An Investigation of Tidal Tails

BGN: 7022R

March 2021

Abstract

The aim of this project is to investigate the formation of tidal tails. Tidal tails are elongated regions of high stellar and interstellar medium (ISM) density that are formed when galaxies strongly interact gravitationally, for example after a close approach between galaxies, which is the situation studied in this project. This project aimed to simulate the formation of tidal tails using using the Python programming language, as well as the associated Numpy, Scipy and Numba libraries. The effect of the direction & radius of asteroid orbit, distance of closest approach and mass of the incoming galaxy were varied and their effects on the formation of tidal tails were measured. Tidal tails were seen to form exclusively from asteroids travelling in the same orbital direction as the approaching galaxies, and the majority of the matter was seen to originate from the outermost orbits around the galaxies, except at very close inter-galactic approaches. The amount of tidal tail material was seen to be maximal at an intermediate inter-galactic close approach distance, with minimal tidal tail formation at very large and small approach distances. Larger perturbing galaxy masses were seen to reduce the formation of tidal tails at intermediate approach distances.

1 Introduction

The approach to this problem was to attempt to solve an Initial Value Problem (IVP) that would simulate tidal tail formation. The system was set up such that a perturbing galaxy makes a close approach to a second galaxy which has many asteroids orbiting it. The resulting path of these asteroids is then calculated for a range of initial conditions. Specifically, the response of the system to changes in the mass of the perturbing galaxy, the distance of closest approach and the direction & radius of orbit of the asteroids with respect to that of the perturbing galaxy. This report outlines the computational background of the research, the implementation of the simulation that was developed and the results and conclusions drawn from the computation that was performed.

2 Methods

2.1 Background

The IVP to be solved is of the form shown in equations 1 and 2. Note that i is an arbitrary index of the objects' position and velocity components. In this project the asteroids were treated as massless i.e. they did not feature in any objects' equation of motion; thus the summation over 'large' masses in equation 2. All of the simulations were performed in 2D, as although expanding the problem to 3D would be simple computationally, the number of extra possible initial conditions to vary would be beyond the scope of this project. Further investigation into the perturbation of 'disc-like' galaxies by objects that are not in the same galactic plane could be done in the future, as most galaxies are to some extent flattened and will not necessarily collide in their own galactic plane, as is assumed by this project.

$$\begin{aligned} dy_i/dt &= f(t, y_i) \\ y_i(0) &= y_i^0 \end{aligned} \tag{1}$$

$$\begin{aligned} f(t, \underline{x}) &= \underline{v} \\ f(t, \underline{v}) &= \sum_{\text{'large' masses}} -\frac{GM}{r^3}\underline{r} \end{aligned} \tag{2}$$

The methods used to solve a particular IVP depend on the type of problem being solved, and other factors such as available computation time. In general, numerical integration methods are either *explicit* or *implicit*. Explicit methods solve an equation of the form $y_{i+1} = F(y_i)$ to calculate y_{i+1}. Implicit methods solve an equation of the form $G(y_{i+1}, y_i) = 0$ to calculate y_{i+1}.

Generally, implicit methods require more computation *per step* than explicit methods as the implicit equations are more difficult to solve. Implicit methods are often more useful when solving stiff problems, ie. those which become numerically unstable at all but very small step-sizes. Here, implicit numerical methods can sometimes provide more accurate results while performing fewer operations overall.

2.2 Implementation

The proposed Initial Value Problem was best solved using the inbuilt `integrate.solve_ivp()` function in the Scipy library. This allows an IVP to be solved using a range of inbuilt (or user-created) solvers. The source code that was written for this project can be found in appendix A. This file was subsequently imported into other scripts and its functions called to compute and plot desired solutions.

A wrapper function called `gal_solver()` was created which takes initial conditions of the objects (all but the centre galaxy), the length of time over which to solve the IVP from 0, the requested solver, the mass of the perturbing galaxy and the number of points to return the evaluated solution at. These are passed to the solver after some manipulation and the calculation galaxy 1's initial velocity to place the observer in the ZMF.

The function `saturn_function()` is passed to the solver as required in the Scipy documentation [1], which returns the value of $f(t, y_i)$ for a given y_i as in equation 2. Importantly for the performance of stiff solvers, `saturn_function()` is designed in a vectorised manner to allow multiple evaluation points in a single function call, and hence `saturn_function()` calls `integrate.solve_ivp()` with the `Vectorised` parameter set to `True`.

The function `jacobian()` returns the Jacobian matrix, as recommended in the documentation, and is also passed to `integrate.solve_ivp()`. The Jacobian matrix $f(t, j_i)$ is defined in equation 3.

$$J_{ij} = \frac{df_i}{dy_j} \tag{3}$$

The Numba package was used to accelerate the calculation of the Jacobian by defining several functions using Numba's `@jit` decorator, which performs just-in-time machine code compilation of the functions [4]. The performance benefits of this usage were not tested explicitly in this project.

In this project, the units used are arbitrary. Given that $G = 4\pi^2 \cdot AU^3 \cdot yr^{-2} \cdot M_\odot$, it is clear that the calculations performed scale arbitrarily, to any size, with larger masses and distances corresponding to longer timescales when scaled up. All analysis was performed in arbitrary units corresponding to $G = 1$ and the masses of the galaxies used are of order 1.

2.3 Testing

2.3.1 Accuracy

Testing was performed to evaluate the different solvers for use in solving this IVP. A stable two-body orbit was simulated over approximately 200 orbits, with objects' mass ratio $r_m = 0.1$, and the total energy of the system was measured throughout. The results are shown in Figure 1. This test aimed to evaluate the precision of simulations across solvers, by measuring a quantity that should remain constant at all times.

