -\sqrt{3}/2\right)$$

In the algorithm for this section, the collinear points are determined using scipy. optimize.newton(l_i,1); where l_i is the function for the ith-Lagrange point. An alternative version of the code sets the equations of motion at which $\frac{\partial \Omega}{\partial x} = \frac{\partial \Omega}{\partial y} = 0$ and solves it using Odeint. This version included in *Appendix B* was not adopted since it was not effective.

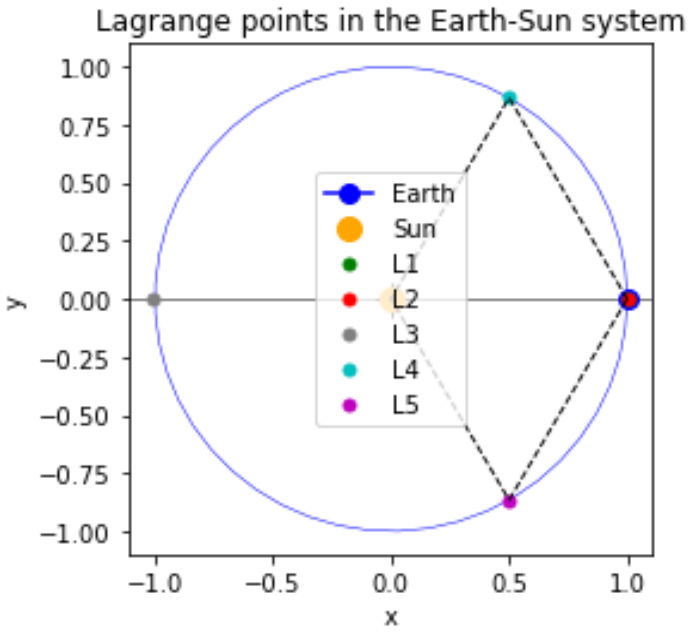

Figure 9. Plot the orbit in a reference frame that is rotating with the Earth's orbit around the center of mass of the Earth-Sun system.

- Stability analysis

From the animation "Lagrange points", it is observed that the position of the Lagrange points follows the trajectory of the Earth and the initial geometry is maintained along the path. It is also seen from the figure above that L_2 is a collinear point and L_4 is one of the points forming the apex of the triangle. According to the theory, L_2 is an unstable point and L_4 is a stable point since for this system, the ratio of the Earth-Sun mass is greater than 24.96 [8]. In order to determine the stability if these points, a small change in momentum in the satellite (p) is determined and shown in the animation "Lagrange points_offsetmomentum." A 20% offset in the satellite velocity is introduced and it is seen that L_2 is more unstable than L_4 at the beginning; however, as time progresses, both points are not stable and move away of the Earth-Sun system. Theoretically, if the small satellite gets a small change in momentum, the resulting momentum will be given as $p_2 = p_1 + F_{12}\Delta t$ where F_{12} is the force between the point and the center of mass.

3-body scattering

The behavior of 3 body scattering was investigated. The initial binary star system consisted of two stars ("the orange star" and "the yellow star"), each with 1-solar mass, 2 AU apart, in a circular orbit around their center of mass (i.e., the origin). Their initial velocities were calculated using Equation (24), where M is the reduced mass and r is the distance between the two stars [9].

$$v = \sqrt{\frac{GM}{r}} \tag{24}$$

Then, the third star ("the blue star"), which was a 2-solar mass star, approached from the right. Its initial vertical velocity was always set to $v_y = 0$ m/s, while its horizontal velocity varied from $v_x = -1$ km/s to $v_x = -100$ km/s. Its initial position was also varied with respect to the binary stars' orbit.

On the plots, the colours and sizes of the stars were arbitrary; they were chosen just to make the stars visually distinct.

