Simulating the Orbital Stability of Circumbinary Planets

Peyton Grimm and Andrew Richter

Abstract

The goal of our research was to simulate the orbit of a circumbinary planet, and determine for what parameters of periapsis, eccentricity, and mass would the system behave like a single-star system. There does not exist an analytical solution for the motion of three or more bodies under gravity. However, there are some scenarios where a three body system behaves like a two body system. One of these scenarios is a planet orbiting far away from its host stars. We built a three body simulation in Java to test when a circumbinary planet behaves like it is in a single-star system. We incrementally changed the planet's starting periapsis, eccentricity, and mass and recorded the instability (defined as the change in orbital angular momentum divided by the planet's mass and initial period.) for each incremental change. We found that instability is strongly inversely related to semi-major axis, directly related to eccentricity, and has no relation to mass. We generated several plots of this parameter space, and discuss the results.

1. Introduction

1.1 N-body Kinematics

The motion of planets about a single star system has been well documented throughout history. Kepler's laws tell us that planets orbit in ellipses with their host star at one of the foci, and Newton's laws describe the gravitational forces on planets.

(1.1)
$$\vec{F} = -\frac{GM_1M_2}{r^2}\hat{r}$$

There exists an analytical solution describing the motion of a planet about a single star as shown by the following equation (Carroll and Ostlie 2017) where *r* is the distance from the star, *a* is the semi-major axis, and *e* is the eccentricity. A diagram of a single-star system is shown in Figure 1.

(1.2)
$$r = \frac{a(1-e^2)}{1 + e\cos(\theta)}$$

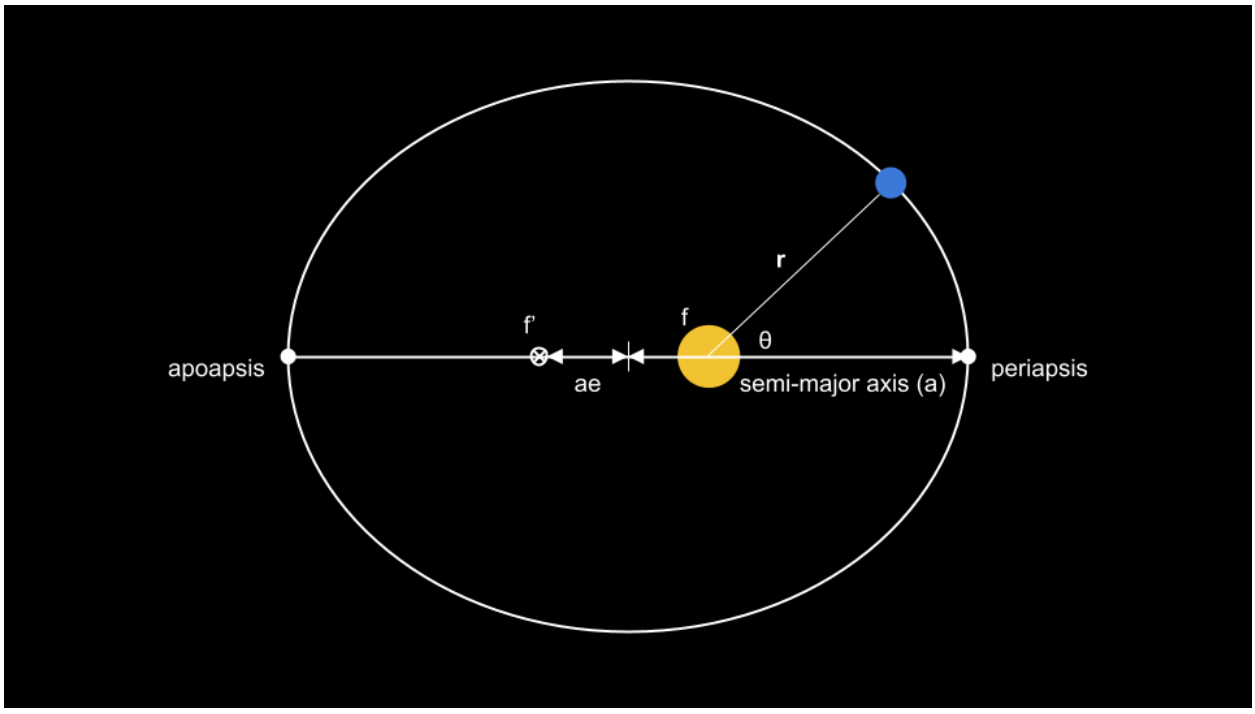

Figure 1. A diagram labeling the orbit of a planet in a single-star system.

However, There exists no analytical solution that describes the motion of three or more bodies under mutual gravitational attraction. They appear to move chaotically. Depending on the configuration of the system, planets can be accreted, ejected, or moved to more stable orbits (Musielak et al. 2005; Chen et al. 2019). This is commonly referred to as the 3-body problem or N-body problem.

