THE FORMATION OF TIDAL TAILS AND OTHER GALACTIC STRUCTURES FROM INTERACTIONS BETWEEN GALAXIES

(Word Count - 2626)

Abstract

The problem of two interacting galaxies is reduced to multiple three body problems by modelling the orbiting particles as a disk of non-interacting test particles. This was investigated numerically using the Euler method to find the galactic structures formed through these interactions. Tidal tails and other structures were seen for some geometries of encounter. When a perturbing galaxy flew by in a parabolic orbit with its velocity against the spin of the test particles, no significant distortions occurred. However, when the galaxy flew with the spin of the particles, various structures such as 'tails' at the bottom of the central galaxy and 'bridges' between the two galaxies were formed, as well as particles being captured by the perturbing galaxy. For these close parabolic encounters, the tails were found to be permanent relics. The mass of the perturbing galaxy and its velocity were varied to see how this affected the structures formed. As its mass was increased, more test masses were captured, and there were more test masses in the tidal tails. As its speed was increased, the amount of disruption to the central galaxy decreased, and the tails that had formed soon dissipated back into the central galaxy.

1 Introduction

Tidal tails are remnants of close encounters between galaxies, interacting via gravity as one flies by another. The tidal forces experienced by the galactic matter in an initially disk shaped galaxy can be intense, either pushing the matter into a long slender tail, creating bridges between the two galaxies, or even being captured by the perturbing galaxy.

There are many observed tidal tail structures in the galaxy. These include the tadpole galaxies and mice galaxies, as shown in figure 1.

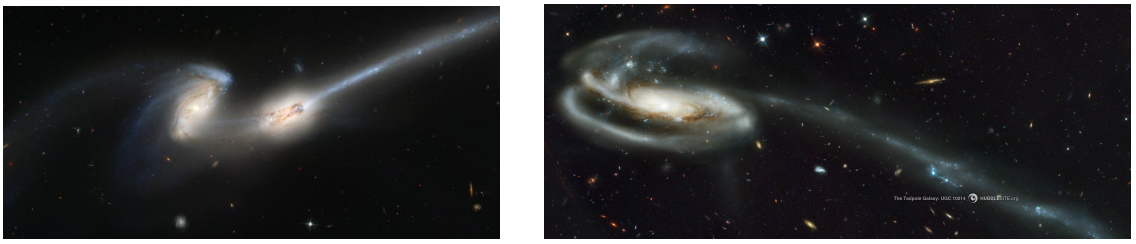

Figure 1: Two observed galaxies with their tidal tails imaged by the Hubble telescope. These are the long slender trails of particles arcing away from the galaxy. The left hand image shows the mice galaxy [1] and the right hand image shows the tadpole galaxy [2].

Structures between multiple galaxies were first observed and studied in the 1950s, [3, 4, 5]. In 1972, simulations by Toomre and Toomre showed that the tail like structures found were indeed remnants of tidal interactions between passing by galaxies in parabolic orbits [6].

In this project, the formation of tidal tails is simulated and reviewed for initial conditions, adjusting the perturbing galaxies mass and trajectory. Section 2 outlines the analysis of the computation required to solve them problem, section 3 outlines how the program was implemented to solve the problem and how well the program performed. Section 4 analyses the structures formed by

interacting galaxies for various different initial conditions, and conclusions are presented in section 5.

2 Analysis

This problem involves simulating the motion of N light masses within a galaxy due to the central mass of the galaxy that they are a part of and to the mass of a perturbing galaxy. Under the assumption that the interactions between the orbiting particles are very weak compared to their interactions between them and the large central masses, we can treat the orbiting masses as non-interacting test masses. A value of $M_{test} = 10^{-6}M_{central}$ was chosen using our Milky Way as a guide, with its central black hole Sgr A* being of order $10^6 M_\odot$ [7].

With the N test particles being non interacting, this problem becomes N 3-body problems with equations of motion

$$\ddot{\mathbf{r}}_\mathbf{i} = -\frac{GM_1(\mathbf{r}_i - \mathbf{R_1})}{|\mathbf{r}_i - \mathbf{R_1}|^\mathbf{3}} - \frac{GM_2(\mathbf{r}_i - \mathbf{R_2})}{|\mathbf{r}_i - \mathbf{R_2}|^\mathbf{3}} \tag{1}$$

$$\ddot{\mathbf{R}}_\mathbf{1} = -\frac{GM_2(\mathbf{R_1} - \mathbf{R_2})}{|\mathbf{R_1} - \mathbf{R_2}|^\mathbf{3}} \tag{2}$$

$$\ddot{\mathbf{R}}_\mathbf{2} = -\frac{GM_1(\mathbf{R_2} - \mathbf{R_1})}{|\mathbf{R_2} - \mathbf{R_1}|^\mathbf{3}} \tag{3}$$

where i-subscript represents the i-th test particle, 1-subscript represents the central galaxy and 2-subscript represents the perturbing galaxy. These require numerical methods to be solved. The Euler method was initially chosen to solve these equations of motion, due to its high speed enabling calculations to be computed quickly. This approximates solutions to the equations of motion as

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \dot{\mathbf{r}}(t)\Delta t \tag{4}$$

$$\dot{\mathbf{r}}(t + \Delta t) = \dot{\mathbf{r}}(t) + \ddot{\mathbf{r}}(t)\Delta t \tag{5}$$

where Δt is the time step between each iteration of the Euler method, and $\ddot{\mathbf{r}}(t)$ is known from equation 1.

As discussed in section 4, a higher order method would have been later implemented to solve these equations more accurately with time permitting. The time step for the Euler method was instead lowered to make sure energy was conserved sufficiently well. The galaxy simulation with interactions, as discussed in figure 4, was run for time steps between 0.001 and 0.1 units. The deviation of energy from the mean was calculated and plotted for each time step, shown in figure 3. $\Delta t = 0.01$ was found to be a good medium between computational accuracy and speed, as it produced a value for total energy that deviated by less than 0.1% for a single galaxy, but still had a fast run time. $\Delta t = 0.001$, although accurate, took 733s to run which was far too long when doing multiple simulations.

Fractional Error in Energy as a Function of Time-Step Length for an Isolated Galaxy

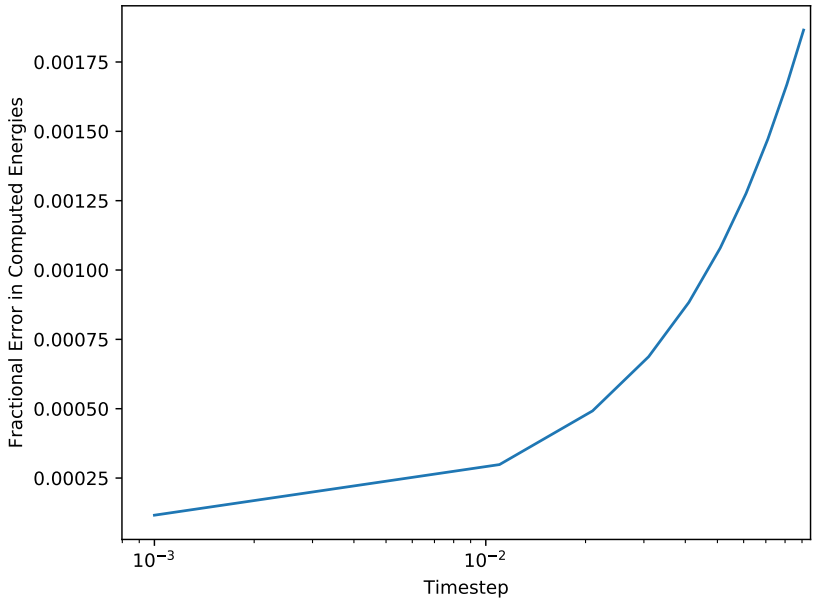

Figure 2: Fractional Error in Energy as a function of time-step length. The fractional error increases as time-step increases, but the time taken to perform the computation also decreases as time-step increases. $\Delta t = 0.01$ was chosen as a compromise between computational accuracy and speed.

