Assessment 1: Coupled Harmonic Oscillator

Benji Tigg

December 2024

1 Introduction

Whilst solved for the 2 body problem and partially solved for 3 bodies, there exists no general solution outside of numerical methods to predict the motion for n bodies interacting with each other. To ensure that during the propagation of the solution, constants are conserved, we have to use a symplectic integrator, like the leap frog integrator, etc.. and a method for calculating the force on each body, the most accurate method is direct integration however this becomes quite a costly calculation as n increase due to having a big O of n squared, other methods like Barnes-Hut exist that limit the number of bodies each body has to check against.

2 Code description

This code produces a solution for the n - body problem using the symplectic velocity verlet integrator. The force calculation is done by using a direct force calculation method, however to speed up time per step, the integrator is multi-threaded for each body, with a thread barrier in each step, to stop the code progressing onto the force calculation until each body has updated it's position. The integrator then outputs to a observer object that is shared between all the intergrator's that calculates the total energy (kinetic + potential) and the angular momentum of the system and then outputs that along with the positions of each body to a file; this calculation is done separately to the integrator as it has no impact of the motion of the bodies and it allows the integrator to continue on running with no impact due to the time required to do other calculations and write to the file. The code also takes in a file where you can specify the starting conditions of the system and the other parameters (time step, number of time steps, gravitation softening constant and the output file name). This file can be loaded into the program as a command line argument, if not specified the program runs the defulat one which is a two body orbit. This can either use the There is then a python file which outputs the paths of the two orbits.

3 Validation

3.1 Two body problem

To validate my code against the two body, I chose to validate it against the orbit of the earth around the sun. To set this system up I used data about the aphelion of Earths orbit around the sun [1]. At the perihelion (indicated by the dotted green line in figure 1) the Earth is $147.1 * 10^6$

km's away from the sun and is moving at a speed of ~30.29km/s and at the aphelion (indicated by the dotted orange line in figure 1), earth is $152*10^6$ km's away and is moving at a speed of of ~29.29 km/s.

Initial Values		
Body 1: Earth	Position.x (km)	0.0
	Position.y (km)	$147.1*10^6$
	Position.z (km)	0.0
	Velocity.x (km/s)	30.29
	Velocity.y (km/s)	0.0
	Velocity.z (km/s)	0.0
	Mass (kg)	$5.9722*10^{24}$
Body 2: Sun	Position.x (km)	0.0
	Position.y (km)	0.0
	Position.z (km)	0.0
	Velocity.x (km/s)	0.0
	Velocity.y (km/s)	0.0
	Velocity.z (km/s)	0.0
	Mass (kg)	$1.9891*10^{30}$

Table 1: Initial values of for the simulation of the orbit of earth around the sun

These speeds and radii where also approximately predicted by the code from the initial conditions as shown in Figure 1 and Table 2.

position	radius(km)	energy(J)	gravitational(J)	kinetic(J)	velocity(m/s)
Aphelion	$152.065*10^6$	$5.30827*10^{29}$	$-2.60697*10^{33}$	$2.60751*10^{33}$	29550.22
Perihelion	$147.099*10^6$	$5.30827*10^{29}$	$-2.69498*^1033$	$2.69517*10^{33}$	30042.86

Table 2: Radii, Energy and Velocity at the perihelion and aphelion for earth

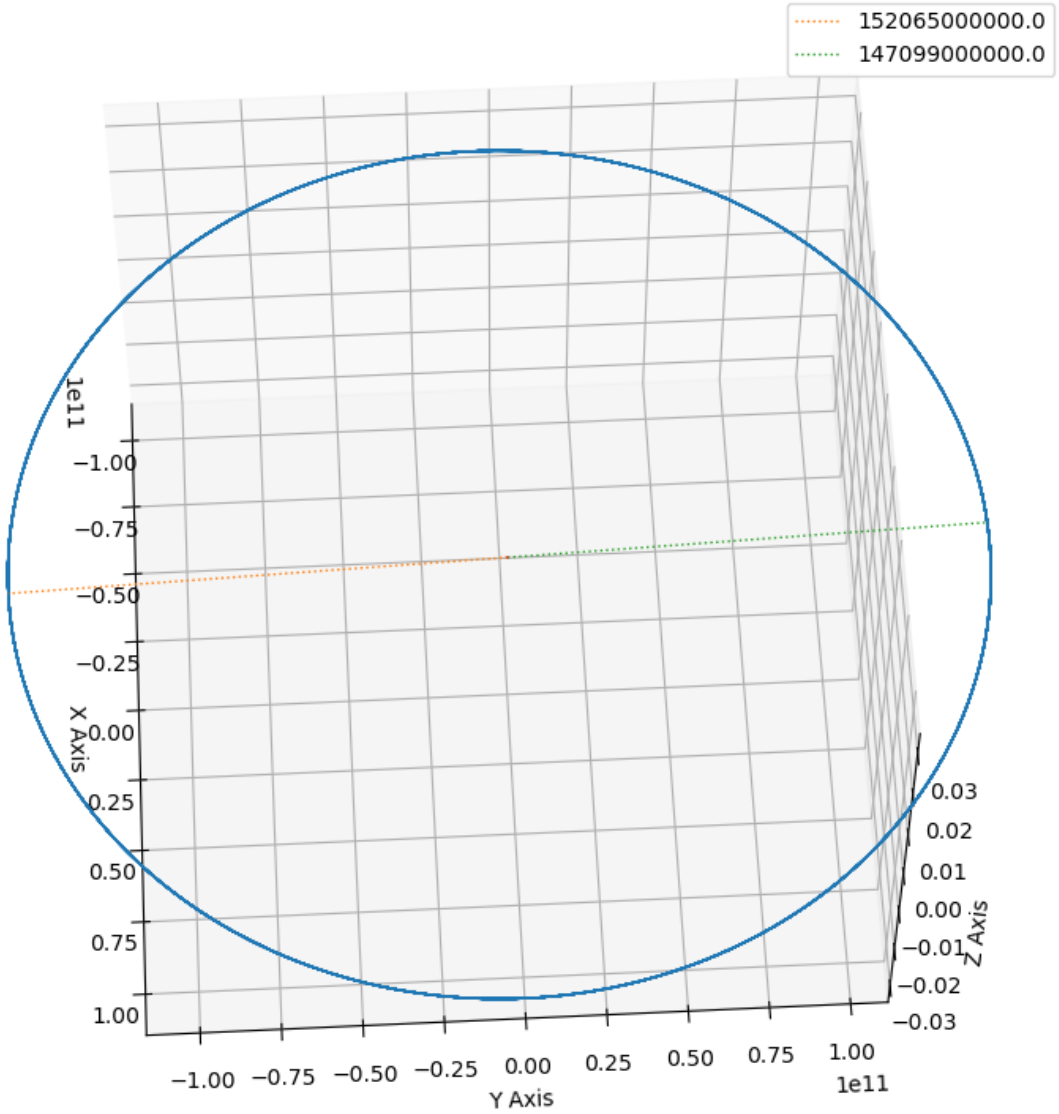

Figure 1: Elliptical orbit of the earth around the sun

3.1.1 Energy conservation

If a integrator is symplectic then that means it will conserve energy over time compared to a integrator like a 4th order RungeKutta which will initially appear to conserve energy, however over time, the energy will begin to drift leading to larger orbits. In Figure 2, we can see that the change in energy over time is periodic with the orbits, of which there is 10 complete orbits which is the correct number of orbits for the time step and max number of steps I put in. Whilst initially I don't have any energy drift as shown in Table 2, in figure 2, the total change in energy from start to finish is $\sim 2 * 10^{27}$. The source of this drift in energy will be the gravitational softening coefficient used, the closer that gets to zero the less the energy will drift, however the orbits would change quite significantly if the bodies got close to each other as the forces would tend to infinity. This small amount of energy drift can be observed in figure 1, where the line is

thicker in places, indicating multiple orbit paths.

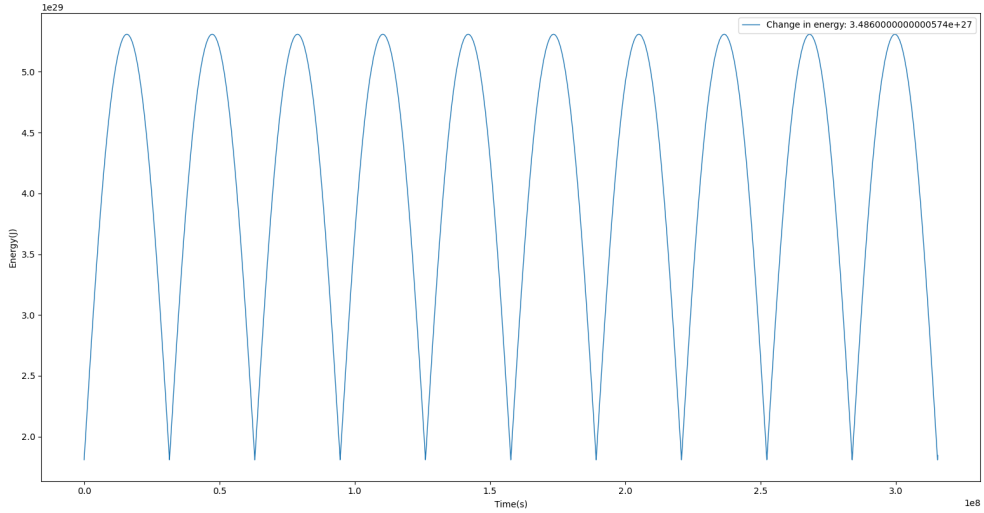

Figure 2: Graph showing the change in energy in the system over time

3.2 Angular Momentum

The angular momentum stayed constant during this entire simulation as expected.

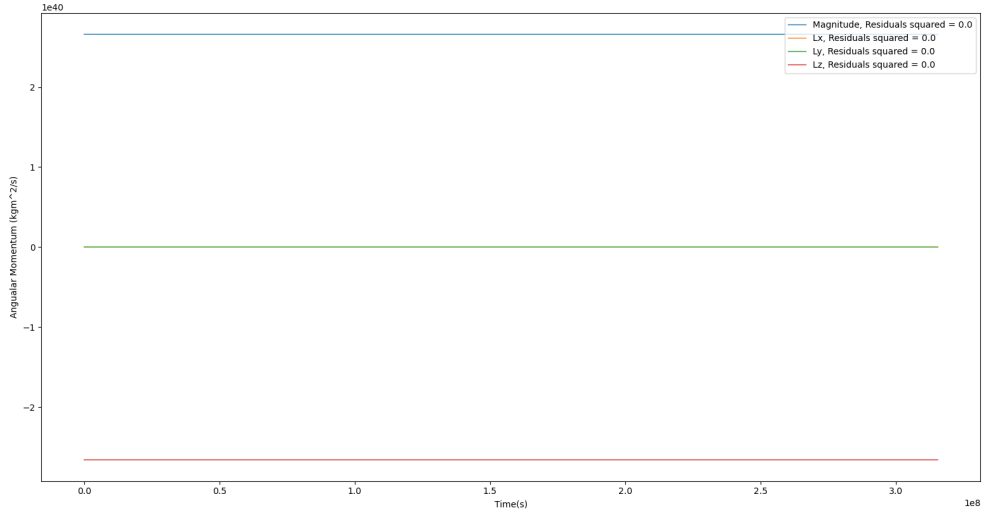

Figure 3: Graph showing the change in angular momentum in the system over time

4 Reproduction of 3 body periodic orbits

A reproduction of various 3 body orbits

4.1 YingYang 2b

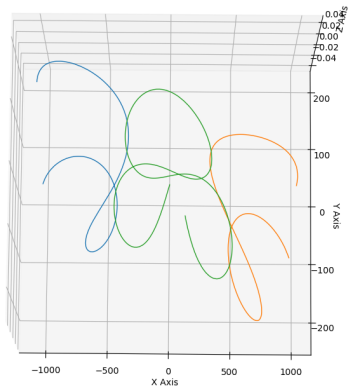

Figure 4: YingYang 2b orbit: mass = $1*10^{13}$kg, timestep = 0.01s, steps = 100000

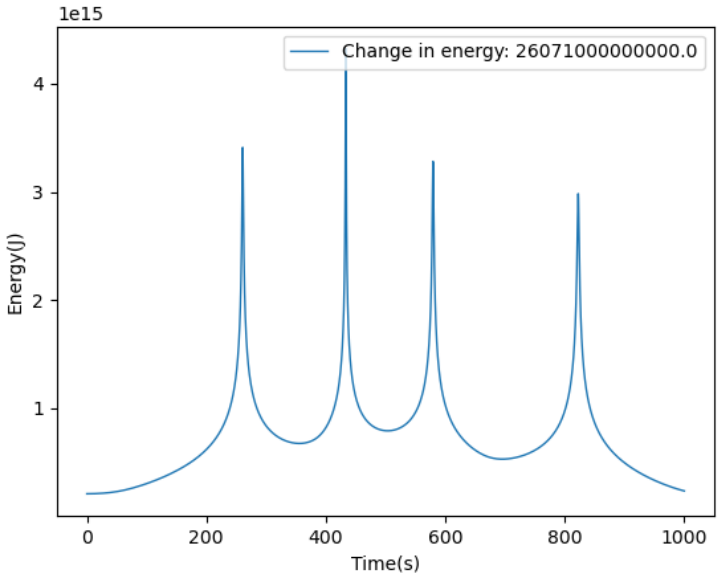

Figure 5: YingYang 2b energy change with time

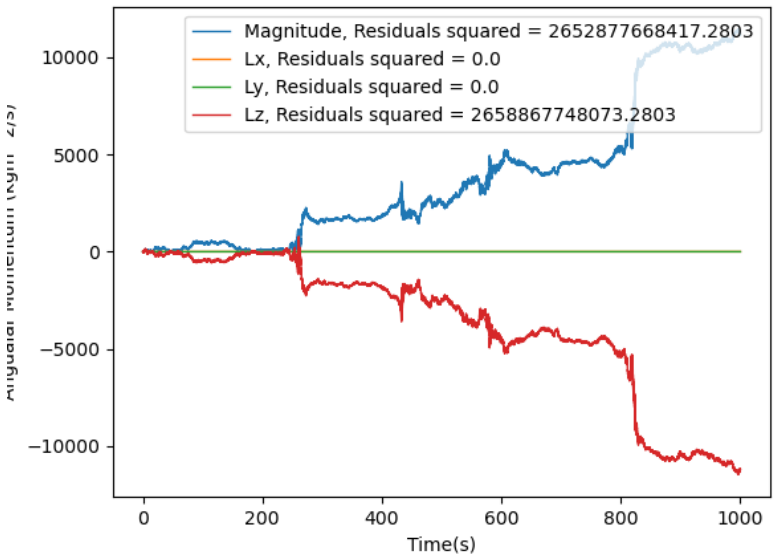

Figure 6: YingYang 2b angular momentum change with time

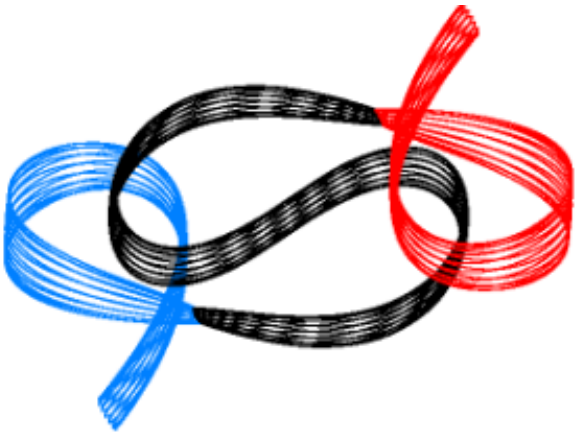

Figure 7: YingYang 2b actual plot

4.2 Bumblebee

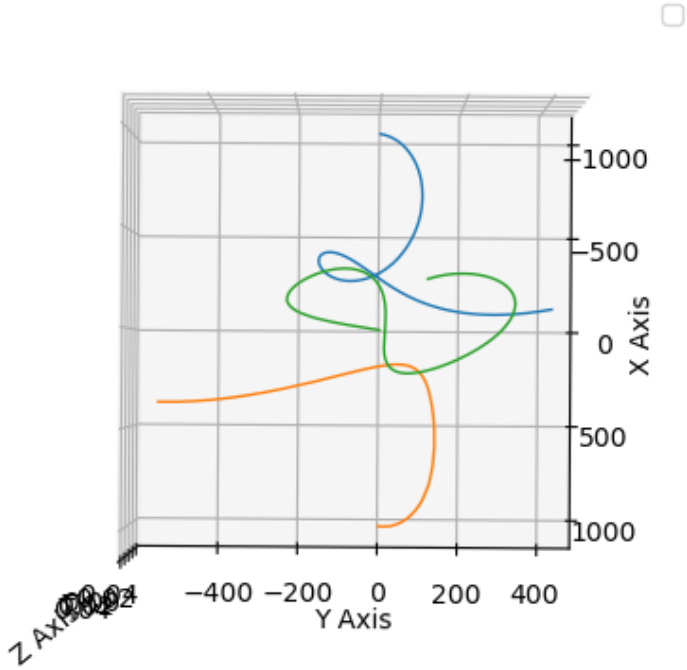

Figure 8: Bumblebee orbit: mass = $1*10^{13}$kg, timestep = 0.01s, steps = 600000

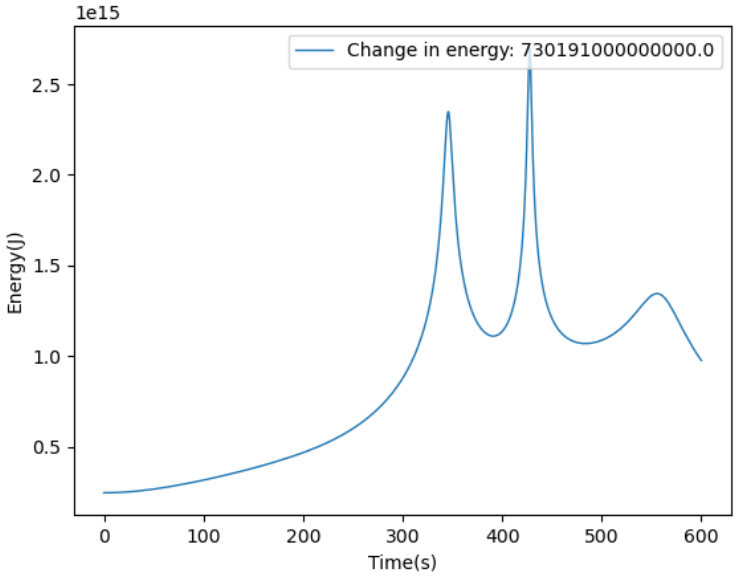

Figure 9: Bumblebee energy change with time

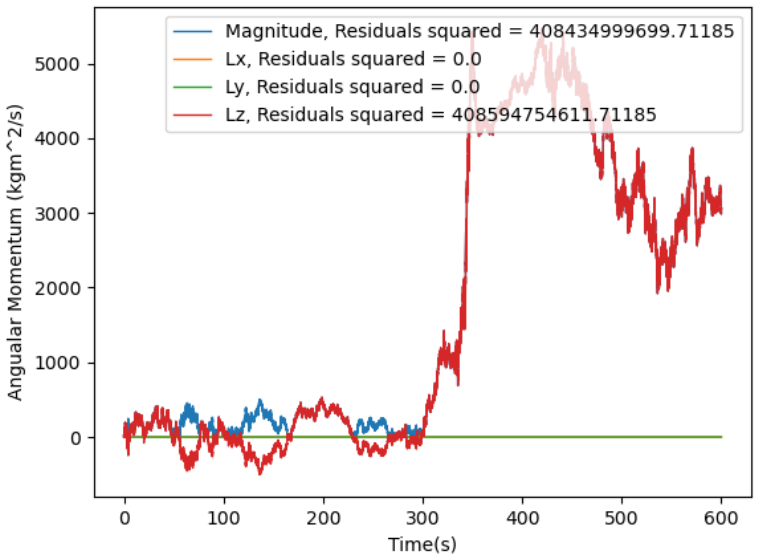

Figure 10: Bumblebee angular momentum change with time

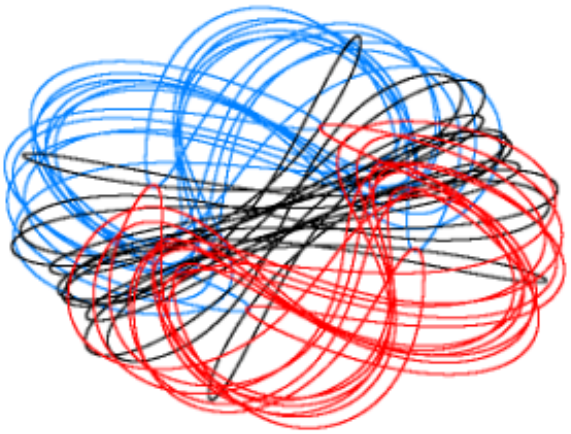

Figure 11: Bumblebee actual plot

4.3 Butterfly

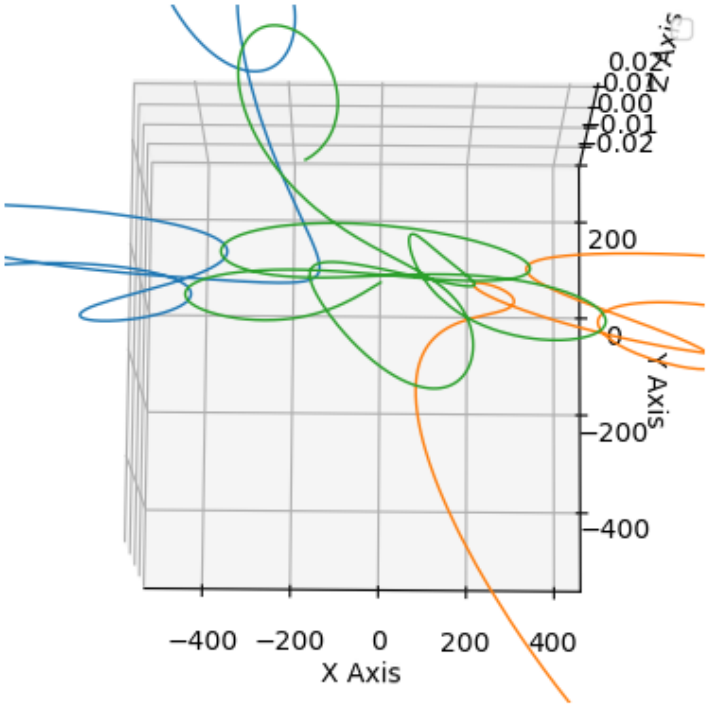

Figure 12: Butterfly 3 orbit: mass = $1*10^{13}$kg, timestep = 0.01s, steps = 200000

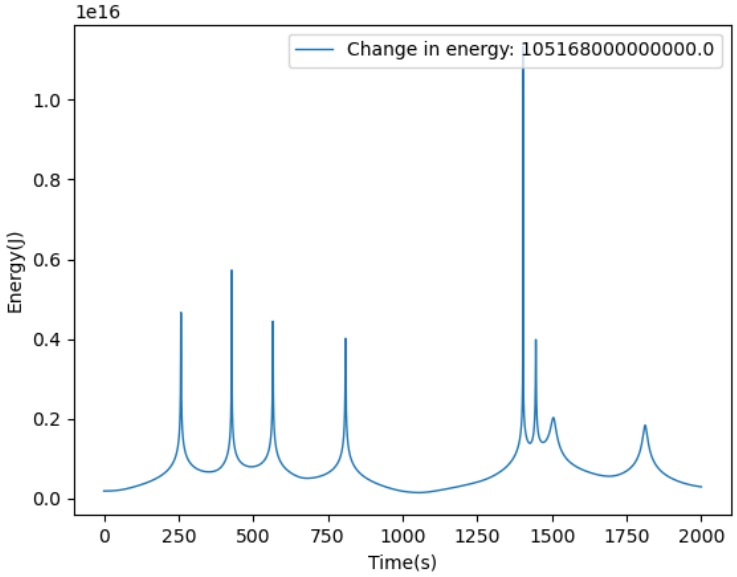

Figure 13: Butterfly 3 energy change with time

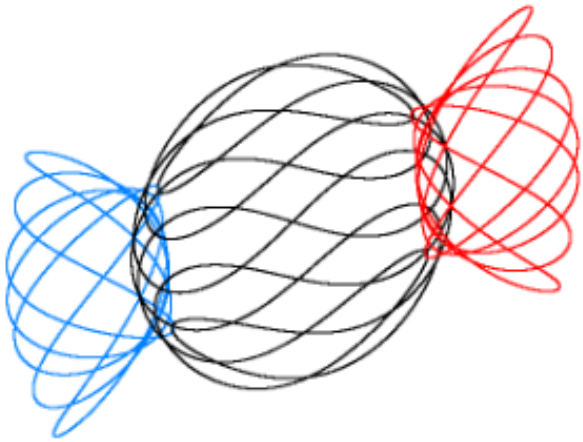

Figure 14: Butterfly 3 actual plot

4.4 Discussion

As the results show, the plots produced by the code didn't manage to reproduce the desired output as the energy conversation and angular momentum are all over the place, this is likely due to the inclusion of quite a aggressive softening factor (around 0.5-1) and the system not behaving as expected and particles flying off in various directions when they come close to each other. However in some cases such as the butterfly 3 orbit, the code managed to capture part of the code before the bodies flew off, further fine tuning of the gravitational softening parameter and time steps would be required to prevent that from happening.