There do exist some special cases where the kinematics of a binary-star system behave like a single-star system: S-type, P-type, and T-type orbits (Musielak et al. 2005). In an S-type orbit, the planet orbits one of the stars closely. If the planet orbits closely enough to its host star with respect to the binary stars' separation, it can effectively ignore the other star. In a P-type orbit, the planet orbits outside its host stars' orbit. If the planet orbits far enough away from its host stars' orbit, it can effectively treat the pair as a point mass. T-type planets orbit via the L4 and L5 Lagrange points (points where each body in the system rotates at the same rate). A diagram of S-type, P-type, and T-type planets is shown in Figure 2, and A diagram of the inner region of a binary star system is shown in Figure 3.

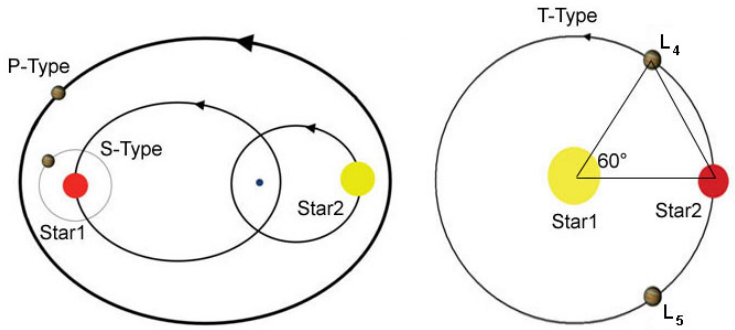

Figure 2. A diagram showing S-type, P-type, and T-type planets. Diagrams are not drawn to scale or accuracy.

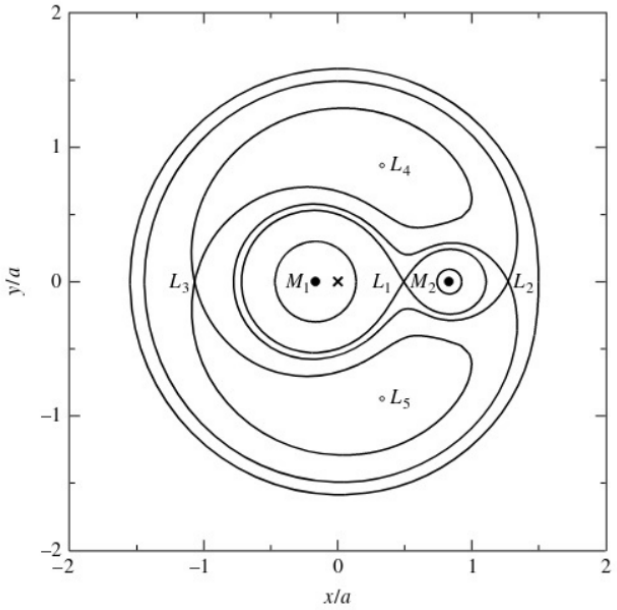

Figure 3. Contours of equal potential energy of the inner region of a binary star system. The diagram is in a reference frame that rotates with the stars. The stars do not have equal mass and the center of mass is located at the x. The Lagrange points are shown (Carroll and Ostlie 2017).

Our research goal is to answer the question, "What parameters for a P-type planet will allow a binary star system to behave like a single star system?" We will simulate a P-type planetary orbit, and then measure the stability of this orbit with respect to the planet's periapsis, eccentricity, and mass.

1.2 Computational Approaches

While there is no analytical solution for these configurations, it is a simple problem to solve computationally. This is the job of numerical integrators. Numerical integrators are algorithms that can computationally solve difficult differential equations. The simplest numerical integrator is the Euler algorithm. For a three body problem, we can calculate the instantaneous

force on each body at any point in time using the universal law of gravitation (equation 1.1). The Euler algorithm assumes this force (slope) to be constant and then calculates the next velocity using point-slope form. It then finds the next position using the same process. Afterwards, the algorithm advances time by one "step" and repeats. As long as each time step is small, the resulting motion will approximately be the same as an analytical solution.

While the Euler algorithm is a good start, it loses accuracy with each time step because the force, and therefore acceleration, is not constant. The Euler algorithm can further be improved by adding extra higher-order slope terms to account for non-constant acceleration and velocity.. This is known as the Runge Kutta method. The Runge Kutta method is a family of numerical integration algorithms. The Euler algorithm is simply a first-order Runge Kutta algorithm. Runge Kutta algorithms re-evaluate the differential equation (slope) at various different points throughout the timestep then take a weighted average of each slope. Each subsequent order of the algorithm adds another term. The general form for each algorithm is given by the following set of equations (Press et al. 1992).

