Exploring the creation of galactic tidal tails via computer modelling

A Part II Computing Project, University of Cambridge
Candidate Number: 7022R

April 30, 2018

Abstract

This report acts primarily as a proof (by computational simulation) that galactic tails and bridges, the extended limb-like structures observed attached to many galaxies, are the relics of close galactic encounters where extreme tidal forces ripped stars from their host galaxy. It goes on to study how tidal tails vary in form when the initial conditions and galactic model is varied. Tidal tails were found to form more readily on lighter galaxies and almost exclusively when the rotation of the stars around the galaxy is in the same sense as the rotation of the galaxies around one another. There is also an optimal zone (for the distance of closest approach between the galaxies) for tail formation, above which the tidal forces are insufficient to form tails and below which the interaction is too catastrophic for any tail structure to be distinguishable. When the galactic mass is modelled as central point mass (e.g. a black hole) a large proportion of stars are lost from their host galaxy as they scatter off the singularity - conversely when it is modelled as a uniform, spherical, non-interacting dark matter halo, more diffuse tails are formed with no scattering. Finally, a hybrid model (where the density of the halo decayed with increasing radius according to a Gaussian), was found to blend the features of both these models and provided an additional degree of freedom (the halo *width*) with which to potentially fit these simulations to real astronomical observations.

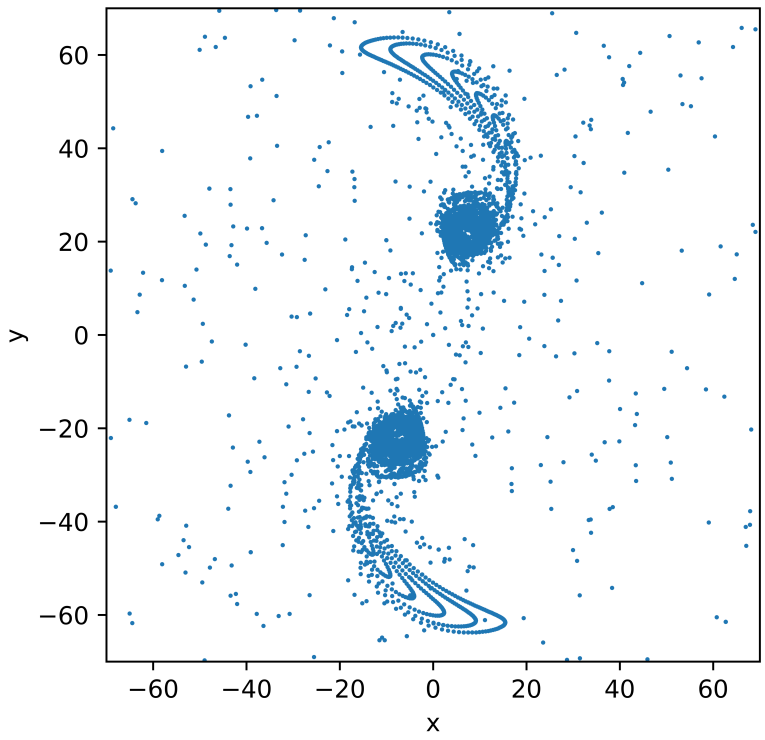


Figure 1: The interaction of two galaxies, each made up of concentric rings of stars (3780 stars in each) from initial radii 2 to 6 units, orbiting a central black hole. The galaxies pass with a closest approach distance of 10 units. Tidal forces create large tidal tails of stars flung from their original galaxy and stars are seen to scatter in all directions, accelerated by the gravitational slingshot effect, as they pass near to the neighbouring galaxy's black hole. Movie file: `tidaltails.mp4`

1 Introduction

Many galaxies in multiple galaxy systems display extended limb-like structures of stars and dust known as *tails* and *bridges*. As argued by Toomre and Toomre [1] in 1972, this report aims to demonstrate, by computational simulation, that these are the relics of extreme tidal forces from close galactic encounters. Indeed, such tidal tails can have very interesting astrophysical consequences, over 1% of stellar formation in the universe is thought to occur in tidal tails [2] and, as argued by Barnes [3], tidal tails can themselves spawn dwarf galaxies.

Tidal forces arise due to the non-constant gravitational force across a body [4] and act to distort the body's original shape. In galactic encounters, the tidal forces can be extreme enough to tear stars away from their original galaxy, creating the dramatic tails this report aims to study. Toomre and Toomre used a simple *black hole galaxy* model for a galaxy with non-interacting stars orbiting a point mass at the centre. This report expands on this by also modelling the galactic mass as a *uniform dark matter halo*, where the mass is distributed uniformly across a spherical halo. A hybrid was also investigated, the *Gaussian dark matter halo*, where the mass density of the halo decreased from the centre according to a Gaussian, with the hope of giving more accurate predictions.

To reduce computing time, the orbiting stars were assumed to be light, such that they did not interact with each other, only with the galactic masses. This assumption is valid if the majority of the mass within the galaxy is contained within the black hole (or the surrounding dark matter), not the stars themselves.

By varying the initial conditions (e.g. galactic masses, rotation direction, distance of closest approach) these models were investigated and visualised as 2D plots. The simulations were also animated and saved as movie files to supplement the plots and these are referred to throughout.

Section 2 analyses the important physics and computational aspects of this investigation, section 3 discusses the implementation of the program, section 4 deals with the performance of the programme as well as the steps taken to optimise it, and section 5 presents and discusses the main results of this report.

2 Analysis

The problem has been scaled by setting G=1 and the masses and length scales of order $10^0 - 10^1$ which produced results with time scales of order 10^2. The problem could easily be rescaled to reflect more realistic galactic motion. From here onwards graph axis labels x and y refer to the distance in arbitrary units (specifically, metres in a universe where G = 1).

The stars were initiated to move in circular motion around their host galaxy, evenly spaced on rings at radii 2, 3, 4, 5 and 6. The number of stars on each ring was set to be 24, 36 48, 60 and 72 respectively (except where stated, e.g. figure (1)). For circular motion the centripetal acceleration is given by:

$$\ddot{\mathbf{r}}_\mathbf{c} = -\frac{v^2}{r}\hat{\mathbf{r}} \tag{1}$$

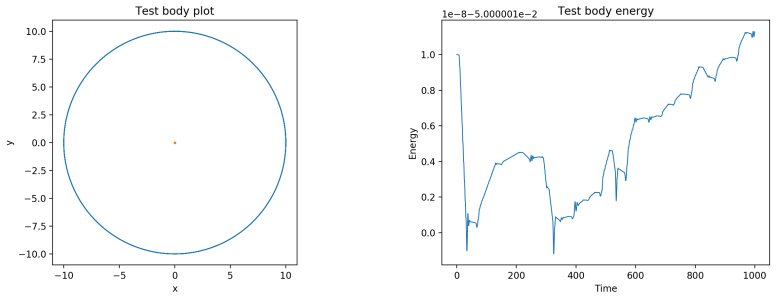
This is provided by the gravitational field due to the central galactic mass:

$$\ddot{\mathbf{r}}_\mathbf{g} = -\frac{GM}{r^2}\hat{\mathbf{r}} \tag{2}$$

For a star at radius r, by Gauss's law, M is just the total mass enclosed, M_{enc}, within a sphere at this radius. For a black hole galaxy, this is the total mass of the galaxy, whilst for the dark matter halo galaxies it may be less.

For the galactic interactions, the motion of *both* galaxies is fully simulated and plotted. The alternative being to perturb a stationary galaxy with a second galactic mass (with no stars orbiting) as suggested in the project handout [5]. Although the latter is simpler and less computationally expensive, the former has the benefit of being symmetric when the two galaxies have equal masses and, if the masses are different, it is easy to see the difference in effect of the encounter on the two galaxies.

SciPy's inbuilt integrator, `odeint`, was used for the integration of the ODEs. It uses the Runge Kutta technique of order 4, known to be one of the better algorithms for gravitational simulations - alternatives could include simple Euler integration or Verlet integration. To check this integrator was good enough for our uses, a simple simulation of one star orbiting a planet was run with condition of the same order of magnitude as were used for the full galactic simulation throughout this report. These were: $G = 1$, galactic $mass \simeq 1$, $r \simeq 10$, $t \simeq 1000$, $integratortimestep \simeq 2$, $orbit \simeq circular$ - this corresponds to approximately 5 circular orbits of a test body. Figure (2) shows this plot and confirms that the deviation from circular orbit over the integration time is too small to see by eye, and that to within 2.4×10^{-5}% energy is conserved.



(a) (b)

Figure 2: Figure (a) shows a plot of a test body orbiting a galactic mass (M=1) in circular orbit at a radius of 10 for a time of 1000 ($\sim$ 5 orbits). The integrator used was SciPy's inbuilt `odeint` which uses a fourth order Runge Kutta algorithm and the number of time steps was 500 - these were chosen as they are typical of those used in the rest of the report. No visible deviation from circular orbit is observed. Figure (b) shows a plot of the energy over the simulation time. The energy fluctuates showing the simulator does not conserve energy but, by considering the maximum and minimum energy, only by 2.4×10^{-5}% of the total energy hence it is concluded that this integrator is easily good enough for the purposes of this study.

3 Implementation

When studying the interaction between two galaxies their total energy was set to zero so that they followed parabolic orbits. In summary, the algorithm used to simulate two interacting galaxies was as follow:

1. Calculate the initial conditions (position and velocity) of the galaxies required for them to pass each other in parabolic orbit (E=0) with distance of closest approach, R_{CA}.
2. Integrate galactic motion.
3. Calculate and set the initial conditions for the stars to orbit the two galaxies in concentric rings with circular motion. This requires knowledge of the initial position and velocity of both galaxies.
4. For each star in turn, integrate to find its motion. At any given time, the position of the two galactic centres (needed to evaluate the force on the star) is given by that found in step 2.
5. Pass the star positions to the animate function to animate the simulation.