Natural units for gravitation (i.e. setting G = 1) and a central mass $M_1 = 1$ unit were chosen when solving this problem for two reasons. These simplify the equations in the problem, since factors of G can now be removed, and also reduce the round off errors accumulated during the simulation.

3 Implementation

This program was written in Python. When implementing this problem, elements were added gradually to different programs, and checked to see whether they behaved as expected.

Firstly, an orbital problem was numerically simulated between a light test mass and a stationary heavy central mass, and the test mass was made to go into a circular orbit, using the formula

$$\mathbf{v}_{\text{circular}} = \sqrt{\frac{\text{GM}_1}{|\mathbf{r}|}}(-\sin\varphi, \cos\varphi, 0) \tag{6}$$

where φ is the angle between the particle and the x axis. This was then duplicated for an arbitrary number of particles distributed in rings with arbitrary radii using nested for-loops, to generate 5 rings of test masses at R = 2, 3, 4, 5 6 units, with 12, 18, 24, 30 36 masses in each ring. The motion of the particles was simulated using the Euler method as outlined in section 2.

Next, the motion of the central and perturbing galaxies were added to the simulation. These were set up with initial position and velocity parameters that can be varied, to change the initial conditions of the orbit. Their positions were then updated also using the Euler method.

The full program has the following structure:

- The input variables for the simulation function are initialized by the user within the program, in the test function. These include the initial positions and velocities of the two galaxies, and the time step for the numerical integration. These were specified within the program so the simulation could be run multiple times, changing these input variables for each run.
- While the simulation runs, the Euler numerical integrator solves the motion of the masses, in steps of length dt which has been specified by the user. The state of the system (i.e. the positions and velocities of the particles, and the total energy of the system) are stored in arrays during each step.
- The output from the simulation is fed into various functions to be used. The fraction of masses in the tail, closest distance between the galaxies and variation of the energy can be calculated. Also, the evolution of the system can be displayed as either an animation or snapshots.
- The results from these functions are fed into a testing function, used to run the simulation multiple times with different initial parameters, and generate several sets of results to be compared.

4 Results and Discussion

In each of the following simulations, the orbital motion of the perturbing galaxy was confined to the same plane as the test masses. This changes the problem from a 3D problem to a 2D problem, so the evolution of the system can be completely seen on the x-y plane.

4.1 Conservation of Energy of the Simulation

The simulation was originally tested for an isolated galaxy, testing how well the energy of the system is conserved to see whether it is physically accurate. For this isolated galaxy, the total energy was calculated at each integration step, with $\Delta t = 0.01$, and then plotted in figure 3. Here, although we see the energy fluctuating about its initial correct value, the deviation never exceeds 0.1% over 100 units of time.

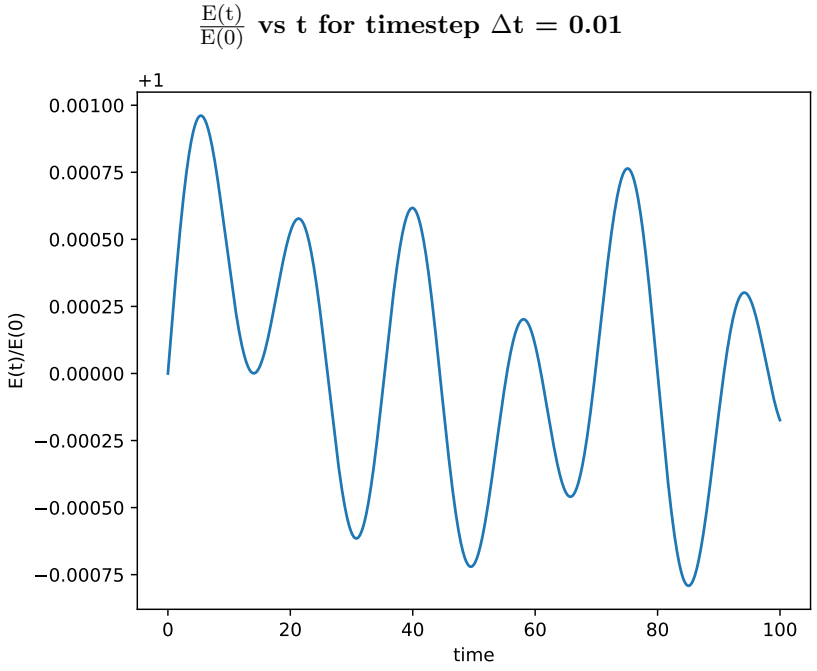

Figure 3: The ratio of energy at a time t with the mean energy as a function of time for a single isolated galaxy. Here, the energy stays within 0.1% of its original value, a suitable accuracy whilst keeping run times not too long.

An ideal simulation would have an error in energy no greater than that in 10^{-12}, as this is the rounding error due to the finite precision arithmetic performed by the computer. However, this would require either a much smaller time step for the Euler method, or implementing a higher order method such as a higher order Runge-Kutta method. Had there been more time this would have implemented, but the deviations from the Euler method were considered satisfactorily small and any graphical errors would be insignificant due to the size of the error.

4.2 Evolution due to Perturbing Galaxy of Equal Mass

Having established that the program works to a sufficient accuracy, the evolution of the system for simple geometries was simulated. Firstly, a perturbing galaxy of equal mass to the central galaxy was flown by in a parabolic orbit both with and against the spin of the test particles. The outputs were plotted in a series of snapshots shown in figures 4 and 5.

Evolution of the central galaxy as the perturbing galaxy passes by in the plane of the test masses, against the direction of their spin