(1.3), (1.4) $$\frac{dy}{dt} = f(t, y) \qquad y(t_0) = y_0$$

(1.5) $$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$$

Each k_i is found by

(1.6.1) $$k_1 = f(t_n, y_n)$$

(1.6.2) $$k_2 = f(t_n + c_2 h, y_n + h(a_{21} k_1))$$

(1.6.3) $$k_3 = f(t_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2))$$

$$\dots$$

(1.6) $$k_s = f(t_n + c_s h, y_n + h(a_{s1} + a_{s2} k_2 + \dots + a_{s,s-1} k_{s-1}))$$

In the equations above, $f(t, y)$ is the differential equation, y_n is the current solution, y_{n+1} is the next solution, h is the time step, s is the order, k_i is one of the slopes (which will depend on prior slopes), c_i is the place along the time step where the k_i is being evaluated (sometimes referred to as a node), b_i is the weight for k_i, and lastly, $a_{s,i}$ is an arbitrarily defined coefficient. Note that $c_i h <= h$ and $\Sigma b_i = 1$ (Press et al. 1992). This collection of equations is commonly arranged in a table called the Butcher Tableau as shown in Figure 4.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

Figure 4. The Butcher Tableau for Runge Kutta algorithms. c_i is the place in the time step where the slope term k_i is being evaluated (read top to bottom), b_i is the weight applied to k_i (read left to right), and a_{ij} is the coefficient applied to previous k terms (read left to right). ("Runge-Kutta Methods")

Higher-order algorithms don't necessarily mean that they are better. The truncation error associated with Runge Kutta algorithms scales as $O(h^{s+1})$ where $O()$ is some function of the time step. Each incremental order has more accuracy but less efficiency and more truncation error. The most popular Runge Kutta algorithm is the classic fourth-order Runge Kutta algorithm (Often confusingly dubbed RK4). It strikes a good balance of accuracy, efficiency, and truncation error (Press et al. 1992). RK4 uses the following system of equations. We visualize RK4 in Figure 5.

(1.7)

$$y_{n+1} = y_n + h\left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right)$$

(1.8.1), (1.8.2)

$$k_1 = f(t_n, y_n) \qquad k_2 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right)$$

(1.8.3), (1.8.4)

$$k_3 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \qquad k_4 = f(t_n + h, y_n + hk_3)$$

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

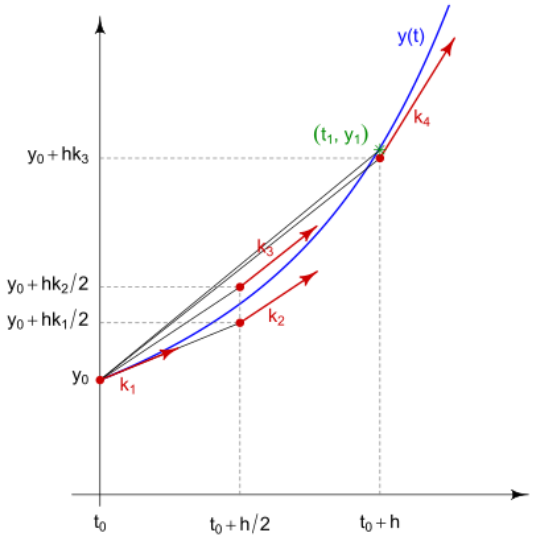

Figure 5. A visualization of RK4. RK4's Butcher Tableau is displayed on the left. The graph on the right contains the slope terms and the points where they are evaluated at. The final position is shown in green and the true function *y(t)* is shown in blue. ("Runge-Kutta Methods")

To achieve accurate results, we used the Runge Kutta Fehlberg algorithm (often called RK45). RK45 compares the difference between a fourth-order and a fifth-order Runge Kutta algorithm to some desired accuracy. RK45 then determines the time step necessary to produce the desired accuracy. It has a rather complicated Butcher Tableau shown below in Figure 6. The fourth-order (y) and fifth-order (y') algorithms use the following system of equations (Press et al. 1992; Fehlberg 1958).

(1.9) $$y_{n+1} = y_n + h(0.116k_1 + 0.549k_3 + 0.535k_4 - 0.200k_5)$$

(1.10) $$y'_{n+1} = y'_n + h(0.119k_1 + 0.519k_3 + 0.506k_4 - 0.180k_5 + 0.036k_6)$$

0						
$1/4$	$1/4$					
$3/8$	$3/32$	$9/32$				
$12/13$	$1932/2197$	$-7200/2197$	$7296/2197$			
1	$439/216$	-8	$3680/513$	$-845/4104$		
$1/2$	$-8/27$	2	$-3544/2565$	$1859/4104$	$-11/40$	
	$16/135$	0	$6656/12825$	$28561/56430$	$-9/50$	$2/55$
	$25/216$	0	$1408/2565$	$2197/4104$	$-1/5$	0

Figure 6. The Butcher Tableau for the Runge Kutta Fehlberg algorithm. The bottom two rows give the weights to find y' and y. Both algorithms use the same slope terms, but the fourth-order algorithm ignores the last term k_6 by setting a coefficient of 0 (Press et al. 1992; Fehlberg 1958).

We define the discrepancy between *y* and *y'* to be **Δ** as shown in equation 2.3 below. We label our desired accuracy **Δ**'. Accuracy here scales as h^5 if our time step is small. We can then calculate the time step needed to produce the desired accuracy *h'* using equation 2.4.