To find the initial star conditions the velocities are found by solving equations (1) and (2) and the stars are dispersed evenly around a circle at the chosen radius. To calculate the initial conditions of the galaxies such that they have zero total energy *and* pass with closest approach distance of R_{CA} the starting velocity for both galaxies (set equal and opposite) required to satisfy the zero energy requirement ($1/2M_1v^2 + 1/2M_2v^2 = GM_1M_2/r$) was first found. Then, the angle at which they are set off with respect to horizontal (galaxy 1 starts in the top left, whilst galaxy 2 starts in the bottom right) is varied in steps of 1°, the motion is integrated for each angle, and the distance of closest approach is found. The angle is then selected which gives the closest approach nearest to that desired. While this may seem an arduous way to go about it when, for two bodies, it can be solved analytically, it actually only takes of the order 1 second in computing time when written as a function (`theta_for_closest_approach`) and leaves open the possibility of generalising to more than two bodies in the future, something which cannot be solved analytically.

The code is written in a procedural way (rather than object oriented) so no classes were introduced. This was chosen because the problem in hand was sufficiently simple that it can be solved by manipulating and passing tensors between five basic functions (see `#SIMULATION FUNCTIONS`, section 7.1). Each type of function has $\sim$6 similar versions which perform the same task (e.g. setting initial values, passing the derivative to the ODE for integrating etc.) for the different types of simulation needed in this study - single galaxy, double galaxy black hole, double galaxy dark matter halo etc.

4 Performance

This is at most a two body simulation when integrating the motion of the two galaxies. Integrating the star motion is only a 1 body simulation since the stars are non-interacting. The overwhelming majority of computing time was

used to integrate the star motion for all N stars. Since this is actually N successive one-body simulation it scales linearly with $\mathcal{O}(N)$ complexity, see figure (3), compared to a full blown N-body simulation which has complexity $\mathcal{O}(N^2)$.

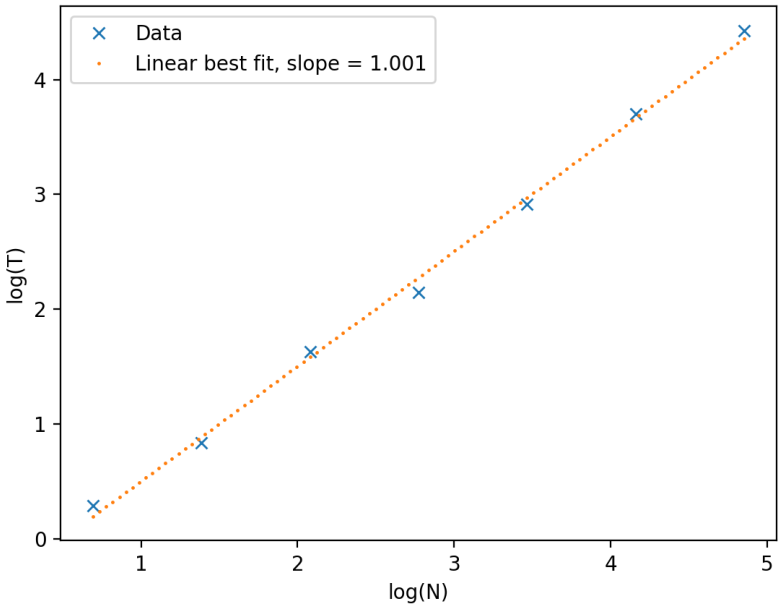


Figure 3: Figure showing log(N) against log(T) where N is the total number of stars simulated and T is the total computing time. The simulation used was the interaction of two galaxies, each with a single ring of N/2 stars at a radius of 4, passing at closest approach of 10 units. The number of integration time steps used was 400. T is taken to be the mean of three repeat simulations. The gradient is 1.001 which indicates that T scales linearly with N, as expected.

The performance of the programme was optimized in a few ways - firstly, by approximating the stars as non-interacting, secondly, by vectorizing code where possible and thirdly, by optimizing the number of time steps so as not to be unnecessarily accurate. After some experimentation, it was concluded that an integration time step of ~ 3 was suitable and as large as possible whilst still giving smooth star motion. The only exception to this was for the black hole galaxy interactions where stars coincidentally passing near to the neighbouring galaxy's singularity were accelerated by the gravitational slingshot effect to very high speeds, occasionally glitching.

With an integration time step of 2, over a time of 400 the interaction of two galaxies each with 240 stars took $\sim$5 minutes to simulate on a modern mid-range laptop. Within an hour it is possible to simulate the interaction of two galaxies, each with over 3000 stars (e.g. figure (1), this took 55 minutes to simulate and contains 7560 stars in total).

I have identified the "rate limiting steps" in my simulation routine - this is the process of identifying the position of the two galaxies at every integration time step in order evaluate the force on the stars. This was done by passing the the *current* time, the *full* time array and the galaxy position array in to the ODE derivative functions, `ODE_double_galaxy`. The time array was then searched for the current time, the index of which was then used to find the current galaxy position - this was found to be much faster than passing an increasing index into the derivative function although could possible be optimised further still - one possibility would be to completely rearrange the code and, instead of trying to pass the pre-determined galaxy positions into the star integration routine, run N *three-body* simulations (one non-interacting star and two massive galactic masses). Secondly, the need to plot every single frame of the interaction in order to form a smooth animation (done automatically by the animation function). Finally, the process of saving plots as `.png` and movies as `.mp4` files took up to a minute so this was made optional, as a boolean `True`/`False` value, in the `#PARAMETRES` section.

5 Results and Discussions

5.1 Single Galaxy Models

Three simple galactic models have been developed for testing and are displayed in figures (4), (5) and (6). Respectively, they are: black hole with concentric rings of stars, dark matter halo with concentric rings of stars and black hole surrounded by stars randomly perturbed off a circular orbit. In this report, the black hole and dark matter

halo galaxies with stars arranged in a ring formation will be used, although less realistic than the random model, it is easier to see patterns created from distorted rings after close encounters with another such galaxy.

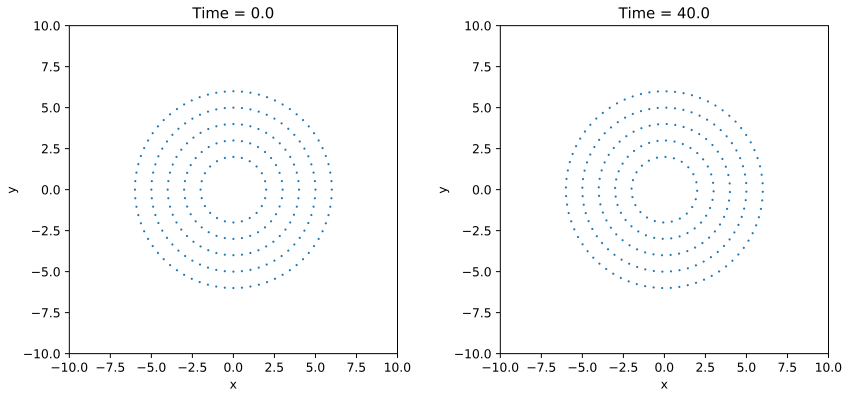


Figure 4: Plots at t=0 and t=40 for a black hole galaxy formed of concentric rings of stars at 2, 3, 4, 5 and 6. The galaxy simulates well and retains its structure even over long times. Movie file: `blackhole.mp4`

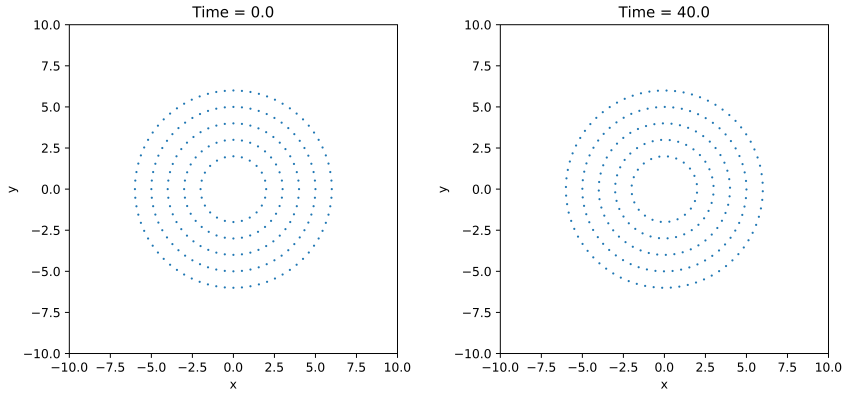


Figure 5: Plots at t=0 and t=40 for a dark matter halo galaxy (halo radius = 6) formed of concentric rings of stars at 2, 3, 4, 5 and 6. It is essentially the same as the galaxy plotted in figure (4) except all stars orbit with the same time period so the galaxy stars rotate like a rigid disk. The galaxy simulates well and retains its structure even over long times. Movie file: `darkmatterhalo.mp4`

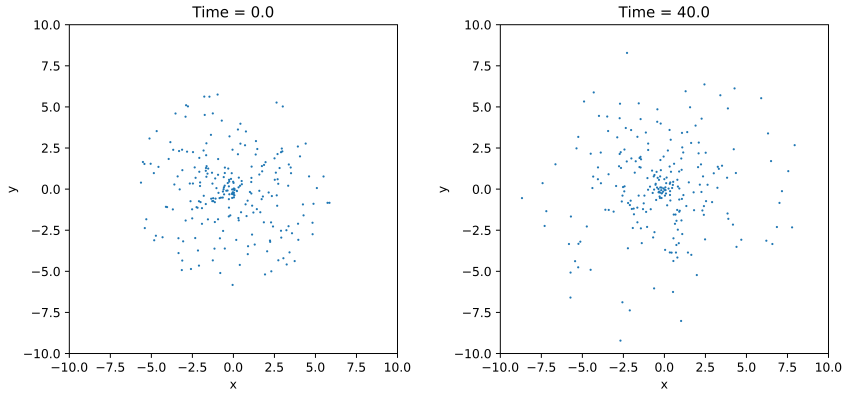


Figure 6: Plots at t=0 and t=40 for a black hole galaxy formed of 240 stars randomly perturbed off circular orbits by a small angle. It starts off contained within a radius of 6 but the perturbed orbits cause it to diffuse slightly as time progresses, at t = 40 the galaxy has reached a steady state. The galaxy simulates well and retains its structure even over long times. Movie file: `random.mp4`

5.2 Galactic tail formation - Black hole model test

A black hole galaxy interaction was simulated with the galaxies initially 40 units apart and passing with a closest approach distance of 10 units. Each galaxy contains 432 stars in 9 layers from radius 2 to radius 6 units and are rotating clockwise. Figure (7) plots the resulting interaction at six successive times and clearly demonstrates the formation of tidal tails and bridges. The bridges quickly diffuse whilst the tails escape to infinity.

