Scientific computing III final project: 4. Restricted three-body problem

Vili Oja 014339369

May 31st, 2017

Contents

1 Introduction

The assignment was to study the restricted three-body problem. The restricted three-body problem consists of two massive objects (called primaries), and a massless third object. The primaries move around their mutual centre of mass on circular orbits in the same plane. Thus their motion is known exactly for all times. Being massless, the third body does not affect the motion of the primaries. Thus the problem is reduced to the study of the motion of only the small test particle in the field of the two co-orbiting primaries.

While there are obviously no massless satellites in real life cases, the problem is actually a fairly good approximation to various astronomical situations. First, most planetary orbits around the Sun, as well as many satellite orbits around their planets, are nearly circular. The same is true of primary star orbits, and the orbits of many stars in galaxies. Furthermore mass ratios of solar system trios are are often such that one object essentially does not influence the orbit of the two primaries that dominate its motion. For example, the orbits of many asteroids are dominated by Jupiter and the Sun but the asteroids do little to modify the motions of those gigantic masses.

The motivation for studying this problem is that we humans can use the information to explain the motion of many asteroids in our solar system, and also use the information when we want to send satellites to an orbit where they don't need to constantly burn fuel to stay at rest with respect to the primaries.

2 Methods

Studying the problem is easiest in a rotating frame. The origin of the coordinate frame is at the centre of mass of the primaries. The x-axis goes through both primaries and the y-axis is perpendicular to it. Thus the coordinate frame is rotating with the primaries at a constant speed.
I chose the units in the following way to simplify the equations:
The unit of mass is chosen so that the total mass of the primaries is 1. The mass of the planet is denoted by μ and thus the mass of the Sun is 1-μ.
The distance between the primaries is used as the unit of length. Thus the coordinates of the primaries are (x=-μ , y=0) and (x=1-μ , y=0).
Finally, the unit of time is chosen so that the mean motion of the primaries is n=1, i.e. the period is P=2π.
When the units are chosen this way, also the gravitational constant is G=1.

Thus the only parameter describing the system is μ. We'll denote the distance of the satellite

from the Sun by r_1 and from the planet by r_2. The final equations of motion of the satellite are

$$\frac{d^2x}{dt^2} - 2\frac{dy}{dt} = \frac{\partial\Omega}{\partial x} \tag{1}$$

$$\frac{d^2y}{dt^2} + 2\frac{dx}{dt} = \frac{\partial\Omega}{\partial x} \ , \tag{2}$$

where Ω is kind of effective potential

$$\Omega = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} \tag{3}$$

The equations of motion cannot be solved analytically. Since there are two second order differential equations, four integration constants are needed for the complete solution. Unfortunately, only one can be found, the Jacobi integral:

$$C = 2\Omega(x, y) - v^2 \ , \tag{4}$$

where v is the velocity of the satellite with respect to the primaries.

The Jacobi integral C gives restraints for the motion of the satellite. Since the squared velocity cannot be negative, the motion of the satellite is constrained to the region where $\Omega \geq \frac{C}{2}$, and the curve $\frac{C}{2} = \Omega$ serves as a barrier, creating unattainable regions of space. These curves are called zero-velocity curves, since if the satellite were to reach them, it would have zero velocity there. And since particles cannot travel along zero-velocity curves (otherwise they wouldn't have zero velocity), they must approach the curves along directions perpendicular to those curves.

There are five special points where the satellite can remain at rest with respect to the primaries. These are called the Lagrangian points, L1 through L5. These points can be found from the equations

$$\frac{\partial\Omega}{\partial x} = \frac{\partial\Omega}{\partial x} = 0 \tag{5}$$

Two of the solutions form equilateral triangles with the primaries. The other three points are on the x-axis, one between the primaries, one to the left and one to the right of the primaries. Technically one should also consider the third dimension, z, when thinking about the locations of these points, but fortunately only the position in the z=0 plane have no z gradient and could thus be candidates for the Lagrangian points.

For these three points we get the equations

$$x - \frac{1-\mu}{(x+\mu)^2} - \frac{\mu}{(x-(1-\mu))^2} = 0 \ , \ 1-\mu < x \tag{6}$$

$$x - \frac{1-\mu}{(x+\mu)^2} + \frac{\mu}{(x-(1-\mu))^2} = 0 \ , \ -\mu < x < 1-\mu \tag{7}$$

$$x + \frac{1-\mu}{(x+\mu)^2} + \frac{\mu}{(x-(1-\mu))^2} = 0 \ , \ x < -\mu \tag{8}$$

Each of these has only one root in the given range.

The location of the points L4 and L5 can be easily derived geometrically. They are located at (x,y) = $(\frac{1}{2} - \mu, \pm\frac{\sqrt{3}}{2})$.

If one considers the potential topography of the system, the first three Lagrangian points are saddle points, and thus aren't stable. Because there are places with larger potential, objects at these points have still some zero-velocity surfaces they cannot pass. L4 and L5 on the other hand are potential maxima, but thanks to Coriolis effects, objects at these points are stable. Objects at these have points the maximum potential, so they don't have any zero-velocity surfaces, so they can *theoretically* get to any place in the system.

I used a simple leapfrog integrator to investigate the motion of a satellite in the Sun-Jupiter system. In leapfrog integration one updates the positions and velocities of the objects at interleaved time points, staggered in such a way that they "leapfrog" over each other. The integrator itself uses inertial coordinates, but when the data is drawn, it's then transformed into a rotating frame.