(1.11) $$\Delta = |y - y'|$$

(1.12) $$h' = h\left(\Delta'/\Delta\right)^{0.2}$$

RK45 will calculate *y* and *y'*, evaluate the best time step, and then find the next solution using *y*. At locations where the solution doesn't change much with respect to time, RK45 increases efficiency by taking large time steps (**Δ**'/**Δ** > 1). At locations where the solution changes greatly with respect to time, RK45 increases accuracy by taking short time steps (**Δ**'/**Δ** < 1). This algorithm is useful for situations where orbits change suddenly (Musielak et al. 2005; Press et al. 1992).

2. Computational Method

2.1 Experimental Setup

Our goal is to "map out" the stability of a P-type orbit with respect to the planet's parameters and determine when the system behaves like a single-star system. The system simulated is a binary star system with stars of equal mass (1 M_{sol}) in a circular orbit with a separation of 2 AU. We chose to put our parameters in units of Astronomical Units (AU), Solar Mass (M_{sol}), and Earth years (yr), such that the gravitational constant G equals $4\pi^2$. This makes our results and parameters easier to read. We inserted a planet and allowed each simulation to run for 100 initial planetary periods. To calculate the initial period (*P*), we assumed a single-star system and used Kepler's third law.

(2.1)
$$P^2 = a^3$$

The system is initialized in a straight-line configuration on the x axis. At each time step, we recorded the time, velocity, position, instantaneous orbital angular momentum, and total energy of the system. We outputted these tables to .csv files. Time steps were calculated using a tolerance (Δ') of 1 meter. This means that our fourth and fifth-order solutions had to agree by 1 meter which is dramatically small when compared to the size of 1 AU.

The goal of this experiment is to measure the stability of the orbit of the planet and determine how the parameters of the planet affect its stability. We are interested in the planet's instantaneous orbital angular momentum because we use it to calculate instability. We defined instability, *I*, as the magnitude of the change in instantaneous orbital angular momentum divided

by the planet's mass and initial period. It is a measure of how much a planet's orbit changes with each orbit.

(2.2)
$$I = \frac{|\Delta L|}{mP} = \frac{|\Delta(r \times v)|}{P}$$

In this case, L is orbital angular momentum, v is velocity, and r is defined as the vector from the center of mass to the planet. We chose to use orbital angular momentum as our measure for instability because it shouldn't change in the case of a single-star system due to Kepler's second law. Because the force exerted on the planet always points radially inward, there is no torque.

We incrementally changed the planet's parameters and ran the simulation for each incremental change. For eccentricity, we used a range of 0.0 - 0.8 in increments of 0.05. For periapsis, we used a range of 2 - 11 AU in increments of 0.05 from 2 - 5 AU and in increments of 0.2 from 5 - 11 AU. From preliminary results, we found that the stability of the system doesn't change much after a periapsis of 4 AU, which is why we chose larger increments in the high periapsis range. We repeated this process using two different values for mass: 10^{-3} and 10^{-6} M_{sol}. This roughly corresponds to a Jupiter-sized planet and an Earth-sized planet. At the end of each simulation, we recorded the mass, periapsis, eccentricity, semi-major axis, period, instability, and instability error in a "master" .csv file. We calculated the initial semi-major axis and period using equations 1.2, and 2.1 respectively. We discuss instability and error calculations in the next section.

2.2 Calibration and Data Analysis

To confirm that our algorithm is accurate, we simulated a single-star system (1 M_{sol} placed at the origin) and compared our results to the analytical solutions. We first performed

some basic debugging tests like verifying that the orbit is an ellipse and that energy agrees with analytical solutions. We confirmed that the simulation agrees. We found that the single-star system shows some instability. However, this instability is many orders of magnitude smaller than what is seen in the binary star systems. A plot of the instability in single-star systems is shown in Figure 7.

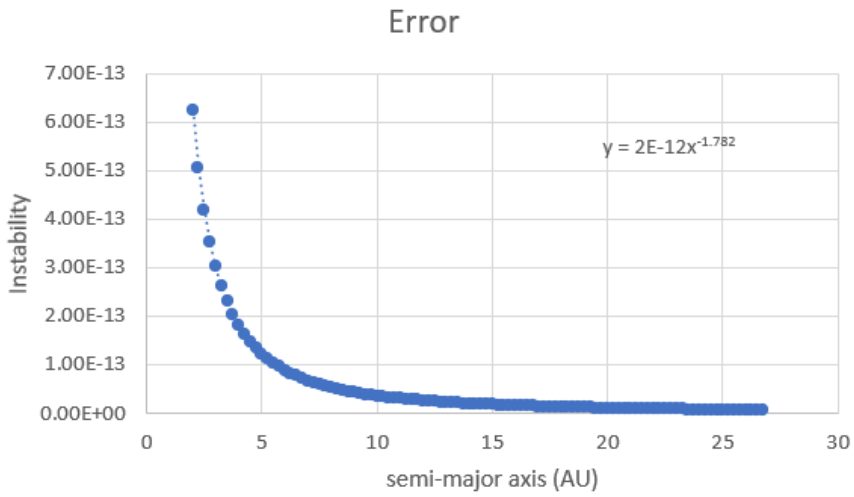

Figure 7. The relationship between instability and semi-major axis plotted in Excel. The "natural" instability drops as a function of $a^{-1.782}$. The motion of the star was not simulated when making this plot.