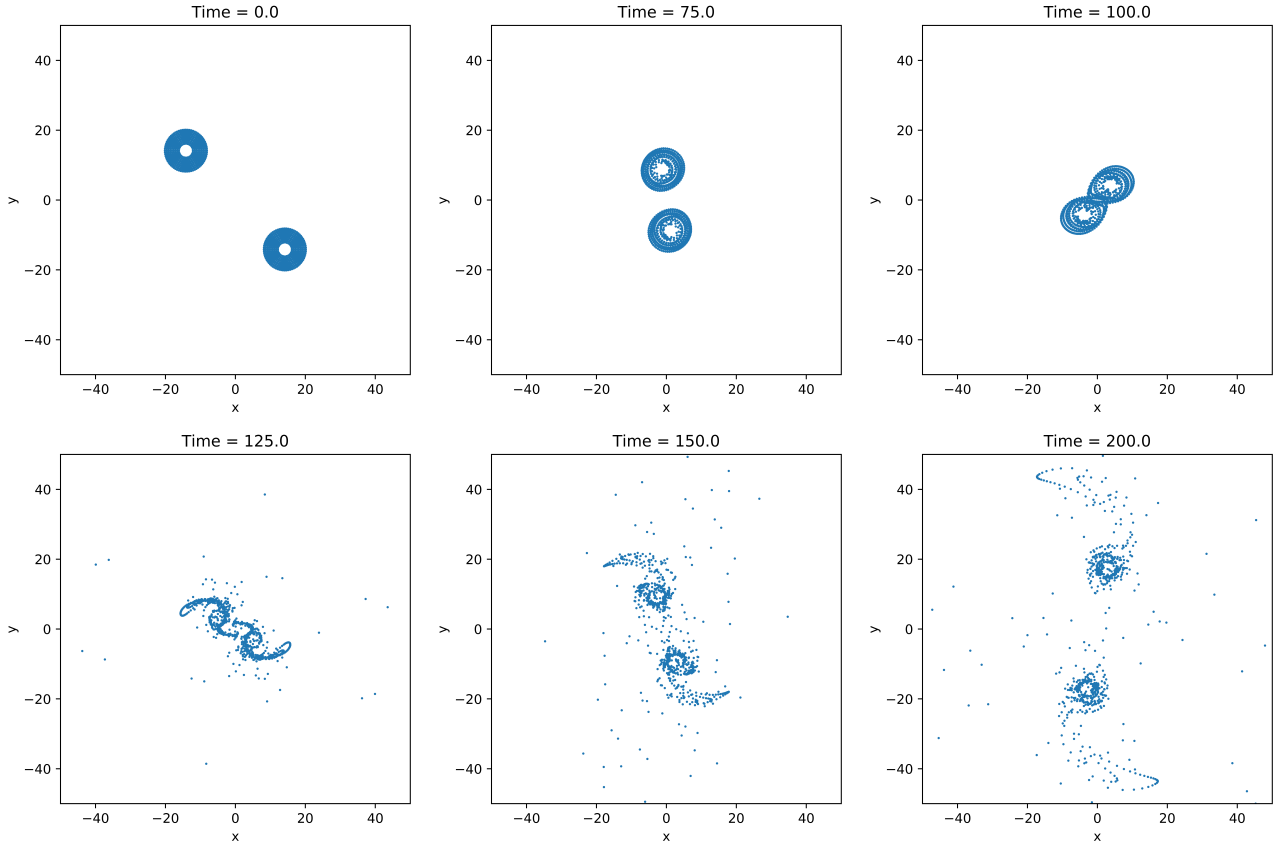


Figure 7: Figure showing the interaction of two black hole galaxies passing with a closest approach distance of 10 units. As the galaxies move closer together, their structure begins to distort (t=100) as stars are perturbed off their initially circular orbit due to the force of the second galaxy. As they reach their closest approach point (t∼125) a shower of stars is emitted in all directions - this is due to stars approaching very close to the opposite galaxy's black hole and being accelerated to high velocities by the gravitational slingshot effect. As the galaxies proceed to move past one another, large arms of stars in the outer layers are thrown away from the centre of mass of the two galaxies. t = 150 also clearly demonstrates secondary arms bridging the gap between the two galaxies which rapidly disperse. At later times the tidal tails have moved far from their host galaxy leaving only the core rings of stars around the galaxies which are not affected significantly by the encounter. It is observed (see movie file) that some stars flung off their initial galaxy are caught and remain bound to the other galaxy, suggesting that interacting galaxies can exchange matter in this way. Movie file: `BHtailformation.mp4`

5.3 Unequal galaxy masses

In this section the effect, on the creation of tidal tails, of the galaxies having non-equal masses is investigated. Figure (8) show the interaction of two galaxies with masses M1 = 1 and M2 = 0.25, 0.5, 1 & 2. For large mass differences there is significant asymmetry in the tidal tails produced from each galaxy, with those connected to the lighter galaxy being much larger and faster moving. It is also observed (as expected) that it is the ratio of the masses of the two galaxies that determines the structure of the tails, *not* the absolute masses.

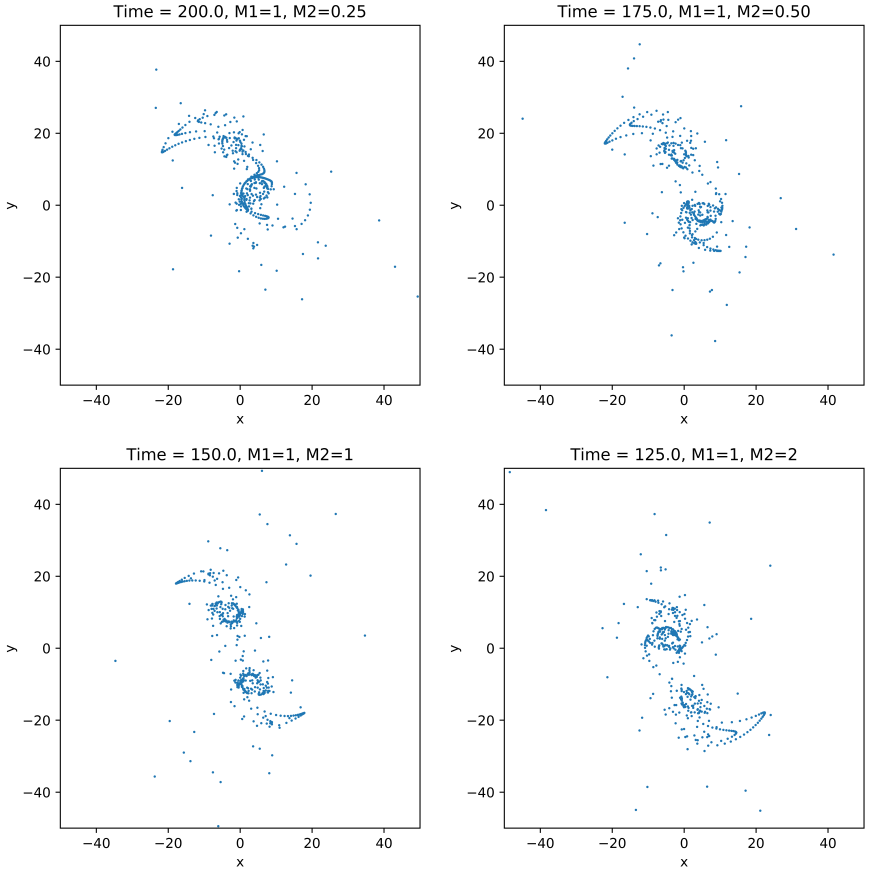


Figure 8: In the figure, M2 refers to the mass of the upper galaxy. The plots show that the lighter of the two galaxies is more susceptible to tailing, this makes sense since the stars orbiting the lighter galaxy will be less strongly bound and more easily perturbed by a nearby heavy galaxy. Also, note the almost perfect symmetry between the M2 = 0.5 plot and the M2 = 2 plot confirming it is not the absolute mass which dictates the form of the tails but the ratio of the masses M1/M2. The plots come at successively earlier times simply because the heavier the total mass of the system the faster they reach the point of closest approach having all been set off from a distance of 40 units apart at t = 0. Each plot was chosen so that the galaxies had receded to $\sim$ 20 units from closest approach so they could be compared fairly. Movie files: `massvariationquarter.mp4`, `massvariationhalf.mp4`, `massvariationequal.mp4`, `massvariationdouble.mp4`

5.4 Clockwise vs anticlockwise approach

Two identical interactions were simulated (initial galaxy separation = 40, closest approach = 10) with the sole difference being that, in the first, the stars rotated around the galaxies in the usual clockwise manner (with the galaxies also orbiting around each other clockwise) whilst in the second, the stars orbit the galaxies anticlockwise. The difference is significant due to the fact that when the stars orbit anticlockwise, the opposite galaxy has a far shorter time period with which to interact with them and, as a result, no tails are formed - contrast this with the clockwise case where large tails are formed and only the very core layers of stars remain bound to their galaxy. These results are visualised in figure (9). Whilst no tails are observed when the stars orbit anticlockwise there is still visible evidence the galaxy has been in a close encounter with another due to the significant distortion of the initially circular star orbits.

For a sufficiently close encounter, some tail-like structure can be observed in galactic encounters where stars orbit the galaxies in the opposite sense to the way the galaxies orbits each other. Even when the galaxies pass as close as 5 units the tails are not nearly as large as for clockwise encounters but rather interesting patterns can be formed, for example those shown in figure (10). Most stars (including many in the tail which largely collapses and falls back into its host galaxy) ultimately remain bound to their original galaxy.

