AST326 Lab #6: Determining Orbital Parameters of 26 Proserpina

Aaryan Thusoo

April 9th 2024

ABSTRACT

Understanding the orbits of celestial bodies is fundamental in astronomy. In this laboratory report, we employ Laplace's method and Newtonian iteration to derive the six orbital parameters necessary for characterizing the orbit of an object in the solar system. Our analysis reveals key parameters defining the orbit's shape: the semi-major axis $a = 2.57^{+0.07}_{-0.07}$ AU and eccentricity $e = 0.09^{+0.02}_{-0.02}$. Additionally, we determine the angles governing the 3D orientation of the orbit: inclination $i = 3.55^{+0.57}_{-0.57}{}^{\circ}$, longitude of ascending node $\Omega = 46.9^{+8.6}_{-12.0}{}^{\circ}$, and argument of perihelion $\omega = 210.9^{+15.9}_{-14.7}{}^{\circ}$. Furthermore, we establish a reference date in Julian Days $\tau = 2455258^{+125}_{-131}$. Using four observation dates, we illustrate the process of parameter determination, reserving one for validation. Our computed position for 26 Proserpina on January 26$^{\text{th}}$, 2012 yields a right ascension of 75.7°and a declination of 29.9°. This investigation enhances our comprehension of orbital dynamics and underscores the precision achievable through mathematical modeling and observational analysis in celestial mechanics.

1. Introduction

Everything in the night sky is moving whether it is noticeable on nightly or century intervals which scientists study to determine physical attributes of these celestial objects. Many celestial bodies, including planets, asteroids, and comets, traverse not in linear paths but in elliptical orbits around larger masses, typically stars. Orbits are the paths object follow repeatedly around another object. These orbits represent the repeated trajectories that objects trace around their central bodies. Keplerian orbits, named after Johannes Kepler, are the precise paths objects repetitively trace around central bodies. Our Earth is the best example as it orbits around the Sun in a repeated pattern in a specific amount of time we have labelled as a year. However, our solar system hosts a myriad of celestial bodies, including seven other planets and over a million asteroids within the Asteroid Belt, each following its unique orbital path.

Observing from Earth, we discern celestial motion against two principal planes. The first, the equatorial plane, slices through Earth's midsection along the equator, forming a fundamental reference for tracking celestial objects. In contrast, the ecliptic plane mirrors Earth's orbital path around the Sun, providing an additional viewpoint for studying cosmic phenomena. We see everything from the equatorial plane due to the tilt Earth's axis sits on so we need to apply calculations to find our positions in the ecliptic plane.

1.1. Keplarian Orbital Parameters

Keplerian orbital paths are defined by six physical parameters that fully characterize the trajectory of an object. Two crucial parameters, the semi-major axis (a) and eccentricity (e), determine the shape of the orbit. The semi-major axis signifies the mean distance traveled by the orbiting body, offering insight into the

orbit's size. Meanwhile, eccentricity delineates the orbit's shape, with a value of 0 indicating a perfect circle and increasing values indicating more elliptical paths. In addition, a single time value known as the epoch of perihelion (τ) serves as the temporal reference point for our calculations, enabling precise determination of the object's position at any given time.

Given that we exist in a three-dimensional space, three rotational angles are essential to define the object's orientation. These angles include inclination (i), longitude of ascending node (Ω), and argument of perihelion (ω). Inclination represents the vertical angle from the reference plane, akin to observing the horizon on Earth. The longitude of ascending node denotes the angle between the reference direction, typically the vernal equinox, and the point where the orbit intersects the reference plane from below. Similarly, the argument of perihelion signifies the angle between the ascending node and the orbit's closest approach to the Sun, measured within the orbital plane.

To determine these six parameters, we employ a set of equations detailed in Table 1. Armed with these parameters, we aptly describe the Keplerian orbit of any celestial object orbiting the Sun. Refer to Section 7 for a visual representation of the three angle parameters in Figure 3, aiding in understanding their spatial configuration and functionality.