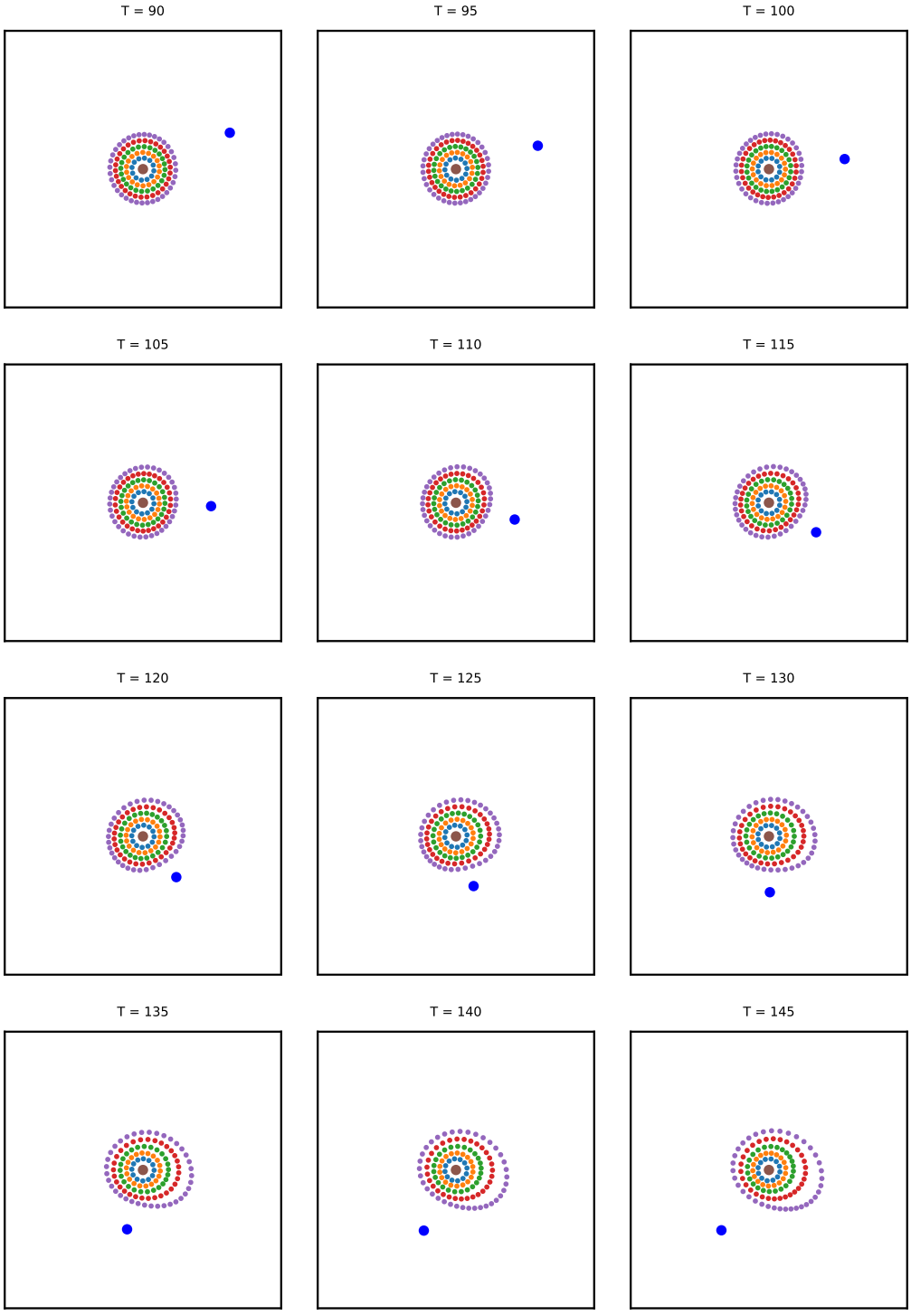

Figure 4: Plots showing the evolution of the initially stationary central galaxy in its frame of reference. The perturbing galaxy flies by in a parabolic orbit, in the plane of the test masses, against the direction of their spin. The distance of closest approach between the two galaxies was 9.46 units. The test particles are spinning anticlockwise. The rings of test particles are of different colours, at R = 2, 3, 4, 5 and 6 units. Here, no significant tail-like structure is formed, and no particles are captured by the perturbing galaxy. At t = 0, $\mathbf{R_2} = (20, 40, 0)$ and $\dot{\mathbf{R}}_\mathbf{2} = (0, –0.31, 0)$. The x and y axes are 50 units in length each. The time to produce this figure was 89s.

Evolution of the central galaxy as the perturbing galaxy passes by in the plane of the test masses, with the direction of their spin

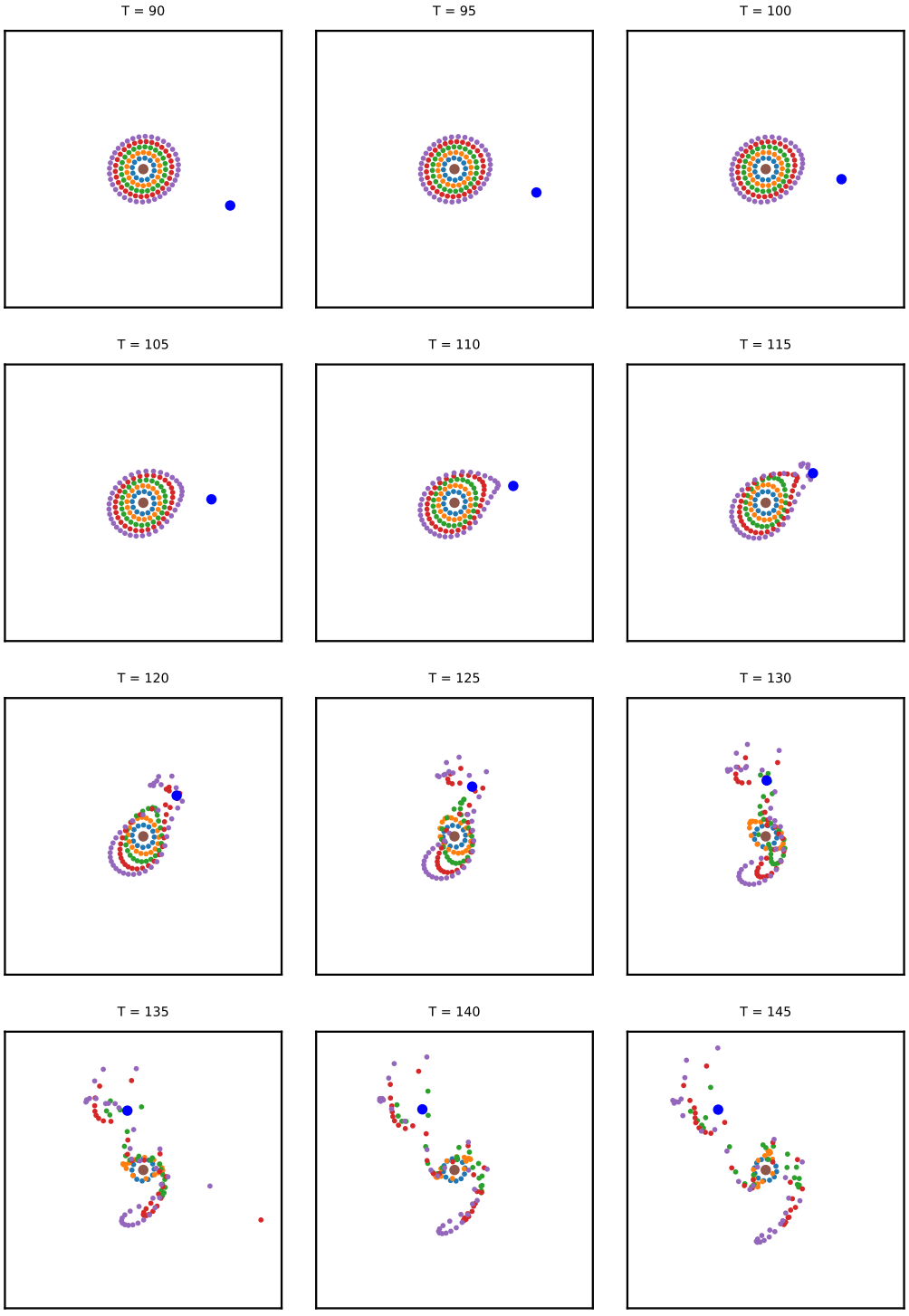

Figure 5: Plots showing the evolution of the initially stationary central galaxy as the perturbing galaxy flies by in a parabolic orbit, in the plane of the test masses, with the direction of their spin. The test particles are spinning anticlockwise. The distance of closest approach between the two galaxies was 9.46 units. Here a long tail like structure is formed below the central galaxy. At T = 180 units, 15% of test masses are in the tail for this simulation, mainly from the outer two rings at R = 5 and 6 units. 19% of test masses are seen to have been captured by the perturbing galaxy, and there are also masses in a 'bridge' in between the two galaxies. At t = 0, $\mathbf{R_2} = (20, -40, 0)$ and $\dot{\mathbf{R}}_\mathbf{2} = (0, 0.29, 0)$. The x and y axes are 50 units in length each. The time to produce this figure was 89s.

It is clear from these figures that passage of the perturbing galaxy against the spin of the test particles has a much smaller effect on the test masses than when the perturbing galaxy moves with the spin. This is expected, as when the perturbing galaxy flies close by to the central galaxy its velocity is matched to the orbital velocities of the test masses. When the perturbing galaxy moves in the same direction of the spin, it exerts a large force on the near-side test masses, midway between it and the central galaxy for a long period of time. When it moves with the opposite direction to the spin it matches the far-side test masses, exerting a much smaller force on them. This is why the central galaxy in figure 4 gets slightly distorted, but no material is torn away from it.

Test material is seen to be from the near side of the disk in figure 5, from the outer rings at R = 5 and 6 units. Here, the gravitational force from the perturbing galaxy overwhelms that from the central galaxy, tearing material from it. 19% of test masses were calculated to have been captured. Some of these test masses, from the top side of the central galaxy, are seen to form a bridge linking the two galaxies together. The forces on these test particles were strong enough to tear them away from the central galaxy, but not strong enough to be completely captured.