The differential equations for the stars' positions were solved using the 4th order Runge-Kutta method, which was selected due to it correctly solving the validation test (i.e., the Earth's orbit around the sun) and it being the highest order method of those that were looked at. Based on the validation test, the 2nd order Runge-Kutta method could have also been used. The efficiency of each method should also be considered, as RK2 is theoretically faster than RK4 due to calculating fewer steps. However, for these purposes, efficiency was not a serious issue as either way the differential equations took less than a second to solve.

For each frame of the animations, the kinetic, potential, and total energies were calculated (using Equation (24), and using the convention that gravitational potential energy is 0 at $r = \infty$ and negative when r is finite.)

In general, there were two types of behaviours that were commonly observed. These behaviours were found using trial and error.

With some initial conditions, the binary stars were knocked out of orbit with each other; this was often due to sharply colliding with the blue star, but sometimes it was due to the blue star's gravity disturbing their trajectories and creating an unstable orbit. Then, all three stars would drift away in different directions. One example of this is when the blue star starts at a position of $x = 40.0$ AU and $y = 0.0$ AU, with an initial horizontal velocity of $v_x = -100$ km/s (Fig. 10, "ThreeBodyScattering - Fast.mp4"). When it reaches the binary system, it travels between the two stars, its gravitational force shifting their trajectories, causing them to escape from their stable orbit. Keeping all the variables the same but varying the blue star's initial vertical position to $y = 0.1,\ 0.5,\ 1.0$ AU created similar situations where the three stars drift off in different directions. However, their final speeds vary. For example, when $y = 1.0$ AU, a sharp collision can be seen between the blue star and the yellow star, resulting in the yellow star moving very fast off screen. Similar behaviours were observed when the blue star started at slow speeds. One example is when it had an initial position of $x = 15.0$ AU and $y = 0.0$ AU, and an initial horizontal speed of $v_x = -1$ km/s (Fig. 1110, "ThreeBodyScattering - Slow.mp4"). As it approached from off screen, the binary star system could be seen drifting towards it. Like before, the blue star disturbed the binary orbit, and all three stars moved away in different directions.

In these situations, the total kinetic energy of the stars appears to be converging as time goes on. This is because the stars approach constant speeds; as they drift apart, their gravitational forces become weaker to the point of being negligible, and they stop accelerating. Meanwhile, the gravitational potential energy approaches zero, due to the stars drifting farther and farther apart.

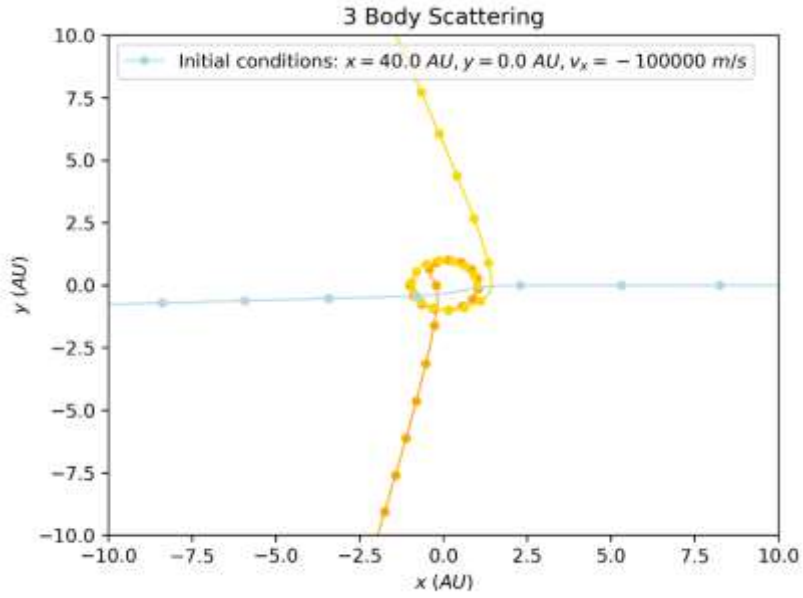

Figure 10. The trajectories of the three stars when the blue star approaches with a large velocity Note that the blue star begins off screen on the right, the orange star begins at $x = -1.0$ AU, and the yellow star begins that $x = +1.0$ AU