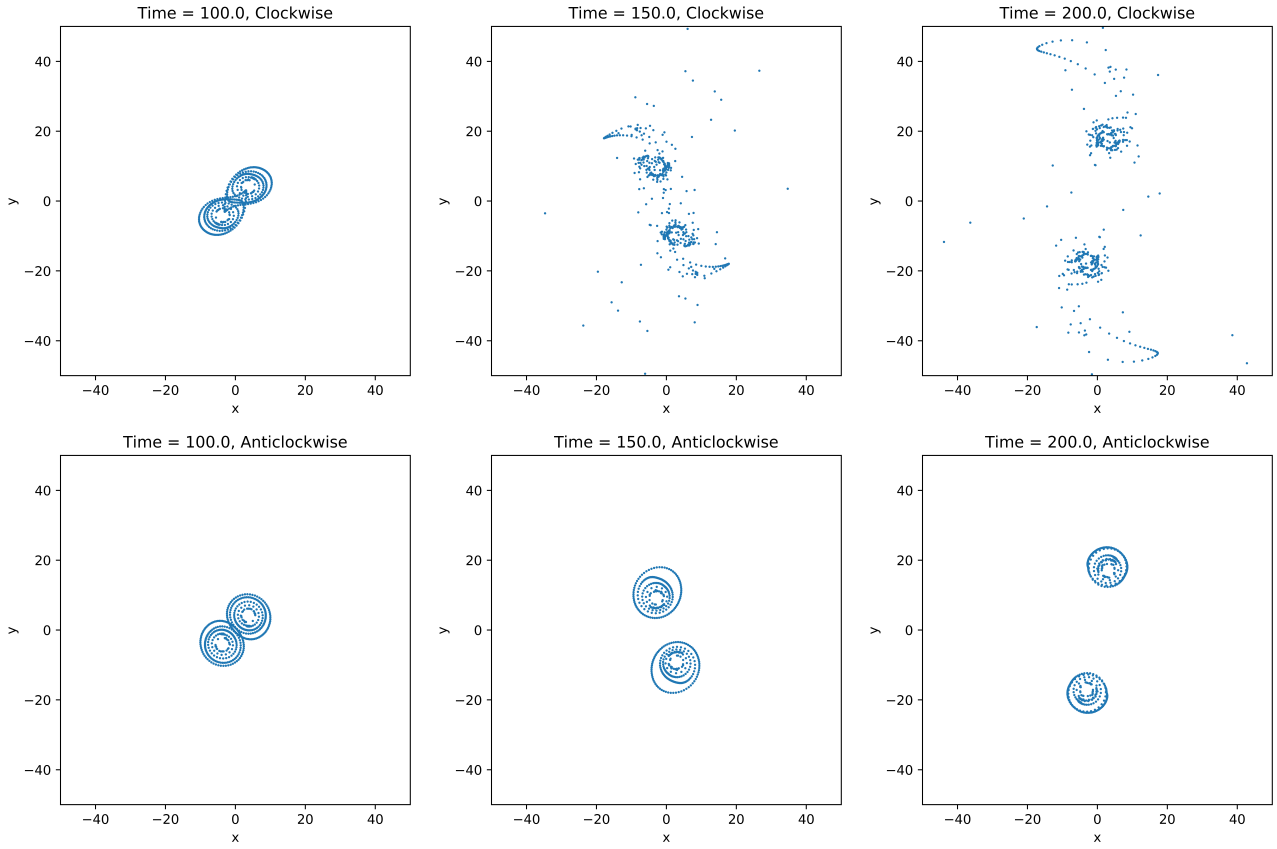


Figure 9: Figures on the top row display a galactic interaction (distance of closest approach: 10) where the stars rotate clockwise around their host galaxy whilst those on the bottom show the same interaction with the stars now orbiting their galaxy anticlockwise. No tidal tails form in the anticlockwise arrangement because the time period with which the second galaxy has to interact with the stars is so short. The outer circular ring is heavily distorted by the interaction whilst the rest aren't too affected. All stars remain bound to their original galaxy. The '(anti)clockwise' in each title refers to the rotation of the stars around their host galaxy, with the two galaxies always orbiting each other clockwise. Movie files: `clockwise.mp4`, `anticlockwise.mp4`

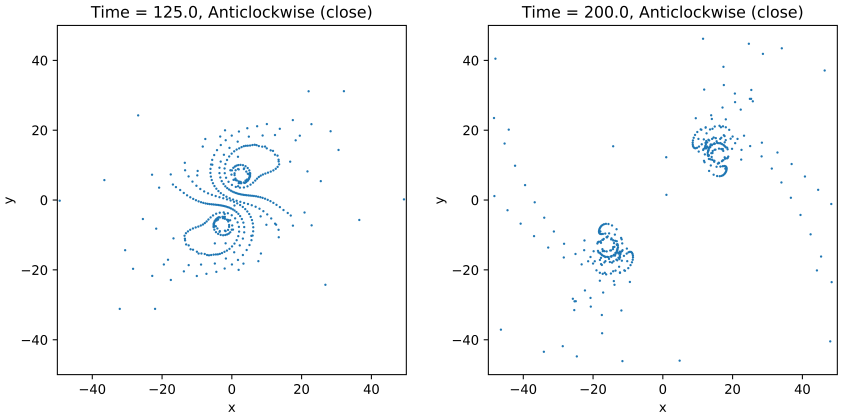


Figure 10: This figure shows the patterns produced when anticlockwise rotating galaxies pass very close to one another (here, 5 units). Despite the proximity of their encounter the tails are still small compared to those formed when the galaxies rotate clockwise and almost all stars remain bound to their original host galaxy. Movie file: `anticlockwiseclose.mp4`

5.5 Closest approach distance

In this section the distance of closest approach (R_{CA}) is varied to see what effect this would have on the formation of tidal tails. Figure (11) displays the results. As expected, those interactions where R_{CA} is small (< 5) are catastrophic and relatively few stars remain bound to their original galaxy. The figure suggests there is an optimum region for tail production, $5 \lesssim R_{CA} \lesssim 12$, above which tidal forces are too small to create tails but below which the devastation is too significant to form any distinct tail structure.

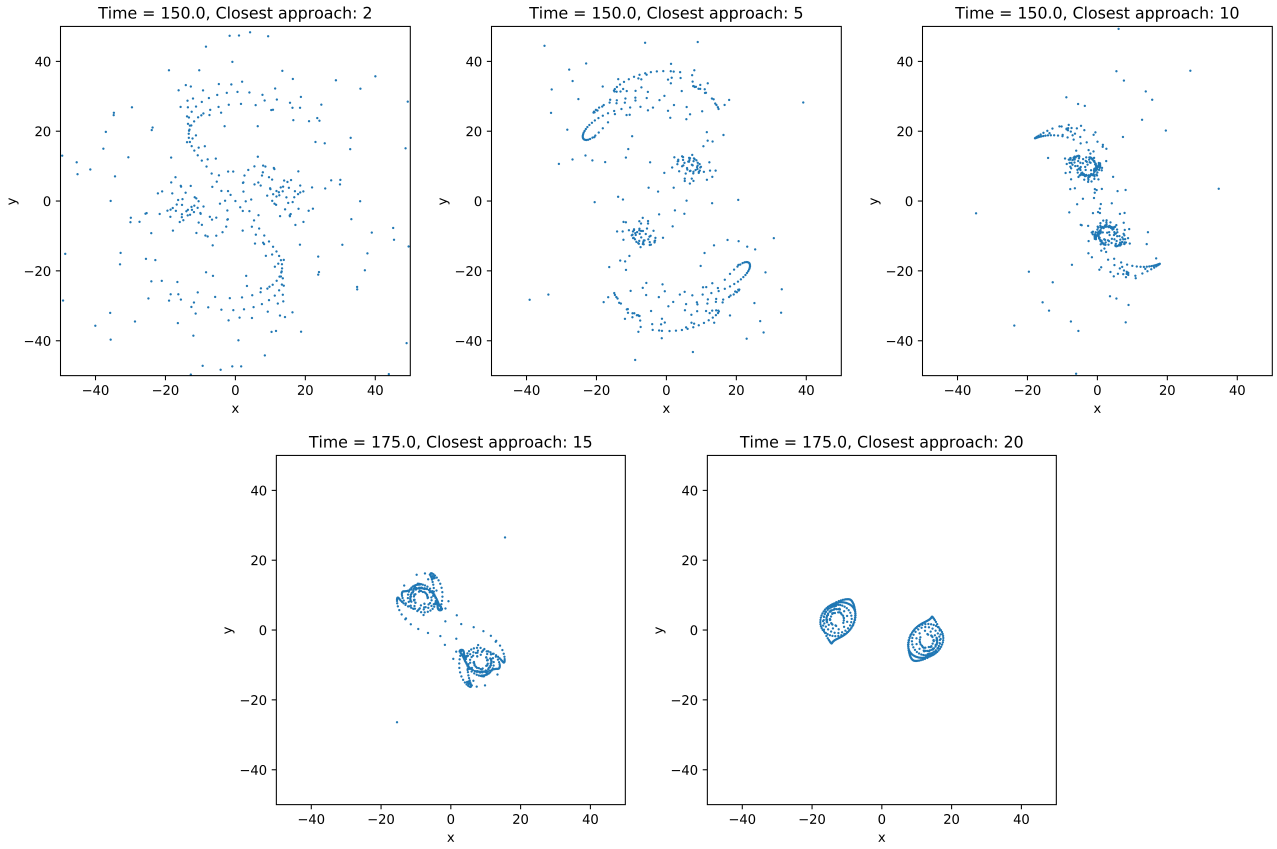


Figure 11: Snapshots of galactic interactions (taken when the galaxies are ~ 30 units apart after closest approach) where the closest approach, R_{CA}, is set to 2, 5, 10, 15 and 20 units respectively. For $R_{CA} = 2$ the interaction is quite devastating, almost no stars remain bound to their host galaxy, the galactic centres are hard to distinguish after the approach and tails are not clearly defined. Some structure is seen particularly along the line of symmetry between the galaxies where stars are brought to a near stand still and pulled equally from both galaxies on either side. For $R_{CA} = 5$ and $R_{CA} = 10$ tidal tails have clearly been formed, where, for $R_{CA} = 5$, they are faster moving and more diffuse than $R_{CA} = 10$. For $R_{CA} = 15$, a small bridge of stars is seen between the two galaxies and what would be small tails can just about be distinguished, however, the tails don't have sufficient energy to escape their host galaxy and grow no larger than this (see movie file). Finally at $R_{CA} = 20$ the galaxies are relatively unaffected by each other. Given the radius of the outer star ring is initially 6, a peak force from the perturbing galaxy of only $6^2/(20-6)^2 = 18\%$ the force from their host galaxy is experienced - this isn't enough, even over the relatively long interaction time, to perturb their orbits significantly, let alone pull them away from their host galaxy. Movie files: `closestapproach2.mp4`, `closestapproach5.mp4`, `closestapproach10.mp4`, `closestapproach15.mp4`, `closestapproach20.mp4`