Tails are seen to form from the test material at the far side of the disk and test masses are seen to be captured in figure 5. The tail shown has mainly masses from the outer two rings, and contains 15% of test masses from the central galaxy. This tail grows bigger with time in later snapshots of the simulation, as the tail test particles are moving in the opposite direction to the central galaxy and they are too far away to be re-captured by it. It is therefore a permanent feature after the interaction.

4.2.1 Single Ring of Test Masses

The interaction with a single ring of test masses was next simulated, to study its distortion in more detail. The central and perturbing galaxies still have equal masses. The ring at R = 6 units was filled with 120 test masses, to get a more accurate estimate of the number of test masses that were in the tail, had been captured, had escape and had remained in the central galaxy. The results are shown in figure 6.

Evolution of the ring at R = 6 units due to a perturbing galaxy with the same mass as the central galaxy

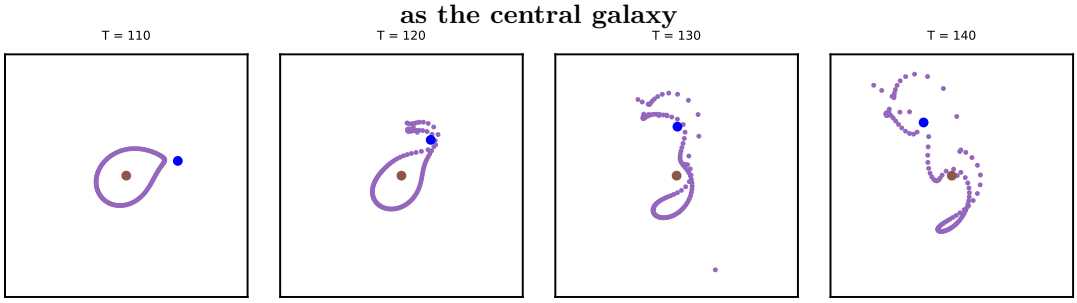

Figure 6: Evolution of the ring at R = 6 units due to a perturbing galaxy of equal mass to the central galaxy. There are 120 test particles in the ring. Here, the particles in the ring contribute to the captured particles, the bridge between the two galaxies and the tidal tail. 24% of test masses are in the tail, 27% of test masses were captured, 8.3% of the test masses escaped from both galaxies and the rest (40%) remained in the central galaxy. The CPU time to produce this figure was 90s.

The particles in the ring contribute to the captured particles, the bridge between the two galaxies and the tidal tail. 24% of test masses are in the tail, 27% of testmasses were captured, 8.3% of the test masses escaped from both galaxies and the rest (40%) remained in the central galaxy. It can also be seen (in the snapshots at T = 130 and 140 in figure 6 that particles from the far side

of the ring are 'sling-shotted' around the central galaxy, overtaking the test masses at the bottom of the galaxy.

4.2.2 Interaction for Increasingly Hyperbolic Orbits

The initial velocity of the perturbing galaxy was increased from $\dot{\mathbf{R}}_\mathbf{2} = (0, 0.3, 0)$ to $\dot{\mathbf{R}}_\mathbf{2} = (0, 0.9, 0)$, so the orbit changes from a parabolic orbit to hyperbolic one. The fractions in different structures were plotted, as shown in figure 7. Increasing the velocity decreased the effect of the perturbing galaxy. This is as expected, as it exerts a large force on the test masses for a shorter amount of time when it travels faster.

In observed galaxies, long arcing tails are therefore permanent relics of very close parabolic encounters, whereas the smaller tails observed are temporary features from more hyperbolic encounters.

Fraction of test particles in the tail, captured, bridged, escaped and in the central galaxy as a function of initial velocity of the perturbing galaxy, at T = 180 units

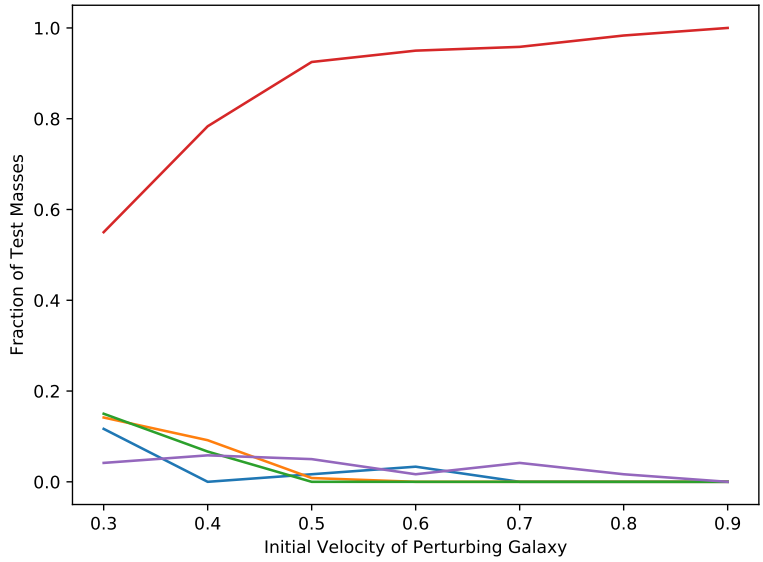

Figure 7: The fraction of test masses in the tail (blue), captured by the perturbing galaxy (orange), escaped from both galaxies (green), in a bridge between the two galaxies (purple) and in the central galaxy (red) as a function of initial speed of the perturbing galaxy. The perturbing galaxy and the central galaxy have the same mass. For a parabolic orbit, at v $\approx$ 0.3 units there is significant disruption to the central galaxy. As speed increases, the disruption goes to zero, and all of the masses remain part of the central galaxy. The CPU time to generate this graph was 692s.

For very hyperbolic orbits with $\dot{\mathbf{R}}_\mathbf{2}(0) > (0, 0.7, 0)$ plots of the evolution of the system showed that small tails did form for a short amount of time, but the test masses within these tails did not have the escape velocity from the central galaxy. This resulted in the masses re-joining the orbits around the central galaxy. An example of this for $\dot{\mathbf{R}}_\mathbf{2}(0) = (0, 0.9, 0)$ is shown in figure 8.

Snapshots showing the formation and de-formation of tidal tails for a highly hyperbolic orbit

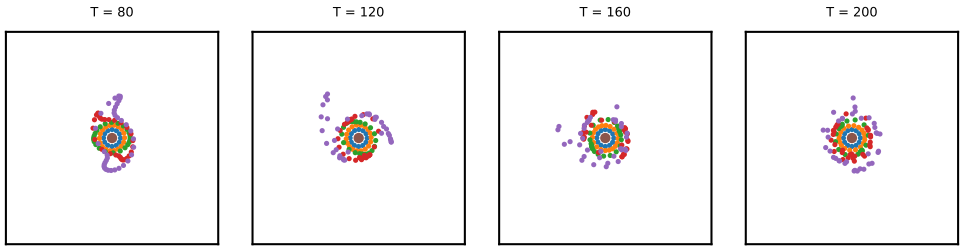

Figure 8: Snapshots of the evolution of the central galaxy for a highly hyperbolic orbit, with $\dot{\mathbf{R}}_\mathbf{2}(0) = (0, 0.9, 0)$. Tidal tails are seen to be formed and then collapse back into the rings of orbits, as the test particles do not have sufficient energy to escape the central galaxy.

4.3 Evolution due to a Perturbing Galaxy of Varying Mass

Real interactions will not all have galaxies of the same mass, so it is important to study the extent of the interaction as mass is varied. The simulation was run from M_2 varying from a $0.2M_1$ to $5M_1$, whilst maintaining that the orbits remained approximately parabolic, with distances of closest approach of between 9 to 12 units.

Snapshots of the evolution of the system were found for $M_2 = 0.2$ units and $M_2 = 4$ units, shown in figures 10 and 11 respectively. The fraction of test masses in different tidal structures was plotted while varying the mass of the perturbing galaxy is shown in figure 9.

Fraction of test particles in the tail, captured, bridged, escaped and in the central galaxy as a function of the mass of the perturbing galaxy, at T = 180 units

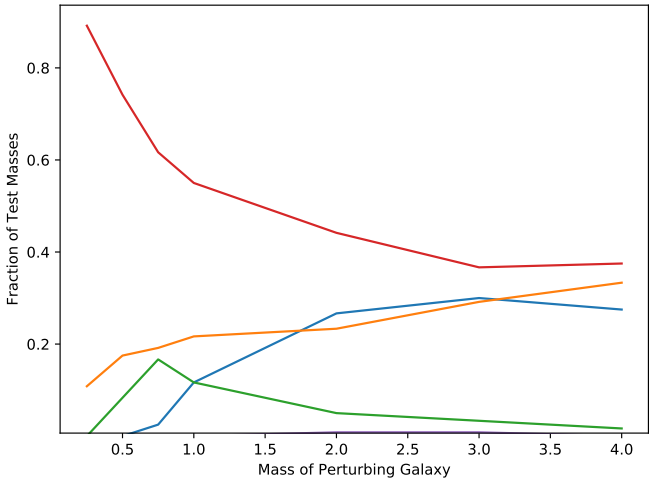

Figure 9: The fraction of test masses in the tail (blue), captured by the perturbing galaxy (orange), escaped from both galaxies (green), in a bridge between the two galaxies (purple) and in the central galaxy (red) as a function of mass of the perturbing galaxy. The perturbing galaxy was kept in a roughly parabolic orbit for each mass, with distance of closest approach kept between 9 and 10 units by adjusting the its speed. Here we can see that the fraction of tail masses and the fraction of captured masses increases as the mass increases. To account for this, the fraction of masses in the central galaxy decreases. Only a few particles ever escape completely, and bridges between the two galaxies have dissipated by t = 180 units. The CPU time to generate this graph was 709s.

Evolution of the central galaxy as the perturbing galaxy with $M_2 = 0.2M_1$ passes by with its path in the same direction as the spin of the test masses.

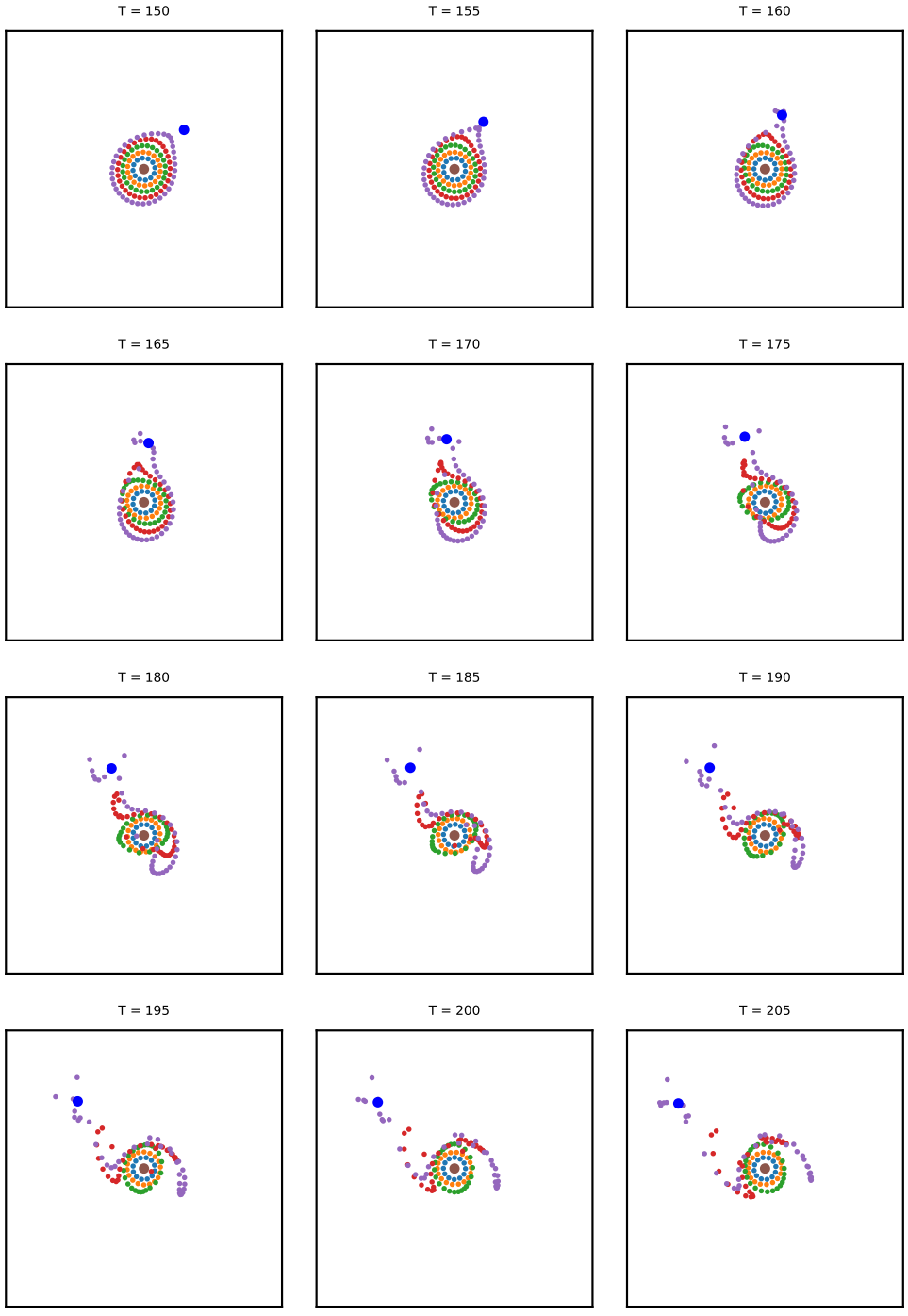

Figure 10: Evolution of the central galaxy when a perturbing galaxy of mass $M_2 = 0.2M_1$ passes by in a parabolic orbit. A tidal tail is seen to have formed and masses have been captured by the perturbing galaxy, but not to the extent as in the equal mass configuration. The bridge between the two galaxies is much more significant for this configuration than when M1 = M2, however this dissipates as the simulation continues.

Evolution of the central galaxy as the perturbing galaxy with $M_2 = 5M_1$ passes by with its path in the same direction as the spin of the test masses.