3 Implementation

For the first task of finding the coordinates of the Lagrangian points for different values of μ I used Matlab. The script I used is in the file `solver.m`. It plots the places for the first thee Lagrangian points along with the respective positions of the planet and the sun with different values of μ. It also calculates the position for L1 through L3 for the Sun-Jupiter system where $\mu = \frac{1}{1047}$.

For the second task I coded a simple orbital integrator with Fortran. The program is split into two files, `functions.f90` and `integrator.f90`. The program can be easily compiled with the make file I've included. So simply typing the command `make` will create the program, `integrator.exe`. Unneeded files can be removed with the command `make clean`. Running the program is done with the following command line arguments:

./integrator.exe [input file] [output file] [time period] [saving density] [time step]

Input file is the file from which the program reads the initial data for the objects. I've included the ones I've used in the run-folder. The output file is the name of the output. Time period is the time for how long the simulation will be run. It is in units of Jovian years, i.e. if the time period is set to 10, the simulation will run for 10 whole periods. Saving density tells how often the program will write down the results of the integration. So if it's 1, every one will be written down. If it's 1000, every 1000th will be written down. Time step tells how long the time step of the simulation will be. It's in units of radians, where 2π would be a whole orbit of Jupiter. So if the time step is for example 0.0001, it would equal to about 1.5 hours.

An example:
./integrator.exe inputL4.dat out_L4_1000Ja.dat 1000 1000 0.0001

I also used python to make a simple program that will draw the orbits of the objects in a rotating frame. Running it is simple:

python draw.py [input file]

Input file is the name of the file in which the data is stored. This is essentially the same as the output file from the integrator. So for example:

python draw.py out_L4_1000Ja.dat

4 Results

4.1 Task 1

As seen in the figures 1 and 2, L1 and L2 are very close to the planet at small values of μ. As μ gets greater values, the system would physically be more like a binary star system than a sun and a planet, but here the smaller object is named planet for convenience. As seen in figure 2, when μ becomes larger than 0.5, the system essentially flips since it's equivalent to the scenario where we flip the sun and the planet around.

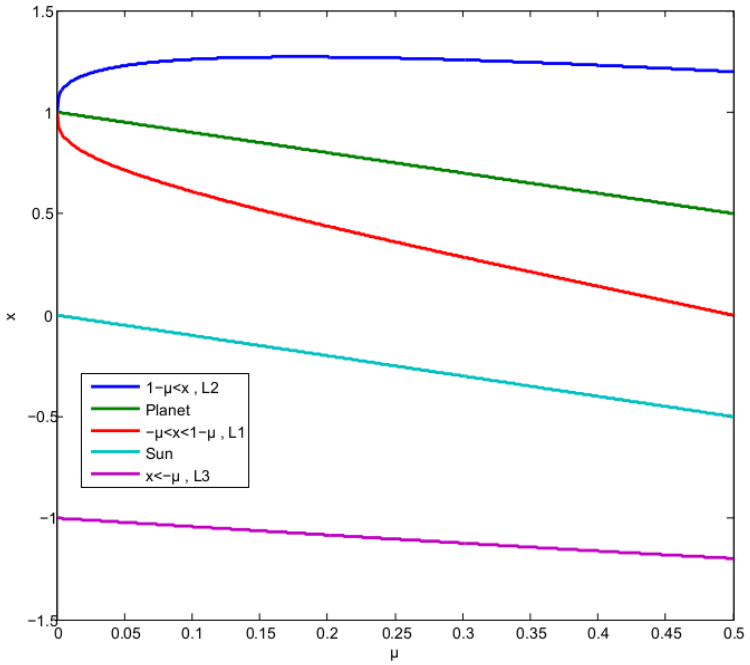

Figure 1: Coordinates of the Lagrangian points, a sun and a planet for different values of μ. Here μ goes only from 0 to 0.5, since after 0.5 the planet technically becomes the sun, so the image would just mirrored and flipped, as seen in figure 2.

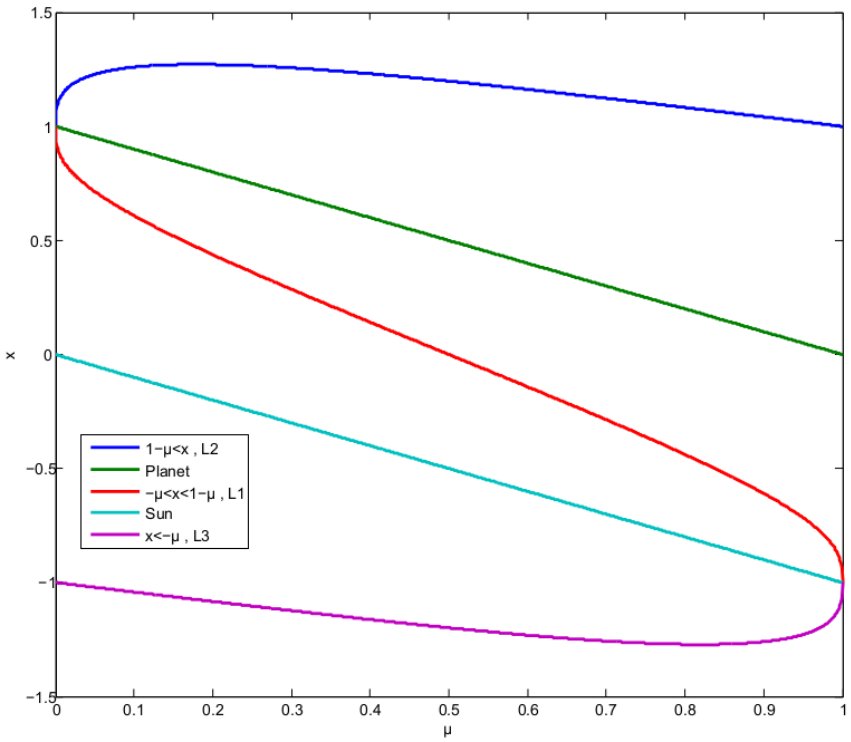

Figure 2: Coordinates of the Lagrangian points, a sun and a planet for different values of μ. Here μ goes from 0 to 1.