5.6 Uniform dark matter halo

The uniform dark matter halo treats the galaxy as though all its mass is stored in a non-interacting sphere of radius 6 with uniform density. The enclosed mass, required to calculate the forces on the stars, is given by :

$$M_{enc}(r) = \begin{cases} M_{gal}, & \text{if } r > r_0 \\ (\frac{r}{r_0})^3 M_{gal}, & \text{if } r < r_0 \end{cases}$$

where M_{gal} is the total mass of the dark matter halo and r_0 is the radius of the halo. This has the consequence that the circular orbit time periods for all stars within the halo are equal and the stars orbit much like a wheel. When two such galaxies collide, since there are no singularities in either, there is no scattering of stars due to the gravitational slingshot effect. This is demonstrated in figure 12 and, as such, the interaction appears less catastrophic than for the black hole galaxies but the extent of the tidal tails produced are somewhat similar. Figure (12) shows the interaction of two such galaxies ($M1 = M2 = 1$, $R_{CA} = 10$) plotted at 6 times, each galaxy is initiated with 432 stars in 9 concentric rings from radius 2 to 6 and is to be compared with its equivalent plot using the black hole galactic model, figure (7).

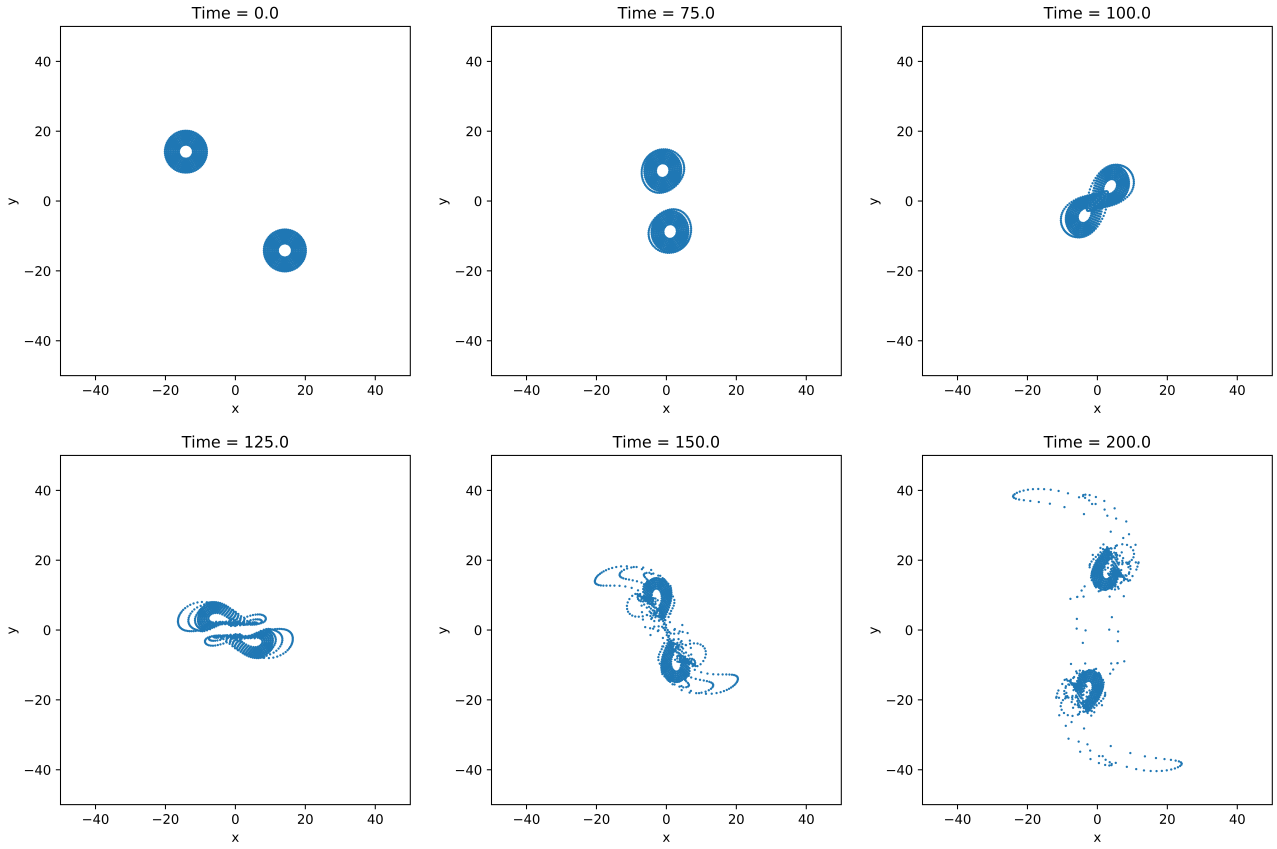


Figure 12: This figure shows the interaction of two uniform dark matter halo galaxies, with closest approach distance of 10. Since there are no black holes there is no scattering. The distortion of the galactic structure in the early stages of the interaction is more significant (e.g. t = 125) than for the black hole model and the galaxies distort in a more elastic manner. Compared to the black hole model the tidal tails are more diffuse and less knife-like in shape and there is also some bridging visible between the two galaxies at t = 200. Since there is no black hole scattering the stars, the galaxies maintain a much larger proportion of their mass compared to figure (7) after the encounter. Movie file: `DMHtailformation.mp4`

5.7 Hybrid model: Gaussian dark matter halo

In this section the galaxy is modelled with a dark matter halo but, rather than the density of the halo being uniform, it follows a Gaussian profile and extends to infinity:

$$\rho(r) = M_{gal}\frac{1}{(\sqrt{2\pi}\sigma)^3}exp(-\frac{1}{2}\frac{r^2}{\sigma^2}) \tag{3}$$

Therefore the total mass contained within a radius r can be found using the error function:

$$M_{enc}(r) = M_{gal}erf(\frac{r}{\sigma}) \tag{4}$$

Where erf is the *Gauss error function*. Since the galactic halo now (theoretically) stretches to infinity the assumption we made that they are point masses (when calculating the motion of the galaxy itself) is no longer valid, although, for sufficiently small Gaussian width, the real motion will not be very different so we can assume the approximation holds. This model is, in some sense, a hybrid of the two models tested above. For small σ (where σ = Gaussian standard deviation := "halo width") the mass is all contained very near the centre, much like a black hole. For much larger σ the mass is more diffusely distributed across a large volume, like the original uniform dark matter halo. One limit of this model would be that, for large sigma the halos significantly overlap. To avoid this an upper bound of $\sigma \leq R_{CA}$ should be set.

Figure (13) plots galactic interaction for widths 0.01, 1, 3, 5 and 10.