The size of this instability strongly indicates that our algorithm is working as intended (at least for a single-star case). We also found that the momentum of a planet tends to naturally fluctuate throughout each orbit. A plot of this is shown in Figure 8. When calculating the instability of the planet, we subtracted the planet's initial momentum from the average over the last period of the simulation. Figure 7 gives us a low estimation for the error in this instability.

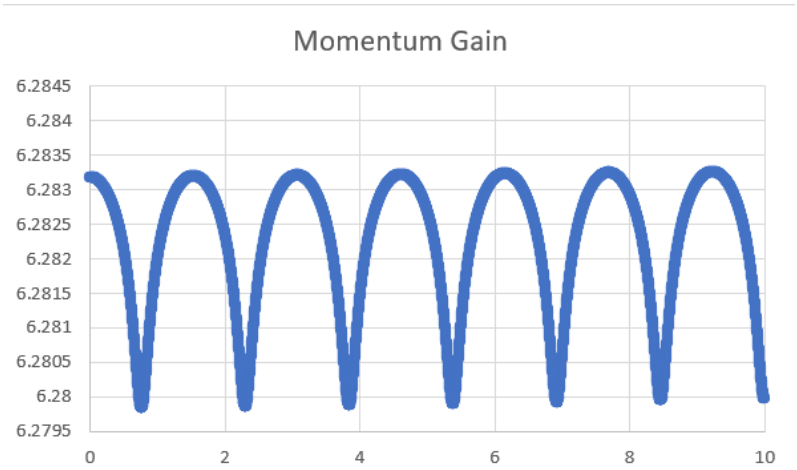

Figure 8. An example of the "natural" fluctuations in a planet's momentum with each orbit. Time in years is on the x axis, and L/m in units of AU^2/yr is on the y axis. This plot was taken early in the debugging process, so the information is not accurate.

While the system behaves as expected in the case of a single-star system, we encountered several problems in a binary-star system. The problems arise not from having three bodies but from the proximity of the two stars in the numerical integration process. With a separation of 2 AU, the stars tended to spiral apart from each other. Energy is numerically not conserved in this process. While the Runge-Kutta Fehlberg algorithm helps with functions that change rapidly, it does not provide us with enough accuracy to simulate a binary star system with this separation. We could obtain more accurate results by lowering the tolerance to something exceptionally small (~ 1 μm), but this is too costly for our computer resources. We attempted to correct the

simulation by using an energy constraint. One example we tried was scaling all bodies by a factor A.

(2.3), (2.4) $$\frac{E_0}{E_i} = A \qquad E_0 = \frac{1}{2}m\left(v\sqrt{A}\right)^2 - \frac{GMm}{r\left(\frac{1}{A}\right)} = AE_i$$

This would guarantee that energy is conserved, and worked well for the binary stars. However, correcting every body by the same factor would "consume" the planets' momentum. We tried to remedy this by using a weighted factor, but we then encountered a problem of choosing the correct weights and maintaining energy conservation.

Because of these problems, we only simulated the planet's motion. The motion of the stars was determined using the following equations.

(2.5), (2.6) $$x = (1\ \mathrm{AU})\cos(\theta) \qquad y = (1\ \mathrm{AU})\sin(\theta)$$

The second star uses the same equations but is shifted by a phase of $\boldsymbol{\pi}$. This approximation will work as long as the mass of the planet is significantly smaller than the mass of the stars. While our results will be less physical, they will be more accurate than if we attempted to simulate the stars' positions. Because we do not simulate the positions of the stars, we believe that Figure 7 is too low of an estimation for the error in each simulation. However, it is the only method we have readily available.

3. Results

Figure 9 reveals the majority of our results. Instability often differs by several orders of magnitude, so we decided to plot the $\log_{10}$ of instability. We found that instability is most strongly related to semi-major axis from a quick visual inspection of the data. This is expected as

distance is the factor that distinguishes a P-type planet. From Figure 9, we see that instability is inversely correlated to periapsis and semi-major axis. We also see that instability is directly correlated to eccentricity. Shockingly, the $\log_{10}$ of instability is not linearly correlated to semi-major axis. This means that instability depends on some exponent raised to a function of semi-major axis.