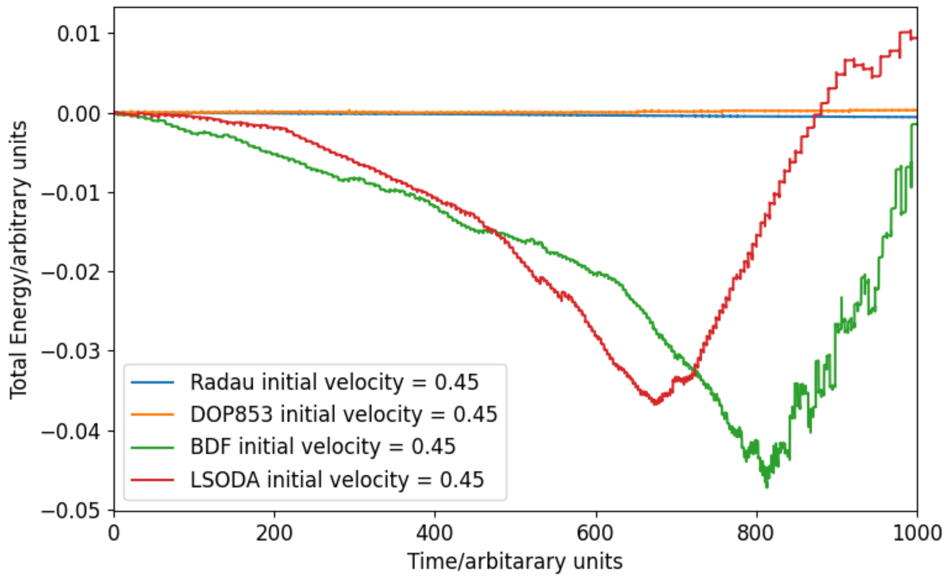

Figure 1: The energy of the two-body system over approximately 200 orbits with objects' mass ratio $r_m = 0.1$. Neither the 'RK45' or 'RK23' methods are included here as they both performed substantially poorer than the others, and so were quickly disregarded. The total energies are normalised to zero at $t = 0$.

This test was repeated across a number of initial velocities of the two bodies, and all tests performed in this way gave very similar results: the Radau and DOP853 solvers gave the most precise results over this long time period simulation. The Radau solver is an implicit Runge-Kutta method while DOP853 is an explicit Runge-Kutta method [1].

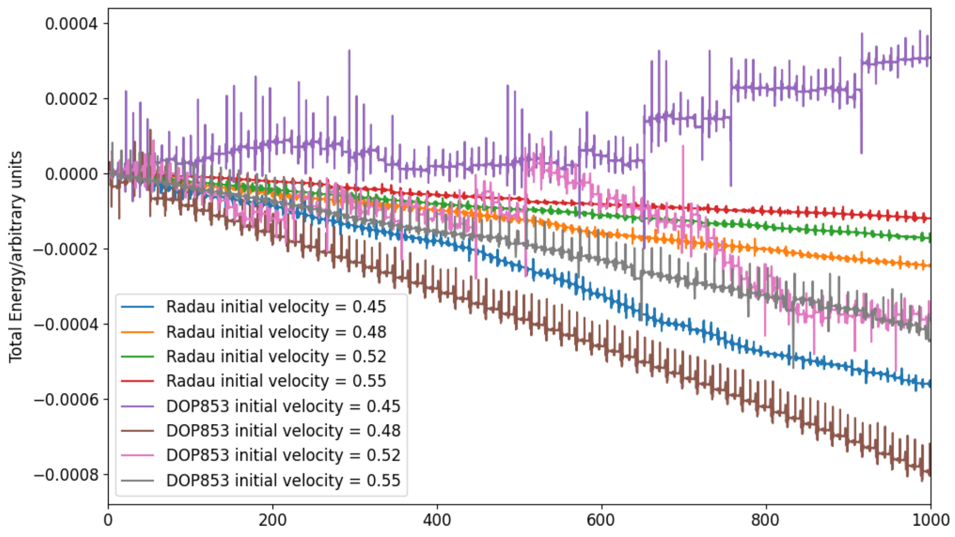

Figure 2: The energy of the two-body system over approximately 200 orbits with objects' mass ratio $r = 0.1$. Now only Radau and DOP853 are included each for a range of smaller-body initial velocities. The total energies are normalised to zero at $t = 0$.

As can be seen in Figure 2, in general DOP853 produced slightly larger fluctuations about the initial total energy over the whole time period. It also produced larger fluctuations on a per-orbit basis, as can be seen by the larger amplitude of short-timescale irregularity on the DOP853 plots. Both produced maximum total energy deviations over 200 orbits on the order of 10^{-3}, while the initial kinetic energy of each body was of order 10^{-1}. However the per-orbit energy deviation was approximately an order of magnitude smaller than this. This can be seen on figure 3.

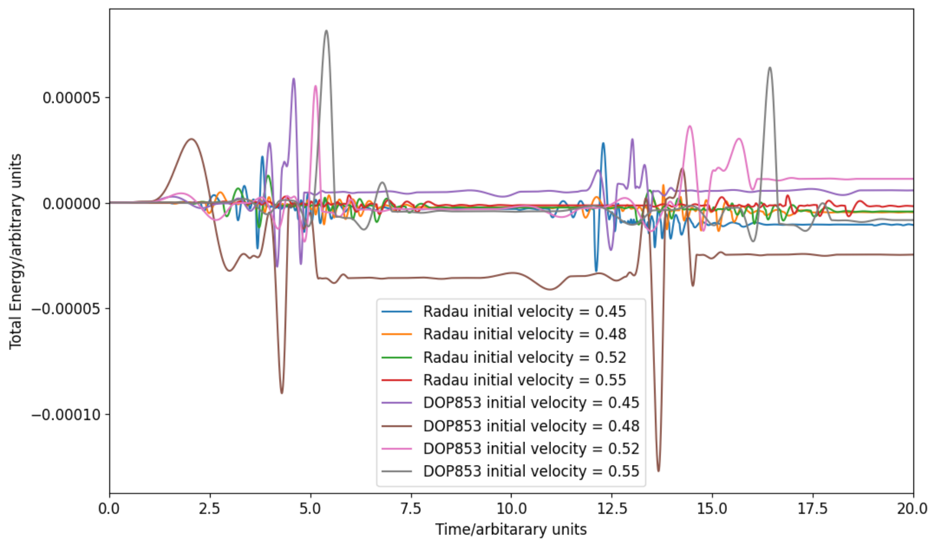

Figure 3: The energy of the two-body system over 2 orbits with objects' mass ratio $r = 0.1$ for Radau and DOP853 solvers for a range of initial smaller-body velocities over 2 orbits. The total energies are normalised to zero at $t = 0$.

2.3.2 Performance

The solvers were also tested to compare simulation times. The time that the solvers took to perform the same simulation was measured using an interval between two calls of the Python inbuilt `time.perf_counter_ns()` function [2]. The solvers performed an identical simulation to previously, this time at 20 different stable orbits. The results are shown in figure 4. DOP853 performed better than all of the other solvers being considered in terms of solve time, with average solve times a factor of 6.00 smaller than using Radau.