4.2 Task 2

The instructions were quite vague about what exactly was wanted in this part, so I chose to study the behaviour of the satellite at the theoretical Lagrangian points, and also how it behaves if its speed is altered one way or the other at those points.

In figures 3, 4 and 5 is shown how an object starting from rest at L1 will behave over time. It is immediately obvious that the point isn't stable since the object starts to orbit the Sun in the rotating frame. It isn't clear here, but there is a zero-velocity surface around the Sun beyond the orbits of the satellite and at the back of Jupiter. The surface is open at the original L1 point, so the object can get to an orbit around Jupiter, as seen in in figure 5.

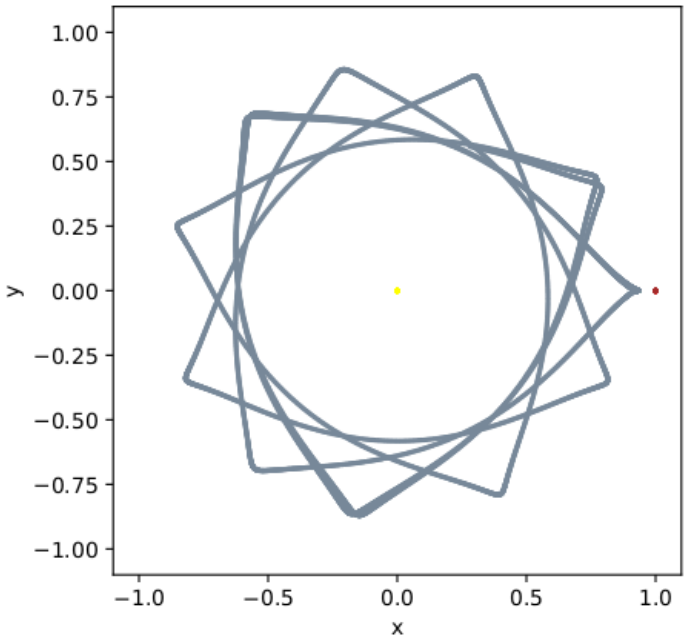

Figure 3: Behaviour at L1. Simulation ran for 10 Jovian years. Input file was `inputL1.dat`, output was `out_L1_10Ja.dat`.

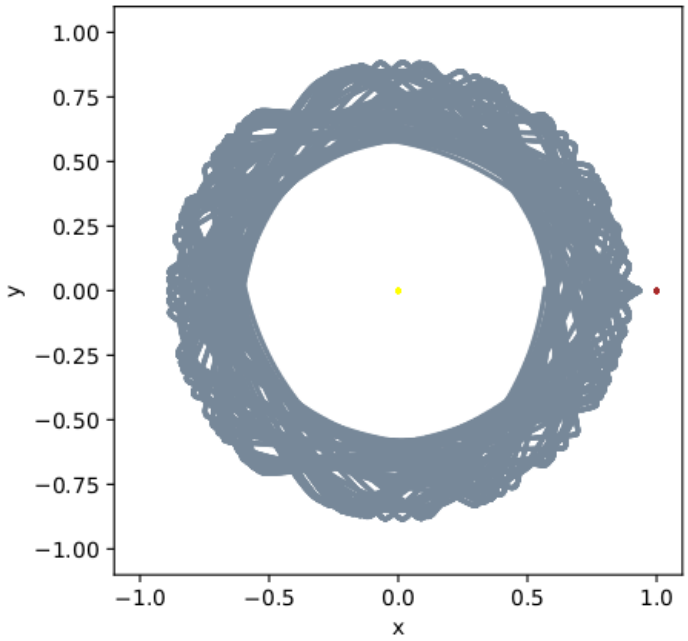

Figure 4: Behaviour at L1. Simulation ran for 100 Jovian years. Input file was `inputL1.dat`, output was `out_L1_100Ja.dat`.

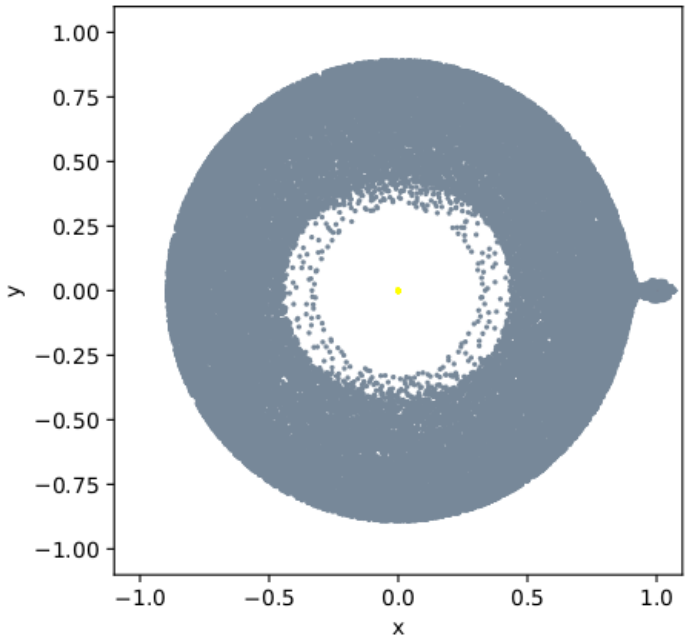

Figure 5: Behaviour at L1. Simulation ran for 1000 Jovian years. Input file was `inputL1.dat`, output was `out_L1_1000Ja.dat`.

In the figures 6 through 10 one can see behaviour around the L2 point. Again the point is clearly not stable. But instead of going straight to orbiting the Sun, the satellite first orbits Jupiter for a while, before jumping to an orbit around the Sun. In the longer simulations the satellite occasionally switches back to orbiting Jupiter. In figure 9 is shown how the satellite escapes from the system if given enough speed. Here I gave the satellite a speed boost of 0.32. Less than that and the satellite would still be bound to the system. The spiral image comes from the fact that the locations of the satellite weren't saved often enough to show a constant orbit.

In figure 10 one can see the zero-velocity surface very clearly. Here the speed of the satellite was increased by a mere 0.0001 compared to being at rest at L2. The satellite manages at some point to go through the hole in the zero-velocity surface that is around Jupiter and switch from orbiting only the Sun or Jupiter to orbiting the whole system. In the image Jupiter is hidden at the connection point of the inner and outer regions.

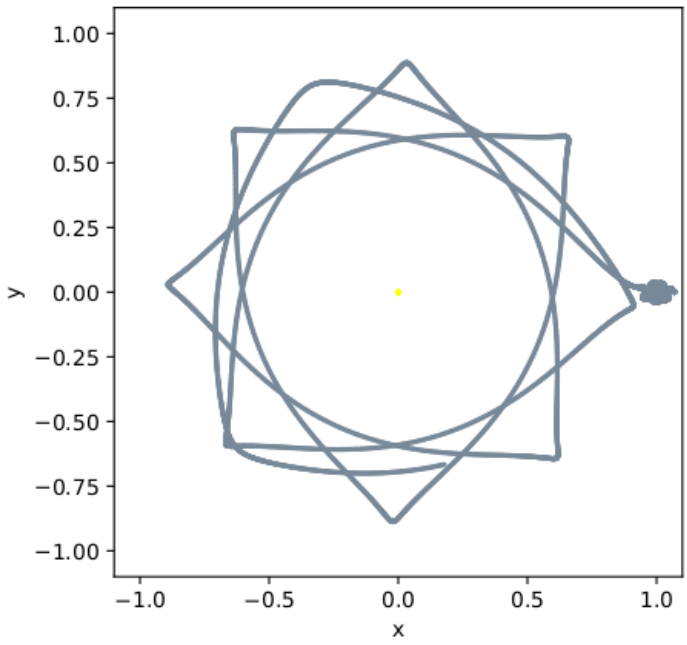

Figure 6: Behaviour at L2. Simulation ran for 10 Jovian years. Input file was `inputL2.dat`, output was `out_L2_10Ja.dat`.

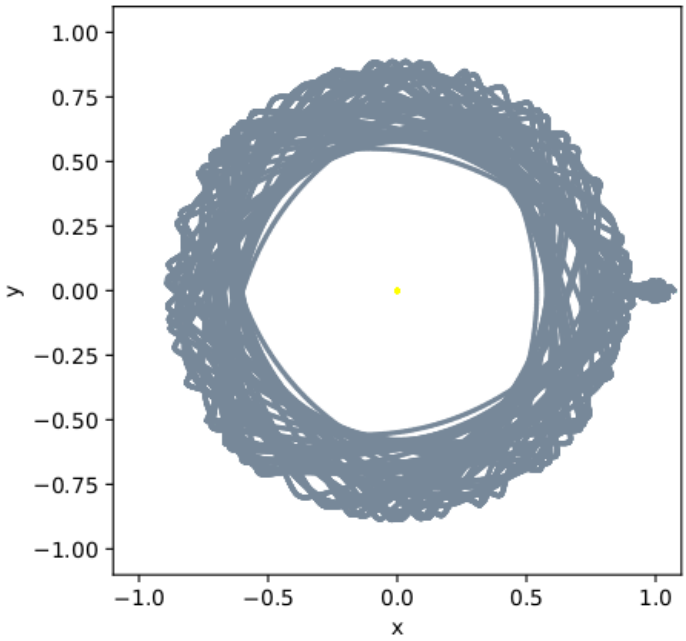

Figure 7: Behaviour at L2. Simulation ran for 100 Jovian years. Input file was `inputL2.dat`, output was `out_L2_100Ja.dat`.

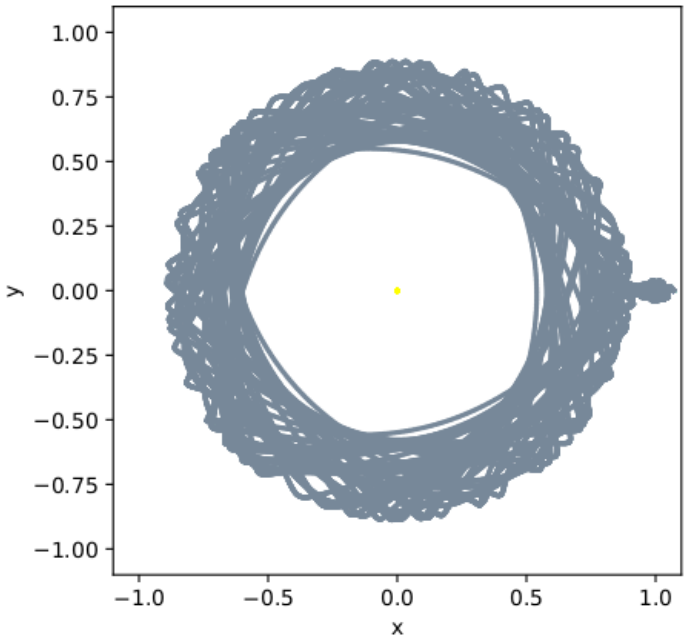

Figure 8: Behaviour at L2. Simulation ran for 1000 Jovian years. Input file was `inputL2.dat`, output was `out_L2_1000Ja.dat`.

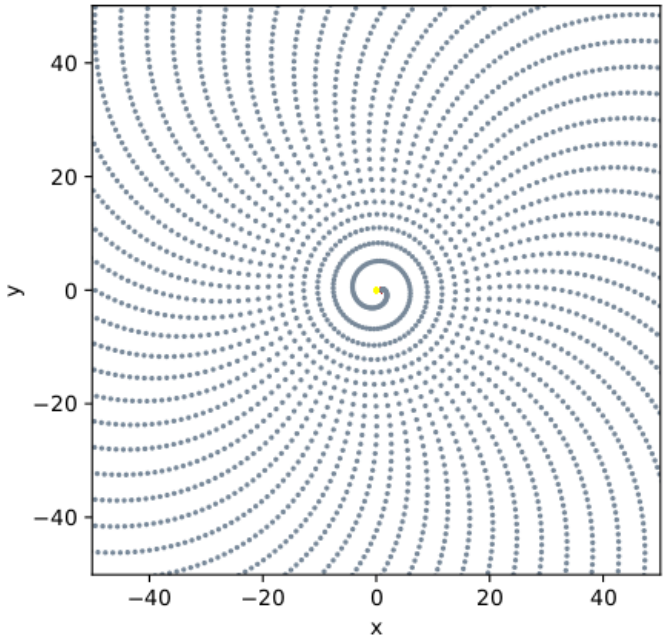

Figure 9: Escaping from L2. Simulation ran for 100 Jovian years. Input file was `inputL2_escape.dat`, output was `out_L2_escape_100Ja.dat`.

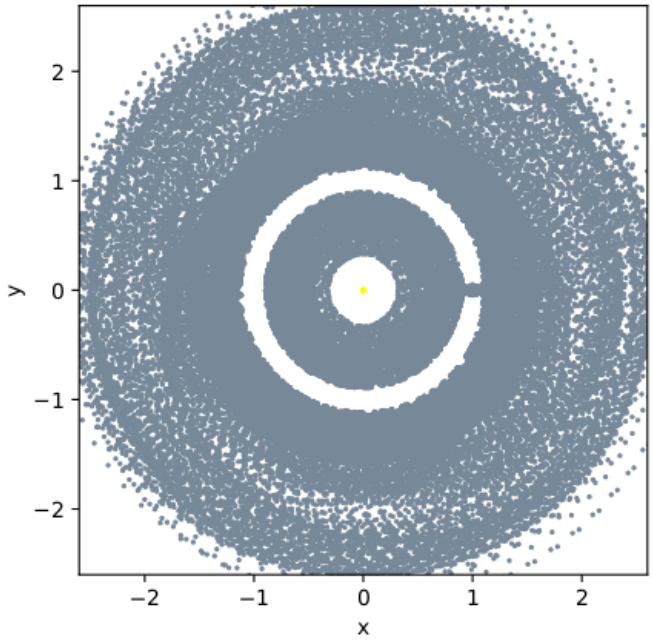

Figure 10: Speeding up at L2. Simulation ran for 1000 Jovian years. Input file was `inputL2_speedup.dat`, output was `out_L2_speedup_1000Ja.dat`.

In figure 11 through 14 one can see behaviour around the L3 point. This point isn't stable either. Here the satellite doesn't start orbiting the Sun in the same way that it did at the previous two points. Instead it starts to do a loop that resembles a horse shoe. This type of orbit is generated because in the inertial frame the satellite orbits the sun at a slightly different distance than the planet. So for example if the satellite's distance is slightly greater than the planet's, it will move slightly slower than the planet. Eventually the planet will catch up to the satellite, but this interaction will change the satellite's orbit so that it will move faster than the planet. Then it will escape from the planet, only to catch back up to it at a later orbit. Then the satellite will lose some energy and start going slower, starting the process again. In the rotating frame this is seen as a horse shoe.

Figure 13 shows the satellite escaping the system from L3. Here the satellite needed more speed than at L2, a boost of 0.42, to escape. In figure 14 the satellite has been slowed down by 0.017. The satellite doesn't make nice horse shoe patterns any more, but instead it at some point interacts with Jupiter and manages to get to an orbit around the system.

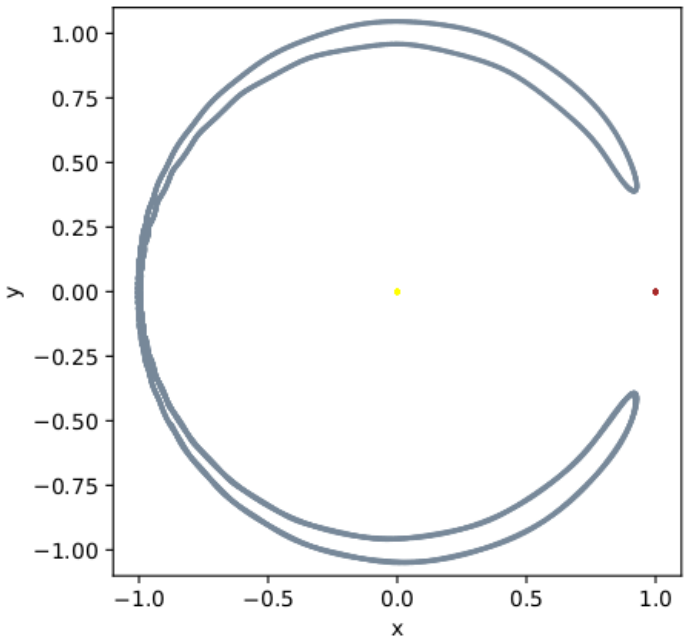

Figure 11: Behaviour at L3. Simulation ran for 100 Jovian years. Input file was `inputL3.dat`, output was `out_L3_100Ja.dat`.

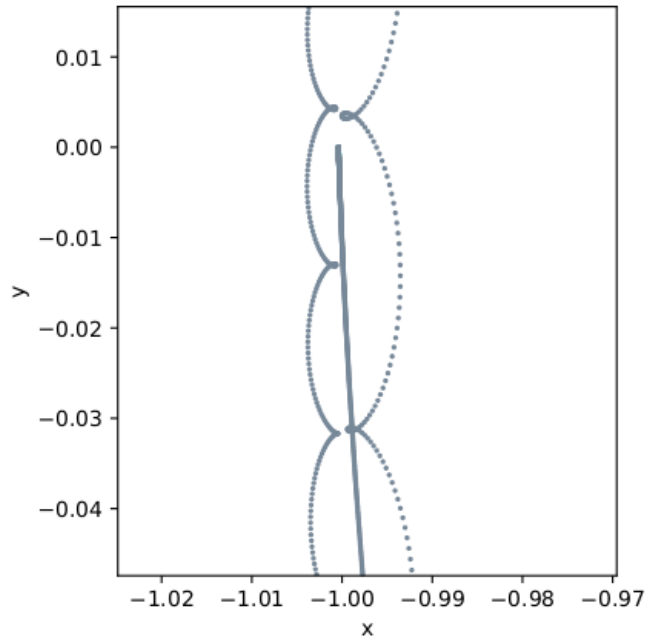

Figure 12: Zoomed in version of figure 11. The solid line is from the start of the simulation.

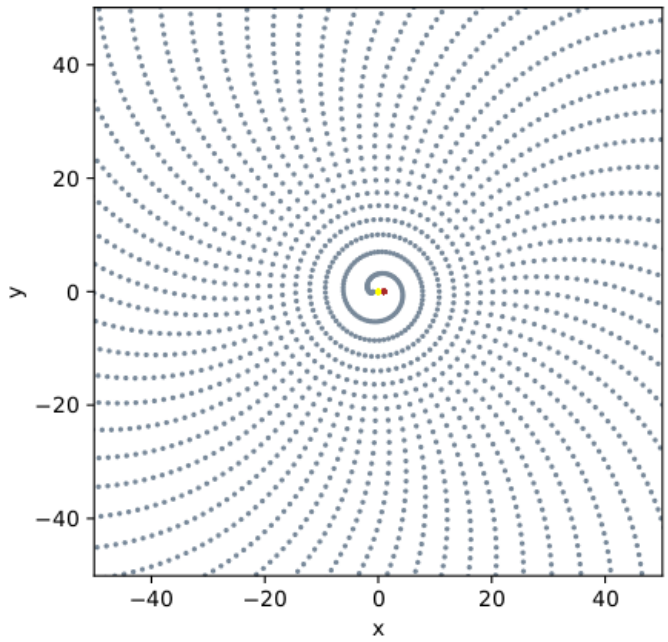

Figure 13: Escaping from L3. Simulation ran for 100 Jovian years. Input file was `inputL3_escape.dat`, output was `out_L3_escape_100Ja.dat`.

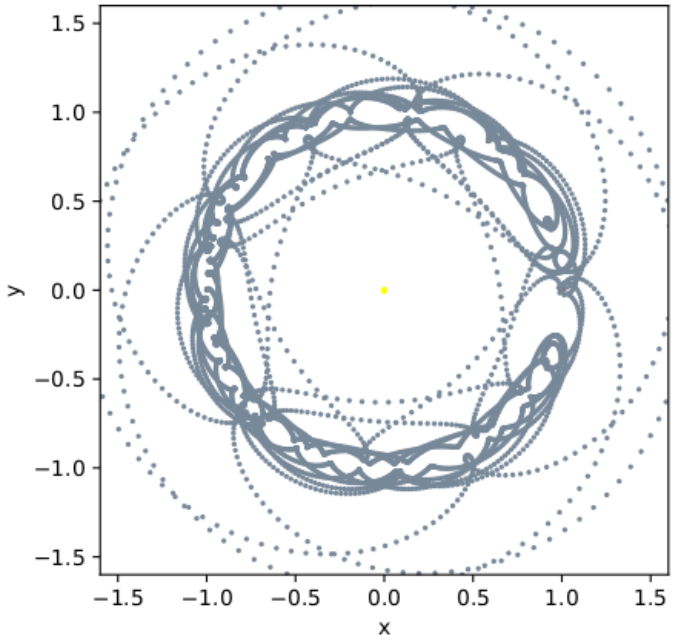

Figure 14: Slowing down at L3. Simulation ran for 100 Jovian years. Input file was `inputL3_speeddown.dat`, output was `out_L3_speeddown_100Ja.dat`.

Figure 15 through 19 show the motion of a satellite about the point L4. In figure 15 one can see that the orbit at L4 is stable. In figure 16 the satellite's speed has been multiplied by a factor of 0.98. This causes the satellite to go about an orbit that resembles a tadpole. The loops into the middle of the tadpole are the result of the satellite moving periodically outward from the stable orbit but Coriolis forces push the satellite back. Same kind of movement could be seen in the previous figure too if zoomed close enough.

In figure 17 the speed has been multiplied by a factor of 0.97 instead. Now the satellite can cross over the L3 point also and starts to make a familiar horse shoe shape. In figure 18 the speed has been multiplied by a factor of 1.02. This also causes a tadpole shaped orbit. In figure 19 the speed was multiplied by a factor of 1.03. This caused the satellite to gain all sorts of orbits around the system, even going to orbit only the Sun for some time instead of the whole system.

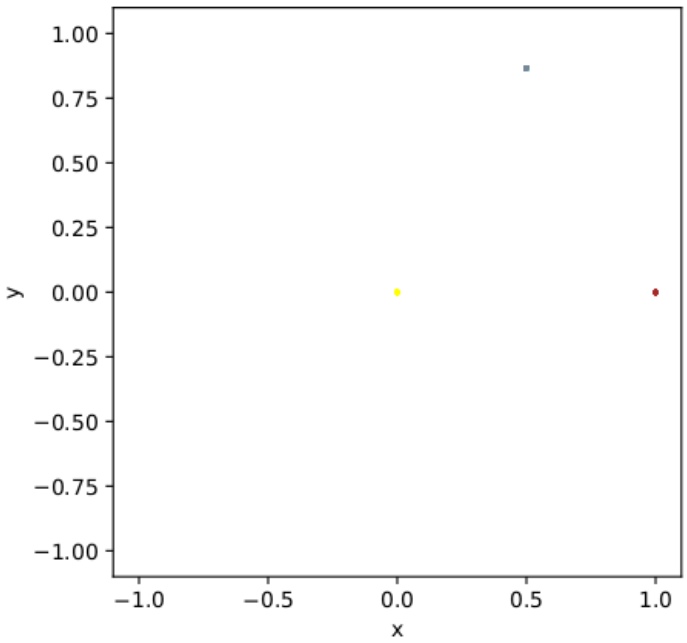

Figure 15: Behaviour at L4. Simulation ran for 1000 Jovian years. Input file was `inputL4.dat`, output was `out_L4_1000Ja.dat`.

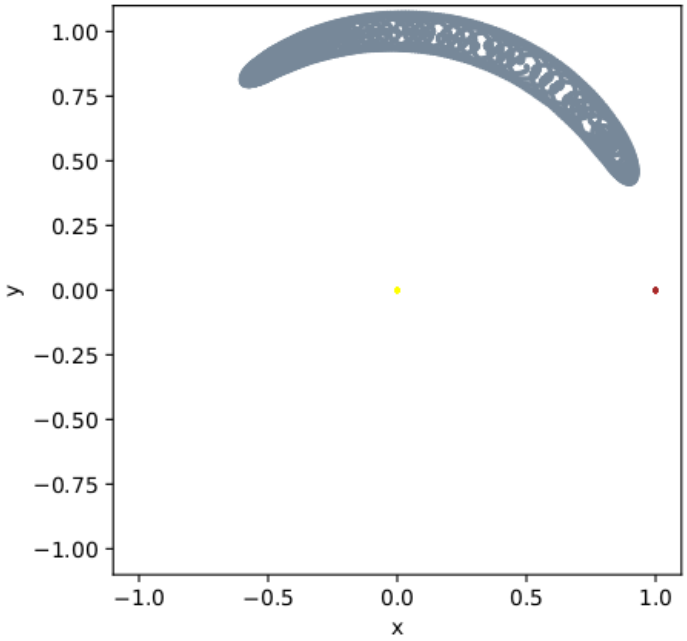

Figure 16: Slowing down at L4. Simulation ran for 1000 Jovian years. Input file was `inputL4_speeddown_toadie.dat`, output was `out_L4_speeddown_1000Ja_toadie.dat`.

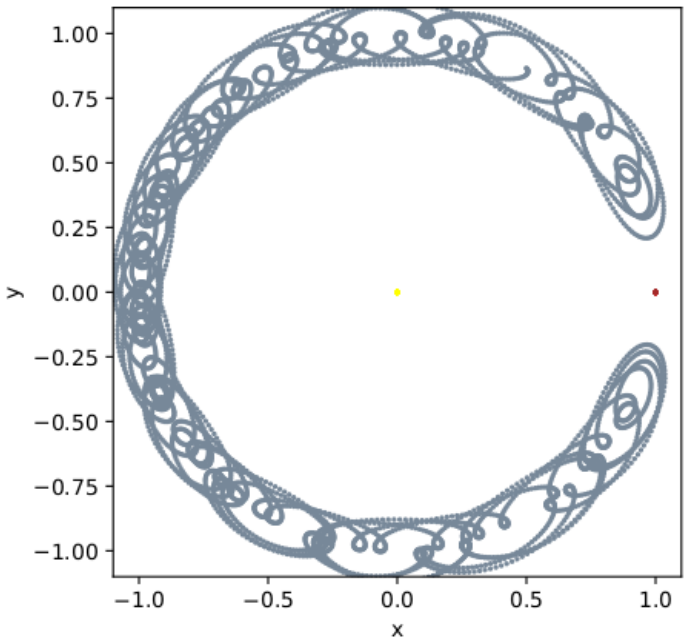

Figure 17: Slowing down more at L4. Simulation ran for 1000 Jovian years. Input file was `inputL4_speeddown_horsie.dat`, output was `out_L4_speeddown_1000Ja_horsie.dat`.

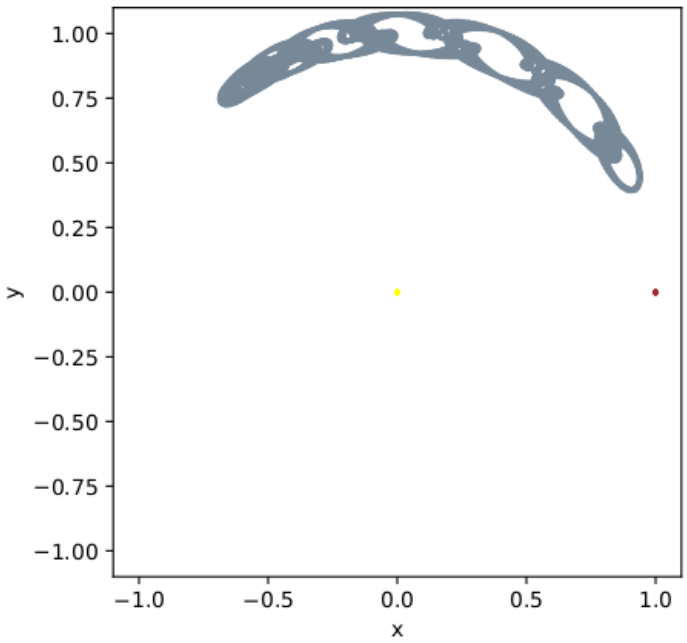

Figure 18: Speeding up at L4. Simulation ran for 1000 Jovian years. Input file was `inputL4_speedup_toadie.dat`, output was `out_L4_speedup_1000Ja_toadie.dat`.

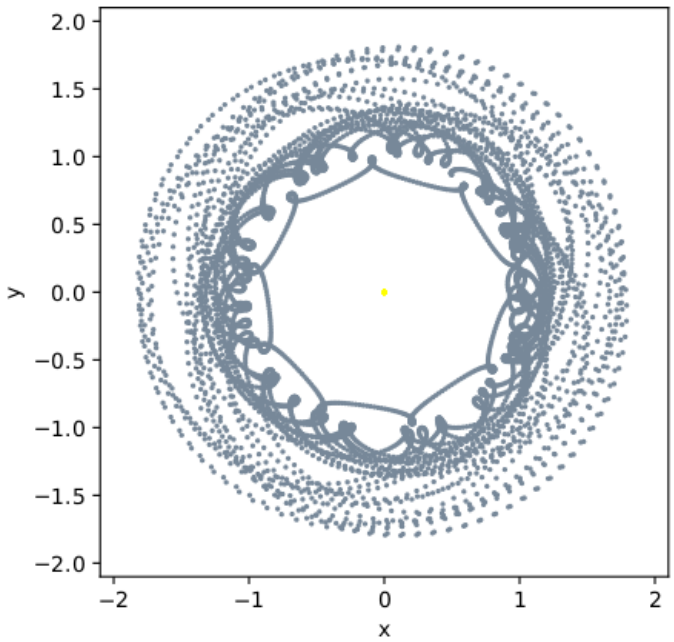

Figure 19: Speeding up more at L4. Simulation ran for 1000 Jovian years. Input file was `inputL4_speedup_around.dat`, output was `out_L4_speedup_1000Ja_around.dat`.

Figure 20 shows that the orbit at L5 is stable, too.

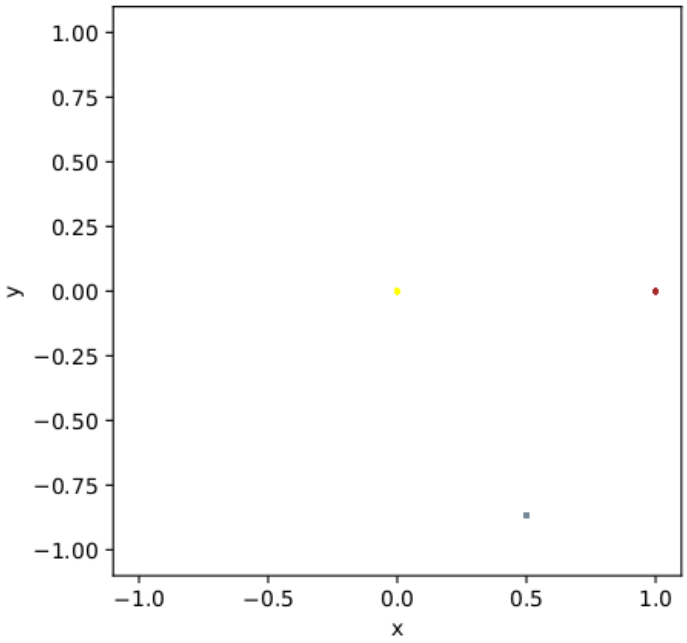

Figure 20: Behaviour at L5. Simulation ran for 1000 Jovian years. Input file was `inputL5.dat`, output was `out_L5_1000Ja.dat`.

5 Conclusions

Overall the project succeeded well.

The results were close to the expected. Orbits at L4 and L5 were stable and there wasn't anything wholly unexpected about the orbits at L1 through L3. While theoretically the satellite would remain stationary if placed at exactly one of the saddle points, it is in reality extremely difficult, so it's not surprising that they weren't stable. I tried many various starting conditions to get the kind of orbit I got in figure 10, so I'm glad that finally one of them gave some results.

The methods I used were quite efficient. A leapfrog integrator is a very simple, yet extremely effective way to integrate orbits of celestial bodies. It is also time-reversible, meaning that one can integrate forward n steps, and then reverse the direction of integration and integrate backwards n steps to arrive at the same starting position. It is also symplectic, which implies that it conserves the (slightly modified) energy of dynamical systems.

It would be easily possible to integrate many different starting positions and speeds to study many more different kinds of scenarios, but I'm happy with the ones I did.