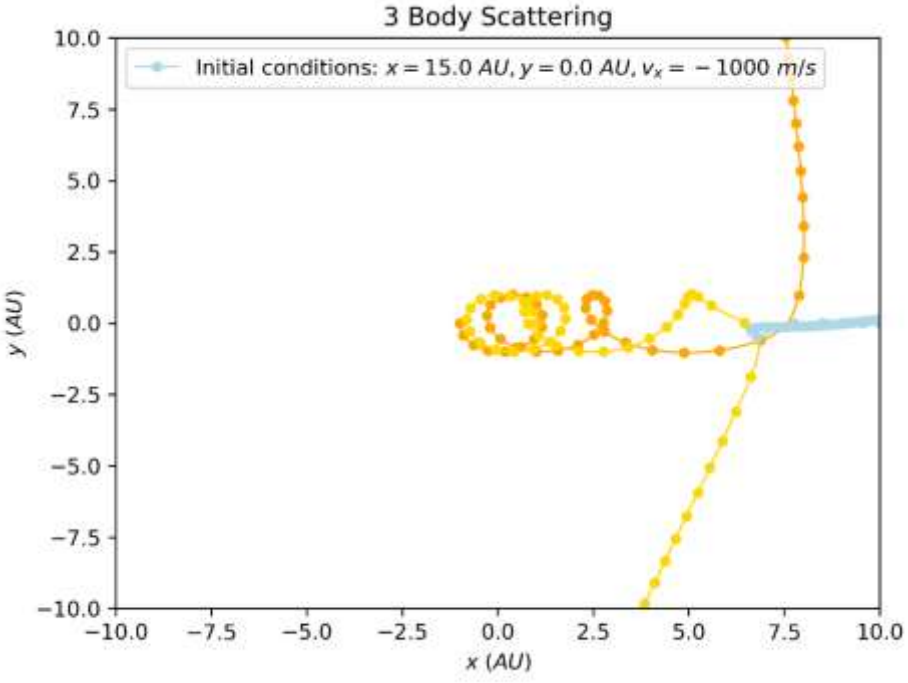

Figure 11. The trajectories of the three stars when the blue star approaches with a small velocity

Meanwhile, some initial conditions resulted in one star being knocked away while the remaining two formed their own new binary star system. Initial conditions that caused this were generally rarer and more difficult to find through trial and error. This is because appropriate speeds are needed such that gravitational bodies form stable (circular or elliptical) orbits. If the stars move too fast, they will experience parabolic or hyperbolic orbits and escape. If they are too slow, then the gravitational forces will cause the stars to collide.

One set of initial conditions that was investigated was $x = 15.0$ AU and $y = 0.45$ AU, and an initial horizontal speed of $v_x = -20$ km/s (Fig. 1211, "ThreeBodyScattering - NewBinarySystem.mp4"). In this case, the orange star escapes, while the blue and yellow stars become a new binary system. Slightly changing the blue star's initial horizontal velocity by 1-2 m/s, without changing anything else, resulted in different behaviour. For $v_x = -20.001$ km/s, the blue and yellow stars briefly become a new binary system, however, the yellow star is visibly drifting into the blue star, before it finally collides with it and all three stars drift away from each other. A similar behaviour is seen when $v_x = -19.998$ km/s. Another interesting set of initial conditions for the blue star is $x = 15.0$ AU, $y = 1.0$ y = 1.0 AU, and $v_x = -20$ km/s (Fig. 1311, "ThreeBodyScattering - WeirdCollisions.mp4"). Here, the blue and orange stars form their own binary system. Then the yellow star, rather than moving away from the new binary system as it would in similar cases, moves towards it. It then collides with the binary system, much like the blue star did in the earlier cases, and all three stars move in different directions.