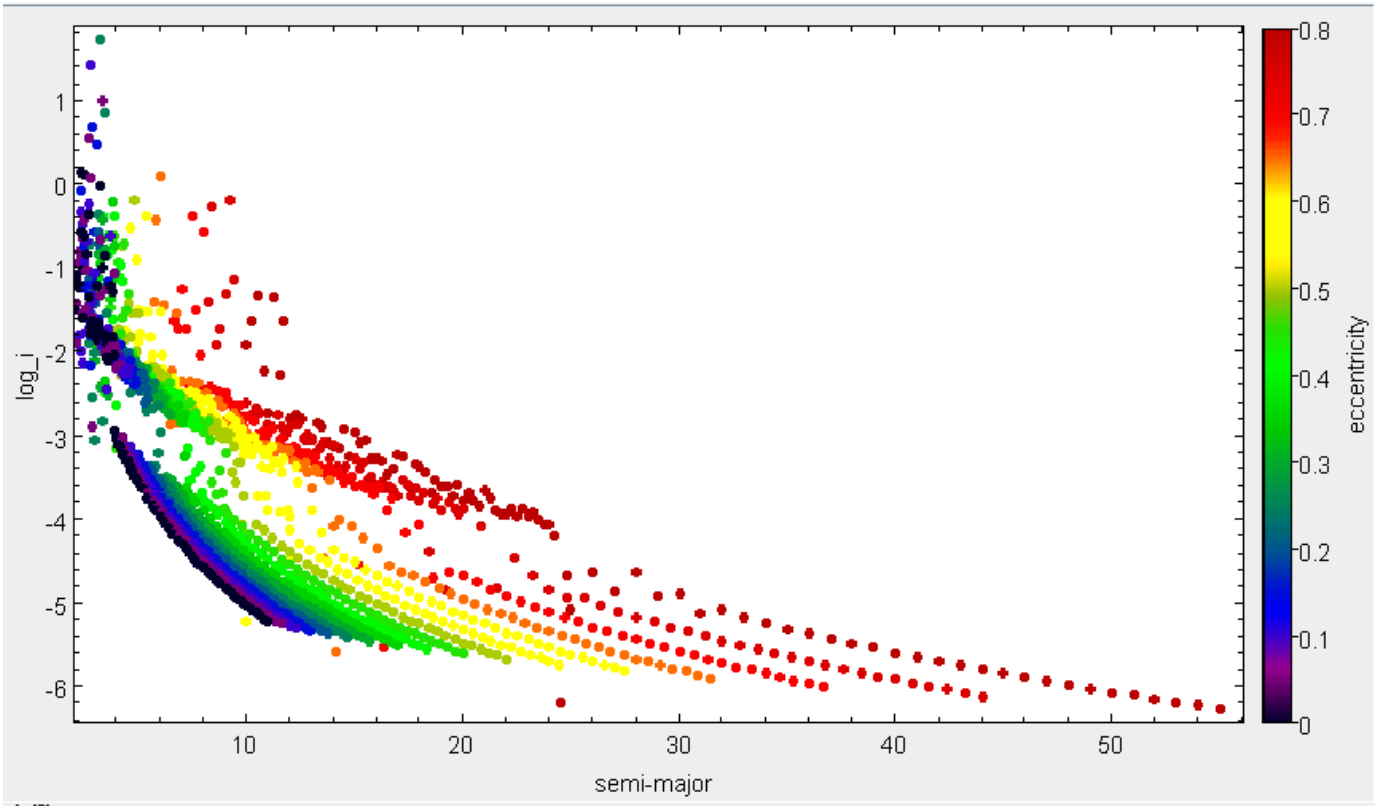

Figure 9. A plot of the $\log_{10}$ of instability versus the initial semi-major axis in AU. The colors correspond to different values of eccentricity as shown.

One interesting development from this plot is the gap in instability starting at roughly -3. Below this gap, instability and semi-major axis are related by some well-defined function. Above this gap, instability seems to behave more chaotically with respect to semi-major axis. We sampled some of the data points above and below this instability gap to understand what is

happening. Our results are shown in Figure 10. The gap appears to be the threshold for when a planet is ejected from the system or is about to be ejected. When a planet is ejected, it gets one last massive boost from the stars. It is likely that this gap depends on the simulation time. In the 4.35 AU periapsis case of Figure 10, Most of the planet's orbits stay consistently close to the star. The planet is nearly ejected only during the last few orbits. If the simulation was stopped a little earlier, we would not have seen this behavior. This suggests that the instability gap moves downward with simulation time.

