Investigating the Stability of Planetary Dynamics within Binary Star Systems in Search of Habitable Planets

Dominic van der Pas (879132)

Theoretical Physics - Swansea University.

April 27, 2020

Abstract

This computational project investigates the stability of an exoplanet's orbit within the habitable zone of a binary star system, and concludes that binary star systems can support habitable exoplanets. However, there is not enough evidence to support substantially redirecting the search away from solitary stars. The results suggested that S-type orbits were *more* stable when the planet was in orbit around the star of smaller mass. Stable S- and P-type orbits were successfully simulated, but only S-type orbits were found to maintain stability within the habitable zone. T-type orbits were not considered because no examples of stable orbits have been discovered to date.

Contents

1 Introduction

It is known from Newton's third law that the total momentum of a system in isolation is always conserved, and therefore there must exist a point of concentration that represents the mass of the whole system. This point of balance is known as the *centre of mass* or *barycenter*, and must be positioned on the line that connects the centres of two or more bodies. Throughout this report, it will be assumed that the both planets and stars are spherically symmetrical, and act gravitationally as point masses.[1] In addition, all orbits have been confined to a common plane on the x- and y-axes, with no component of motion in the z-direction.

Note that throughout this report, Earth mass and Solar mass will be represented by $M_\oplus$ and $M_\odot$ respectively.

1.1 Restricted Three Body Problem

The restricted three-body problem is the simplest variation of all three-body problems due to one of the bodies having an infinitesimal mass. Since such a negligible body leads to essentially zero perturbation of the other two massive bodies, the two massive bodies will appear and react exactly as they would in a two body problem.[2]

Rigil Kentaurus, otherwise known as Alpha Centauri, is technically a tertiary star system, but acts more like a binary system consisting of two similarly massive and relatively-close stars, Alpha Centauri A and Alpha Centauri B. The third star, Proxima Centauri, is much further away from the other two stars, and its far lower mass means that it has little effect on the binary part of the system. In fact, Alpha Centauri A and B are only separated by 23 AU, whereas Proxima Centauri is roughly 12000 AU away.[3, p.1] As a result, this report will focus specifically on treating Alpha Centauri as a restricted three-body system; if the mass of Proxima Centauri is taken to be negligible, then Alpha Century is the closest binary star structure to the Solar System.[4]

1.2 Habitable Zone

Within this report we define a habitable planet as one that already harbours simple/complex life, or one that has a suitable environment with the potential to sustain future development of life.

There has been some variance in definitions over the past 60 years, but the circumstellar habitable zone, known simply as the habitable zone, has generally been defined as the circumstellar region of space around a star in which a terrestrial planet ($0.1 \leqslant M \leqslant 10M\oplus$) has an orbit stable enough to allow for the existence and maintenance of liquid water on its surface, given sufficient atmospheric pressure; such planets are known as 'aqua planets'.[5, p.4] [6, p.443] [7, p.2]

Aqua planets that steer too far away from a star tend to experience a freezing effect, while others such as Venus come too close, experiencing a boiling effect that leads to the filling up of the atmosphere with water vapour, which in turn amplifies the greenhouse effect and results in further evaporation of the oceans.[6, p.443] Therefore, the search for planets with the potential to sustain complex life is narrowed down to habitable zones throughout the universe.

Although the habitable zone is relatively small, it could potentially be scaled in size by roughly a factor of three (see figure 1) to encompass 'land planets', should they be considered habitable. These are typically 'dry' planets with no oceans and vast deserts, for which their habitability depends on the presence of locally abundant water.[6, p.443]

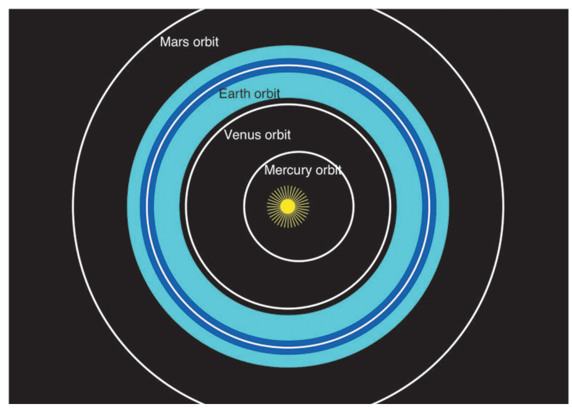

Figure 1: A section of the Solar System depicting the region in which liquid water can be maintained within an atmospheric pressure of 1 bar. The dark blue represents the habitable zone for 'aqua planets', and the light blue represents the habitable zone for 'land planets', which is approximately three times the radius. The yellow represents the Sun, and the white circular lines correspond to the orbits of Mercury, Venus, Earth and Mars.[6, p.459]

1.3 Exoplanets

The term exoplanet, sometimes interchanged with the term extrasolar planet, refers to a planet that orbits any given star other than the Sun, and is therefore located outside of our Solar System.[8] There are a total of 4151 exoplanets that have been discovered and confirmed as of 16th April 2020, whereby the majority of exoplanet recordings have either a radius $2R\oplus <$

$R < 6R\oplus$ or a mass $M > 300M\oplus$.[9] These statistics are subject to an exoplanet being a free-floating mass that is no greater than 30 Jupiter masses.[10]

The existence of exoplanets orbiting binary star systems has been proven by observation, but records suggest that exoplanets are far more likely to be orbiting a *singular* star, with only 144 and 36 within binary and higher multiple-star systems respectively.[11, p.1]

Discovered in 2016 within Rigil Kentaurus, the nearest exoplanet, *Proxima Centauri b*, was identified as the only planet orbiting the closest star to the Solar System, Proxima Centauri. With a mass of $1.3M_\oplus$ and a period of 11.2 days, it is believed to be a rocky-based planet that is located within the habitable zone of Proxima Centauri, which is only around 4.2 light-years away from the Sun; very close when considering the scale of the universe.[4] Despite orbiting a singular star, Proxima Centauri b is of particular interest, given the fact that it is not only a rocky-based, roughly Earth-massed exoplanet within a habitable zone, but is also the closest exoplanet to the Solar System on record.

Most exoplanets have been found to orbit main-sequence stars similar to that of the Sun, but the majority of these discoveries have been observed to exist under significantly different conditions. Exoplanets are usually far more massive than the Earth, but this evidence is partially distorted by the fact that the most commonly chosen methods of detection, such as Doppler technique and transit, favour the discovery of large planets in close orbit to a star. The composition of exoplanets also differs to that of the rocky terrestrial planets since they are usually gaseous, which aligns with their larger mass when we observe that giant planets within the Solar System are all gaseous. Other regular properties of exoplanets include largely elliptical orbits, which would suggest that the Solar System is not particularly common in that planetary orbits within it are near circular.[8] However, for the simplification of this computational project, and the purpose of identifying *habitable* planets, we shall assume that all orbits are perfectly circular throughout. It is expected that the distance of a habitable planet from its star(s) would experience little fluctuation, thus avoiding extreme temperature changes.

1.4 Binary Stars

There is some uncertainty within the scientific community when it comes to the estimation of the percentage of star systems that are binary, but the most common assertion is that over 50% of all main sequence stars exist as binary or multiple star systems.[12, p.1] However, these assertions may be subject to significant sample bias in that binary star systems are often much more massive and bright than solitary stars, and are therefore more easily detected. Analysis with greater precision suggests that the more common fainter stars are likely to be singular, and could potentially compose up to two thirds of all stellar systems within the universe.[13, p.L63]

Approximately 50-60% of binary systems allow for both the formation and long-term stability of Earth-like planets, with the majority, 40-50%, being wide enough to support S-type orbits, and 10% being narrow enough to support P-type orbits (S- and P-type orbits explained in the following section). Given that the Milky Way contains in excess of 100 billion star systems, an abundance of Earth-like planets is expected to exist within our own galaxy.[14, p.282]

The size of a star plays an important role when it comes to the search for complex life. In particular, The Sun is relatively large, but not so massive that it has too short a life-span. Many super-massive stars would not live long enough to allow the evolution towards complex life, even if the planet were perfectly habitable.[15]

1.5 Types of Stable Orbits

From a dynamical point of view, there are *three* types of stable orbits within binary star systems, which are commonly referred to as either S-type, P-type or T-type (see figure 2). The Satellite-type (S-type) involves a planet in stable orbit around only one of the two stars, whereas the Planet-type (P-type) involves a planet being in a distant, yet stable orbit around both stars.[16, p.4] The third type of stable orbit occurs only in binary systems where the masses of each star are significantly different. This T-type orbit involves a (Trojan) planet sharing an orbit with the smaller star whilst being gravitationally locked into a fixed position. These positions are known as Lagrangian points L4 and L5, which occur 60° in front of the smaller star, and 60° behind. However, since no planet has yet been observed to maintain such an orbit, this report will focus solely on the possible habitability of exoplanets with Satellite- and Planet-type orbits.[17]

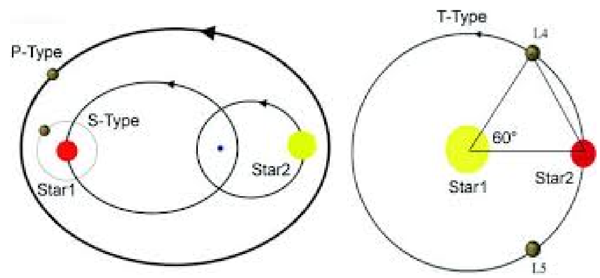

Figure 2: Two binary star systems portraying three different types of stable orbits (all anti-clockwise).
Left: displays a P-type orbit around both stars, and an S-type orbit around the red star 1. The black dot represents the system's centre of mass.
Right: displays a T-type orbit along with the Lagrangian points L4 and L5. [11, p.2]

In order to evaluate the habitability of an exoplanet, it must first be considered that a planet is only stable if it is capable of constantly sustaining orbital parameters such as inclination, semimajor axis and eccentricity. More precisely, a planet is stable if the slight variations in orbital parameters fluctuate in a sinusoidal fashion as opposed to growing exponentially. A planet will incur some instability if perturbations cause the orbital parameters to change significantly with the risk that it either overcomes the system's gravitational field, or undergoes a collision. [16, p.4]

It is already known that the rigid limits of orbital range for higher multiple star systems (three or more) mean that it would be unlikely to find stable orbits of Earth-like exoplanets within the habitable zone, and so this report will not pursue any further the evaluation of higher multiple star systems. [15]

1.6 Aim of Project

This project aims to model the celestial mechanics of binary star systems, and to investigate the feasibility of planets orbiting within the habitable zone. Furthermore, the aim is to address the two most fundamental questions when considering the feasibility of habitable planets existing within *binary* star systems; can an exoplanet maintain a stable orbit within a binary star system, and if so, could such an orbit remain stable within the habitable zone?