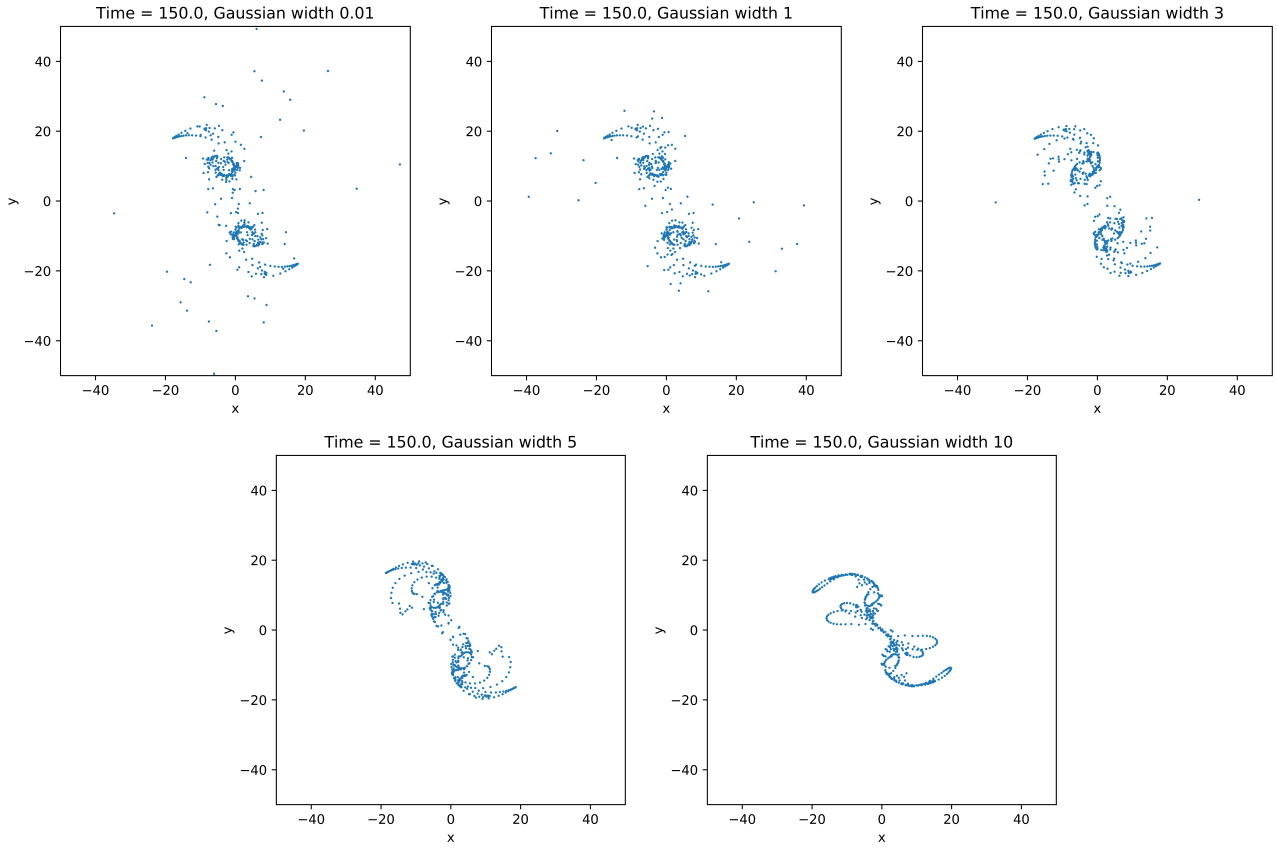


Figure 13: Five plots (all at t = 150) of interacting galaxies using the Gaussian halo model are displayed. The mass of the galaxy is entirely contained in a non-interacting dark matter halo, the density of which takes on a Gaussian profile with standard deviation σ (aka "Gaussian width"). $\sigma = 0.01$ and 1 are nearly identical (and, importantly, identical to figure (7)) since almost the entirety of the galactic mass is contained *within* the radius of even the closest star ring hence, according to Gauss's law, the dark matter halo is essentially a black hole. $\sigma = 5$ & 10 show similar features (lack of scattering, more diffuse and less 'knife-like' tails) to the uniform dark matter halo investigated in section 5.6 (see figure (12)). In these two plots, the Gaussian halo is wide enough that the density within the outer star radius is almost uniform however this density will be much lower for $\sigma = 10$ meaning the stars are more weakly bound explaining the stronger distortions. Finally, $\sigma = 3$ is a true hybrid, it shows characteristics of both the black hole model and the uniform dark matter halo. Movie files: `gaussian001.mp4`, `gaussian1.mp4`, `gaussian3.mp4`, `gaussian5.mp4` & `gaussian10.mp4`

6 Conclusion

Tidal tails and bridges were successfully observed by simulating close encounters of galaxies modelled in three different ways and, with a variety of different starting conditions. It was observed that the lighter of the two galaxies produced the largest and fastest moving tidal tails since the stars were most weakly bound. Galactic encounters where the rotation of the stars around the galaxy was in the opposite sense to the rotation of the galaxies around each other showed very little tailing since the time period for interaction with the other galaxy was so much shorter. It was also concluded that (for galaxies of mass 1) there is a range of closest approach distances for which tidal tail production is optimised, this zone is given by: $5 \lesssim R_{CA} \lesssim 12$, above which tidal forces aren't large enough to pull the galaxy apart and below which the encounter is catastrophic and little tail structure can be distinguished. A uniform dark matter halo model was tested. Since there are no longer any singularities, star scattering was reduced to zero. The model produced more diffuse tidal tails and the galaxies were distorted more significantly but ultimately retained a larger proportion of their original stars. Final a hybrid Gaussian dark matter halo was tested and showed good results. By increasing the width, σ, the interaction smoothly transitioned from the black hole model to the uniform dark matter halo model, where $\sigma = 3$ convincingly showed hybrid properties of both regimes. I would propose the variable σ could be optimised to most closely match the results seen in real galactic collisions.

Word count: 2926

References

[1] Toomre and Toomre. “Galactic bridges and tails”, The Astrophysical Journal 178 (1972): 623-666

[2] Naeye, R. “Shot in the Dark’ Star Explosion Stuns Astronomers”, NASA, Goddard Flight Center (2007)

[3] Barnes, J & Hernquist, L. “Formation of dwarf galaxies in tidal tails”, Institute of Astronomy Hawaii (1992)

[4] Green, D. IB Classical Dynamics Notes, The Cavendish Laboratory (2017)

[5] Buscher, D. Part II Computational Physics Project, The Cavendish Laboratory (2018)

7 Appendices

7.1 Code Listing

```
import numpy as np
import numpy.random as rand
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from numpy import cos, sin, pi, sqrt
from scipy.integrate import odeint
from scipy.special import erf
import time
import datetime
import os



#CODE INSTRUCTIONS

#Set the parameters below as required. If you wish to save a .mp4 movie and .png plots change plotfigures = savemovie to True (ensure the subdirectories 'Figures/' & 'Movies/' are already made in the directory this programme is saved in).
#Scroll to the bottom where there are calls to seven functions, each simulating a different thing along with instructions about input variables.









#PARAMETRES (set as required)

#physical parameters
G = 1
M = 1 #single galaxy mass
M1 = 1 #double galaxies masses 1 and 2. M1/M2 must be in the range 0.2-->5
M2 = 1
halo_radius = 6 #radius of uniorm density dark matter halo
sigma = 3 #gaussian halo standard deviation (width)
clockwise = True #true = clockwise rotation of galaxies, false = anticlockwise


#Animation/plotting parameters
xy_max = 50 #+/- plotting range
N = 500 #no. integration time steps
animation_length = 10 #movie length (in seconds) - scale animation time step so movie is this long.
t_max = 500
t_selection = [0,50,100,150,200,300,400,499] #time's you wish to plot at (must be integer and < t_max)

#If plotfigures = savemovies = True, the code will save an .mp4 movie and .png plots of the simulation into the Movies/ & Figures/ subdirectories which MUST already exist in the same parent directory as this code is saved.
current_directory = os.path.dirname(__file__)
figure_directory = os.path.join(current_directory, 'Figures/')
movie_directory = os.path.join(current_directory, 'Movies/')
plotfigures = savemovie = False #If False, only an animation will show, no movie file/plots will be saved permanently. options: False/True