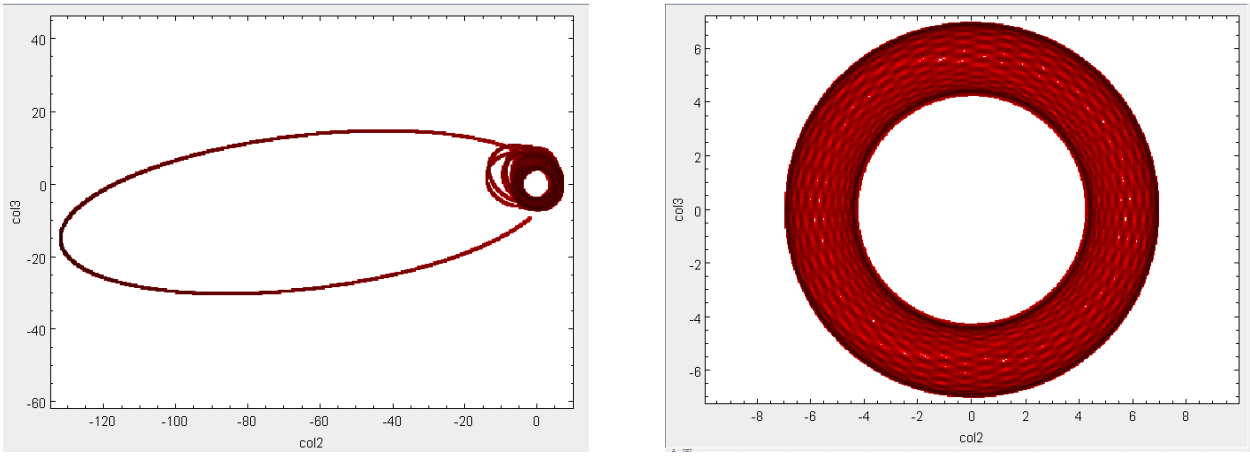

Figure 10. An xy plot of two simulations. On the left, the planet started with a periapsis of 4.35 AU and an eccentricity of 0.2. On the right, the planet started with a periapsis of 4.4 AU and an eccentricity of 0.2. This shows how sensitive the orbit is to changes in periapsis.

In Figure 9, each "band" of eccentricity has a consistent shape. It appears that eccentricity stretches out the x axis of this plot. In Figure 11, we attempt to fit a function to one of these plots. This function serves as a rough estimate of the shape of the graph and is by no means accurate. Because we can not have a negative semi-major axis, we would expect some asymptotic behavior as the semi-major axis approaches zero. Our fit does not account for that nor

does it account for the instability gap. We tried a variety of fitting functions, and the one that worked best was an exponential decay function.

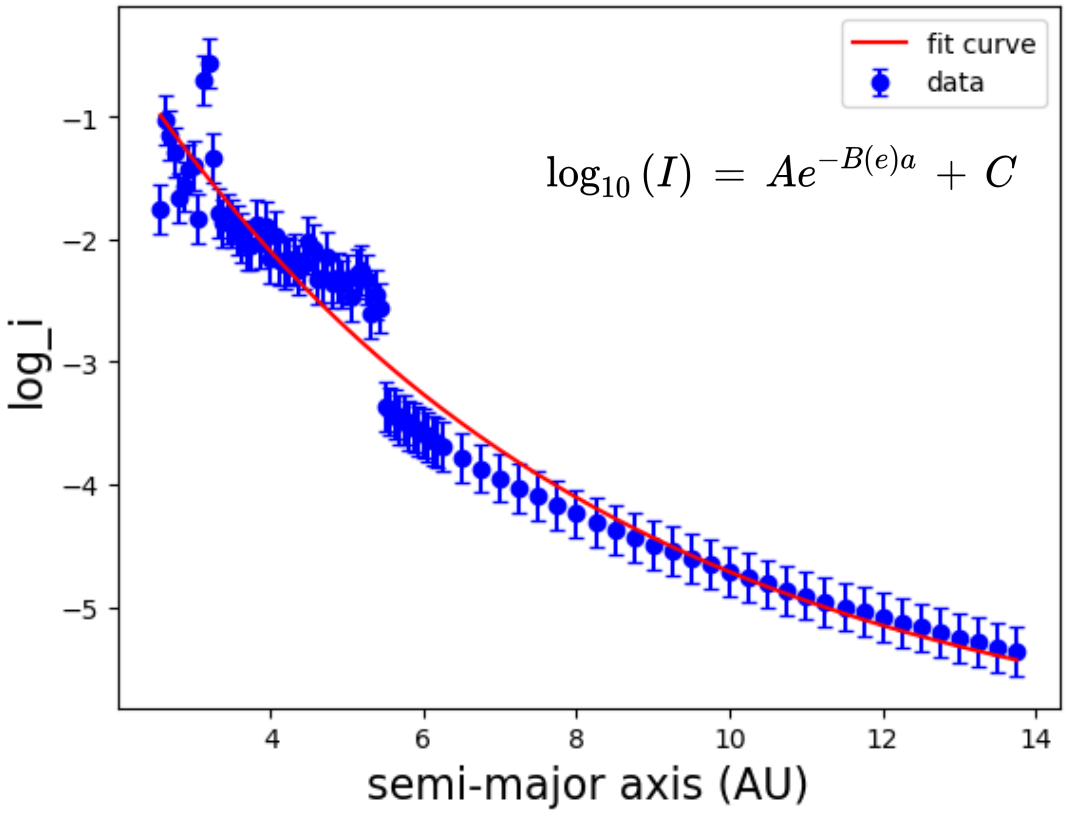

Figure 11. An estimated fit to the $\log_{10}$ of the instability. We only used data points with an eccentricity of 0.2. Errors are arbitrarily defined as the program needed errors to run. The parameters are as follows: A = 8.05 +/- 0.20, $B(e)$ = 0.164 +/- 0.018, and C = -6.28 +/- 0.29. This fit has a reduced chi-squared value of 2.13.