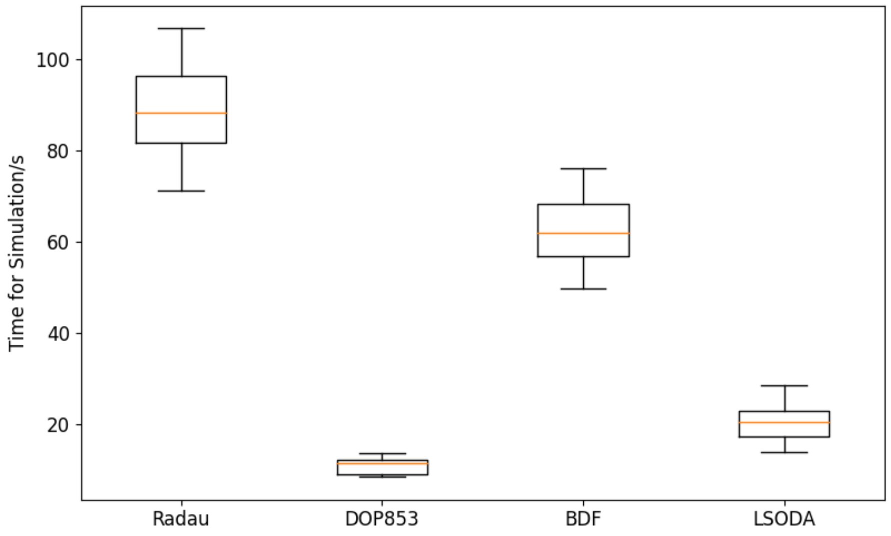

Figure 4: A box plot comparing simulation times across solvers for (approximately) a 200-orbit simulation. Data from 20 distinct sets of stable-orbit initial conditions.

Given that DOP853 performed similarly to Radau with respect to accuracy of simulation, and also performed significantly better with respect to compute time, it was chosen to undertake the computation for the project. Additionally, some problems were experienced with the Radau solver when solving certain sets of initial conditions; the simulation would 'hang' and have to be terminated manually, having already taken many orders of magnitude longer to compute the solution. This appeared to happen frequently during the simulations, and hence Radau was further disregarded despite its slightly higher solve accuracy.

When performing simulations, the nature of the IVP was not truly that of an n-body problem, as the asteroids were considered 'massless' with respect to the galaxies and each other. Thus, the paths of the galaxies were invariant with respect to the presence or initial conditions of the asteroids, and it was possible to calculate only a small number of asteroid paths in one call of the `gal_solver()` function, and then to make multiple function calls and concatenate the results. This method was seen to be more efficient, as the solvers' execution time seemed to scale faster than linearly with respect to the number of simultaneous equations ($\propto$ number of objects simulated), and hence it was more efficient to simulate small sets of several objects and merge the results after execution than to perform the whole simulation in one function call. However, this involves recalculating the paths for the galaxies for each set, and hence this implementation was not necessarily justified; the exact relationship between simulation time and number of objects per set was not studied in detail, and in general the optimum set size would depend on a multitude of factors. A side-benefit of the set approach was that it allowed feedback to be passed to the user on how long execution was likely to take during execution. The efficiency of the program would be further increased by the implementation of multithreading, which was not done in this project's code.

The program was able to simulate a 3000-asteroid simulation such as that performed to produce figure 6 in $\sim 5148s$.

3 Results and Analysis

Simulations were carried out using a larger central galaxy 'galaxy 1' of mass $m_0 = 1$, starting at the origin, which was surrounded by asteroids at random azimuthal angles. The asteroids had negligible mass with initial radii $r_{ast} \in [1, 6]$ and were on initially circular orbits. A smaller perturbing galaxy 'galaxy 2' of mass m_p began at $x_0 = 60$ units and y_0 variable, with no initial y-velocity, and x-velocity $v_x = -0.25$. With correctly chosen m_p and y_0 this gives a simple two-body parabolic orbit with a well defined distance of closest approach, the form of which can be seen in figures 5 and 6.

The tidal tails formed do not resemble the usual tail-like structure that might be expected from images of tidal tails such as those seen in images of NGC 4038-4039 [3], as can be seen in figure 6. This is likely due to the lack of viscosity in the simulation, which would be present in a real galaxy due to the relatively dense ISM in these collision regions. The effects of viscosity on this simulation are complex and beyond the scope of this project, however, we will assume that the asteroids which transfer to a close orbit around galaxy 2 would be those that form the 'tail' material.

3.1 Direction & Radius of Orbit

Figure 5 shows the paths of 200 asteroids, half of which are going clockwise and anticlockwise respectively, and plotted separately. The orbit of the galaxies is *anticlockwise*, and so it is expected that asteroids travelling in opposite directions will behave differently. As can be seen in figure 5, the asteroids travelling anticlockwise (with galaxy 2) are likely to be moved into unstable orbits; some are 'picked up' by galaxy 2 and form tidal tails. However, the clockwise-travelling asteroids are much less strongly affected, and although their orbits are somewhat perturbed they do not form tidal tails. This qualitative result was seen consistently across almost every simulation performed, with some exceptions observed at very close approaches for the asteroids with the largest orbit radii.

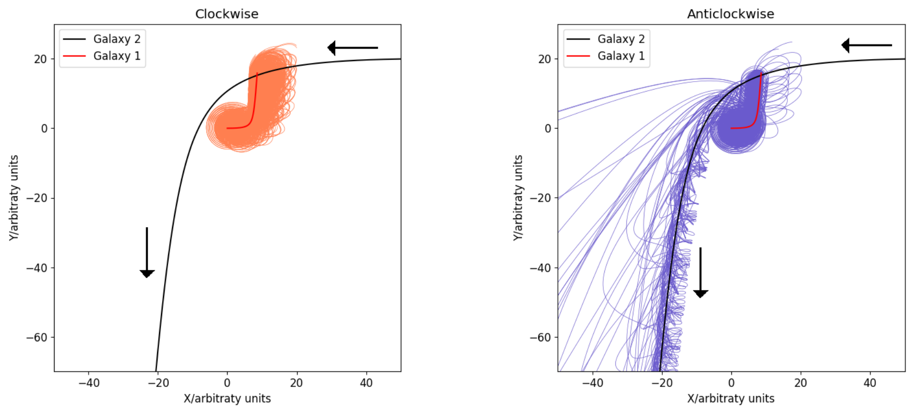

Figure 5: Plots of object paths for perturbing galaxy velocity $y_0 = 20$ and mass ratio $r_m = 0.1$, showing the paths for 200 asteroids travelling clockwise and anticlockwise respectively. Note that Galaxy 2 approaches from the right and hence 'orbits' galaxy 1 anticlockwise, as shown by the arrows. The closest approach of the galaxies was 11.6 units.

As might be expected the inner orbits are less likely to be perturbed than the outer orbits, as seen in figure 6. Figure 7 shows the asteroids' final distances from galaxy 1 as a function of their initial radii for $y_0 = 20$, $r_m = 0.1$.

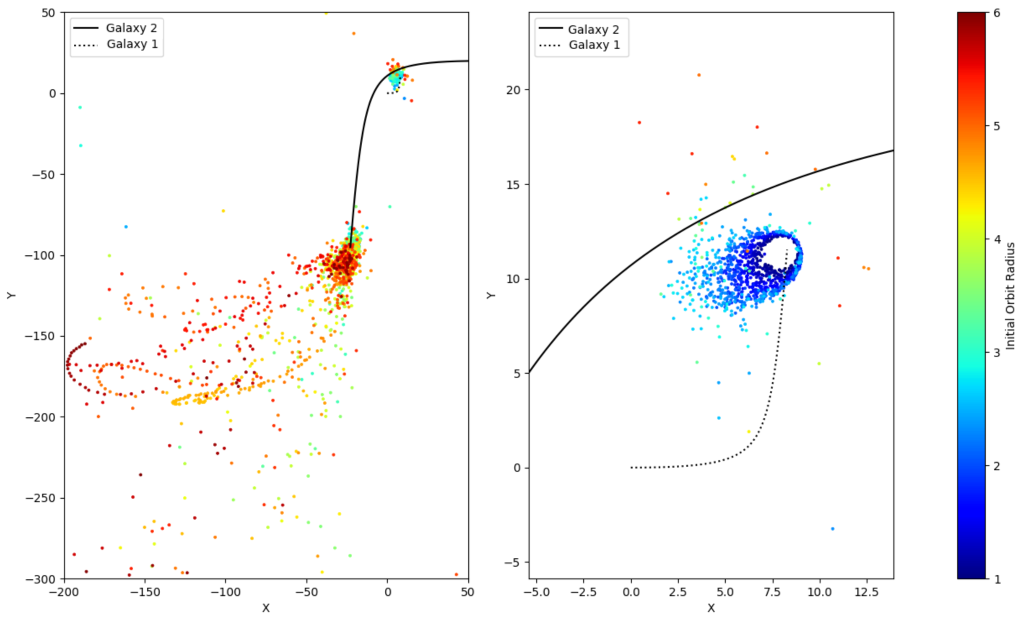

Figure 6: Scatter plot of final positions of 3000 asteroids, at two scales, coloured to show the initial radii of the asteroids. $y_0 = 20$, $r_m = 0.1$. Closest approach at 11.37 units. The inner asteroids are much more likely to be retained by galaxy 1, while the outer asteroids are more likely to be transferred, as shown by the colours in the plot.

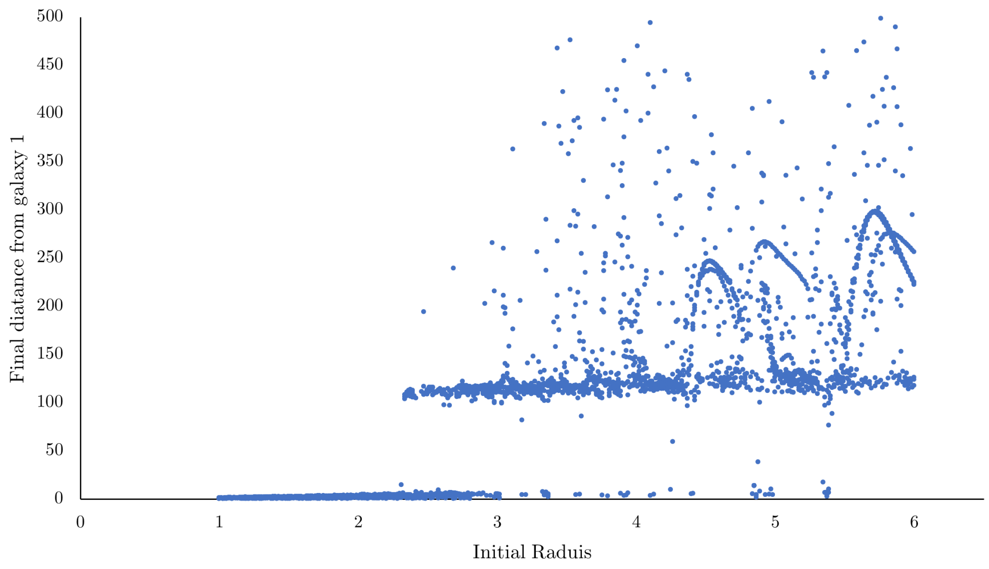

Figure 7: Plot of asteroids' initial orbital radii and final distances from galaxy 1 for 3000 asteroids. $y_0 = 20$, $r_m = 0.1$. Closest approach at 11.37 units. Two distinct groups of perturbed and unperturbed asteroids are clearly visible, and it can be seen that the larger-radius asteroids are more likely to be disturbed and form tidal tails, while inner asteroids are more likely to be retained. However, there is significant overlap between the groups between initial radii 2 - 4 units.

Clearly, other factors influence the likelihood of an individual asteroid being perturbed, such as the angular position of the asteroid in its orbit at closest approach. Indeed, at very small closest-approaches we see the breakdown of this trend, as seen in figure 8. As can be seen in figure 9, there is no clear relationship between initial orbital radii and final distances from galaxy 1 at these close approaches. The significance of results at these very close approaches is discussed in the next section.

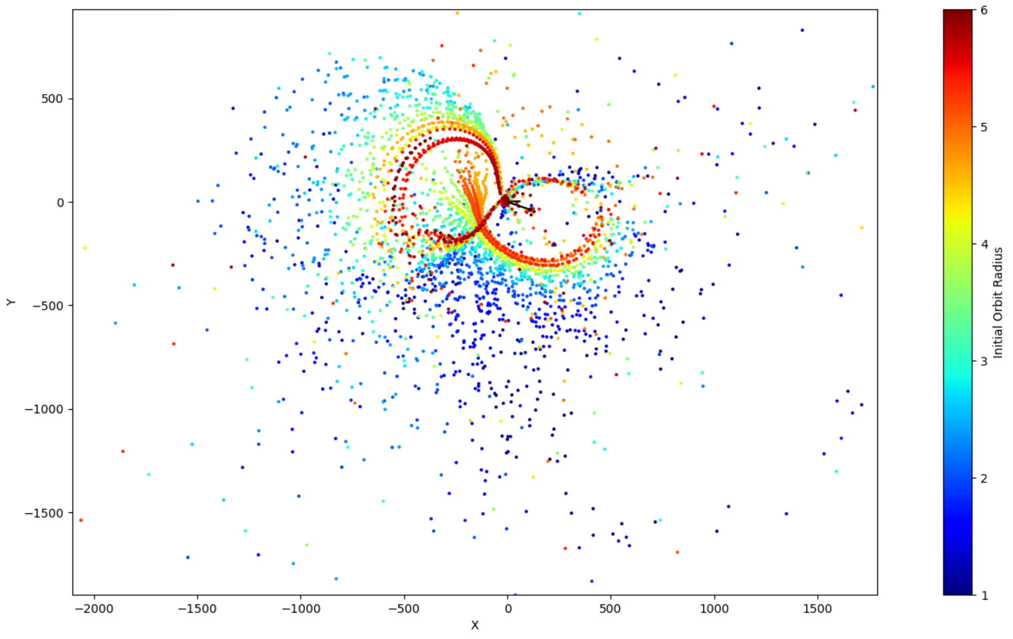

Figure 8: Scatter plot of final positions of 3000 asteroids, coloured to show the initial radii of the asteroids. The scale is large and so the galaxies' paths are not plotted. $y_0 = 3$, $r_m = 0.1$. Closest approach at 1.8 units. At these very close approaches the scattering of the asteroids becomes increasingly isotropic in direction, and the relationship between initial radius and final position of the asteroids becomes less well-defined.

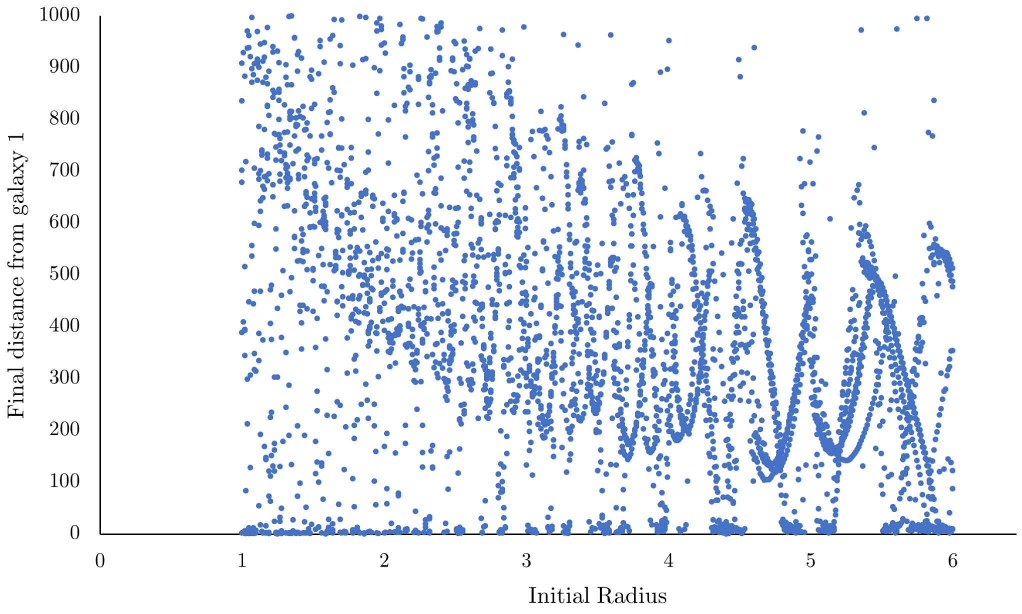

Figure 9: Plot of asteroids' initial orbital radii and final distances from galaxy 1 for 3000 asteroids. $y_0 = 3$, $r_m = 0.1$. Closest approach at 1.8 units. Again, no clear relationship is visible due to the large number of asteroids that are scattered into unstable orbits and not retained by either galaxy.

3.2 Closest Approach Distance

Varying the distance of closest approach of the galaxies changes the extent of the interaction between the asteroids and galaxy 2. A qualitative comparison between relatively small and large closest approaches is shown in figure 10. Generally, closer approaches produce more transfer of asteroids to galaxy 2 as can be seen at ② and more asteroids being scattered into unstable orbits as seen at ①. At this close approach some anticlockwise asteroids are still retained by galaxy 1 as seen at ③ (along with all of the clockwise-travelling asteroids).

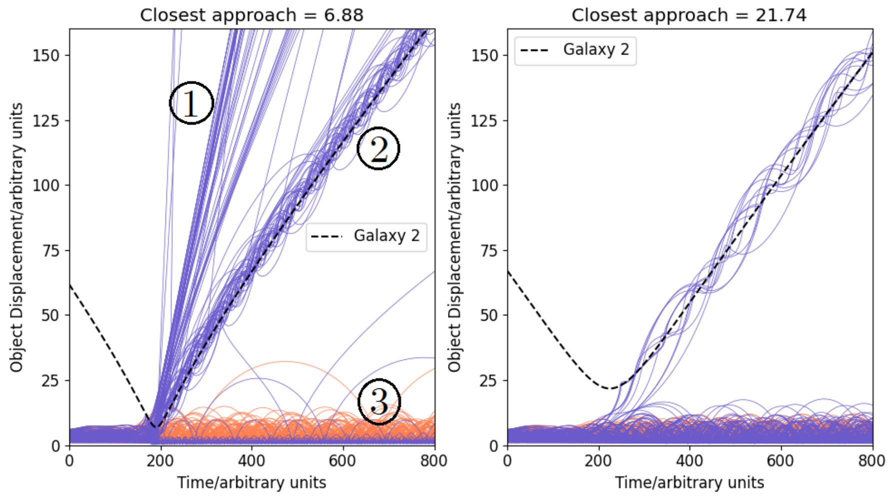

Figure 10: Plots of object distances from Galaxy 1 throughout the simulation, with galaxies' closest approach at 6.88 and 21.74 units. ①, ②, ③ label the main features of the plot as described above. As before, the clockwise-travelling asteroids are orange while the anticlockwise-travelling are purple.

To obtain a quantitative result, the asteroids must be characterised as either retained by galaxy 1 or not. Considering figures 10 and 11, it is clear that this distinction can be performed by considering the final positions of the asteroids from galaxy 1 at large t.

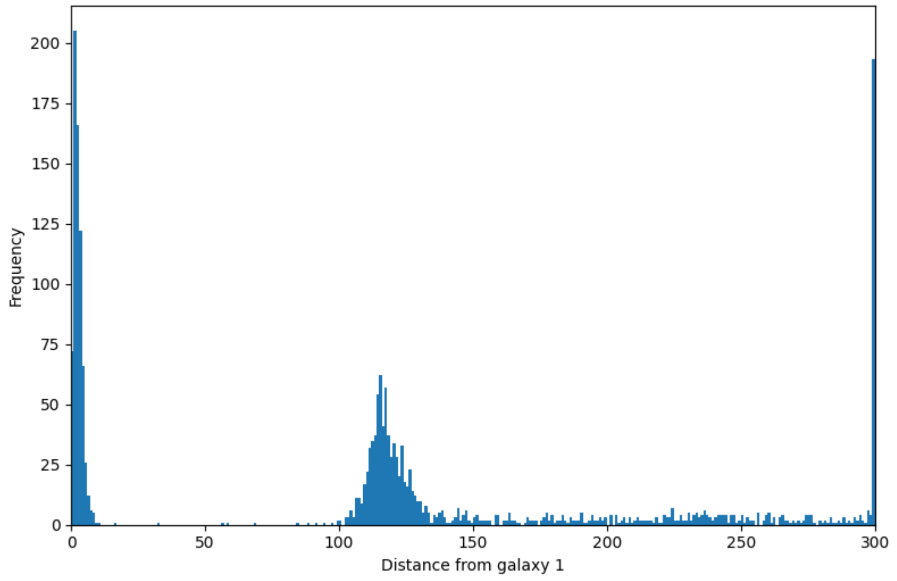

Figure 11: A histogram showing the distances of asteroids from galaxy 1 at $t = 600$. The distances over 300 are placed in an overflow bin at 300. A clear spike can be seen near 0 which represents the asteroids retained by galaxy 1, whole the asteroids in orbits around galaxy 2 are seen as a spike at around 120 units.

The histogram in figure 11 is truncated at an appropriate distance (50 units) to allow the number of retained asteroids to be counted. 4000-asteroid simulations with $r_m = 0.1$ and $y_0 \in [15, 30]$ were performed and the number of retained asteroids was recorded along with the distance of closest approach. The results are shown in figure 12. As expected the overall trend is positive, and approximately linear for closest approach $\in [5, 20]$ with $R^2 = 0.9990$. There is some unexpected trend-reversal at very close approach approaches which is discussed below.

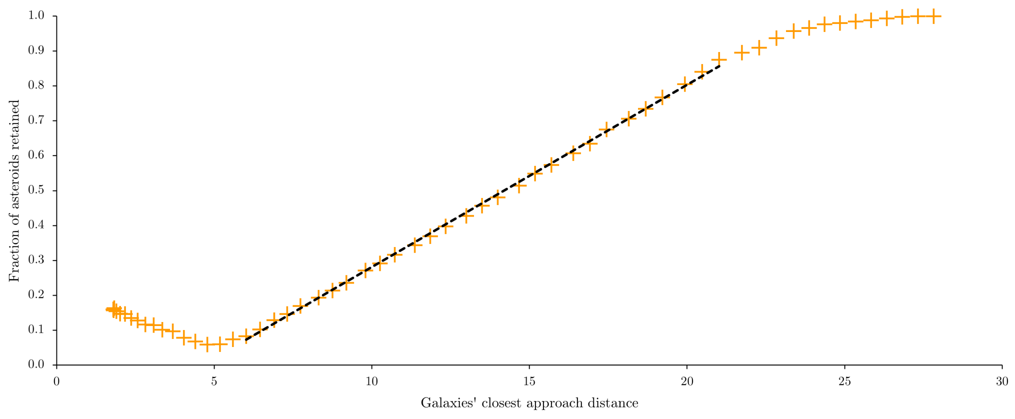

Figure 12: A plot of the fraction of asteroids retained by galaxy 1 against the galaxies' closest approach distance, each point coming from a 4000-asteroid simulation. A trendline is fitted for the approximately linear part of the plot with $R^2 = 0.9990$, $y = 0.0522x - 0.2407$. The constant of proportionality for the linear part is 0.0522 ± 0.0003 arbitrary distance unit^{-1}. All 4000 asteroids were retained at the largest closest approaches (around 28 units).

Similar analysis was performed for counting the number of asteroids transferred to galaxy 2, with all asteroids finishing within 30 units being counted. This methodology is less well suited for this application, as the identification of those that are in stable orbits around galaxy 2 is less clear, as was seen in figure 11. As expected, at large close approach distances the transfer rate tends to zero, as the interaction becomes weak. At small close approach distances the number of asteroids transferred to orbits around galaxy 2 decreases to zero. This is due to a large number of asteroids being scattered into unstable orbits, rather than retained by either galaxy, as was shown in figure 8. Again, the validity of results for these close approaches is discussed below.

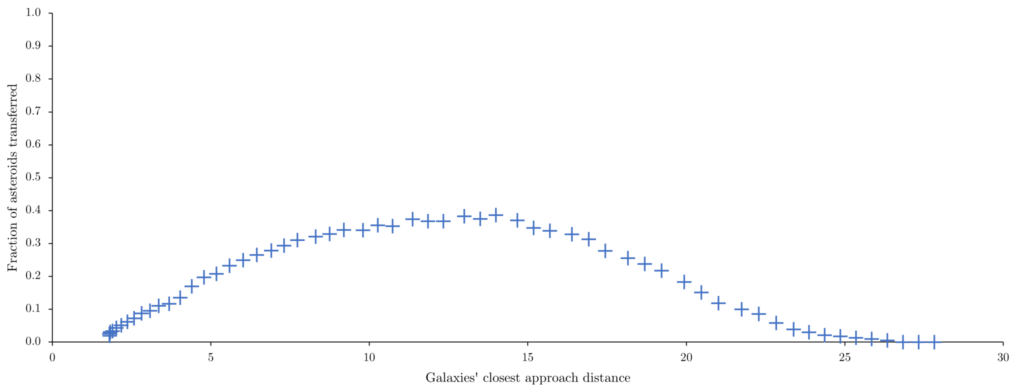

Figure 13: A plot of the fraction of asteroids transferred to galaxy 2 against the galaxies' closest approach distance, each point coming from a 4000-asteroid simulation. A broad peak is visible, with maximal transfer of 38.6% at a close approach of 13.99 units.

There is some unexpected behaviour at close approaches (< 5 units), as was seen in the previous section. However, this result is largely ignorable in terms of its real-world implications. At these very close approaches, galaxy 2 passes closer to galaxy 1 than most of its surrounding asteroids, and the galaxies essentially collide. This makes the zero-viscosity assumption of the simulation significantly invalid, as we may expect to see the galaxies lose much of their kinetic energy and even merge together if the action of dissipative forces were considered.

This highlights a more fundamental issue with the approach that has been taken in this project, as in reality the galaxies are composed of asteroids, and do not behave as separate dense objects as assumed in this model. This assumption breaks down in the limit of close approach, as the behaviour of the galaxies during collision is not represented well by this model. More complex models and the simulation of the galaxies and ISM as fluids, governed by the equations of fluid dynamics (and indeed magnetohydrodynamics) would be more appropriate to study these galactic collisions.

3.3 r_m

Identical simulations to above were performed, with $y_0 = 20$ and $r_m \in [0.05, 0.9]$. The retention and transfer rates were measured in the same way as before. The closest approach distances, though not perfectly constant, remained in the range $[11, 14.5]$, increasing with increasing r_m. The results can be seen in figure 14.

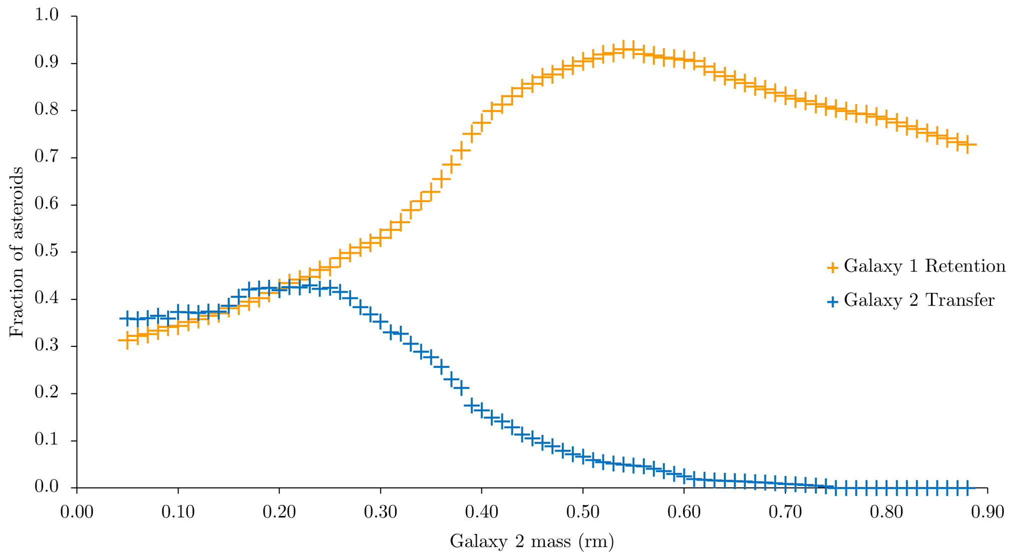

Figure 14: A plot of the fraction of asteroids transferred to galaxy 2 and fraction retained by galaxy 1, each point coming from a 4000-asteroid simulation. As the mass of the perturbing galaxy was increased, the fraction of asteroids transferred to galaxy initially increases with mass, but then decreases to zero by $r_m = 0.8$. The fraction of asteroids retained by galaxy 1 also increases, with a broad maximum around $r_m = 0.54$, and subsequently decreases linearly.

The galaxies take a more shallow parabolic path as their masses become more similar. This causes a low transfer rate to galaxy 2, as it spends much less time in the vicinity of galaxy 1 and then asteroids. However, given that the closest approach distance is not small, the excessive 'scattering' of asteroids seen before does not occur, and hence the retention rate by galaxy 1 remains high. However, as r_m increases further, the amount of scattering increases and so the retention rate is seen to decrease above a certain value. These results are likely to be more accurate than the previous ones, as the galaxies never come into close enough approach to collide, and hence the assumptions made by the model are more valid.

4 Conclusions

Tidal tails were seen to form exclusively from asteroids that orbited in the same direction as the galaxies, in this case anticlockwise. They were also seen to be composed mainly from asteroids that had been orbiting at large radii around galaxy 1, with the exception of tails formed after very close approaches. The implications of this anomalous behaviour at close-approach is largely ignorable, as the model being used is not well suited for simulations where the galaxies 'overlap', given then the simulation is collisionless and has no viscosity.

The fraction of asteroids removed from galaxy 1 was seen to vary linearly with closest approach distance, with constant of proportionality 0.0522 ± 0.0003 *arbitrary distance unit*$^{-1}$. Again, some anomalous behaviour was seen at very close approaches and conclusions drawn here again are likely inaccurate given the limitations of the model. The fraction of asteroids transferred to galaxy 2 was seen to be zero at very close and far approaches, with a broad peak at intermediate approaches, and a maximum transfer of 38.6% at a closest approach of 13.99 units.

The fraction of asteroids transferred to galaxy 2 was seen to decrease with r_m, while the fraction retained by galaxy 1 was seen to increase up to a maximum around $r_m = 0.54$, before decreasing. This was due to the shorter interaction time between the galaxies as their masses became comparable, resulting in lower asteroid transfer rates.

word count: 2950

References

[1] `scipy.integrate.solve_ivp()` documentation, https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html

[2] `time.perf_counter_ns()` documentation, https://docs.python.org/3/library/time.html

[3] Hubble NGC 4038-4039 Image, https://hubblesite.org/contents/media/images/2006/46/1995-Image.html, 16 October 2006

[4] Numba Overview, https://numba.readthedocs.io/en/stable/user/overview.html

A Source Code

Note that in the source code, galaxy 1 is referred to as 'sun' while galaxy 2 is referred to as 'saturn'. This convention was retained after initial creation for lexical clarity when writing/reading the code.

```
import numpy as np
import scipy.integrate as integrate
import numpy.random as rand
from numba import jit

G = 1

def asteroid_generator(radii,n=10, CW= False):
    """
    Creates initial positions and velocities of asteriods
       at radii given with
    n being the number of asteroids at each given radius.
    returns tuple of vectors. CW is boolean that sets the
       direction of the asteroids
    """
    initial_vectors = []

    rand.seed(seed=1234) #Set seed for consistency

    CW_factor = 1 if CW else -1

    for r in radii:
        for theta in rand.rand(n)*2*np.pi:
            initial_vectors.extend([
                r*np.cos(theta),
                r*np.sin(theta),
                CW_factor*np.sin(theta)*np.sqrt(G/r),
                -CW_factor * np.cos(theta)*np.sqrt(G/r)])

    return np.array(initial_vectors)

def saturn_function(t,y, saturn_mass=0.1):
    """
```

```
The function that is passed to the solver to calculate
   the accelereations of the galaxies and the asteroids
   .
galaxies affect all other objects but asteroids are not
    concidered to affect the acceleration of others.
y can be vectorised ie. the function can evaluate at
   multiple points in one function call.
"""

saturn = y[None,0:4]
sun = y[None,4:8]

asteroids = y[8:len(y)].reshape((int((len(y)-8)/4), 4,
   -1)) if len(y) > 8  else None

saturn_sun_displacement = saturn[:,0:2] - sun[:,0:2]

if not(isinstance(asteroids, type(None))): #checks
   wether to evaluate any asteroids or not
    asteroid_sun_displacement = asteroids[:,0:2] - sun
       [:,0:2]
    asteroid_saturn_displacement = asteroids[:,0:2] -
       saturn[:,0:2]

    asteroids_derivative = np.concatenate((
        asteroids[:,2:4],
        -G * asteroid_sun_displacement / np.expand_dims
           (np.linalg.norm(asteroid_sun_displacement,
           axis=1 ), axis=1)**3 + -G *
           asteroid_saturn_displacement / np.
           expand_dims(np.linalg.norm(
           asteroid_saturn_displacement, axis=1 ), axis
           =1)**3
    ), axis=1)

    output = np.concatenate((
        np.concatenate((saturn[:,2:4],-G *
           saturn_sun_displacement / np.expand_dims(np.
           linalg.norm(saturn_sun_displacement, axis=1
           ), axis=1)**3), axis=1),
            np.concatenate((sun[:,2:4], -saturn_mass * -G *
                 saturn_sun_displacement / np.expand_dims(np
                .linalg.norm(saturn_sun_displacement, axis=1
                 ),axis=1 )**3), axis=1),

            asteroids_derivative
            ), axis=0).reshape(y.shape)
    else:
        output = np.concatenate((
            np.concatenate((saturn[:,2:4],-G *
                saturn_sun_displacement / np.expand_dims(np.
                linalg.norm(saturn_sun_displacement, axis=1
                ), axis=1)**3), axis=1),

            np.concatenate((sun[:,2:4], -saturn_mass * -G *
                 saturn_sun_displacement / np.expand_dims(np
                .linalg.norm(saturn_sun_displacement, axis=1
                 ),axis=1 )**3), axis=1)
            ), axis=0).reshape(y.shape)

    return output

# the three following functions are defined to simplify the
    jacobian, and are compiled using Numba
# using the @jit decorator to compile them as C code at
   runtime

@jit
def j1 (a,b,c):
    return G*(-c**2 * G * b + 2*(b - a)**2)/c**4

@jit
def j2 (a,b,c):
    return -2*G*(b[1]-a[1])*(b[0]-a[0])/c**4
@jit
def j3 (a,b,c):
    return -2*G*(b[1]-a[:,1])*(b[0]-a[:,0])/c**4

def jacobian(t,y, saturn_mass=0.1):
```

```
"""
The Jacobian is a function passed to the solver and is
   used in 'Stiff' solvers
returns derivatives of f(y) with respect to each y.
"""
sat = y[0:4]
sun = y[4:8]

ast = y[8:].reshape((-1,4)) if len(y) > 8  else None

sat_sun_dist = np.linalg.norm(sun[0:2] - sat[0:2])

if not(isinstance(ast, type(None))):
    ast_sun_dist = np.linalg.norm(ast[:,0:2] - sun
       [0:2].T, axis = 1)
    ast_sat_dist = np.linalg.norm(ast[:,0:2] - sat
       [0:2].T, axis = 1)

saturn_jac = np.zeros((4, y.size))

saturn_jac[0,2] = 1
saturn_jac[1,3] = 1
saturn_jac[2,0] = j1(sat[0], sun[0], sat_sun_dist)
saturn_jac[3,0] = j2(sat, sun, sat_sun_dist)
saturn_jac[2,1] = j2(sat, sun, sat_sun_dist)
saturn_jac[3,1] = j1(sat[1], sun[1], sat_sun_dist)

saturn_jac[2:4,4:6] = -saturn_jac[2:4,0:2]

sun_jac = np.zeros((4, y.size))

sun_jac[0,2] = 1
sun_jac[1,3] = 1

sun_jac[2,0] = saturn_mass * j1(sun[0], sat[0],
   sat_sun_dist)
sun_jac[3,0] = saturn_mass * j2(sun, sat, sat_sun_dist)
sun_jac[2,1] = saturn_mass * j2(sun, sat, sat_sun_dist)
sun_jac[3,1] = saturn_mass * j1(sun[1], sat[1],
   sat_sun_dist)
```

```
    sun_jac[2:4,4:6] = -sun_jac[2:4,0:2]


    if not(isinstance(ast, type(None))):
        ast_jac = np.zeros((y.size-8, y.size)).reshape
           ((-1,4,y.size)) if not(isinstance(ast, type(None
           ))) else None

        ast_jac[:,0,2] = 1
        ast_jac[:,1,3] = 1

        ast_jac[:,2,0] = j1(ast[:,0], sun[0], ast_sun_dist)
            + saturn_mass * j1(ast[:,0], sat[0],
           ast_sat_dist)
        ast_jac[:,3,0] = j3(ast, sun, ast_sun_dist) +
           saturn_mass * j3(ast, sat, ast_sat_dist)
        ast_jac[:,2,1] = j3(ast, sun, ast_sun_dist) +
           saturn_mass * j3(ast, sat, ast_sat_dist)
        ast_jac[:,3,1] = j1(ast[:,1], sun[1], ast_sun_dist)
            + saturn_mass * j1(ast[:,1], sat[1],
           ast_sat_dist)


        ast_jac[:,2:4,4:6] = -ast_jac[:,2:4,0:2]


        ast_jac = ast_jac.reshape(-1,y.size)

    output = np.concatenate((saturn_jac,sun_jac), axis=0)
       if isinstance(ast, type(None)) else np.concatenate((
       np.concatenate((saturn_jac,sun_jac), axis=0),
       ast_jac))

    return output

def gal_solver(t_max,saturn_initial_cond,
   asteroid_initial_cond= None, num_eval=1000,method='
   DOP853',saturn_mass=0.1):
    """
```

```
Solves the path of the Sun, Saturn and asteroids.
   Essentially a wrapper for the np.sovlve_ivp()
   function that takes arguments that are useful for
   calculating in a loop via another script.
Default method is DOP853
"""
y0 = np.concatenate((saturn_initial_cond, [0,0,-
   saturn_mass * saturn_initial_cond[2],-saturn_mass *
   saturn_initial_cond[3]],asteroid_initial_cond)) if
   not(isinstance(asteroid_initial_cond, type(None)))
   else np.concatenate((saturn_initial_cond, [0,0,-
   saturn_mass * saturn_initial_cond[2],-saturn_mass *
   saturn_initial_cond[3]]))


return integrate.solve_ivp(
            fun = saturn_function,
            t_span = [0,t_max],
            t_eval=np.linspace(0,t_max,num=num_eval),
            y0 = y0,
            method= method,
            vectorized=True,
            jac=jacobian,
            args=(saturn_mass,)
            )
```