In answering these questions this report aims to conclude upon whether it is more effective to focus attention on both singular and binary star systems, or exclusively the former, in search of life outside the Solar System.

2 Theory

2.1 From Kepler to Newton

Johannes Kepler was the first to empirically determine the three laws of planetary motion, but it was Isaac Newton who later attempted and succeeded in defining the underlying physical processes that govern motion. Newton (among others) identified that, for a body in uniform circular motion, its acceleration must point towards the centre of the circle, and further deduced that the gravitational forces of two massive bodies were inversely proportional to the square of their separation. [18] Newton made this known as the law of universal gravitation, which states that the force acting on body 2 from body 1,

$$\boldsymbol{F}_{12} = \frac{Gm_1m_2}{r_{12}^2}\hat{\boldsymbol{r}}, \tag{1}$$

where G is Newton's gravitational constant, m_1 and m_2 are the masses of body 1 and 2 respectively, r_{12} is the magnitude of the separation between the two bodies, and $\hat{\boldsymbol{r}}$ represents a unit vector.

This gravitational force that acts on a planet orbiting a star is equal to the centripetal force,

$$\boldsymbol{F}_c = \frac{mv^2}{\boldsymbol{R}}, \tag{2}$$

where m is the mass of the planet, R is its orbital radius, and v is its velocity, which is at right-angles to the orbital trajectory. [1]

2.2 Gravitational Constant

By taking a two-body system consisting of identical masses M that are equal and opposite in distance from the origin (see figure 3), the gravitational and centripetal forces, defined via equations (1) and (2) respectively, can be equated such that

$$\frac{Mv^2}{\boldsymbol{R}} = \frac{GM^2}{r^3} \cdot \boldsymbol{r}, \tag{3}$$

and rearranged for the gravitational constant

$$G = \frac{v^2r^3}{M(\boldsymbol{R}\cdot\boldsymbol{r})} = \frac{v^2(2R)^3}{2M(\boldsymbol{R}\cdot\boldsymbol{R})} = \frac{4v^2R}{M}, \tag{4}$$

where R is radius of orbit, and v is velocity. Note that the separation was taken to be $r = 2R$, as seen within figure 3.

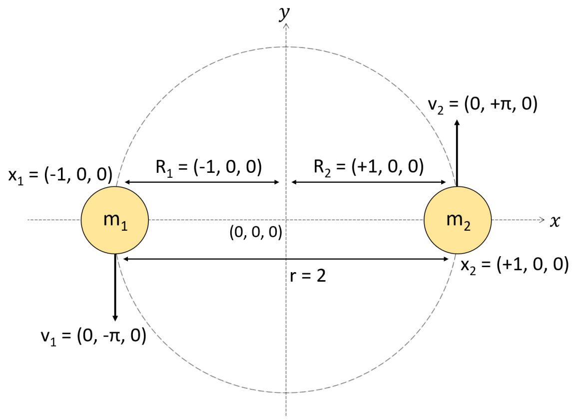

Figure 3: A simple binary system consisting of two bodies that are equidistant from the origin, and moving solely in the xy-plane, with equal and opposite initial velocities of pi. R_1 and R_2 are the radii of orbit, m_1 and m_2 are the masses of each body, r is the seperation of the two bodies, x_1 and x_2 are 3d position vectors, v_1 and v_2 are 3d velocity vectors.

By taking distance to be the circumference of a *circular* orbit $2\pi R$, and time to be the period of time elapsed over one complete orbit $2T$, the speed-distance-time formula,

$$v = \frac{\pi R}{T}, \tag{5}$$

can be substituted into equation 4 to express the gravitational constant in terms of orbital radius R, bodily mass M and time period T,

$$G = \frac{4\pi^2 R^3}{MT^2}. \tag{6}$$

After expressing the bodily mass, orbital radius and time period as solar masses, astronomical units and years respectively, such that M, R and T are numerically equal to 1, the universal gravitational constant

$$G = 4\pi^2 \text{ AU}^3 \ M_\odot^{-1} \text{ year}^{-2}. \tag{7}$$

2.3 Velocity

The gravitational and centripetal forces, defined via equations (1) and (2) respectively, can be equated such that

$$\frac{m_1 v_1^2}{\boldsymbol{R_1}} = \frac{G m_1 m_2}{r^2} \cdot \hat{\boldsymbol{r}}, \tag{8}$$

where $r = |R_2 - R_1|$ is clear in figure 3.

Upon rearrangement for velocity, it is found that

$$v_1 = \frac{\sqrt{G m_2 |R_1|}}{r}, \tag{9}$$

and by similar logic

$$v_2 = \frac{\sqrt{G m_1 |R_2|}}{r}. \tag{10}$$

3 Project Resources

Hardware: Lenovo ThinkPad X1 laptop (Intel i7 core, 16GB RAM) connected to a Dell 24inch monitor and computer mouse.

Software: Spyder (Python 3.7) for computational simulations, and Atom (LaTeX) for producing this report.

Research: Articles (via Google Scholar), online books, printed books and Google's Search Engine.

4 Method and Code Validation

4.1 Initial Method

The first consideration made before even starting to write the code was about the time it would take to run each simulation. In an effort to optimise convergence and computation time, all units were taken to be astronomical, such that mass was measured in solar masses, position in astronomical units (distance between The Sun and Earth), and time in years.

Python's SciPy library was chosen as an all encompassing import for mathematical operations since it offered most of the required NumPy functions for this project and more.

In order to increase the efficiency of validating the code, the most simple of cases was targeted first; a two-body system of equal masses and positions equidistant from the origin, portrayed by figure 3. The velocities were determined by inserting $m_1 = m_2 = |R_1| = |R_2| = 1$, $G = 4\pi^2$ and $r = 2$ into equations (9) and (10), such that $v_1 = -\pi$ and $v_2 = +\pi$. For simplicity, there is only initial velocity in the y-direction.

After the time period was then determined as $T = 2$ via equation (5), all constants and initial vectors were inserted into the program. An array that joined the position and velocity arrays together was then created and later flattened to a single dimension after realising that the Ordinary Differential Equation (ODE) solver wouldn't process the initial parameters in multi-dimensional form.

Next, a function was created, within which the initial position and velocity arrays were sliced and assigned to variables. The magnitude of the separation of the two stars and their individual accelerations were also defined within the function, and lastly the velocity and acceleration arrays were concatenated before returning their value as the output of the function.

In order to manually set and vary the duration of the orbit simulation, as well as the number of interval points, a linearly spaced vector for time was generated. The actual 'meat and bones' of the simulator, the ODE solver, relied upon Python's very useful library function, sci.integrate.odeint(). The role played by the odeint() function was to return an array containing position and velocity vectors for each individual time point, which would then be sliced since only the first six columns were required.

4.2 Plotting Initial Setup

With calculations successfully made, the MatPlotLib library was used extensively to create and customise a specific graphical display of the results, as illustrated in figure 4.

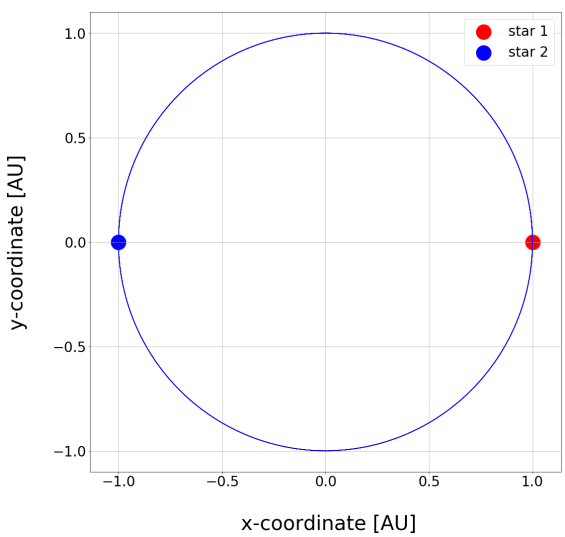

Figure 4: A two-body system consisting of equal mass stars that are equidistant from the origin. See table 2 within the appendix for full details of parameters used.

Once the expected output was achieved, it quickly became apparent that a large number of simulations would be produced, and so it made sense to create a way of automatically saving the graphical output from each simulation, along with its corresponding parameters, in a specific folder. The end product (code) of this initial program can be viewed in the '*Full Code: Two Body, Same Mass*' section within the appendix.

4.3 Adjusting Perspective of Plot

In the next phase of validating the code, the program was re-run with different masses ($5M\odot$ and $M\odot$), but as can be seen in figure 5, the system appeared to be 'drifting' due to some overall momentum.

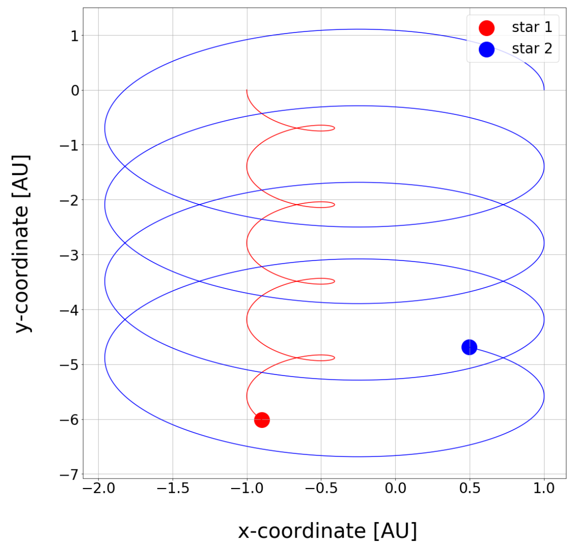

Figure 5: A two-body system consisting of unequal masses that are 'drifting' due to an overall momentum in the system. See table 3 within the appendix for full details of parameters used.

Without actually changing any parameters, the code was updated to incorporate a new centre of mass perspective giving the 'illusion' that the system was not 'drifting' (see figure 6). The specific code used to change the perspective can be viewing in the '*Code Snippet: Centre of Mass Perspective (Two-Body)*' section within the appendix.