The goal of our research was to determine when the system behaves like a single-star system. There is no clear definition for when the system quits behaving like a binary-star system.

We elected to choose $\log_{10}(I) = -5$ as our criteria for single-star behavior. Figure 12 shows the subset of systems that meet our criteria. By plotting periapsis versus eccentricity, we can make contours of equal instability. We can begin to see how periapsis and eccentricity relate in order to constrain instability.

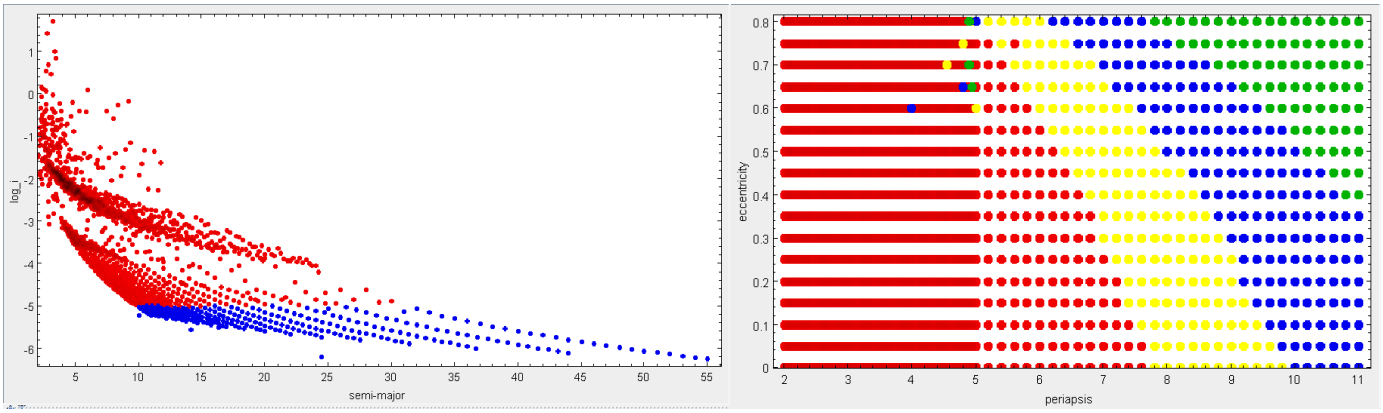

Figure 12. Subsets for varying criteria of stability. The right image is a plot of eccentricity versus periapsis. Yellow corresponds to $\log_{10}(I) = -4.5$, blue corresponds to -5, and green corresponds to -5.5.

Lastly, we found that mass had no effect on the instability of the system. Our results were identical. This is likely because we did not simulate the positions of the stars. A Jupiter mass planet will perturb the positions of the stars more which in turn should perturb the planet's orbit. This behavior is absent from our simulation.

4. Conclusion

Using the Runge-Kutta Fehlberg algorithm, we simulated several binary star systems with P-type planets. We varied the planet's periapsis, eccentricity, and mass then measured that planet's instability after 100 initial periods. Instability varies with both periapsis, and eccentricity. Using a criteria of ($\log_{10}(I) = -5$), Low-end eccentricity planets are stable when their periapsis is roughly 10 AU (5 times the star separation), and high-end eccentricity planets are stable when their periapsis is roughly 7 AU (3.5 times the star separation). From Figure 9, we found that instability is strongly indirectly related to the planet's semi-major axis. This relationship is so strong that $\log_{10}(I)$ appears to have a decaying exponential relationship to the semi-major axis. We found that instability is directly related to the planet's eccentricity. Eccentricity appears to "stretch-out" the x axis in Figure 9. We found that instability has no relationship to mass. However, we did not simulate the motion of the stars, which likely caused this absent relationship. When a planet is ejected, we found that its instability considerably increases as shown by the "instability gap" in Figure 9. This gap likely depends on simulation time, as not all systems may have had enough time to eject their planets.

References

Carroll and Ostlie 2017, An Introduction to Modern Astronomy, DOI: 10.1017/9781108380980

Chen et al. 2019, Orbital Dynamics of Circumbinary Planets, MNRAS 490, 5634–5646

Fehlberg 1958, Eine Methode zur Fehlerverkleinerung beim Runge-Kutta-Verfahren, ZAMM 38, 11-12

Musielak et al. 2005, Stability of Planetary Orbits in Binary Systems, A&A 434, 355-364

Press et al. 1992, Numerical Recipes in C, ISBN 0-521-43108-5

“Runge-Kutta Methods.” Wikipedia, https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods