Lagrange Points

Lucy O

September 2024

1 Introduction

My aim for this investigation was ***How can a convergent algorithm be used to find the location of Lagrange points and how do the predictions made by an algorithm compare to the predictions made using theoretical equations and empirical data?*** I chose this investigation because I knew that I wanted to do something space related, and it has real life applications in space exploration, particularly for satellite locations. Lagrange points are points in a two body system where an object of negligible mass can stay in the same position relative to the other objects in the system. Lagrange points can be found in any system that involve two bodies orbiting around their barycentre in a stable system. Examples of systems with Lagrange points include the sun-earth system and earth-moon system.

In my initial research I found out about the historical context of Lagrange points - Euler is credited with discovering three of the Lagrange points, however he did not mention them many times. Rather he is credited with them because he developed the equations that allowed for the calculation of the position of three co-linear bodies that were attracted to each other with a force proportional to their mass and the inverse of the distance between them. This paved the way for French-Italian mathematician and physicist Joseph-Louis Lagrange's later research into finding specific points that would be stable for a two body system with a much smaller third mass.

I also found out about how one of the Lagrange points (known as L3) can be found by extending the line through the body with the smaller mass to the body with the larger mass to the other side. This means that theoretically, a body of equal mass could be orbiting the sun opposite the earth at the same rate, meaning that we may not be able to directly observe it. This led to the science-fiction trope of Planet X, or counter-earth, which began with a movie where an astronaut thought they had crash landed on earth, but really another planet opposite earth where everything way backwards. While this may not be the case, it's interesting how Lagrange points have worked their way into fiction.

2 Mathematical Strategies

A number of mathematical strategies were used to create the simulation and validate its results at all stages of the program. In this section there are some of the mathematical techniques I used in this investigation.

2.1 Equations and Algebra

I used algebra in this investigation for rearranging, understanding and checking physical equations. For this project, I used the equations in Newton's Laws, and Kepler's laws of planetary motion, using Newton's Law of Universal Attraction to calculate gravity rather than Einstein's Theory of General Relativity. This is mainly because Newton's Gravitation gives nearly the same results, especially when the objects are moving at non-relativistic speeds (as planets usually do). Some equations I used and modified are:

Newton's Law of Universal Gravitation

This is Newton's law that describes how every object is attracted to one another. This was used for calculating how points around the barycentre would be attracted to each other. The equation for this law is:

$$F_g = G\frac{m_1 m_2}{R^2}$$

Where F_g is the force of gravity m_1 and m_2 are the masses of the two objects, R is the distance between them and G is the gravitational constant, experimentally determined to be $6.67 \times 10^{-11}\ Nm^2kg^{-2}$. However, as we are calculating the gravitational field rather than the force for a specific object, and $F = ma$, we can divide both sides of the equation by the mass of the object, and we are left with:

$$a_g = G\frac{m}{R^2}$$

Where m is the mass of one of the objects in the two body system, and a_g is the acceleration of the third object due to the gravitational force exerted on it by the mass of the object in the system.

In order to correctly calculate the velocity of objects in orbit, I had to calculate the gravity exerted on body 1 by body 2 (and vice versa) as though it was coming from the barycentre, in order to correctly simulate their orbits. This meant creating an apparent mass called m_{app}, which each body behaved as though it was orbiting. This equivalence can be represented as:

$$a_1 = G\frac{m_2}{(d_1 + d_2)^2} = G\frac{m_{app}}{d_1^2}$$

Where a_x represents acceleration of body x, d_x represents the distance of body x from the barycentre, and m_x represents the mass of body x. This can be solved for m_{app}, resulting in:

$$m_{app} = \frac{m_2 d_1^2}{(d_1 + d_2)^2}$$

This means that you can now calculate the acceleration of body 1 as though the only object exerting a force on it is a mass of m_{app} coming from the barycentre of the system. This can be used to calculate the orbital velocity of body 1.

Centripetal Acceleration

This determines the acceleration of an object moving in uniform circular motion, directed towards the centre of the circle. The equation for centripetal acceleration is:

$$a_c = \frac{v^2}{R}$$

Where a_c is the acceleration of the body directed towards the center of the circle, v is the velocity of the object around the circle and R is the distance from the center of the circle to the object. This is a powerful equation that can be used to calculate orbits of objects. If you substitute in acceleration due to gravity as a_c, you can find the velocity of the object if you know its distance from the barycentre, or the distance from the barycentre if you know its velocity.

In this investigation, I used centripetal acceleration to check if the point is a Lagrange point. For any given point, you can determine its velocity and distance from the barycentre, meaning you can find what its centripetal acceleration should be if it follows a circular orbit. Then you can calculate the acceleration due to the gravitational forces from both of the bodies and compare that to the value for centripetal acceleration. If they are the same, then that point is a Lagrange point. If they aren't, you can use the difference to help point you to the Lagrange point.

Period of Orbit

In order to find the period of the orbit, I needed to use formulas that would calculate the orbit and then calculate the speed of the orbit. The equations I used for this were $v = \frac{d}{t}$ and $C = 2\pi r$. First I calculated the orbital velocity of one of the two main bodies using a modified version of Newton's Law of Gravitation, and used that body's distance from the barycentre to calculate the circumference, or the path of its orbit. From there, I used the rearranged velocity equation to calculate the time.

This was important to calculate because in a system such as this, the third point must have the same orbital period. This is because it must be in order to have a stable orbit. If it didn't have the same orbital period, it would eventually be sucked into one of the bodies because of its gravity (If it was going too slow, from its perspective the bodies would come from behind and it would be sucked into their gravitational pull. If it was going too fast, it would "catch up" with the bodies and also be sucked into their gravitational pull.)

2.2 Geometry and Trigonometry

Geometry is used in almost any problem involving physical modelling, including this one. During this investigation, I used circles to model the orbits of bodies, trigonometry to calculate vectors and the distance formula to calculate the distance between bodies and find their gravitational pull.

In order to calculate the distance between bodies, I assigned each body a Cartesian coordinate so that I could use the regular distance formula, or:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

It was important to be able to use a formula like this because the distances between bodies really affected their gravitational attraction to each other.

One of the applications of using this distance formula was triangulating the position of a point so I could calculate its acceleration in two dimensions. Using uniform circular motion to model the orbit of a point, I calculated the magnitude of the acceleration towards the centre needed to keep the object in a stable orbit. However this gave a magnitude, but not a direction. This meant I had to find an angle so I could use trigonometry to resolve the vector into component form. In order to do this:

$$\theta = \tan^{-1} \frac{bary_y - point_y}{bary_x - point_x}$$

$$\overrightarrow{a} = a\cos\theta\,\hat{\imath} + a\sin\theta\,\hat{\jmath}$$

Where $\overrightarrow{a}$ is the vector for acceleration, $\hat{\imath}$ and $\hat{\jmath}$ are the unit vectors ($\hat{\imath}$ is in the positive x direction and $\hat{\jmath}$ is in the positive y direction), *bary* refers to the coordinates of the barycentre and *point* refers to the coordinates of a given point. This gives acceleration as a vector directed from the point towards the barycentre.

3 Method

For this project, I decided that I would use Python programming to find the Lagrange points through simulation using a convergent algorithm. First I will set up a binary system of two mutually orbiting bodies (such as the earth and sun) that follows the laws of physics outlined in the last section. Then I'll calculate the net force on a single point in the system, and move the point by a gradually decreasing amount until the net force on it is approximately equal to the centripetal force needed to keep it in a circular orbit.

Once I've done this, I will simulate this for a number of points in random locations in the system and find the locations where they come to rest and compare them with the Lagrange points.

The first step to programming this was creating a function that would return all the physical quantities that I could need for the simulation. This did most of the maths in the code. Using the maths in the last section, I calculated the centripetal force needed to keep the point in orbit, the scalar and vector values of gravity from the larger body, the second body and the total of them both, the velocity of the point and the distance of the point from the barycentre.

After I worked on creating an algorithm that would move individual points to a Lagrange point. I did this by calculating the difference between the centripetal acceleration and the net gravity on the point using the functions I had made earlier. This would give me a vector that pointed towards a point with a smaller difference. Because each time the point moved to a location with a smaller difference, it would mean that if I moved the point by an amount proportional to the size of the difference, the algorithm would naturally converge and move the point by a smaller amount each time. Because the differences were often very small, I found that if I multiplied the difference by 10 million before adding the vector to the position of the point, it would move it at an amount large enough to be substantial, but small enough to converge properly.

When this worked and I had found a good coefficient for the difference, I generated a hundred random points around the binary system and repeated the algorithm for each point. However, these points would only converge to two different locations when there are five known Lagrange points. I found out that this was because the Lagrange points the algorithm had not found were the co-linear Lagrange points - it was only if they were exactly co-linear with the two bodies that they could be found, because any other position would mean that the move to the co-linear position it would have to move *away* from its vertical difference. To solve this problem, I generated 30 random points that were co-linear with the two bodies. Because these points were already co-linear, there was no net gravitational force - and therefore difference - that pointed up or down, meaning that they would remain co-linear during the simulation.

Once I had factored this in, the algorithm roughly found all five Lagrange points. To visualise how the Lagrange points were found, I made a gif, named sample.gif in the files attached.