During situations where the stars formed new binary systems, the energies appeared to oscillate. For example, when the blue star found itself in a stable orbit with the yellow star in Figure 12, their orbit was more elliptical than circular; the stars did not orbit with constant speed. The two stars would speed up when they got closer together and slow down when they were farther apart. This was reflected in the kinetic energy reading, which would spike sharply when the stars got closer together.

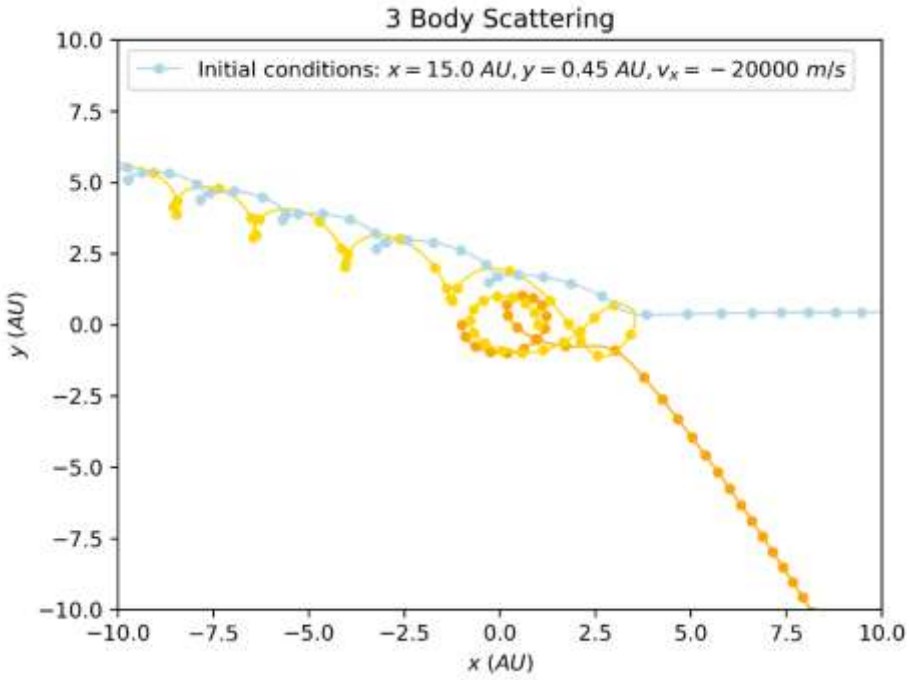

Figure 12. The trajectories of the three stars when the blue star and the yellow star form a new binary system while the orange star escapes.

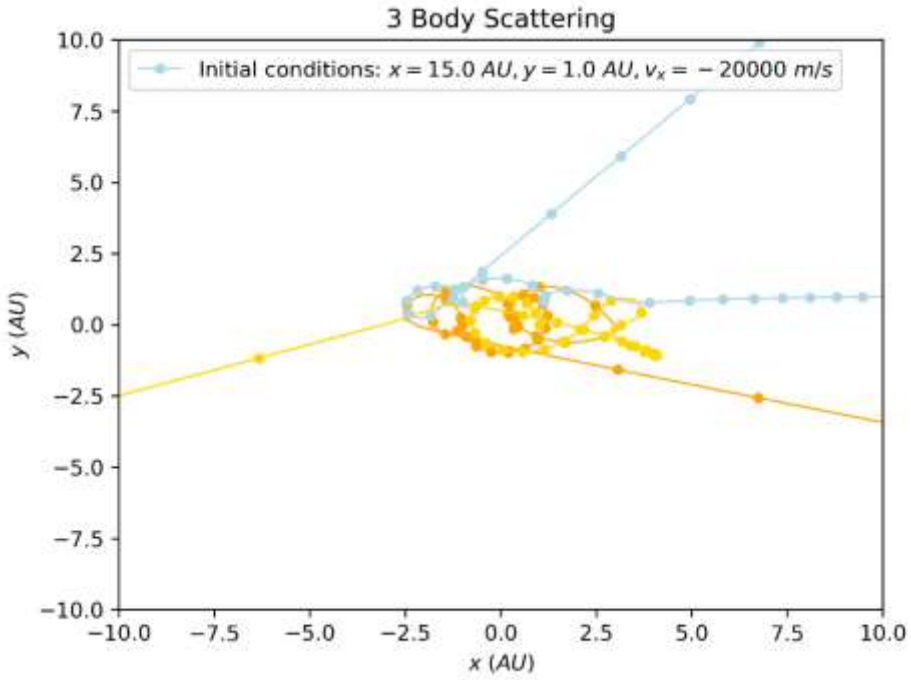

Figure 13. The trajectories of the three stars when the blue star briefly forms a new binary system with the orange star, the yellow star collides with them, and everyone flies off in a different direction.

Discussion

In this work, we compared four different numerical integration methods: Euler, RK2, RK4, and leapfrog. When the Earth's orbit around the sun was used as a validation test, it was found that the Euler method was significantly more inaccurate than the other methods, especially at long periods of time. The theoretical solution to the Earth-sun system was the Earth making 1 circular orbit around the sun in 1 year. However, the Euler plot result showed the Earth slowly drifting away from the Sun, making larger orbits each following year. Aside from this, the other methods that were investigated did yield the expected result. The only differences between them would be the order of error – RK2 and leapfrog are second order methods, while RK4 is fourth order – and time reversibility, since a particular advantage to the leapfrog method is that it is time reversible.

One of the main issues of the gravitational collision simulation was the extensive time needed to generate the results and animations. Perhaps optimizations such as compiling with the JIT module or reducing unnecessary 'for loops' should be investigated in future studies with these simulations. Furthermore, the quality of the animations should be improved. Unfortunately, the python animation savers that were initially investigated did not work properly, and therefore we chose a lower quality animation output format.

In the Lagrange points case, it is noticed that obtaining the equations to solve for each point and using optimize.newton allowed to determine the locations of the collinear points however, the optimize.newton method is operationally less effective if the initial guess is far off of the actual value since it have to run a larger number of iterations. An advantage of using this method was the ease in the implementation of the non-dimensional equations of motion, which improves time efficiency since the units can be recovered at the end allowing less iterations. One potential improvement to the animations of the code is to include interconnecting lines between the three mases of the system which could emulate better the constant geometry maintained as the Sun and Earth rotate around the center of mass. It is also noticed that the numerical integration was accurate for showing the Earth's orbit; however, since the sun is very close to the barycenter, its trajectory was a fairly distorted as the time progressed.

For the 3 Body Scattering simulation, two general patterns of behaviour were examined. Sometimes, introducing the third star caused the binary star system to fall apart, resulting in all three stars moving in different directions, their relative distances approaching infinity. Meanwhile, some initial conditions resulted in a new binary system forming, consisting of the third star and one of the two original binary stars. The most significant factor in the stars' behaviour was the third star's initial velocity.

One possible improvement to the 3 Body Scattering simulation would be to have the energies plotted as a function of time in a separate subplot rather than having the values written down on each frame of the animation. This would make it easier to identify patterns in the energies. For example, when the gravitational potential energy approached zero as the three stars drifted apart, it would have been helpful to watch a plot of this energy rather than watch the number decrease. Similarly, when the blue and yellow stars formed a new elliptical orbit and experienced oscillations in their kinetic energies, it would have been interesting to watch the kinetic energies spike up on a plot.

In the Lagrange points case, it is noticed that obtaining the equations to solve for each point and using optimize.newton allowed to determine the locations of the collinear points however, the optimize.newton method is operationally less effective if the initial guess is far off of the actual value since it have to run a larger number of iterations. An advantage of using this method was the ease in the implementation of the non-dimensional equations of motion, which improves time efficiency since the units can be recovered at the end allowing less iterations. One potential improvement to the animations of the code is to include interconnecting lines between the three masses of the system which could emulate better the constant geometry maintained as the Sun and Earth rotate around the center of mass. It is also noticed that the numerical integration was accurate for showing the Earth's orbit; however, since the sun is very close to the barycenter, its trajectory was fairly distorted as the time progressed.

Conclusion

This project presents an approach to solve the Gravitational N-body problem by using numerical integration methods and investigating its validity thought three study cases: Galaxy collisions, Langrange points, and 3-Body Scattering. We showed that using the Euler method for numeric integration conserved energy poorly in the context of circular orbital trajectories and the leapfrog and RK4 methods were considered for the simulations. In the galaxy collisions study, it was found that galaxies colliding with prograde motion have a larger amount of deformation than those with retrograde motion with respect to the angular momentum of the orbit. Furthermore, some stars lost their orbit entirely or were caught into orbit by the other galaxy it was colliding with. The stars at larger orbits from their galactic centers were pulled off more easily, and the stars belonging to the higher mass galaxy showed less overall deformation. In the Lagrange points case, it was seen that there are three collinear points (L_1, L_2, L_3) formed along the axis of the main masses and two triangular points (L_4, L_5). It was concluded that L_2 is unstable point since and L_4 stable point since it the apex vertex of the triangle.

For the 3 Body Scattering problem, it was found that the initial velocity of the incoming star significantly impacted the behavior of the system. When its initial velocity was very large or very small, the three stars would be ejected in different directions. Meanwhile, at initial velocities of the right magnitude (assuming the star's initial position also allowed it to approach at appropriate angles), the stars were sometimes able to form a new binary system, with one of the original binary stars escaping. This demonstrated the concept of stable and unstable orbits. The gravitational potential and kinetic energies were also calculated and examined over the course of the stars' interactions.

Overall, gravitational N-body simulations is useful to solve complex systems by means of numerical integration methods. This project allowed as to explore accurate methods and considerations and determine future improvements in time efficiency and accuracy.

References

[1] Barnes, J. E. (2012). Gravitational softening as a smoothing operation. *Mon. Not. R. Astron. Soc*.

[2] Athanassoula, E., Fady, E., Lambert, J. C., & Bosma, A. (2000). Optimal softening for force calculations in collisionless N-body simulations. *Monthly Notices of the Royal Astronomical Society*, *314*(3), 475–488. https://doi.org/10.1046/J.1365-8711.2000.03316.X/2/M_314-3-475-EQ027.JPEG

[3] Softening and discreteness noise. (n.d.). Retrieved March 22, 2023, from http://www.cs.toronto.edu/~wayne/research/thesis/depth/node3.html

[4] Gregersen, E. What is Earth's Velocity? Retrieved from https://www.britannica.com/story/what-is-earths-velocity#:~:text=Earth%20orbits%20the%20Sun%20every,or%20107%2C225.3%20kilometers%20per%20hour.

[5] Circular Restricted Three-Body Problem — Orbital Mechanics & Astrodynamics. (n.d.). Retrieved April 22, 2023, from https://orbital-mechanics.space/the-n-body-problem/circular-restricted-three-body-problem.html

[6] B Almeida Prado Member, A. F. (n.d.). Orbital Maneuvers Between the Lagrangian Points and the Orbital Maneuvers Between the Lagrangian Points and the Primaries in the Earth-Sun System. XXVIII(2), 131.

[7] Lagrange Points Derivations | Orbital Mechanics with Python 55 - YouTube. (n.d.). Retrieved April 22, 2023, from https://www.youtube.com/watch?v=bpnEa7d3KBQ

[8] What is a Lagrange Point? | NASA Solar System Exploration. (n.d.). Retrieved April 22, 2023, from https://solarsystem.nasa.gov/resources/754/what-is-a-lagrange-point/

[9] Circular orbit. (2023). *Wikipedia.* https://en.wikipedia.org/wiki/Circular_orbit