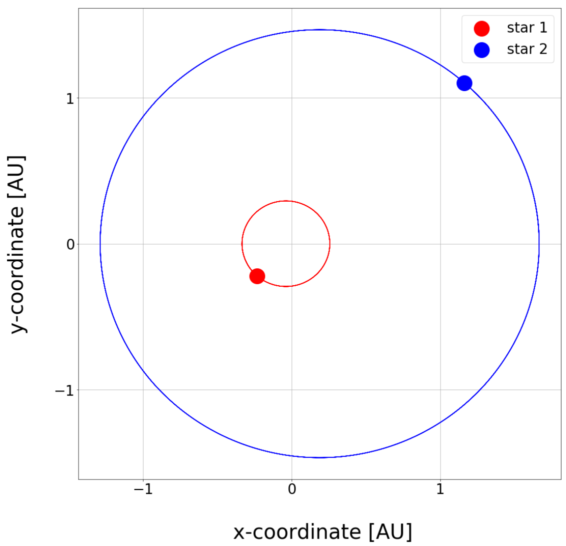

Figure 6: A two-body system consisting of unequal masses without any visible 'drift' from the system; the perspective has been changed to the view the system from the centre of mass. See table 3 within the appendix for full details of parameters used.

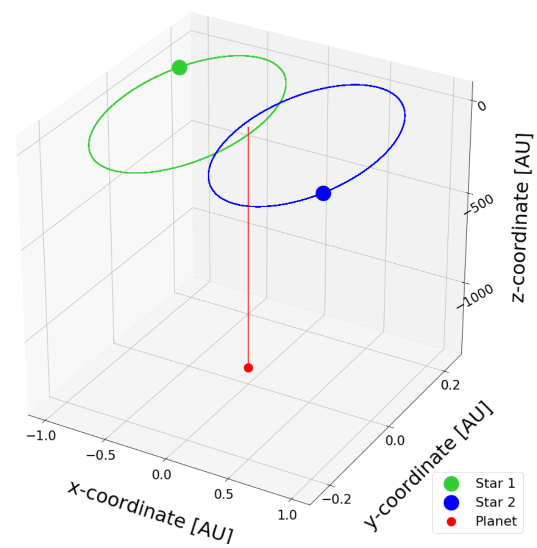

Figure 7: A restricted three-body system consisting of two equal mass stars, and one planet of negligible mass. See table 4 within the appendix for full details of parameters used.

4.4 Testing 3D Plot

The presentation of results was then experimented with to effectively portray a system containing bodies that move in z-direction in addition to the x- and y-directions. The specific code that was used to plot in three-dimensions can be viewed via the '*Code Snippet: 3D Plotting (Three-Body)*' section within the appendix.

It was expected that if a third body of negligible mass was travelling fast enough away from the centre of the system, the remaining system would resemble a simple two-body system, and so figure 7 further validates the code. This is a good example of a restricted three-body system, where a negligible mass has been used instead of a theoretical infinitesimal mass.

4.5 Final 'real' Test: Sun-Earth-Moon

The final stage of code validation involved simulating a real system, Sun-Earth-Moon, with the following known values,

$$r_{\mathrm{EM}} = 0.00257 \text{ AU}^{[19]}, \tag{11}$$

$$\mathrm{M}_{\oplus} \approx 3 \times 10^{-6} \ \mathrm{M}_{\odot}{}^{[20]\,[21,\text{ p.1}]} \tag{12}$$

and

$$\mathrm{M}_{\mathrm{Moon}} = 3.69 \times 10^{-8} \ \mathrm{M}_{\odot}{}^{[22]}, \tag{13}$$

where r_{EM} is the separation between the Earth and Moon, $M_{\oplus}$ is the mass of the Earth, and $\mathrm{M}_{\mathrm{Moon}}$ is the mass of the Moon. The result is illustrated in figure 8.

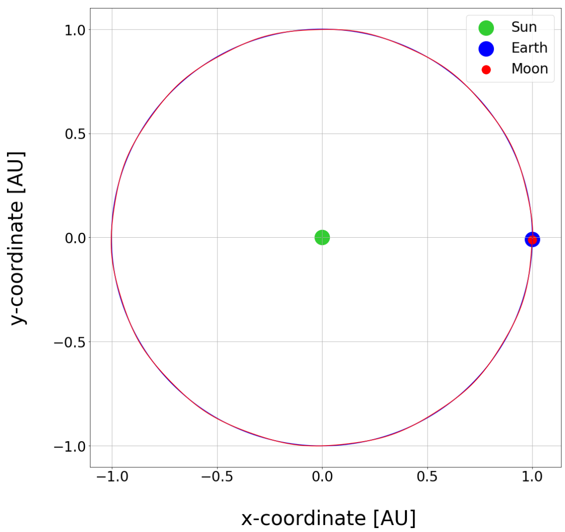

Figure 8: A three-body system consisting of The Sun, Earth and Moon. See table 5 within the appendix for full details of parameters used.

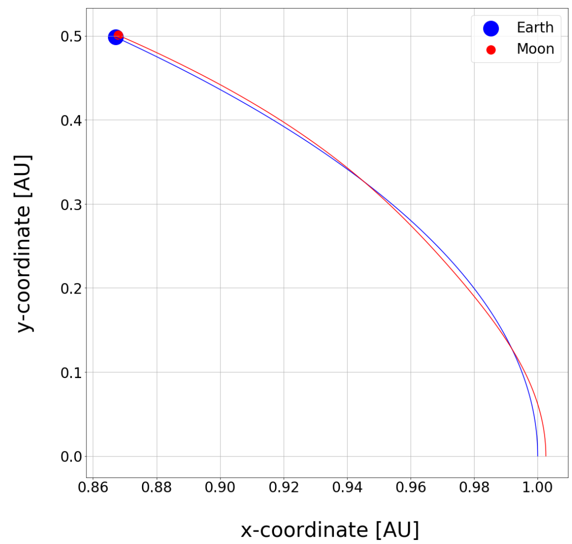

Figure 9: A three-body system consisting of The Sun, Earth and Moon, where The Sun has been excluded from the plot to allow for a magnified view of the Moon completing one full orbit around the Earth. See table 5 within the appendix for full details of parameters used.

The shape was expected since planetary orbits within the Solar System are close to being circular, but in order to see if the Moon is orbiting the Earth correctly, a duplicate 'magnified' plot was created by commenting out the plotting of the Sun, and reducing the time period from 1 to 0.083, which represents *one month* (1 year / 12 months = 0.083). The result is illustrated by figure 9, which clearly shows that the Moon completes one whole orbit around the Earth in one month, and therefore proves that the code written within the appendix of this report for a three-body problem is correct.

It is important to mention however, that this result assumed the Earth-moon orbit to be in the same plane as the Earth-Sun orbit, when in fact the plane of the Moon's orbit is inclined by an average of 5.145° to the plane of Earth's orbit about the Sun.[23] This is clearly portrayed by figure 10.

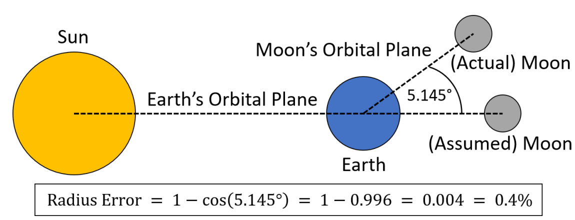

Figure 10: A diagram to show the inclination of the Moon's orbital plane against the Earth's orbital plane, along with the resulting error in radius of the orbit.

Since the resulting radius error is a relatively small 0.4%, it makes sense that figure 9 appeared to present the correct solution; the assumption was a reasonable one.

5 Interpretation of Results

5.1 Unstable Orbit

This is another example of a restricted three-body problem involving two identical massive stars and a planet with negligible mass. The velocities are all of equal magnitude, and the program was initiated in an attempt to observe a typical unstable orbit. It can be clearly observed in figure 11 that the planet permanently leaves the binary star system; in this example, just 3 solar orbits are shown, but larger periods were tested to ensure that the planet did not loop back into orbit.

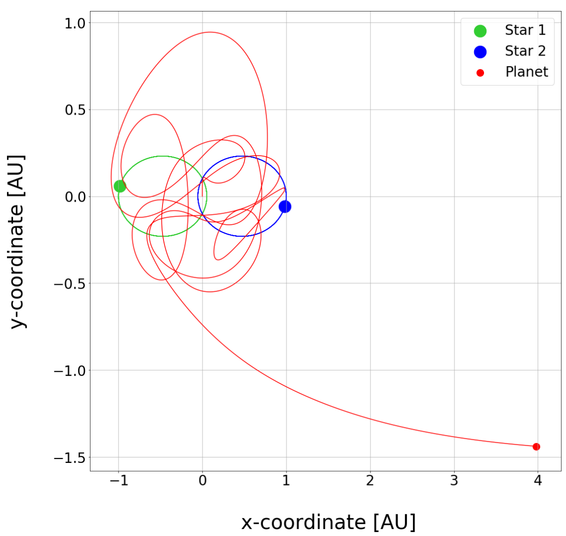

Figure 11: A restricted three-body system consisting of two equal mass stars, and one planet of negligible mass. The planet leaves the system permanently. See table 6 within the appendix for full details of parameters used.

5.2 S-Type Orbit Around Larger Star

This report investigates stable planetary orbits within a known binary star system, Alpha Centauri. Technically Alpha Centauri is a tertiary star system, but it shall be treated as a restricted three-body system since Proxima Centauri is much smaller in mass, and a lot further away from the other two larger stars. The third body within this 'binary' star system is given an Earth-like mass to represent a potentially habitable planet. In order to successfully simulate such a system, the following additional known values need be known,

$$\mathrm{M_A} = 1.100\, M_\odot \text{ [24, p.L11]} \,, \tag{14}$$

$$\mathrm{M_B} = 0.907\, M_\odot \text{ [24, p.L11]} \,, \tag{15}$$

$$T_{\mathrm{AB}} = 79.910 \text{ years [25, p.283]} \tag{16}$$

and

$$r_{\mathrm{AB}} = 23 \text{ AU [3, p.1]} \,, \tag{17}$$

where $\mathrm{M_A}$ and $\mathrm{M_B}$ are the masses of Alpha Centauri A and B respectively, T_{AB} is the time period required for one complete orbit by Alpha Centauri A and B around their corresponding centre of mass, and r_{AB} is the separation between Alpha Centauri A and B.

To begin with, a plot (figure 12) was created using a smaller time period; the aim being to present clearly the orbital pattern of an S-type orbit before going on to obtain more precise results.

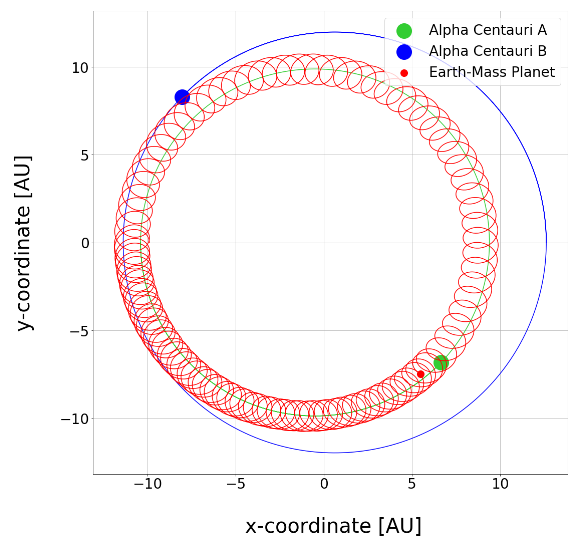

Figure 12: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. Plotted with a relatively low time period for clarity, the planet has an initial velocity of 10.5 AU/year, and is in an S-type orbit around the larger star, Alpha Centauri A. See table 7 within the appendix for full details of parameters used.

The result for an Earth-like planet travelling with an initial velocity of 10.5 AU/year around Alpha Centauri A was then repeated for a much larger time period (figure 13), which made clear that a *stable* orbit occurs under these conditions.

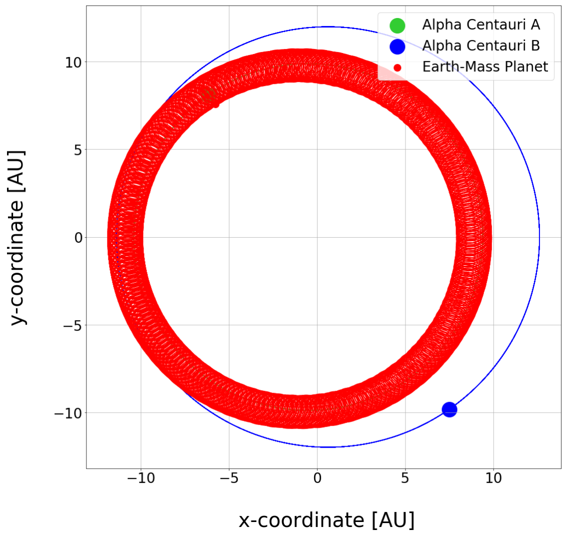

Figure 13: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet has an initial velocity of 10.5 AU/year, and is in an S-type orbit of radius 0.97(5) AU around the larger star, Alpha Centauri A. See table 7 within the appendix for full details of parameters used.

Next, the same conditions were simulated with the planet initially travelling at 11 AU/year, and as can be seen from figure 14, the orbit is more significantly more perturbed by the stars, but still remains a stable.

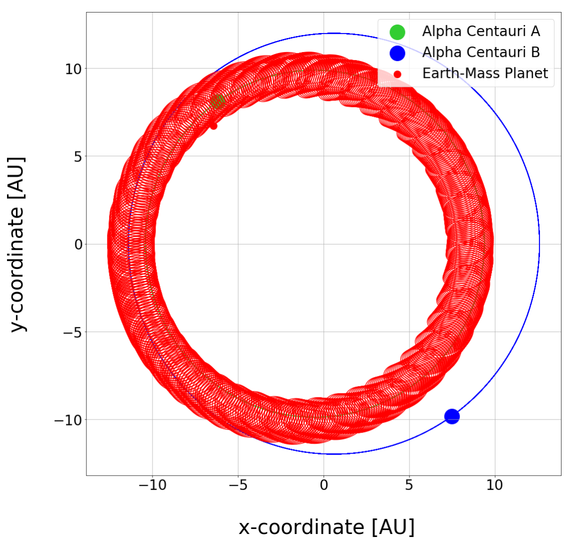

Figure 14: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet has an initial velocity of 11 AU/year, and is in an S-type orbit of radius 1.33(7) AU around the larger star, Alpha Centauri A. See table 8 within the appendix for full details of parameters used.

A final repeat under the same conditions was simulated with the planet initially travelling at 11.5 AU/year, and resulted in an orbit that is clearly *less* stable (see figure 15), but shows little signs of becoming detached from the system. However, if it were possible to increase the time period by a substantial amount, the planet may well show signs of instability.

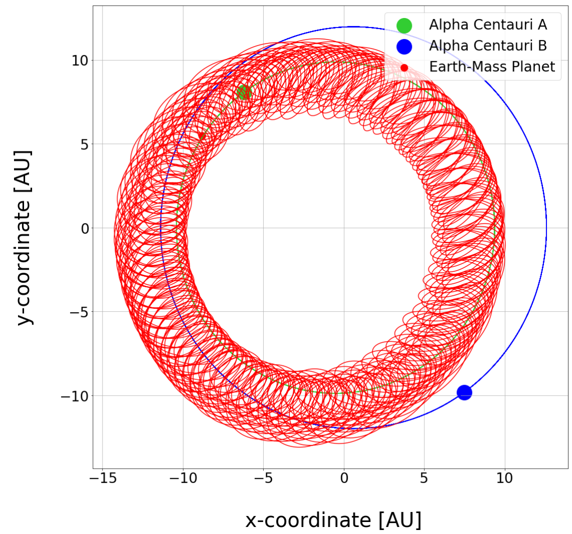

Figure 15: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet has an initial velocity of 11.5 AU/year, and is in an S-type orbit of radius 2.28(3) AU around the larger star, Alpha Centauri A. See table 9 within the appendix for full details of parameters used.

5.3 S-Type Orbit Around Smaller Star

The simulation of an Earth-like planet travelling with an initial velocity of 10.5 AU/year around Alpha Centauri B, produced the *most* stable orbit out of all evaluated results within this report. Since the radius of the S-type orbit in figure 16 is clearly the smallest, it would suggest that the smaller the radius, the more stable the orbit. However, there will be a point at which the radius is too short, resulting in a likely collision between the planet and the star.

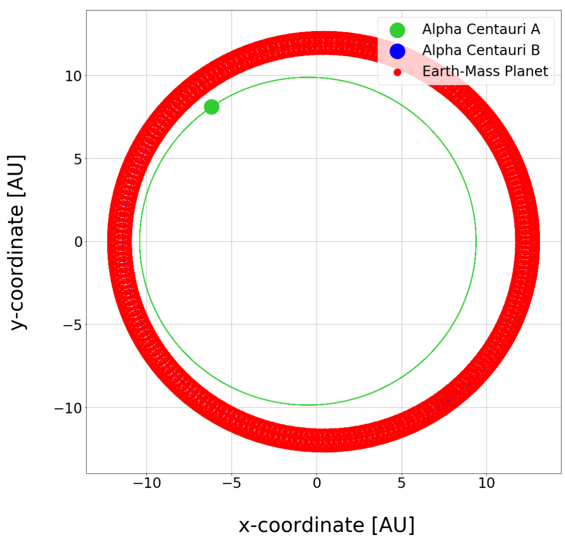

Figure 16: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet has an initial velocity of 10.5 AU/year, and is in an S-type orbit of radius 0.700(2) AU around the smaller star, Alpha Centauri B. See table 10 within the appendix for full details of parameters used.

Next, the same conditions were simulated with the planet initially travelling at 11 AU/year. Figure 17 appears to have become slightly more perturbed, but far less so than that seen in figure 14.

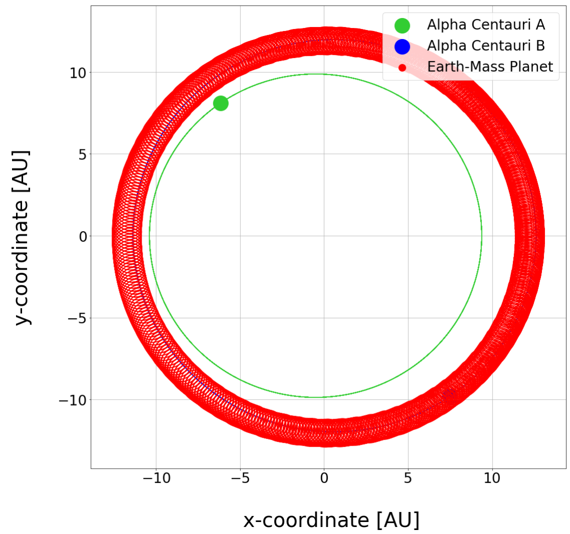

Figure 17: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet has an initial velocity of 11 AU/year, and is in an S-type orbit of radius 0.87(2) AU around the smaller star, Alpha Centauri B. See table 11 within the appendix for full details of parameters used.

Having carried out the final repeat under the same conditions with the planet initially travelling at 11.5 AU/year, it is now clear from the addition of figure 18 that the planetary orbits (S-type) around the smaller mass star, Alpha Centauri B, are significantly more stable than those around Alpha Centauri A.

It is important to mention that these results were based on the significant assumption that all orbits are perfectly circular, which is not even remotely close to the current Alpha Centauri binary system; in reality the orbital eccentricity is 0.52.[26] However, since this report is seeking to determine the potential of stable orbits within binary star systems in general, it does not invalidate the story behind the results.

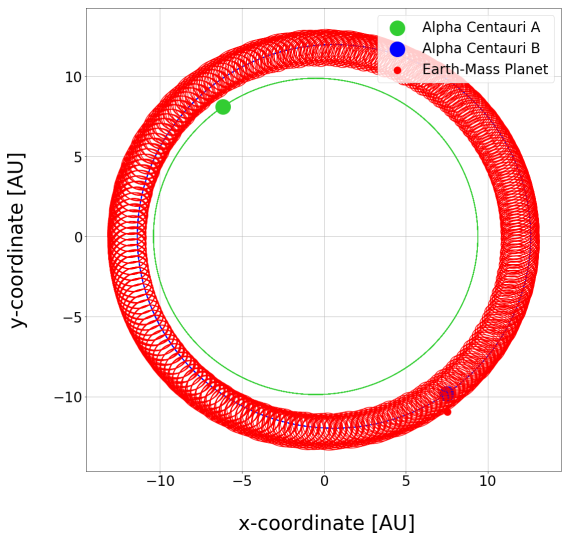

Figure 18: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet has an initial velocity of 11.5 AU/year, and is in an S-type orbit of radius 1.17(1) AU around the smaller star, Alpha Centauri B. See table 12 within the appendix for full details of parameters used.

5.4 Comparing S-Type Orbits

The orbital diameter of planets in S-type orbits were calculated by first finding the average diameter of the entire donut-shape produced by the planet's many orbits, and then subtracting the diameter of the star's orbit. This makes sense because you are left with the orbital radius of the planet on either side of the donut-shape, which combine to make a diameter. However, the diameter was only calculated as a means of identifying the orbital radius (see table 1), and so the diameter was divided by two. The actual code for this can be viewed in the '*Code Snippet: Calculating Radius of S-type Orbits*' within the appendix.

Looking more deeply into the code, the method involved subtracting minimum values from maximum values at points in which either the x or y-coordinates were zero. There were a range of solutions due to the planets not passing through the exact same points upon completion of each orbit, and so the method was an approximation that became sufficiently accurate for large periods of $T = 1000$.

Table 1 provides further evidence that planetary orbits (S-type) are more stable around the smaller mass star, but since this report has focussed on only one real binary star system, it can not be considered conclusive.

Table 1: A direct comparison between S-orbit radii. HZ is the Habitable Zone, v is velocity, 'A Radius' represents the orbital radius of the Earth-mass planet around Alpha Centauri A, and 'B Radius' represents that around Alpha Centauri B.

v [AU/year]	A Radius [AU]	B Radius [AU]
10.5	0.97(5)	0.700(2)
11	1.33(7)	0.87(2)
11.5	2.28(3)	1.17(1)
HZ [AU] [27, p.8]	≈ 1-2	≈ 0.7-1.2

The orbital radii about Alpha Centauri B are all within the predicted range of habitable zone, whereas only two (one if excluding the uncertainty) of the orbital radii about Alpha Centauri A lie within the habitable zone. Also, the uncertainty values for B radius are significantly lower, which agrees with the previously observed conclusion that planetary orbits about Alpha Centauri B are substantially less perturbed than those about Alpha Centauri A.

Another key takeaway from table 1 is that, due to the relatively small variations in initial velocity causing a planet to leave the habitable zone, the likelihood of detecting a planet within such a region has to be very much lower than detecting a planet in some orbit elsewhere.

5.5 P-Type Orbits

After many simulations, it was discovered that the P-type orbits, of an Earth-like planet within a binary star system, appear to only be stable when significantly far from the binary stars, such that they act more like a point mass. As a result, the P-type orbit, illustrated by figure 19, is far from the habitable zone, and so this report finds that P-type orbits should be less of a target than S-type orbits when attempting to detect potentially habitable exoplanets. This might not apply if the stars have extremely high luminosity.

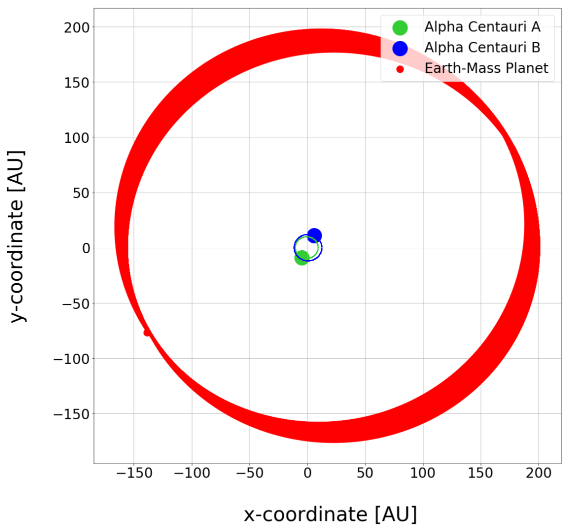

Figure 19: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet is in a P-type orbit around Alpha Centauri A and B. See table 13 within the appendix for full details of parameters used.

Figure 19 appears to suggest that the planet's orbit is gradually changing with time, indicating possible future instability. However, upon re-running the simulation with ten times the period (figure 20), it was very interesting to see that the planet's orbit appeared very symmetrically stable.

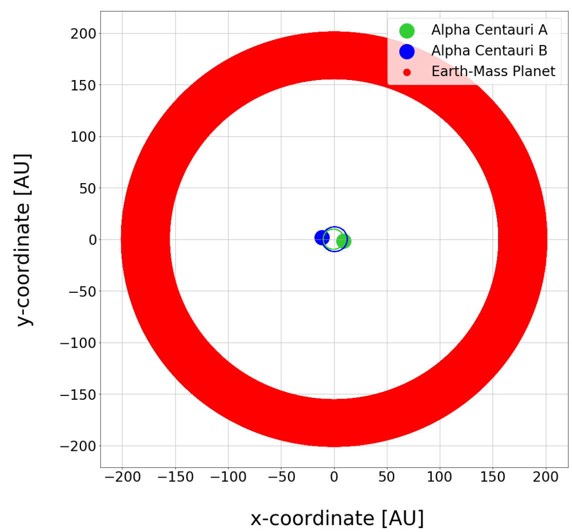

Figure 20: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. The planet is in a P-type orbit around Alpha Centauri A and B. A higher time period has been used here, reaching the limit of the available hardware. See table 13 within the appendix for full details of parameters used.

6 Conclusion

The aim of the project was to ultimately investigate the stability of an exoplanet's orbit within the habitable zone of a binary star system, and conclude upon whether binary star systems are a good place for discovering habitable exoplanets.

In an attempt to achieve these aims, a program was created within Spyder (Python 3.7) to model the celestial mechanics of binary star systems. The code was validated step-by-step (see figures 4, 5, 6 and 7) before being fully verified upon successfully simulating the Sun-Earth-Moon three-body problem (see figure 8). The result was correct in that it portrayed the Moon completing one full orbit over a time period of a month (see figure 9).

After the verification of the code was complete, S-type orbits were simulated for slightly different planetary velocities (see figures 13, 14, 15, 16, 17 and 18). A key takeaway from table 1 is that, due to the relatively small variations in initial velocities causing a planet to leave the habitable zone, the likelihood of detecting a planet within such a region appears to be much less common than detecting a planet in some orbit elsewhere. Furthermore, the results suggested that S-type orbits were *more* stable when the planet was in orbit around the star of smaller mass.

This project successfully simulated more than one stable S-type orbit within the habitable zone (see table 1), and despite successfully simulating a stable P-type orbit (see figure 20), no stable P-type orbits within the habitable zone of Alpha Centauri A or B were identified.

Previous research has suggested that approximately 50-60% of binary systems allow for both the formation and long-term stability of Earth like planets, with the majority, 40-50%, being wide enough to support S-type orbits, and 10% being narrow enough to support P-type orbits. This is consistent with the results obtained, since stable S-type orbits proved much easier to simulate, by trial and error, than P-type orbits.

However, a large number of assumptions were made throughout the project, and so there is plenty of scope for improvement. First, all orbits were assumed to be perfectly circular despite the chosen binary system, Alpha Centauri A and B, being highly elliptical (orbital eccentricity of 0.52); this system was chosen because of it being the closest star system to the Solar System. Additionally, Alpha Centauri is technically a tertiary star system, but was treated as a restricted three-body problem within this report, and since Proxima Centauri is not massless, this treatment is not entirely perfect. It was however, a reasonable treatment, given the fact that Proxima Centauri is 12000 AU from either Alpha Centauri A or B, whose separation is only 23 AU. Another assumption was that the Earth-Moon orbit lies in the same plane as the Earth-Sun orbit, but in actual fact there is an inclination of 5.145° (see figure 10), which would have resulted in a radius error of 0.4%. The final significant limitation within this project was due to the Python program being limited to a set number of intervals because of hardware limits in the computer's memory, CPU and RAM, and so extended time periods could not be evaluated. This limited the ability of determining the length of time that a stable orbit can be sustained.

The project could have been significantly improved by using a computer with much larger memory, more CPU and greater RAM. Perhaps the most major improvement that could be made would be to account for all, or at least some, of the elliptical orbits, particularly that of the Alpha Centauri A and B binary system. In total, there were not many results obtained, relatively speaking, and so a great improvement would be to simply create more simulations, which would increase the reliability of the results.

In conclusion, since 144 out 4151 exoplanets have been discovered within binary star systems, it may be worth considering binary star systems in the search for habitable worlds. Furthermore, since over 50% of all main sequence stars exist as binary or multiple star systems (vast majority of which are binary), and given the fact that galaxies contain approximately 100 billion star systems each, there are clearly lots of exoplanets to be discovered within binary star systems. However, these assertions may be subject to significant sample bias because binary star systems are often much more massive and bright than solitary stars, and are therefore more easily detected. Overall then, this project concludes that binary star systems are good places to be searching for habitable exoplanets in addition to singular star systems, but there is not enough evidence to support substantially redirecting the search away from solitary stars.

7 References

[1] George Ogden Abell. *Exploration of the Universe*, pages 74–77. Holt, Rinehart and Winston, New York ; London, 2nd ed. edition, 1969.

[2] Stanton J. Peale. Celestial mechanics. https://www.britannica.com/science/celestial-mechanics-physics/The-three-body-problem#ref77433. Accessed March 31, 2020.

[3] Paul A. Wiegert and Matt J. Holman. The Stability of Planets in the Alpha Centauri System. *The Astronomical Journal*, 113:1445, Apr 1997.

[4] Erik Gregersen. Alpha centauri. https://www.britannica.com/place/Alpha-Centauri. Accessed: 12-Apr-2020.

[5] Guillermo Gonzalez, Donald Brownlee, and Peter Ward. The Galactic Habitable Zone: Galactic Chemical Evolution. *Icarus*, 152(1):185–200, 2001.

[6] Yutaka Abe, Ayako Abe-Ouchi, Norman Sleep, and Kevin Zahnle. Habitable Zone Limits for Dry Planets. *Astrobiology*, 11(5):443–460, 06 2011.

[7] Ravi Kumar Kopparapu. A Reivsed Estimate of the Occurrence Rate of Terrestrial Planets in the Habitable Zones Around Kepler M-Dwarfs. *The Astrophysical Journal*, 767(1):L8, Mar 2013.

[8] University of Colorado At Boulder. Extrasolar planets. http://lasp.colorado.edu/outerplanets/exoplanets.php. Accessed April 16, 2020.

[9] Nasa Exoplanet Science Institute. Exoplanet and candidate statistics. https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html. Accessed April 16, 2020.

[10] Nasa Exoplanet Science Institute. Exoplanet criteria for inclusion in the archive. https://exoplanetarchive.ipac.caltech.edu/docs/exoplanet_criteria.html. Accessed April 16, 2020.

[11] R. Schwarz, B. Funk, R. Zechner, and Á. Bazsó. New Prospects for Observing and Cataloguing Exoplanets in Well-Detached Binaries. *Monthly Notices of the Royal Astronomical Society*, 460(4):3598–3609, May 2016.

[12] Elisa V. Quintana and Jack J. Lissauer. Terrestrial Planet Formation in Binary Star Systems. *arXiv preprint arXiv:0705.3444*, 2007.

[13] Charles J Lada. Stellar Multiplicity and the Initial Mass Function: Most Stars are Single. *The Astrophysical Journal Letters*, 640(1):L63, 2006.

[14] N. Haghighipour. *Planets in Binary Star Systems*, page 202. Astrophysics and Space Science Library. Springer Netherlands, 2010.

[15] Stars and habitable planets: Stable orbits in binary star systems. http://www.solstation.com/habitable.htm. Accessed: 4-Apr-2020.

[16] Nader Haghighipour, Rudolf Dvorak, and Elke Pilat-Lohinger. Planetary Dynamics and Habitable Planet Formation in Binary Star Systems. *Astrophysics and Space Science Library*, page 285–327, 2010.

[17] Alison Klesman. Can solar systems exist in a binary star system? https://astronomy.com/magazine/ask-astro/2020/01/can-solar-systems-exist-in-a-binary-star-system. Accessed: 27-Mar-2020.

[18] Stanton J. Peale. Celestial mechanics. https://www.britannica.com/science/celestial-mechanics-physics/Keplers-laws-of-planetary-motion#ref77427. Accessed March 30, 2020.

[19] Dr. David R. Williams. Planetary fact sheet - ratio to earth values. https://nssdc.gsfc.nasa.gov/planetary/factsheet/planet_table_ratio.html. Accessed: 15-Mar-2020.

[20] Dr. David R. Williams. Sun fact sheet. https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html. Accessed: 09-Apr-2020.

[21] Astrophysical constants and parameters. http://pdg.lbl.gov/2019/reviews/rpp2018-rev-astrophysical-constants.pdf. Accessed: 12-Feb-2020.

[22] Dr. David R. Williams. Moon fact sheet. `https://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html`. Accessed: 16-Mar-2020.

[23] Fred Espenak. Eclipes and the moon's orbit. `https://eclipse.gsfc.nasa.gov/SEhelp/moonorbit.html`. Accessed: 18-Mar-2020.

[24] F. Thévenin, J. Provost, P. Morel, G. Berthomieu, F. Bouchy, and F. Carrier. Asteroseismology and Calibration of Alpha Centuri Binary System. *Astronomy & Astrophysics*, 392(1):L9–L12, Aug 2002.

[25] Dimitri Pourbaix, David Nidever, Chris McCarthy, and Butler. Constraining the Difference in Convective Blueshift Between the Components of α Centauri with Precise Radial Velocities. *Astronomy & Astrophysics*, 386(1):280–285, 2002.

[26] `http://www.solstation.com/orbits/ac-absys.htm`. Accessed: 12-Apr-2020.

[27] Lisa Kaltenegger and Nader Haghighipour. Calculating the Habitable Zone of Binary Star Systems. I. S-Type Binaries. *The Astrophysical Journal*, 777(2):165, 2013.

[28] Gaurav Deshmukh. Modelling the three body problem in classical mechanics using python. `https://towardsdatascience.com/modelling-the-three-body-problem-in-classical-mechanics-using-python-9dc270ad7767`. Accessed February 28, 2020.

8 Appendix

8.1 Tables

Table 2: Two-body system with stars of identical mass, orbiting each other (sharing same orbit). See figure 4 for graphical representation.

Initial Parameters	Star 1	Star 2
Mass [$M_\odot$]	1	1
Position [AU]	(-1, 0, 0)	(1, 0, 0)
Velocity [AU/year]	(0, -3.14159, 0)	(0, 3.14159, 0)
Start Time: 0	End Time: 5	Points: 10000

Table 3: Two-body system with different mass stars and centre of mass perspective. See figures 5 and 6 for graphical representation.

Initial Parameters	Star 1	Star 2
Mass [$M_\odot$]	5	1
Position [AU]	(-1, 0, 0)	(1, 0, 0)
Velocity [AU/year]	(0, -3.14159, 0)	(0, 7.02000, 0)
Start Time: 0	End Time: 4	Points: 10000

Table 4: Three-body system consisting of two identical mass stars, and one planet of negligible mass. See figure 7 for graphical representation.

Initial Parameters	Star 1	Star 2	planet
Mass [$M_\odot$]	1	1	3.69441e-08
Position [AU]	(-1, 0, 0)	(1, 0, 0)	(0, 0, 0)
Velocity [AU/year]	(0, -1, 0)	(0, 1, 0)	(0, 0,10)

Start Time: 0 End Time: 100 Points: 100000

Table 5: A three-body system consisting of The Sun, Earth and Moon. See figures 8 and 9 for graphical representation.

Initial Parameters	Sun	Earth	Moon
Mass [$M_\odot$]	1	3.00273e-06	3.69441e-08
Position [AU]	(0, 0, 0)	(1, 0, 0)	(1.00257, 0, 0)
Velocity [AU/year]	(0, 0, 0)	(0, 6.28319, 0)	(0, 6.49311, 0)

1. Start Time: 0 End Time: 1 Points: 100
2. Start Time: 0 End Time: 0.08300 Points: 10000

Table 6: A restricted three-body system consisting of two equal mass stars, and one planet of negligible mass. See figure 11 for graphical representation.

Initial Parameters	Star 1	Star 2	planet
Mass [$M_\odot$]	1	1	1.00000e-10
Position [AU]	(-1, 0, 0)	(1, 0, 0)	(0, 0, 0)
Velocity [AU/year]	(0, -1, 0)	(0, 1, 0)	(0, 1, 0)

Start Time: 0 End Time: 3 Points: 10000

Table 7: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. S-orbit around Alpha Centauri A with an initial velocity of 10.5 AU/year. See figures 12 and 13 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(-11, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 10.50000, 0)

1. Start Time: 0 End Time: 100 Points: 10000
2. Start Time: 0 End Time: 1000 Points: 100000

Table 8: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. S-orbit around Alpha Centauri A with an initial velocity of 11 AU/year. See figure 14 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(-11, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 11, 0)

Start Time: 0 End Time: 1000 Points: 100000

Table 9: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. S-orbit around Alpha Centauri A with an initial velocity of 11.5 AU/year. See figure 15 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(-11, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 11.50000, 0)

Start Time: 0 End Time: 1000 Points: 100000

Table 10: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. S-orbit around Alpha Centauri B with an initial velocity of 10.5 AU/year. See figure 16 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(12, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 10.50000, 0)

Start Time: 0 End Time: 1000 Points: 100000

Table 11: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. S-orbit around Alpha Centauri B with an initial velocity of 11 AU/year. See figure 17 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(12, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 11, 0)

Start Time: 0 End Time: 1000 Points: 100000

Table 12: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. S-orbit around Alpha Centauri B with an initial velocity of 11.5 AU/year. See figure 18 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(12, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 11.50000, 0)

Start Time: 0 End Time: 1000 Points: 100000

Table 13: A three-body system consisting of an Earth-mass planet within the Alpha Centauri binary star system. P-orbit around Alpha Centauri A and B. See figures 19 and 20 for graphical representation.

Initial Parameters	Alpha Centauri A	Alpha Centauri B	Earth-Mass Planet
Mass [$M_\odot$]	1.10000e+00	9.07000e-01	3.00000e-6
Position [AU]	(-11.50000, 0, 0)	(11.50000, 0, 0)	(200, 0, 0)
Velocity [AU/year]	(0, -0.90400, 0)	(0, 0.90400, 0)	(0, 0.50000, 0)

1. Start Time: 0, End Time: 100000, Points: 10000000
2. Start Time: 0, End Time: 1000000, Points: 100000000

8.2 Full Code: Two Body, Same Mass

```python
#--------------------------------------------------------------------------------
#-----------------------------     IMPORTS      ---------------------------------
#--------------------------------------------------------------------------------

# Import required functions from the SciPy library
from scipy import pi, array, linalg, linspace, integrate, concatenate

# Import required functions from the MatPlotLib library
from matplotlib.pyplot import plot, scatter, figure, xlabel, xticks, ylabel, \
yticks, legend, grid, show, savefig, locator_params

#Import the Time library and required functions from the OS library to facilitate
# saving of plot and parameter files
from os import path, mkdir
import time

#--------------------------------------------------------------------------------
#---------------------------      CONSTANTS      --------------------------------
#--------------------------------------------------------------------------------

#Define universal gravitational constant [AU^3 / solar mass / years^2]
G = 4*pi**2

#Define mass of stars [solar mass]
m1 = 1
m2 = 1


#--------------------------------------------------------------------------------
#---------------    INITIAL CONDITIONS AND TIME PARAMETERS      -----------------
#--------------------------------------------------------------------------------

#Define initial position vectors as arrays [AU]
r1 = array([-1,0,0], dtype="float64") #Place star 1 on x-axis at -1 AU
r2 = array([+1,0,0], dtype="float64") #Place star 2 on x-axis at +1 AU

#Define initial velocity vectors as arrays [AU / year]
v1 = array([0,-pi,0], dtype="float64") #Assign star 1 with y-velocity of -pi
v2 = array([0,+pi,0], dtype="float64") #Assign star 2 with y-velocity of +pi

#Create an array of initial parameters and flatten to 1D
init_params = array([r1, r2, v1, v2]).flatten()

#Define parameters for independent variable time [years]
t_start = 0 #Begin simulation at time zero
t_end   = 1 #End simulation at time one (half a period in this example)

#Define number of time points (time intervals plus 1)
t_points = 10000


#--------------------------------------------------------------------------------
#--------------------------      FUNCTION      ----------------------------------
#--------------------------------------------------------------------------------

#Define a function that returns velocity and accleration vectors from given
# position and velocity vectors.
# Variable A represents an array of size 12 that stores the 3d vector values
# for the position and velocity each body.
def TwoBody(A, t, G, m1, m2):
    r1 = A[:3]   #Slice A for index values 0,1,2
    r2 = A[3:6]  #Slice A for index values 3,4,5
    v1 = A[6:9]  #Slice A for index values 6,7,8
    v2 = A[9:12] #Slice A for index values 9,10,11

```

```
    #Calculate 3d vector magnitude of star seperation [AU]
    r_mag = linalg.norm(r2 - r1)

    #Calculate acceleration vectors [AU / years^2]
    a1 = G*m2*(r2 - r1) / (r_mag)**3 #dv1 by dt
    a2 = G*m1*(r1 - r2) / (r_mag)**3 #dv2 by dt

    #Create an array of size 12 that stores the 3d vector values for the
    # velocity and acceleration of each body.
    r_dependents = concatenate((v1, v2, a1, a2))

    return r_dependents


#--------------------------------------------------------------------------------
#----------------------------      SOLVER      ----------------------------------
#--------------------------------------------------------------------------------

#Define duration [AU] of simulation, and number of calculated time points
t_span = linspace(t_start, t_end, t_points)

#Call ODE solver where odeint returns an array of dimensions [t_points, 12],
# of which we only need the 1st 6 "columns", containing x1, y1, z1, x2, y2, z2.
two_body_sol = integrate.odeint(TwoBody, init_params, t_span, args=(G, m1, m2))
r1_sol       = two_body_sol[:,:3]
r2_sol       = two_body_sol[:,3:6]


#--------------------------------------------------------------------------------
#----------------------------      PLOTTING      --------------------------------
#--------------------------------------------------------------------------------

#Create figure
figure(figsize=(17,17))

#Plot star trajectories
plot(r1_sol[:,0], r1_sol[:,1], color="r")
plot(r2_sol[:,0], r2_sol[:,1], color="b")

#Plot final positions of stars and display as red/blue blobs
scatter(r1_sol[-1,0], r1_sol[-1,1], color="r", marker="o", s=900, label="star 1")
scatter(r2_sol[-1,0], r2_sol[-1,1], color="b", marker="o", s=900, label="star 2")

#Print final position coordinates
print(r1_sol[-1,0], r1_sol[-1,1])
print(r2_sol[-1,0], r2_sol[-1,1])

#Label axes
xlabel('\n x-coordinate [AU]', fontsize=40)
ylabel('y-coordinate [AU] \n', fontsize=40)

#Set number of ticks to five
locator_params(axis='x', nbins=5)
locator_params(axis='y', nbins=5)

#Increase size of tick labels
xticks(fontsize=28)
yticks(fontsize=28)

#Display grid
grid()

#Display legend
legend(loc='upper right', fontsize=28)

```

```python

#-------------------------------------------------------------------------------
#------------------------      SAVE PARAMETERS      ----------------------------
#-------------------------------------------------------------------------------

#Save an image of each plot along with its given input parameters as files in a
# sub-directory 'Physics_py' with date and time stamp in the file name for ease
# of future reference.
#Use Python f-string formatting to write the input parameter values in suitable
# formats for LaTeX inclusion.
if not path.exists('Physics_py'):
        mkdir('Physics_py')
dt_str=time.strftime("%Y%m%d-%H%M%S")
savefig(f'Physics_py/{dt_str}_test_pyplot#1.png')

with open(f'Physics_py/{dt_str}_test_pyplot#1.txt','w') as text_file:
    text_file.write(f'\
\
Mass      &  \
{m1:{".5e" if m1!=int(m1) else ">2.0f"}}  &  \
{m2:{".5e" if m2!=int(m2) else ">2.0f"}}  \\\\ \n\
\
Radius    &  (\
{r1[0]:{">8.5f" if r1[0]!=int(r1[0]) else ">2.0f"}},\
{r1[1]:{">8.5f" if r1[1]!=int(r1[1]) else ">2.0f"}},\
{r1[2]:{">8.5f" if r1[2]!=int(r1[2]) else ">2.0f"}} )  &  (\
\
{r2[0]:{">8.5f" if r2[0]!=int(r2[0]) else ">2.0f"}},\
{r2[1]:{">8.5f" if r2[1]!=int(r2[1]) else ">2.0f"}},\
{r2[2]:{">8.5f" if r2[2]!=int(r2[2]) else ">2.0f"}} ) \\\\ \n\
\
Velocity  &  (\
{v1[0]:{">8.5f" if v1[0]!=int(v1[0]) else ">2.0f"}},\
{v1[1]:{">8.5f" if v1[1]!=int(v1[1]) else ">2.0f"}},\
{v1[2]:{">8.5f" if v1[2]!=int(v1[2]) else ">2.0f"}} )  &  (\
\
{v2[0]:{">8.5f" if v2[0]!=int(v2[0]) else ">2.0f"}},\
{v2[1]:{">8.5f" if v2[1]!=int(v2[1]) else ">2.0f"}},\
{v2[2]:{">8.5f" if v2[2]!=int(v2[2]) else ">2.0f"}} ) \\\\ \n\
\
Time      &  \
{t_start:{">8.5f" if t_start!=int(t_start) else ">2.0f"}}  &  \
{t_end:{">8.5f" if t_end!=int(t_end) else ">2.0f"}}  &  \
{t_points:{">8.5f" if t_points!=int(t_points) else ">2.0f"}}  \\\\ \
\
')

show()
```

8.3 Code Snippet: Centre of Mass Perspective (Two-Body)

```python
#--------------------------------------------------------------------------------
#----------------------------      PERSPECTIVE      -----------------------------
#--------------------------------------------------------------------------------

#Calculate centre of mass (COM) location
r_com_sol = (m1*r1_sol + m2*r2_sol) / (m1 + m2)

#Calculate location of star 1 from COM perspective
r1_com_sol = r1_sol - r_com_sol

#Calculate location of star 2 from COM perspective
r2_com_sol = r2_sol - r_com_sol


#--------------------------------------------------------------------------------
#----------------------------      PLOTTING      --------------------------------
#--------------------------------------------------------------------------------

#Create figure
figure(figsize=(17,17))

#Plot star trajectories
plot(r1_com_sol[:,0], r1_com_sol[:,1], color="r")
plot(r2_com_sol[:,0], r2_com_sol[:,1], color="b")

#Plot final positions of stars and display as red/blue blobs
scatter(r1_com_sol[-1,0], r1_com_sol[-1,1], color="r", marker="o", s=900, label="star 1")
scatter(r2_com_sol[-1,0], r2_com_sol[-1,1], color="b", marker="o", s=900, label="star 2")

#Print final position coordinates
print(r1_com_sol[-1,0], r1_com_sol[-1,1])
print(r2_com_sol[-1,0], r2_com_sol[-1,1])
```

8.4 Code Snippet: 3D Plotting (Three-Body)

```
#--------------------------------------------------------------------------------
#-----------------------------     IMPORT      ----------------------------------
#--------------------------------------------------------------------------------

# Import required functions from the MatPlotLib library
from matplotlib.pyplot import plot, scatter, figure, xlabel, xticks, ylabel, \
yticks, legend, grid, show, savefig, locator_params

#Import 3D axes
from mpl_toolkits import mplot3d


#--------------------------------------------------------------------------------
#----------------------------      PLOTTING      --------------------------------
#--------------------------------------------------------------------------------

#Create figure
figure(figsize=(17,17))

#Create 3D axes
ax = axes(projection='3d')

#Plot trajectories for the two stars and the planet
ax.plot(r1_com_sol[:,0], r1_com_sol[:,1], r1_com_sol[:,2], color='limegreen')
ax.plot(r2_com_sol[:,0], r2_com_sol[:,1], r2_com_sol[:,2], color='b')
ax.plot(r3_com_sol[:,0], r3_com_sol[:,1], r3_com_sol[:,2], color='r')

#Plot final positions of the two stars and the planet
ax.scatter(r1_com_sol[-1,0], r1_com_sol[-1,1], r1_com_sol[-1,2], color='limegreen', marker='o', s=600, label='Star 1')
ax.scatter(r2_com_sol[-1,0], r2_com_sol[-1,1], r2_com_sol[-1,2], color='b', marker='o', s=600, label='Star 2')
ax.scatter(r3_com_sol[-1,0], r3_com_sol[-1,1], r3_com_sol[-1,2], color='r', marker='o', s=200, label='Planet')

#Label axes
ax.set_xlabel('\n\n\n x-coordinate [AU]', fontsize=32)
ax.set_ylabel('\n\n\n y-coordinate [AU]', fontsize=32)
ax.set_zlabel('\n\n\n z-coordinate [AU]', fontsize=32)

#Increase size of tick labels
xticks(fontsize=21)
yticks(fontsize=21)

#Set number of ticks on axes to five
locator_params(axis='x', nbins=5)
locator_params(axis='y', nbins=5)
locator_params(axis='z', nbins=5)

#Display legend
ax.legend(loc='lower right', fontsize=22)

#Rotate z-axis tick labels and increase size
setp( ax.zaxis.get_majorticklabels(), rotation=30, fontsize=21)
```

8.5 Full Code: Three Body, Sun-Earth-Moon

```python
#--------------------------------------------------------------------------------
#-----------------------------     IMPORTS      ---------------------------------
#--------------------------------------------------------------------------------

# Import required functions from the SciPy library
from scipy import pi, array, linalg, linspace, integrate, concatenate

# Import required functions from the MatPlotLib library
from matplotlib.pyplot import plot, scatter, figure, xlabel, xticks, ylabel, \
yticks, legend, grid, show, savefig, locator_params

#Import the Time library and required functions from the OS library to facilitate
# saving of plot and parameter files
from os import path, mkdir
import time


#--------------------------------------------------------------------------------
#---------------------------      CONSTANTS      --------------------------------
#--------------------------------------------------------------------------------

#Define universal gravitational constant [AU^3 / solar mass / years^2]
G = 4*pi**2

#Define mass of Sun, Earth and Moon [solar mass]
m_sun   = 1
m_earth = 3.00273e-6
m_moon  = 3.69432e-8


#--------------------------------------------------------------------------------
#---------------    INITIAL CONDITIONS AND TIME PARAMETERS      -----------------
#--------------------------------------------------------------------------------

#Define initial position vectors as arrays [AU]
r1 = array([0      , 0, 0], dtype='float64') #Place Sun 1 at origin
r2 = array([1      , 0, 0], dtype='float64') #Place Earth on x-axis at +1 AU
r3 = array([1.00257, 0, 0], dtype='float64') #Place Moon on x-axis at +1.00257 AU

#Starting with the Moon at its furthest point from the Sun, approximate its
# initial velocity as the radial component around the Sun plus the radial
# component around the Earth.
v3_y = 2*pi*( r3[0] + (r3[0]-1)*12 )

#Define initial velocity vectors as arrays [AU / year]
v1 = array([0, 0   , 0], dtype='float64') #Assign Sun as static
v2 = array([0, 2*pi, 0], dtype='float64') #Assign Earth with y-velocity of +2*pi
v3 = array([0, v3_y, 0], dtype='float64') #Assign Moon with y-velocity of +v3_y

#Create an array of initial parameters and flatten to 1D
init_params = array([r1, r2, r3, v1, v2, v3]).flatten()

#Define parameters for independent variable time [years]
t_start = 0 #Begin simulation at time zero
t_end   = 1 #End simulation at time one (period of one complete orbit in this example)

#Define number of time points (time intervals plus 1)
t_points = 10000


#--------------------------------------------------------------------------------
#--------------------------      FUNCTION      ----------------------------------
#--------------------------------------------------------------------------------

#Define a function that returns velocity and accleration vectors from given
```

```python
# position and velocity vectors.
# Variable A represents an array of size 18 that stores the 3d vector values
# for the position and velocity each body.
def ThreeBody(A, t, G, m_sun, m_earth, m_moon):
    r1 = A[:3]    #Slice A for index values 0,1,2
    r2 = A[3:6]   #Slice A for index values 3,4,5
    r3 = A[6:9]   #Slice A for index values 6,7,8
    v1 = A[9:12]  #Slice A for index values 9,10,11
    v2 = A[12:15] #Slice A for index values 12,13,14
    v3 = A[15:18] #Slice A for index values 15,16,17

    #Calculate 3d vector magnitudes of three body seperations [AU]
    r12_mag = linalg.norm(r2-r1)
    r13_mag = linalg.norm(r3-r1)
    r23_mag = linalg.norm(r2-r3)

    #Calculate acceleration vectors [AU / years^2]
    a1 = ( G*m_earth*(r2-r1) / (r12_mag)**3 ) + ( G*m_moon*(r3-r1)  / (r13_mag)**3 ) #dv1 by dt
    a2 = ( G*m_sun*(r1-r2)   / (r12_mag)**3 ) + ( G*m_moon*(r3-r2)  / (r23_mag)**3 ) #dv2 by dt
    a3 = ( G*m_sun*(r1-r3)   / (r13_mag)**3 ) + ( G*m_earth*(r2-r3) / (r23_mag)**3 ) #dv3 by dt

    #Create an array of size 18 that stores the 3d vector values for the
    # velocity and acceleration of each body.
    r_dependents = concatenate((v1, v2, v3, a1, a2, a3))

    return r_dependents


#--------------------------------------------------------------------------------
#----------------------------      SOLVER      ----------------------------------
#--------------------------------------------------------------------------------

#Define duration [AU] of simulation, and number of calculated time points
t_span = linspace(t_start, t_end, t_points)

#Call ODE solver where odeint returns an array of dimensions [t_points, 18],
# of which we only need the 1st 9 "columns", containing x1, y1, z1, x2, y2, z2, x3, y3, z3.

three_body_sol = integrate.odeint(ThreeBody, init_params, t_span, args=(G, m_sun, m_earth, m_moon))
r1_sol         = three_body_sol[:, :3]
r2_sol         = three_body_sol[:,3:6]
r3_sol         = three_body_sol[:,6:9]


#--------------------------------------------------------------------------------
#----------------------------      PERSPECTIVE      -----------------------------
#--------------------------------------------------------------------------------

#Calculate centre of mass (COM) location
r_com_sol = (m_sun*r1_sol + m_earth*r2_sol + m_moon*r3_sol) / (m_sun + m_earth + m_moon)

#Calculate location of Sun from COM perspective
r1_com_sol = r1_sol - r_com_sol

#Calculate location of Earth from COM perspective
r2_com_sol = r2_sol - r_com_sol

#Calculate location of Moon from COM perspective
r3_com_sol = r3_sol - r_com_sol
```

```python
#-------------------------------------------------------------------------------
#----------------------------      PLOTTING      -------------------------------
#-------------------------------------------------------------------------------
#Create figure
figure(figsize=(17,17))

#Plot trajectories for The Sun, Earth and Moon
plot(r1_com_sol[:,0], r1_com_sol[:,1], color='limegreen')
plot(r2_com_sol[:,0], r2_com_sol[:,1], color='b')
plot(r3_com_sol[:,0], r3_com_sol[:,1], color='r')

#Plot final positions of The Sun, Earth and Moon
scatter(r1_com_sol[-1,0], r1_com_sol[-1,1], color='limegreen', marker='o', s=900, label='Sun')
scatter(r2_com_sol[-1,0], r2_com_sol[-1,1], color='b', marker='o', s=900, label='Earth')
scatter(r3_com_sol[-1,0], r3_com_sol[-1,1], color='r', marker='o', s=300, label='Moon')

#Label axes
xlabel('\n x-coordinate [AU]', fontsize=40)
ylabel('y-coordinate [AU]\n', fontsize=40)

#Set number of ticks to five
locator_params(axis='x', nbins=5)
locator_params(axis='y', nbins=5)

#Increase size of tick labels
xticks(fontsize=28)
yticks(fontsize=28)

#Display grid
grid()

#Display legend
legend(loc='upper right', fontsize=28)


#-------------------------------------------------------------------------------
#------------------------      SAVE PARAMETERS      ----------------------------
#-------------------------------------------------------------------------------

#Save an image of each plot along with its given input parameters as files in a
# sub-directory 'Physics_py' with date and time stamp in the file name for ease
# of future reference.
#Use Python f-string formatting to write the input parameter values in suitable
# formats for LaTeX inclusion.
if not path.exists('Physics_py'):
        mkdir('Physics_py')
dt_str=time.strftime("%Y%m%d-%H%M%S")
savefig(f'Physics_py/{dt_str}_test_pyplot#1.png')

with open(f'Physics_py/{dt_str}_test_pyplot#1.txt','w') as text_file:
    text_file.write(f'\
\
Mass      &  \
{m_sun:{".5e" if m_sun!=int(m_sun) else ">2.0f"}}  &  \
{m_earth:{".5e" if m_earth!=int(m_earth) else ">2.0f"}}  &  \
{m_moon:{".5e" if m_moon!=int(m_moon) else ">2.0f"}}  \\\\ \n\
\
Radius    &  (\
{r1[0]:{">8.5f" if r1[0]!=int(r1[0]) else ">2.0f"}},\
{r1[1]:{">8.5f" if r1[1]!=int(r1[1]) else ">2.0f"}},\
{r1[2]:{">8.5f" if r1[2]!=int(r1[2]) else ">2.0f"}} )  &  (\
\
{r2[0]:{">8.5f" if r2[0]!=int(r2[0]) else ">2.0f"}},\
{r2[1]:{">8.5f" if r2[1]!=int(r2[1]) else ">2.0f"}},\
```

```
{r2[2]:{">8.5f" if r2[2]!=int(r2[2]) else ">2.0f"}} )  &  (\
\
{r3[0]:{">8.5f" if r3[0]!=int(r3[0]) else ">2.0f"}},\
{r3[1]:{">8.5f" if r3[1]!=int(r3[1]) else ">2.0f"}},\
{r3[2]:{">8.5f" if r3[2]!=int(r3[2]) else ">2.0f"}} )  \\\\ \n\
\
Velocity  &  (\
{v1[0]:{">8.5f" if v1[0]!=int(v1[0]) else ">2.0f"}},\
{v1[1]:{">8.5f" if v1[1]!=int(v1[1]) else ">2.0f"}},\
{v1[2]:{">8.5f" if v1[2]!=int(v1[2]) else ">2.0f"}} )  &  (\
\
{v2[0]:{">8.5f" if v2[0]!=int(v2[0]) else ">2.0f"}},\
{v2[1]:{">8.5f" if v2[1]!=int(v2[1]) else ">2.0f"}},\
{v2[2]:{">8.5f" if v2[2]!=int(v2[2]) else ">2.0f"}} )  &  (\
\
{v3[0]:{">8.5f" if v3[0]!=int(v3[0]) else ">2.0f"}},\
{v3[1]:{">8.5f" if v3[1]!=int(v3[1]) else ">2.0f"}},\
{v3[2]:{">8.5f" if v3[2]!=int(v3[2]) else ">2.0f"}} )  \\\\ \n\
\
Time      &  \
{t_start:{">8.5f" if t_start!=int(t_start) else ">2.0f"}}  &  \
{t_end:{">8.5f" if t_end!=int(t_end) else ">2.0f"}}  &  \
{t_points:{">8.5f" if t_points!=int(t_points) else ">2.0f"}}  \\\\ \
\
')

show()
```

8.6 Code Snippet: Calculating Radius of S-type Orbits

```
#Determine the diameter of the planet's orbit about the x and y planes by doing the following:
# ( diameter of entire donut-shape produced by planet's orbits  ) minus ( diameter of Alpha Centauri A's orbit )
diameter_x = ( max(r3_com_sol[:,0]) - min(r3_com_sol[:,0]) ) - ( max(r1_com_sol[:,0]) - min(r1_com_sol[:,0]) )
diameter_y = ( max(r3_com_sol[:,1]) - min(r3_com_sol[:,1]) ) - ( max(r1_com_sol[:,1]) - min(r1_com_sol[:,1]) )

#Calculate the average radius of the planet's orbit about Alpha Centauri A (star 1)
mean_radius = (diameter_x + diameter_y) / 4

#Print the average radius so that it can be later evaluated
print(mean_radius)
```

9 Acknowledgements

Project Supervisor: Prof. D.C. Dunbar (Swansea University)

Author of Reference Code (online source) [28]: Gaurav Deshmukh (Purdue University)