```

```
52
53
54  #SIMULATION FUNCTIONS
55  #These are the 5 function types which carry out the integration: ODE, integrate, SIV, GetData &
        animate. They are explained below.
56
57  #ODEs: Returns the derivative to be integrated.
58  def ODE_single_galaxy(r,t):
59      return [r[2],r[3],-G*M*r[0]/(r[0]**2+r[1]**2)**(3/2),-G*M*r[1]/(r[0]**2+r[1]**2)**(3/2)]
60
61  def ODE_single_galaxy_halo(r,t):
62      Mass = (M*(sqrt(r[0]**2+r[1]**2)/halo_radius)**3 if (sqrt(r[0]**2+r[1]**2)/halo_radius) < 1
            else M)
63      return [r[2],r[3],-G*Mass*r[0]/(r[0]**2+r[1]**2)**(3/2),-G*Mass*r[1]/(r[0]**2+r[1]**2)**(3/2)
            ]
64
65  def ODE_galaxy_motion(R,t): # R = [ x1 , y1 , vx1 , vy1 , x2 , y2 , vx2 , vy2] for galaxy 1 and
        galaxy 2. returns [vx1, vy1, d(vx1)/dt, d(vy1)/dt, vx2, vy2, d(vx2)/dt, d(vy2)/dt] aka the
        derivatives
66      x , y = R[0]-R[4] , R[1]-R[5]
67      return [ R[2] , R[3] , -G*M2*x/(x**2+y**2)**(3/2) , -G*M2*y/(x**2+y**2)**(3/2) , R[6] , R[7]
            , G*M1*x/(x**2+y**2)**(3/2) , G*M1*y/(x**2+y**2)**(3/2) ]
68
69  def ODE_double_galaxy(r,t,R,t_array):
70      index = np.searchsorted(t_array,t)
71      cgp = [] #current galaxy position
72      for i in range(4):
73          cgp.append(R[i][index-1])
74      x1 , y1 , x2 , y2 = r[0]-cgp[0] , r[1]-cgp[1] , r[0]-cgp[2] , r[1]-cgp[3]
75      return [  r[2]  , r[3] ,  -G*M1*x1/(x1**2+y1**2)**(3/2) + -G*M2*x2/(x2**2+y2**2)**(3/2)  , -
            G*M1*y1/(x1**2+y1**2)**(3/2) + -G*M2*y2/(x2**2+y2**2)**(3/2)  ]
76
77  def ODE_double_galaxy_halo (r,t,R,t_array):
78      index = np.searchsorted(t_array,t)
79      cgp = [] #current galaxy position
80      for i in range(4):
81          cgp.append(R[i][index-1])
82      x , y  = [r[0]-cgp[0],r[0]-cgp[2]], [r[1]-cgp[1],r[1]-cgp[3]]
83      Mass = [M1,M2]
84      for i in range(len(Mass)):
85          Mass[i] = (Mass[i]*(sqrt(x[i]**2 + y[i]**2)/halo_radius)**3 if (sqrt(x[i]**2 + y[i]**2)/
                halo_radius) < 1 else Mass[i])
86      return [  r[2]  , r[3] ,  -G*Mass[0]*x[0]/(x[0]**2+y[0]**2)**(3/2) + -G*Mass[1]*x[1]/(x
            [1]**2+y[1]**2)**(3/2)  ,  -G*Mass[0]*y[0]/(x[0]**2+y[0]**2)**(3/2) + -G*Mass[1]*y[1]/(x
            [1]**2+y[1]**2)**(3/2)  ]
87
88  def ODE_double_galaxy_gaussian_halo (r,t,R,t_array):
89      index = np.searchsorted(t_array,t)
90      cgp = [] #current galaxy position
91      for i in range(4):
92          cgp.append(R[i][index-1])
93      x , y  = [r[0]-cgp[0],r[0]-cgp[2]], [r[1]-cgp[1],r[1]-cgp[3]]
94      Mass = [M1,M2]
95      for i in range(len(Mass)):
96          Mass[i] = Mass[i]*erf(sqrt(x[i]**2 + y[i]**2)/sigma)
97      return [  r[2]  , r[3] ,  -G*Mass[0]*x[0]/(x[0]**2+y[0]**2)**(3/2) + -G*Mass[1]*x[1]/(x
            [1]**2+y[1]**2)**(3/2)  ,  -G*Mass[0]*y[0]/(x[0]**2+y[0]**2)**(3/2) + -G*Mass[1]*y[1]/(x
            [1]**2+y[1]**2)**(3/2)  ]
98
99
100
101 #Functions to call the integrator, scipy.integrate.odeint, and pass them the initial conditions
        and a t_array to be integrated over.
102 def integrate_single_galaxy(t,r0):
103     r = odeint(ODE_single_galaxy,r0,t)
104     return r[:,0],r[:,1]
105
106 def integrate_single_galaxy_halo(t,r0):
107     r = odeint(ODE_single_galaxy_halo,r0,t)
108     return r[:,0],r[:,1]
109
110 def integrate_galaxy_motion(t,R0):
111     r = odeint(ODE_galaxy_motion,R0,t)
112     return r[:,0],r[:,1],r[:,4],r[:,5]
113
114 def integrate_double_galaxy(t,r0,R,t_array):
115     r = odeint(ODE_double_galaxy,r0,t,args=(R,t_array))
116     return r[:,0],r[:,1]
```

```
117
118 def integrate_double_galaxy_halo(t,r0,R,t_array):
119     r = odeint(ODE_double_galaxy_halo ,r0,t,args=(R,t_array))
120     return r[:,0],r[:,1]
121
122 def integrate_double_galaxy_gaussian_halo(t,r0,R,t_array):
123     r = odeint(ODE_double_galaxy_gaussian_halo ,r0,t,args=(R,t_array))
124     return r[:,0],r[:,1]
125
126
127 #SIV = Set Initial Values. Returns a vector (or a vector of vectors in the case of multiple stars
        ) with the starting positions and velocites of the stars or galaxies.
128 def SIV_single_galaxy_ring(r,n): #sets initial conditions for n particles evenly around a circle
        of radius r. r and n are equal length vectors to allow for multiple rings
129     r0 = []
130     for j in range(len(n)):
131         for i in range(n[j]):
132             r0.append([r[j]*cos(i*2*pi/n[j]),r[j]*sin(i*2*pi/n[j]),sqrt(G*M/r[j])*sin(i*2*pi/n[j
                ]),-sqrt(G*M/r[j])*cos(i*2*pi/n[j])])
133     return r0
134
135 def SIV_single_galaxy_halo_ring(r,n): #sets initial conditions for n particles evenly around a
        circle of radius r. r and n are equal length vectors to allow for multiple rings
136     r0 = []
137     for j in range(len(n)):
138         Mass = (M*(r[j]/halo_radius)**3 if r[j]/halo_radius < 1 else M)
139         for i in range(n[j]):
140             r0.append([r[j]*cos(i*2*pi/n[j]),r[j]*sin(i*2*pi/n[j]),sqrt(G*Mass/r[j])*sin(i*2*pi/n
                [j]),-sqrt(G*Mass/r[j])*cos(i*2*pi/n[j])])
141     return r0
142
143 def SIV_single_galaxy_random(n,max_r): #sets initial conditions for n with randomly perturbed
        circular orbits within a max starting radius max_r
144     r0 = []
145     for i in range(n):
146         R = max_r*rand.random()
147         theta = 2*pi*rand.random()
148         v = sqrt(G*M/R) + 0.1*rand.uniform(-1,1)*sqrt(G*M/R)
149         delta = rand.uniform(-pi/6,pi/6)
150         r0.append([R*cos(theta),R*sin(theta),v*sin(theta+delta),-v*cos(theta+delta)])
151     return r0
152
153 def SIV_double_galaxy_ring(r,n,R0): #sets initial conditions for n particles evenly around a
        circle of radius r about two galaxies with initial conditions defined by R0
154     k = (1 if clockwise==True else -1)
155     r0 = []
156     for j in range(len(n)): #galaxy 1 stars
157         for i in range(n[j]):
158             r0.append([r[j]*cos(i*2*pi/n[j])+R0[0],r[j]*sin(i*2*pi/n[j])+R0[1],k*sqrt(G*M1/r[j])*
                sin(i*2*pi/n[j])+R0[2],-k*sqrt(G*M1/r[j])*cos(i*2*pi/n[j])+R0[3]])
159     for j in range(len(n)): #galaxy 2 stars
160         for i in range(n[j]):
161             r0.append([r[j]*cos(i*2*pi/n[j])+R0[4],r[j]*sin(i*2*pi/n[j])+R0[5],k*sqrt(G*M2/r[j])*
                sin(i*2*pi/n[j])+R0[6],-k*sqrt(G*M2/r[j])*cos(i*2*pi/n[j])+R0[7]])
162     return r0
163
164 def SIV_double_galaxy_halo_ring(r,n,R0): #sets initial conditions for n particles evenly around a
         circle of radius r about two galaxies with initial conditions defined by R0
165     k = (1 if clockwise==True else -1)
166     r0 = []
167     for j in range(len(n)): #galaxy 1 stars
168         Mass = ((M1*(r[j]/halo_radius)**3) if r[j]/halo_radius < 1 else M)
169         for i in range(n[j]):
170             r0.append([r[j]*cos(i*2*pi/n[j])+R0[0],r[j]*sin(i*2*pi/n[j])+R0[1],k*sqrt(G*Mass/r[j
                ])*sin(i*2*pi/n[j])+R0[2],-k*sqrt(G*Mass/r[j])*cos(i*2*pi/n[j])+R0[3]])
171     for j in range(len(n)): #galaxy 2 stars
172         Mass = ((M2*(r[j]/halo_radius)**3) if r[j]/halo_radius < 1 else M)
173         for i in range(n[j]):
174             r0.append([r[j]*cos(i*2*pi/n[j])+R0[4],r[j]*sin(i*2*pi/n[j])+R0[5],k*sqrt(G*Mass/r[j
                ])*sin(i*2*pi/n[j])+R0[6],-k*sqrt(G*Mass/r[j])*cos(i*2*pi/n[j])+R0[7]])
175     return r0
176
177 def SIV_double_galaxy_gaussian_halo_ring(r,n,R0): #sets initial conditions for n particles evenly
         around a circle of radius r about two galaxies with initial conditions defined by R0
178     k = (1 if clockwise==True else -1)
179     r0 = []
180     for j in range(len(n)): #galaxy 1 stars
181         Mass = M1*erf(r[j]/sigma)
```

```
182         for i in range(n[j]):
183             r0.append([r[j]*cos(i*2*pi/n[j])+R0[0],r[j]*sin(i*2*pi/n[j])+R0[1],k*sqrt(G*Mass/r[j])*sin(i*2*pi/n[j])+R0[2],-k*sqrt(G*Mass/r[j])*cos(i*2*pi/n[j])+R0[3]])
184     for j in range(len(n)): #galaxy 2 stars
185         Mass = M1*erf(r[j]/sigma)
186         for i in range(n[j]):
187             r0.append([r[j]*cos(i*2*pi/n[j])+R0[4],r[j]*sin(i*2*pi/n[j])+R0[5],k*sqrt(G*Mass/r[j])*sin(i*2*pi/n[j])+R0[6],-k*sqrt(G*Mass/r[j])*cos(i*2*pi/n[j])+R0[7]])
188     return r0
189
190 def SIV_galaxy_motion(s_init,s_closest): #sets initial conditions for the two galaxies, starting at s_init separtion to orbit with zero total energy and pass with closest approach distance s_closest
191     h = s_init/(2*sqrt(2))
192     theta = theta_for_closest_approach(s_init,s_closest)
193     print("Starting angle (deg): %.1f" %(theta*180/pi))
194     EffMass = 2*M1*M2/(M1+M2)
195     return [-h,h,sqrt(G*EffMass/(h*2*sqrt(2)))*cos(theta),-sqrt(G*EffMass/(h*2*sqrt(2)))*sin(theta),h,-h,-sqrt(G*EffMass/(h*2*sqrt(2)))*cos(theta),sqrt(G*EffMass/(h*2*sqrt(2)))*sin(theta)]
196
197
198
199 #Given the starting locations and positions of the stars and/or galaxies and passes them, one by one, to the integrator which returns their position throughout
200 #the simulation as a vector. This function then stacks each of these in a form which can be plotted by the animator and returns these stacked vectors x, y.
201 #It all (for sanity) prints the percentage of stars completed so the user can guage how long it will take.
202 def GetData_single_galaxy(r0): #returns a tuple of tensors x,y which contain the x and y positions of all stars at all times
203     t = np.linspace(0,t_max,N)
204     x, y = integrate_single_galaxy(t,r0[0])
205     for i in range(len(r0)-1):
206         datax, datay = integrate_single_galaxy(t,r0[i+1])
207         x = np.column_stack((x,datax))
208         y = np.column_stack((y,datay))
209         print("%.1f %% " %(100*i/len(r0)), end="\r")
210     return x,y
211
212 def GetData_single_galaxy_halo(r0):#returns a tuple of tensors x,y which contain the x and y positions of all stars at all times
213     t = np.linspace(0,t_max,N)
214     x, y = integrate_single_galaxy_halo(t,r0[0])
215     for i in range(len(r0)-1):
216         datax, datay = integrate_single_galaxy_halo(t,r0[i+1])
217         x = np.column_stack((x,datax))
218         y = np.column_stack((y,datay))
219         print("%.1f %% " %(100*i/len(r0)), end="\r")
220     return x,y
221
222 def GetData_galaxy_motion(r0): #returns tuple of vectors x1, y1, x2, y2 for the positions of the galaxies
223     t = np.linspace(0,t_max,N)
224     x1, y1, x2, y2 = integrate_galaxy_motion(t,r0)
225     x = np.column_stack((x1,x2))
226     y = np.column_stack((y1,y2))
227     return x1, y1, x2, y2
228
229 def GetData_double_galaxy(r0,R_galaxies):#returns a tuple of tensors x,y which contain the x and y positions of all stars at all times
230     t = np.linspace(0,t_max,N)
231     x, y = integrate_double_galaxy(t,r0[0],R_galaxies,t)
232     for i in range(len(r0)-1):
233         datax, datay = integrate_double_galaxy(t,r0[i+1],R_galaxies,t)
234         x = np.column_stack((x,datax))
235         y = np.column_stack((y,datay))
236         print("%.1f %% " %(100*i/len(r0)), end="\r")
237     return x,y
238
239 def GetData_double_galaxy_halo(r0,R_galaxies):#returns a tuple of tensors x,y which contain the x and y positions of all stars at all times
240     t = np.linspace(0,t_max,N)
241     x, y = integrate_double_galaxy_halo(t,r0[0],R_galaxies,t)
242     for i in range(len(r0)-1):
243         datax, datay = integrate_double_galaxy_halo(t,r0[i+1],R_galaxies,t)
244         x = np.column_stack((x,datax))
245         y = np.column_stack((y,datay))
```

```
246             print("%.1f %% " %(100*i/len(r0)), end="\r")
247         return x,y
248 
249     def GetData_double_galaxy_gaussian_halo(r0,R_galaxies):#returns a tuple of tensors x,y which
        contain the x and y positions of all stars at all times
250         t = np.linspace(0,t_max,N)
251         x, y = integrate_double_galaxy_gaussian_halo(t,r0[0],R_galaxies,t)
252         for i in range(len(r0)-1):
253             datax, datay = integrate_double_galaxy_gaussian_halo(t,r0[i+1],R_galaxies,t)
254             x = np.column_stack((x,datax))
255             y = np.column_stack((y,datay))
256             print("%.1f %% " %(100*i/len(r0)), end="\r")
257         return x,y
258 
259 
260 
261 
262     #These animation functions tie everything together so the systems can be plotted easily in a
        single line call (See bottom).
263     #They take only the most logical  inputs: radii, number of stars, initial galaxy separation,
        closest approach etc.
264     def animation_single_galaxy_ring(Radii,Numbers):
265         R0 = SIV_single_galaxy_ring(Radii,Numbers)
266         datax, datay = GetData_single_galaxy(R0)
267         animate(datax,datay)
268         plot_figures(datax,datay)
269         return
270 
271     def animation_single_galaxy_random(r_max,n_total):
272         R0 = SIV_single_galaxy_random(n_total,r_max)
273         datax, datay = GetData_single_galaxy(R0)
274         animate(datax,datay)
275         plot_figures(datax,datay)
276         return
277 
278     def animation_single_galaxy_halo_ring(Radii,Numbers):
279         R0 = SIV_single_galaxy_halo_ring(Radii,Numbers)
280         datax, datay = GetData_single_galaxy_halo(R0)
281         animate(datax,datay)
282         plot_figures(datax,datay)
283         return
284 
285     def animation_galaxy_motion(s_init,s_closest):
286         R0 = SIV_galaxy_motion(s_init,s_closest)
287         R_galaxies  = GetData_galaxy_motion(R0)
288         datax = np.column_stack((R_galaxies[0],R_galaxies[2]))
289         datay = np.column_stack((R_galaxies[1],R_galaxies[3]))
290         animate(datax,datay)
291         plot_figures(datax,datay)
292         return
293 
294     def animation_double_galaxy(Radii,Numbers,s_init,s_closest):
295         R0_galaxies = SIV_galaxy_motion(s_init,s_closest)
296         R_galaxies = GetData_galaxy_motion(R0_galaxies)
297         R0_stars = SIV_double_galaxy_ring(Radii,Numbers,R0_galaxies)
298         datax, datay = GetData_double_galaxy(R0_stars,R_galaxies)
299         animate(datax,datay)
300         plot_figures(datax,datay)
301         return
302 
303     def animation_double_galaxy_halo(Radii,Numbers,s_init,s_closest):
304         R0_galaxies = SIV_galaxy_motion(s_init,s_closest)
305         R_galaxies = GetData_galaxy_motion(R0_galaxies)
306         R0_stars = SIV_double_galaxy_halo_ring(Radii,Numbers,R0_galaxies)
307         datax, datay = GetData_double_galaxy_halo(R0_stars,R_galaxies)
308         animate(datax,datay)
309         plot_figures(datax,datay)
310         return
311 
312     def animation_double_galaxy_gaussian_halo(Radii,Numbers,s_init,s_closest):
313         R0_galaxies = SIV_galaxy_motion(s_init,s_closest)
314         R_galaxies = GetData_galaxy_motion(R0_galaxies)
315         R0_stars = SIV_double_galaxy_gaussian_halo_ring(Radii,Numbers,R0_galaxies)
316         datax, datay = GetData_double_galaxy_gaussian_halo(R0_stars,R_galaxies)
317         animate(datax,datay)
318         plot_figures(datax,datay)
319         return
320 
```

```
321
322
323
324
325
326 #OTHER FUNCTIONS
327
328 #Returns an angle the two galaxies (initally a distance s_init apart along the y = -x line) must
      be set of at from the horizontal in order to pass with closest approach s_closest (total
      energy = 0)
329 def theta_for_closest_approach(s_init,s_closest):
330     Theta = [pi*i/180 for i in range(46)]
331     h = s_init/(2*sqrt(2))
332     S = []
333     for theta in Theta:
334         EffMass = 2*M1*M2/(M1+M2)
335         initial_galaxy_data = [-h,h,sqrt(G*EffMass/(h*2*sqrt(2)))*cos(theta),-sqrt(G*EffMass/(h
              *2*sqrt(2)))*sin(theta),h,-h,-sqrt(G*EffMass/(h*2*sqrt(2)))*cos(theta),sqrt(G*EffMass
              /(h*2*sqrt(2)))*sin(theta)]
336         R_gal  = GetData_galaxy_motion(initial_galaxy_data)
337         galaxy_separation = sqrt((R_gal[0]-R_gal[2])**2+(R_gal[1]-R_gal[3])**2)
338         S.append(abs(s_closest - np.ndarray.min(galaxy_separation)))
339     i = S.index(min(S))
340     return pi*i/180
341
342 #The next two functions animates, and saves, the simulation. Takes x and y as vectors containing
      the positions of all the particles. eg:  x = [[x1 x2 ... xn]_t=1  [x1 x2 ... xn]_t=2 ... [x1
      x2 ... xn]_t=n]
343 def update_func(num,datax,datay,line):
344     line.set_data(datax[num,:],datay[num,:])
345     return line,
346
347 def animate(datax,datay):
348     fig1 = plt.figure()
349     line, = plt.plot([],[],".", markersize = 1.5)
350     plt.xlim(-xy_max,xy_max)
351     plt.ylim(-xy_max,xy_max)
352     plt.xlabel('x')
353     plt.ylabel('y')
354     plt.gca().set_aspect('equal')
355     anim = animation.FuncAnimation(fig1, update_func, datax.shape[0], fargs=(datax,datay,line),
          interval=1000*animation_length/N, blit=True)
356     print('Computing time = %.1f s' %(time.time() - T0))
357     plt.show()
358     if savemovie != False:
359         anim.save(movie_directory + '%s.mp4' %datetime.datetime.now().strftime("%I%M%p%B%d"))
360
361 #Plots and saves the simulation at all the times given in the t_selection variable at the top.
362 def plot_figures(datax,datay): #t_selection must be even in length (2,4,6...)
363     if plotfigures != False:
364         for t in t_selection:
365             plt.figure()
366             num = int(round((t/t_max)*N))
367             plt.plot(datax[num,:],datay[num,:],".", markersize = 1.5)
368             plt.xlim(-xy_max,xy_max)
369             plt.ylim(-xy_max,xy_max)
370             plt.title('Time = %.1f' %t)
371             plt.gca().set_aspect('equal')
372             plt.xlabel("x")
373             plt.ylabel("y")
374             plt.savefig(figure_directory + '%s_%g.png' %(datetime.datetime.now().strftime("%I%M%p
                  %B%d"),t), bbox_inches='tight',dpi=500)
375
376
377
378
379
380
381 #CALL LINES
382
383 #Call these one at a time (by unhashing) with the lines below depending on what you would like to
       simulate. There are 6 in total:
384
385 #single_galaxy_ring. A single galaxy of concentric rings of stars around a black hole. Input
      parametres: ([r0,r1,r2...],[n0,n1,n2...])
386 #single_galaxy_halo_ring. A single galaxy of concentric rings of stars around a uniform density
      dark matter halo. Input parametres: ([r0,r1,r2...],[n0,n1,n2...])
387 #single_galaxy_random: A single galaxy with n_total stars randomly perturbed off circular orbits
```

```
    within a radius r_max. Input parametres: (r_max,n_total)
#galaxy_motion: The motion of a two galaxies (no stars) in parabolic orbit. Input parametres: (starting_seperation,closest_approach)
#double_galaxy: Double galaxy interaction with concentric star rings, using the black hole model. Input parametres: ([r0,r1,r2...],[n0,n1,n2...],starting_seperation,closest_approach,clockwise=True/False) false if galaxies rotate anticlockwise
#double_galaxy_halo: Double galaxy interaction with concentric star rings, using the uniform dark matter halo model. Input parametres: ([r0,r1,r2...],[n0,n1,n2...],starting_seperation,closest_approach,clockwise=True/False)
#double_galaxy_gaussian_halo: Double galaxy interaction with concentric star rings, using the gaussian dark matter halo model. Input parametres: ([r0,r1,r2...],[n0,n1,n2...],starting_seperation,closest_approach,clockwise=True/False)


# animation_double_galaxy_halo([2,3,4,5,6],[24,36,48,60,72],40,10)
# animation_double_galaxy_gaussian_halo([2,3,4,5,6],[24,36,48,60,72],40,10)



T0 = time.time()
# animation_single_galaxy_ring([2,3,4,5,6],[24,36,48,60,72])
# animation_single_galaxy_halo_ring([2,3,4,5,6],[24,36,48,60,72])
# animation_single_galaxy_random(6,240)
# animation_galaxy_motion(40,10)
animation_double_galaxy([2,3,4,5,6],[24,36,48,60,72],40,10)
# animation_double_galaxy_halo([2,3,4,5,6],[24,36,48,60,72],40,10)
# animation_double_galaxy_gaussian_halo([2,3,4,5,6],[24,36,48,60,72],40,10)
```