Appendix B: Secondary versions of the code

```
### 3 body scattering ###
%matplotlib notebook
m1 = Msun
m2 = Msun
m3 = 2.0*Msun
M = m1*m2/(m1+m2)
r = 2*AU
v = np.sqrt(G*M/r)
years = 10.0
totaltime = 3.154e7*years
times = np.arange(0., totaltime, 86000*years)
# 0.45 and -20000 and 15AU
# 0.45 and -55000 and 20AU
# 0.45 and -5000 and 8AU

def ThreeBodyScattering(times, xi = 15.0*AU, yi = 0.45*AU, vxi = -20000, method='RK4',
wrap=None):

    '''
    Calculates the positions and velocities for a 3 body scattering system.
    A binary star system starts out in a stable orbit.
    A third star approaches with some velocity.
    The initial conditions of this third star can be varied.

    Parameters
    ----------
    times - an array of the times
    xi - initial x position of the third star (default is 15.0 AU)
    yi - initial y position of the third star (default is 0.45 AU)
    vxi - initial x velocity of the third star (default is -20000 m/s)
```

```
    method - integration method (default is 'RK4')

    Example
    ---------------------------
    >>>years, totaltime = 10.0, 3.154e7*years
    >>>times = np.arange(0., totaltime, 86000*years)
    >>>out = ThreeBodyScattering(times, xi = 15.0*AU, yi = 0.45*AU, vxi = -20000,
method='RK4')
    returns an array with the positions and velocities for each star corresponding to each time step
in *times*.
    '''

    IC_pos = np.array([[-AU, 0., 0.], # position of star 1 in binary system
                       [AU, 0., 0.], # position of star 2 in binary system
                       [xi, yi, 0.]]) # position of third star, disturbing the binary system
    IC_vel = np.array([[0., -v, 0.], # position of star 1 in binary system
                       [0., v, 0.], # position of star 2 in binary system
                       [vxi, 0., 0.]]) # position of third star, disturbing the binary system

    p0 = np.empty((3,2,3))
    for j in range(3):
        p0[j,0,:] = IC_pos[j]
        p0[j,1,:] = IC_vel[j]

    steppers = {'euler': euler_step,'RK2': rk2_step,'RK4': rk4_step}
    stepper = steppers[method]

    m = np.array([m1,m2,m3])
    soft = 0.1

    project = NBody(p0, m, soft=0., dpdt=Gravity, wrap=wrap)
```

```
    return project.integrate(times, stepper)

class AnimatedScattering(object):

    '''
    Creates an animation of the three body scattering problem.

    Parameters
    ----------
    method - integration method (default is 'RK4')
    xi - initial x position of the third star (default is 15.0 AU)
    yi - initial y position of the third star (default is 0.45 AU)
    vxi - initial x velocity of the third star (default is -20000 m/s)
    plotsize - the bounds of the x and y axes that are displayed in the animation (default is 10 AU)

    Returns
    ----------
    The binary star system is animated as a small yellow and a medium orange star in a stable orbit.
    The third star is animated as a large blue star. It's initial conditions are displayed.
    The energies (kinetic, potential, and total) of the system are displayed for each frame/time step.

    Example
    ----------
    >>>AnimatedScattering(xi = 15*AU, yi = 0.45*AU, vxi=-20000)
    >>>plt.show()
    '''

    def __init__(self, method='RK4', xi=15*AU, yi=0.0*AU, vxi=-20000, plotsize=10*AU):
        '''
```

```
    Obtains an array of positions and velocities of the stars using the ThreeBodyScattering()
function.
    Initializes the figure and creates the animation.
    '''

    self.vxi = vxi
    self.xi = xi
    self.yi = yi

    # the actual data
    self.out = ThreeBodyScattering(times, xi=self.xi, yi=self.yi, vxi=self.vxi, method=method)

    self.l = len(self.out[:,0,0,0]) # number of frames (parameter for the animate function)
    #fig, ax = plt.subplots(dpi=300)
    fig, ax = plt.subplots()
    self.fig = fig
    self.ax = ax
    self.ax.set_title('3 Body Scattering')
    self.ax.set_xlim(-plotsize/AU, plotsize/AU)
    self.ax.set_ylim(-plotsize/AU, plotsize/AU)

    plt.xlabel('$x$ (AU)')
    plt.ylabel('$y$ (AU)')

    # set up text to displays the energies
    self.textKE = ax.text(-1.4e12/AU, -0.9e12/AU, '')
    self.textPE = ax.text(-1.4e12/AU, -1.1e12/AU, '')
    self.textE = ax.text(-1.4e12/AU, -1.3e12/AU, '')

    # animation
    self.ani = animation.FuncAnimation(self.fig,
```

```
                                self.update,
                                frames=range(self.l),
                                interval=50, #50
                                init_func=self.setup_plot,
                                blit=True)

    def setup_plot(self):
        '''
        Sets up a plot for each star.
        '''

        # setting up plots
        # the third star gets labelled with its initial conditions
        self.line0, = plt.plot([], [],c='orange', marker='o', ms=10)
        self.line1, = plt.plot([], [],c='gold', marker='o', ms=7)
        self.line2, = plt.plot([], [],c='lightblue', marker='o', ms=15,
                               label=f'Initial conditions: $x={self.xi/AU}$ $AU,y={self.yi/AU}$
$AU,v_x={self.vxi}$ $m/s$')
        self.ax.legend(loc='upper left')
        return self.line0, self.line1, self.line2,

    def update(self, i):

        '''
        Obtains the position of each star at each frame and plots it.
        Calculates the kinetic, potential, and total energy for the system at each frame and displays
it.

        Parameters
        ----------
        i - frame number
```

```
'''

# positions at the ith frame
x1, y1 = self.out[:,0,0,0][i], self.out[:,0,0,1][i]
x2, y2 = self.out[:,1,0,0][i], self.out[:,1,0,1][i]
x3, y3 = self.out[:,2,0,0][i], self.out[:,2,0,1][i]

# plotting the positions at the ith frame
self.line0.set_data([x1/AU, y1/AU])
self.line1.set_data([x2/AU, y2/AU])
self.line2.set_data([x3/AU, y3/AU])

# kinetic energy
KE1 = 0.5*m1*((self.out[:,0,1,0][i])**2 + (self.out[:,0,1,1][i])**2)
KE2 = 0.5*m1*((self.out[:,1,1,0][i])**2 + (self.out[:,1,1,1][i])**2)
KE3 = 0.5*m1*((self.out[:,2,1,0][i])**2 + (self.out[:,2,1,1][i])**2)

r12 = np.sqrt((x2 - x1)**2 + (y2 - y1)**2)
r23 = np.sqrt((x3 - x2)**2 + (y3 - y2)**2)
r31 = np.sqrt((x1 - x3)**2 + (y1 - y3)**2)

# potential energy
PE12 = -G*m1*m2/r12
PE23 = -G*m2*m3/r23
PE31 = -G*m3*m1/r31

KE = KE1 + KE2 + KE3
PE = PE12 + PE23 + PE31
E = KE + PE

self.textKE.set_text(f'Kinetic energy = {KE:0.3E} J') # display the kinetic energy
```

```
        self.textPE.set_text(f'Potential energy = {PE:0.3E} J') # display the potential energy
        self.textE.set_text(f'Total energy = {E:0.3E} J') # display the total energy
        return self.line0, self.line1, self.line2,

    def save_anim(self):
        FFwriter = animation.FFMpegWriter(fps=24)
        self.ani.save('ThreeBodyScattering - NewBinarySystem.mp4', writer = FFwriter)


# xi = 15*AU, yi = 1.0*AU, vxi=-20000 weird
scattering_animation = AnimatedScattering(xi = 15*AU, yi = 0.45*AU, vxi=-20000)
#scattering_animation.save_anim()
plt.show()
```