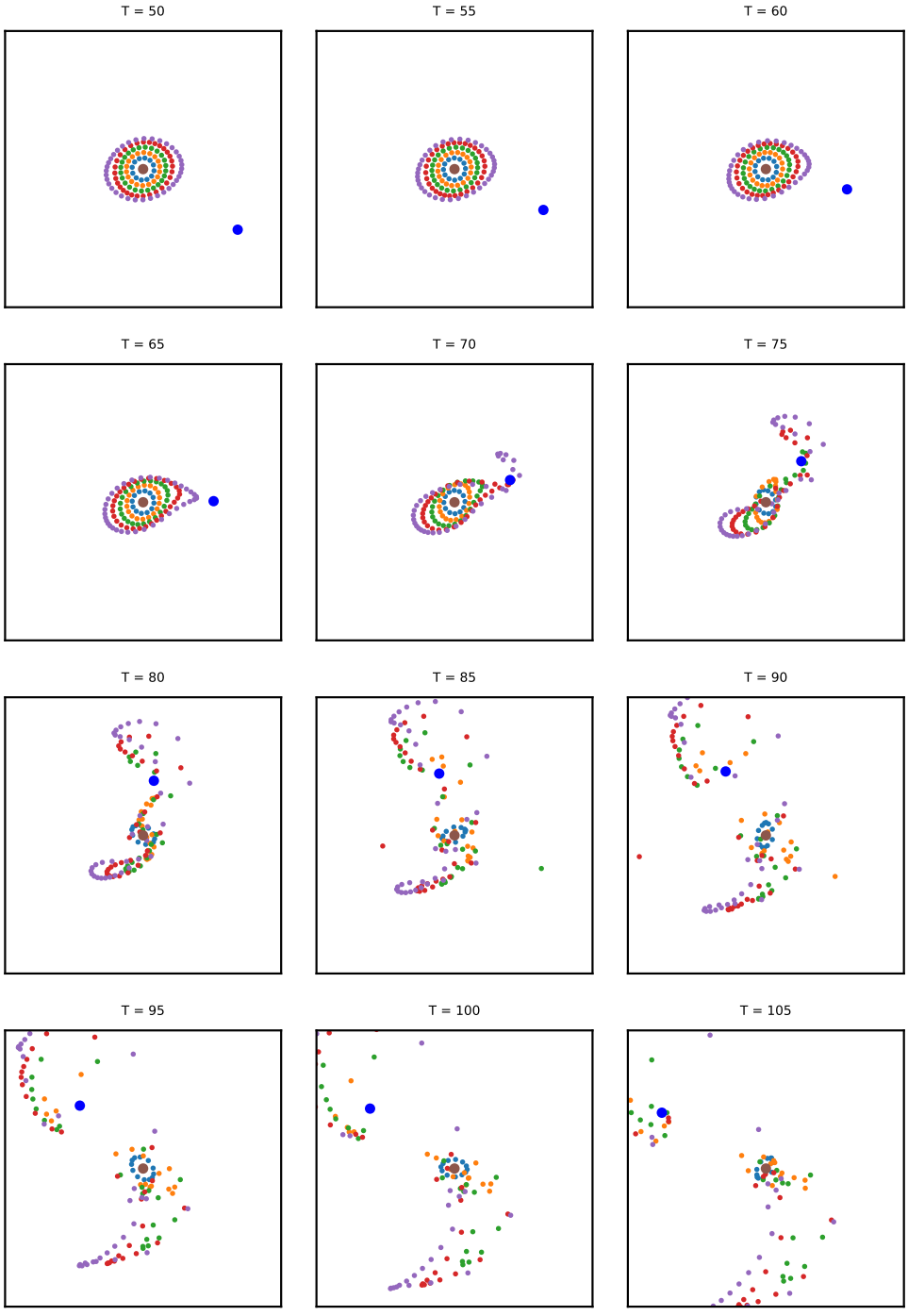

Figure 11: Evolution of the central galaxy when a perturbing galaxy of mass $M_2 = 5M_1$ passes by in a parabolic orbit. A tidal tail is seen to have formed and masses have been captured by the perturbing galaxy, with a greater length than for the equal mass configuration in figure 5. The bridge between the two particles is less significant than when M1 = M2, dissipating very quickly between t = 75 and 90 units.

4.4 Further Work

In all of the previous simulations, we have restricted the motion of the galaxies to the same plane on which the test masses spin - this is obviously not the case for real galactic interactions! To continue investigating galactic interactions, the angle of incident velocity to the x-y plane could be varied from 0^o to 90^o, and the results plotted on the x-z plane or in three dimensions and analysed.

The simulation could also be made more realistic by adding test masses to the perturbing galaxy too. This would also allow us to compare the simulation with observed interactions. If the geometry of observed interactions were known, they could be simulated and compared to observation.

This simulation used the Euler method for numerical integration, known for its instability for larger time steps. To improve computational accuracy whilst maintaining speed, an adaptive time-step higher order numerical method could be implemented, such as the Runge-Kutta-Fehlberg (RK45) method. This method determines the correct stepsize used for each integration step, making it a very accurate numerical method.

5 Conclusions

By simplifying the problem of interacting galaxies to a 3 body problem with non-interacting test masses, the trajectories of the test particles due to interacting was simulated using the Euler method of numerical integration. This was then further reduced to a two dimensional problem, as the perturbing galaxy travelled in the same plane as the central galaxy's disk.

When the perturbing galaxy flew in the opposite direction to the spin of the near side test particles, little disruption to the test masses was found, as the perturbing galaxy did not exert a large force on the test particles for very long. However, when it flew in the same direction to the spin, various structures of test masses were found. Tails of test particles pointing in the direction where the perturbing galaxy came from were found, as well as a bridges of test particles connecting the central and perturbing galaxies. Test particles were also captured by the perturbing galaxy.

Initial parameters were varied and the simulation was run. As the speed of the perturbing galaxy was increased, it was found that the central galaxy became less disturbed, with no tails or bridges forming and no masses being captured. This is as expected as at faster speeds, the perturbing galaxy exerts a large force on the test masses for a smaller amount of time. As the perturbing mass was increased from $0.2M_1$ to $5M_1$ disruption increased, again because greater forces are exerted by the test particles by a larger mass.

The tidal tails formed by interactions were seen in snapshots to get longer as the mass of the perturbing galaxy was increased, but this was not quantified. Also, although trajectories in a two dimensional plane were studied extensively, real interactions take place in three dimensions and could be simulated to further the project.

References

[1] "Hubble's New Camera Delivers Breathtaking Views of the Universe". HubbleSite [online]. 30-04-2002.

[2] "The Tadpole Galaxy: Distorted Victim of Cosmic Collision" . HubbleSite [online]. 30-04-2002.

[3] Zwicky, F; 1953. "Luminous and dark formations of intergalactic matter". *Physics Today*, 6, No. 4, p. 7.

[4] Zwicky, F; 1956. "Multiple Galaxies" *Ergebnisse der Exakten Naturwissenschaften*, 29, p. 344.

[5] Arp, H; 1966 "Atlas of Peculiar Galaxies". California Institute of Technology.

[6] Toomre, Alar; Toomre, Juri; 1972. "Galactic Bridges and Tails". *Astrophysical Journal*, Vol. 178, p. 623.

[7] Binney, J; Tremaine, S; 2008. "Galactic Dynamics". 2nd Edition, *Princeton University Press*, Princeton.

Appendix: Source Code

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from numpy import pi
from numpy import linalg as LA

'''
Arrays that will be filled when the simulation function is ran multiple times
in the test function, to compare what happens when there are different initial
parameters are chosen
'''
E_mean = []
E_std = []
Tailed = []
Captured = []
Escaped = []
Central = []
Bridge = []

def Simulation(dt, G1pos, G2pos, G1vel, G2vel, M2, E_mean, E_std, Tailed, Captured,
    Escaped, Central, Bridge):
    """
    Setting up the problem. The masses of the two galaxies are equal to 1,
    and G has been scaled to 1. The test particles have a mass of 10^-6, as
    using data that our M_sun ~ 10^-6 * M_Sg-A*
    """

    G=1
    M1=1
    Mtest = 10**(-6)

    """
    The radii of the rings of test particles around the galaxy and the number of
    test particles on each ring.
    """

    Number = 1 * np.array([12, 18, 24, 30, 36])
    R = [2, 3, 4, 5, 6]

    """
    Lots of data has to be stored to solve the equations of motion of the masses.
    Here the lists storing information are set up.
    """

    position1 = [] #position of the first galaxy
    position2 = [] #position of the perturbing galaxy
    testmasspositions = [] #positions of the test masses
```

```
    positiontest = []
    #array containing positions of test masses, updated for each time step
    Rvectors = [] #list of vectors from test masses to galaxy 1
    Distances = [] #list of distances from test masses to galaxy 1
    Rvectors2 = [] #list of vectors from test masses to galaxy 2
    Distances2 = [] #list of distances from test masses to galaxy 2
    initialdistances = []
    #array of initial distances of test masses to galaxy 1.
    #Used to see whether test mass has been disturbed
    testpositions = [] #initial test mass positions
    testvelocities = [] #list of test mass velocity vectors
    testmomenta = [] #list of test mass momentum vectors
    testKEs = [] #list of test mass kinetic energies
    testPEs = [] #list of test mass potential energies from galaxies 1 and 2
    totaltestKEs = [] #total kinetic energy from all of the test masses
    totaltestPEs = [] #total potential energy from all of the test masses
    testforces = [] #list of resultant force vectors on test masses
    KE1 = [] #list of kinetic energies of galaxy 1 for each time step
    KE2 = [] #list of kinetic energies of galaxy 2 for each time step
    GalPEs = []
    #list of potential energy for both galaxies, updated for each time step
    galaxydistances = []
    #list of distances between two galaxies, updated for each time step


    """Setup of the galaxies:"""

    #Initial positions of of each galaxy
    galaxyvector = G2pos - G1pos
    galaxydistance = LA.norm(galaxyvector)
    galaxydistances.append(galaxydistance)

    #Initial momenta of each galaxy
    G1mom = G1vel * M1
    G2mom = G2vel * M2

    #Initial Energies of each galaxy
    G1KE = 0.5 * M1 * (LA.norm(G1vel))**2
    G2KE = 0.5 * M1 * (LA.norm(G2vel))**2
    GalPE = -G * M1 * M2 / (galaxydistance)

    #Initial forces on each galaxy
    F12 = - G * M1 * M2 * galaxyvector/(galaxydistance**3)
    F21 = - F12

    #Certain initial values are stored in lists, to be updated at each time step
    position1.append(G1pos)
    position2.append(G2pos)

    KE1.append(G1KE)
    KE2.append(G2KE)
    GalPEs.append(GalPE)

    """Setup of the test masses"""
    #ith ring of test masses
    for i in range(len(R)):

        #jth test mass in the ith ring
        for j in range(Number[i]):

            #trig argument for the initial circular motion of the test particles
            trig_argument = 2*pi*j/Number[i]

            #position vector
            testpos = np.array([R[i] * np.cos(trig_argument),
                                R[i] * np.sin(trig_argument), 0])
            testpositions.append(testpos)
            np.array(testpositions)

            #displacement from galaxy 1
```

```
            Rvector = testpos - G1pos
            Rvectors.append(Rvector)
            np.array(Rvectors)

            #distance from galaxy 1
            Distance = LA.norm(Rvector)
            Distances.append(Distance)
            initialdistances.append(Distance)
            np.array(Distances)

            #displacement from galaxy 2
            Rvector2 = testpos - G2pos
            Rvectors2.append(Rvector2)
            np.array(Rvectors2)

            #distance from galaxy 2
            Distance2 = LA.norm(Rvector2)
            Distances2.append(Distance2)
            np.array(Distances2)

            #velocity of test masses
            testvel = np.sqrt(G*M1/R[i]) * np.array([-np.sin(trig_argument),
                        np.cos(trig_argument), 0])
            testvelocities.append(testvel)
            np.array(testvelocities)

            #momentum for test masses
            testmom = testvel * Mtest
            testmomenta.append(testmom)
            np.array(testvelocities)

            #KE for test masses
            testKE = 0.5 * Mtest * (LA.norm(testvel))**2
            testKEs.append(testKE)
            np.array(testKEs)

            #PE for test masses from both galaxies
            testPE = - G * Mtest * ((M1/Distance) + (M2/Distance2))
            testPEs.append(testPE)
            np.array(testPEs)

            #Forces on test masses from galaxies 1 and 2
            Ftest1 = - G * M1 * Mtest * Rvector/(Distance**3)
            Ftest2 = - G * M2 * Mtest * Rvector2/(Distance2**3)
            testforces.append(Ftest1 + Ftest2)

        positiontest.append(testpositions)

    totaltestKE = np.sum(testKEs)
    totaltestPE = np.sum(testPEs)

    totaltestKEs.append(totaltestKE)
    totaltestPEs.append(totaltestPE)


    """Solving the equations of motion for the test masses"""
    t=0
    DT = int(1/dt)
    Tmax = 180
    T = 1


    """
    Updates the parameters for each time step, time dt after the previous step.
    Certain parameters are stored for each time step, to be used in analysis.
    This program uses an Euler method to solve the equations of motion.
    """
    while t<Tmax:

        for i in range(len(Distances)):
```

```
            testmomenta[i] = testmomenta[i] + testforces[i] * dt

            testvelocities[i] = testmomenta[i] / Mtest

            testpositions[i] = testpositions[i] + testvelocities[i] * dt
            testmasspositions.append(testpositions[i])

            Rvectors[i] = testpositions[i] - G1pos
            Distances[i] = LA.norm(Rvectors[i])

            Rvectors2[i] = testpositions[i] - G2pos
            Distances2[i] = LA.norm(Rvectors2[i])

            totaltestKE += 0.5 * Mtest * (LA.norm(testvelocities[i]))**2

            totaltestPE += - G * Mtest * ((M1/Distances[i]) + (M2/Distances2[i]))
            testPEs.append(totaltestPE)

            testforces[i] = - ((G * M1 * Mtest * Rvectors[i]/(Distances[i]**3))
             + (G * M2 * Mtest * Rvectors2[i]/(Distances2[i]**3)))

        positiontest.append(testmasspositions)
        totaltestKEs.append(totaltestKE)
        totaltestPEs.append(totaltestPE)

        totaltestPE = 0
        totaltestKE = 0
        testmasspositions = []

        #update for galaxies
        G1mom = G1mom + F21*dt
        G2mom = G2mom + F12*dt

        G1vel = G1mom / M1
        G2vel = G2mom / M2

        G1pos = G1pos + G1vel * dt
        G2pos = G2pos + G2vel * dt

        galaxyvector=G2pos-G1pos
        galaxydistance = LA.norm(galaxyvector)
        galaxydistances.append(galaxydistance)

        G1KE = 0.5 * M1 * (LA.norm(G1vel))**2
        G2KE = 0.5 * M1 * (LA.norm(G2vel))**2
        GalPE = -G * M1 * M2 / (galaxydistance)

        F12 = - G * M1 * M2 * galaxyvector/(galaxydistance**3)
        F21 = - F12

        position1.append(G1pos)
        position2.append(G2pos)

        KE1.append(G1KE)
        KE2.append(G2KE)
        GalPEs.append(GalPE)

        t=t+dt

    #conversion to numpy arrays for ease later on
    positiontest = np.array(positiontest)
    position1 = np.array(position1)
    position2 = np.array(position2)
    KE1 = np.array(KE1)
    KE2 = np.array(KE2)
    GalPEs = np.array(GalPEs)
    totaltestKEs = np.array(totaltestKEs)
    totaltestPEs = np.array(totaltestPEs)


```

```
    """
    Function outputs the total energy of the system,
    checking how well energy is conserved
    """
    def TotalEnergy(KE1, KE2, GalPEs, totaltestKEs, totaltestPEs):
        totalenergy = KE1 + KE2 + GalPEs + totaltestKEs + totaltestPEs
        E_mean.append(np.mean(totalenergy))
        E_std.append(np.std(totalenergy))

        print("The maximum value of energy is: %e " %(max(totalenergy)))
        print("The minimum value of energy is: %e " %(min(totalenergy)))
        print("The mean value of energy is: %e " %(np.mean(totalenergy)))
        print("The standard deviation of energy values is: %e " %(np.std(
totalenergy)))
        print("The standard error of energy values is: %e" %(np.std(totalenergy)*np
.sqrt(dt/Tmax)))

        '''
        plt.plot(totalenergy)
        plt.savefig('EnergyCons.pdf')
        '''


    """
    Function outputs the number and fraction of test particles that have been
    disrupted in various ways by the perturbing galaxy
    """
    def TailFraction(initialdistances, Distances):
        EscapeNumber = 0
        TailNumber = 0
        CaptureNumber = 0
        CentralNumber = 0
        BridgeNumber = 0

        for i in range(len(initialdistances)):

            if Distances[i] < 10:
                CentralNumber += 1

            elif Distances2[i] < 20:
                CaptureNumber += 1

            elif (Distances[i] > 20 and Distances[i] < Distances2[i]):
                TailNumber += 1

            elif Distances[i] < galaxydistance and Distances2[i] < galaxydistance:
                BridgeNumber += 1

            else:
                EscapeNumber += 1

        print(
        """Tail Number: %i"""
        %(TailNumber))

        print(
        """Capture number: %i"""
        %(CaptureNumber))

        print(
        """Centre number: %i"""
        %(CentralNumber))

        print(
        """Bridge number: %i"""
        %(BridgeNumber))

        print(
        """Escape number: %i"""
        %(EscapeNumber))
```

```
325
326         frac_in_tail = TailNumber/len(Distances)
327         frac_capt = CaptureNumber/len(Distances)
328         frac_esc = EscapeNumber/len(Distances)
329         frac_central = CentralNumber/len(Distances)
330         frac_bridge = BridgeNumber/len(Distances)
331
332         Tailed.append(frac_in_tail)
333         Captured.append(frac_capt)
334         Escaped.append(frac_esc)
335         Central.append(frac_central)
336         Bridge.append(frac_bridge)
337
338         print(Tailed)
339         print(Captured)
340         print(Central)
341         print(Bridge)
342         print(Escaped)
343
344
345     """
346     Function outputs the closest distance from galaxy 1 to galaxy 2 for simulation
347     """
348     def ClosestDistance(galaxydistances):
349         print("The closest distance is: %f " %(min(galaxydistances)))
350
351
352     """
353     Function outputs snapshots of the evolution of the system
354     """
355     def Snapshot(positiontest, position1, position2, Tmax):
356
357         #if (G2vel[1]*100) % 5 == 0:
358
359         plt.figure()
360         T = 50
361         T_plus = T + 60
362         i = 1
363         NumCum = np.array(np.cumsum(Number))
364         NumCum = np.insert(NumCum, 0, 0)
365
366         while T < T_plus:
367
368             plt.subplot(4, 3, i, aspect='equal')
369
370             for j in range(len(NumCum)-1):
371                 P = NumCum[j]
372                 Q = NumCum[j+1]
373                 plt.plot(positiontest[T*DT, P:Q, 0],
374                 positiontest[T*DT, P:Q, 1],
375                 marker='o', linestyle='', markersize=1, linewidth='0.1')
376             plt.plot(position1[T*DT,0], position1[T*DT,1], marker='o', markersize
    =3)
377             plt.plot(position2[T*DT,0], position2[T*DT,1], marker='o', markersize
    =3, color='b')
378             plt.xlim(position1[T*DT,0]-25, position1[T*DT,0]+25)
379             plt.ylim(position1[T*DT,1]-25, position1[T*DT,1]+25)
380             plt.title('T = %i' %(T), fontsize='5')
381             plt.xticks([])
382             plt.yticks([])
383             #plt.axis('off')
384             #plt.xlabel('distance / units', fontsize='5')
385             #plt.ylabel('distance / units', fontsize='5')
386             #plt.tick_params(labelsize=8)
387             i += 1
388             T += 5
389
390         plt.subplots_adjust(left=0.2, hspace=0.0)
391         plt.show()
392
```

```
393
394     """Function that animates the paths of the galaxies and test masses"""
395     def FuncAnimation(Distances, positiontest, position1, position2, Tmax):
396
397         fig, ax = plt.subplots()
398         plt.xlim(-30,70)
399         plt.ylim(-50,50)
400
401         line = []
402
403         for i in range(len(Distances)):
404                 line.append((ax.plot(positiontest[0::DT, i, 0], positiontest[0::DT,
     i, 1],
405                 'b', marker='o', linestyle='', markersize=1))[0])
406         line.append((ax.plot(position1[0::DT,0], position1[0::DT,1], marker='o'))
    [0])
407         line.append((ax.plot(position2[0::DT,0], position2[0::DT,1], marker='o'))
    [0])
408
409
410         def update(num, positiontest, position1, position2, line):
411             for i in range(len(Distances)):
412                 line[i].set_data(positiontest[0::DT, i, 0][num], positiontest[0::DT
    , i, 1][num])
413             line[len(Distances)].set_data(position1[0::DT,0][num], position1[0::DT
    ,1][num])
414             line[len(Distances)+1].set_data(position2[0::DT,0][num], position2[0::
    DT,1][num])
415
416
417         ani = animation.FuncAnimation(fig, update, len(positiontest[0::DT, 0, 0]),
418                                       fargs=[positiontest, position1, position2,
    line],
419                                       interval=100, blit=False)
420
421         plt.gca().set_aspect('equal', adjustable='box')
422
423         ani.save('simulationeuler %f %f.mp4' %(M1, G2vel[1]))
424
425     TailFraction(initialdistances, Distances)
426     ClosestDistance(galaxydistances)
427     TotalEnergy(KE1, KE2, GalPEs, totaltestKEs, totaltestPEs)
428     Snapshot(positiontest, position1, position2, Tmax)
429     FuncAnimation(Distances, positiontest, position1, position2, Tmax)
430
431 def Tests():
432
433
434     def EnergyCons():
435         G1pos = np.array([0, 0, 0])
436         G2pos = np.array([20, 40, 0])
437         G1vel = np.array([0, 0, 0])
438         G2vel = np.array([0, -0.31, 0])
439         M2=1
440         for dt in np.arange(0.001, 0.1, 0.01):
441             Simulation(dt, G1pos, G2pos, G1vel, G2vel, M2, E_mean, E_std, Tailed,
    Captured, Escaped, Central, Bridge)
442         print(E_std)
443         print(E_mean)
444         E_frac = np.array(E_std)/np.array(E_mean)
445         plt.plot(np.arange(0.001, 0.1, 0.01), E_frac)
446         plt.show()
447
448     def SameMass():
449         M2 = 1
450         Simulation(dt, G1pos, G2pos, G1vel, G2vel, M2, E_mean, E_std, Fractions)
451
452     def ChangingVelocities():
453         G1pos = np.array([0, 0, 0])
454         G2pos = np.array([20, -40, 0])
```

```
455         G1vel = np.array([0, 0, 0])
456         G2vel = np.array([0, 0.31, 0])
457         M2=1
458         dt=0.01
459 
460         for i in range(30, 100, 10):
461             G2vel[0] = 0
462             G2vel[1] = i
463             G2vel = G2vel/100
464             Simulation(dt, G1pos, G2pos, G1vel, G2vel, M2, E_mean, E_std, Tailed,
    Captured, Escaped, Central, Bridge)
465 
466         plt.plot(np.arange(0.30,1.00,0.10), Tailed)
467         plt.plot(np.arange(0.30,1.00,0.10), Captured)
468         plt.plot(np.arange(0.30,1.00,0.10), Escaped)
469         plt.plot(np.arange(0.30,1.00,0.10), Central)
470         plt.plot(np.arange(0.30,1.00,0.10), Bridge)
471         plt.show()
472 
473 
474     def ChangingMasses():
475         Initials = [25, 27, 29, 31, 38, 45, 52]
476         J=0
477         G1pos = np.array([0, 0, 0])
478         G2pos = np.array([20, -40, 0])
479         G1vel = np.array([0, 0, 0])
480         G2vel = np.array([0, 0, 0])
481         dt=0.01
482 
483         for i in np.arange(0.25, 1, 0.25):
484             G2vel[1] = Initials[J]
485             G2vel[0] = 0
486             G2vel = G2vel/100
487             M2 = i
488             Simulation(dt, G1pos, G2pos, G1vel, G2vel, M2, E_mean, E_std, Tailed,
    Captured, Escaped, Central, Bridge)
489             J += 1
490 
491         for i in np.arange(1, 5, 1):
492             G2vel[1] = Initials[J]
493             G2vel[0] = 0
494             G2vel = G2vel/100
495             M2 = i
496             Simulation(dt, G1pos, G2pos, G1vel, G2vel, M2, E_mean, E_std, Tailed,
    Captured, Escaped, Central, Bridge)
497             J += 1
498 
499         plt.plot([0.25, 0.5, 0.75, 1, 2, 3, 4], Tailed)
500         plt.plot([0.25, 0.5, 0.75, 1, 2, 3, 4], Captured)
501         plt.plot([0.25, 0.5, 0.75, 1, 2, 3, 4], Escaped)
502         plt.plot([0.25, 0.5, 0.75, 1, 2, 3, 4], Central)
503         plt.plot([0.25, 0.5, 0.75, 1, 2, 3, 4], Bridge)
504         plt.show()
505 
506     EnergyCons()
507 
508 Tests()
```