Keplarian Parameter	**Equation**	**Equation Number**
Semi-Major Axis	$a = \frac{k^2 r}{2k^2 - rv^2}$	(1)
Eccentricity	$e = \sqrt{1 - \left(\frac{h^2}{ak^2}\right)}$	(2)
Longitude of Ascending Node	$\Omega = \arctan\left(\frac{-h_x}{h_y}\right)$	(3)
Inclination	$i = \arccos\left(\frac{h_z}{h}\right)$	(4)
Epoch of Perihelion	$\tau = t - \frac{M}{n}$	(5)
Argument of Perihelion	$\omega = \arccos\left(\frac{r_x \cdot \cos(\Omega) + r_y \cdot \sin(\Omega)}{r}\right) - \nu$	(6)

Table 1:: This table holds the 6 important equations needed for finding the parameters of any Keplarian orbit

1.2. Required Background

The goal of this lab is to apply Right Ascension and Declination positions to determine the Keplerian motion of asteroid 26 Proserpina. To demonstrate the methodology's efficacy, data from Table 4 is utilized to analyze the orbital motion of Ceres, a dwarf planet in the Asteroid Belt. True anomaly of Ceres is derived by calculating its mean anomaly (M) and eccentric anomaly (E), representing the mean motion and projection onto a circular track, respectively. While Equation 7 directly correlates these values, their analytical solution is unfeasible. Therefore, a Newtonian iteration using Equations 8 and 9 is employed to derive converging solutions, as detailed in Section 7.2.2. Subsequently, the solved mean and eccentric anomalies facilitate the determination of positions at various times, enabling a comprehensive visualization of Ceres' trajectory.

$$M = E - e\sin(E) \tag{7}$$

$$E_{n+1} = E_n + \frac{M_i - (E_n - e\sin(E))}{1 - e\cos E_n} \tag{8}$$

$$M_{i+1} = M_i + \sqrt{\frac{k^2}{a^3}}\delta t \tag{9}$$

After demonstrating the orbital motion using the six parameters, we proceed to analyze asteroid 26 Proserpina, which was previously observed in a lab experiment. In our prior report, we obtained the right ascension and declination of the asteroid on January 20$^{\text{th}}$, 21$^{\text{st}}$, 23$^{\text{rd}}$, and the 26$^{\text{th}}$ of 2012. The data originates from the Jet Propulsion Lab at NASA (NASA 2024). By utilizing the observations from the first three dates, we can determine the six parameters, enabling us to predict the position of the asteroid on the fourth day and any day throughout the year. Given that the asteroid orbits the Sun, but our observations are Earth-based, it's crucial to establish the correlation between the positions of the Earth, Sun, and 26 Proserpina.

This process involves manipulating Cartesian vectors to represent the positions and distances between the three celestial bodies, ultimately determining the distance between the Sun and the asteroid. We denote the distance between the Sun and Proserpina as $\vec{r}$, while the distance from the Sun to Earth is $\vec{R}$, with a magnitude of 1 AU. Similarly, the distance from Earth to the asteroid is denoted as ρ, also expressed in astronomical units. Computing the distance values for r and ρ begins with identifying the directional components of the vectors to facilitate vector addition.

Equation 10 depicts the conversion of right ascension (α) and declination (δ) values into Cartesian coordinates, resulting in a unit vector, $\hat{s}_{eq}$, with a magnitude of 1. This conversion is performed for each of the four observation dates. Subsequently, we need to adjust for Earth's tilt of $\epsilon = 23.44°$, applying the transformation matrix detailed in Equation 11. This yields the unit vectors of the Earth directed towards 26 Proserpina for the four observation dates, all situated within the ecliptic plane.

$$\begin{aligned} x_{eq} &= \cos\alpha\cos\delta \\ y_{eq} &= \sin\alpha\cos\delta \\ z_{eq} &= \sin\delta \end{aligned} \tag{10}$$

$$\hat{s} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\epsilon & \sin\epsilon \\ 0 & -\sin\epsilon & \cos\epsilon \end{pmatrix} \begin{pmatrix} x_{eq} \\ y_{eq} \\ z_{eq} \end{pmatrix} \tag{11}$$

To proceed further, we will employ Laplace's Method for orbit estimation, which relies on the vector triangle formed by the vectors connecting the Earth to the Sun, the orbiting object, and the vector from the Sun to the object. By iteratively differentiating, a sequence of equations was derived to determine the orbital parameters. Utilizing Laplace's method for asteroid motion prediction necessitates not only the unit vectors from Earth to 26 Proserpina but also their first two time derivatives. Since there is no direct equation available for obtaining the derivatives of the unit vectors, we resort to using the first three observations.

Through a Taylor Expansion, we solve for $\hat{s}_2$, $\dot{\hat{s}}_2$, and $\ddot{\hat{s}}_2$ using $\hat{s}_1$ and $\hat{s}_3$. Additionally, we require the Julian dates for these observations to employ Equation 12, where $\tau_1 = t_2 - t_1$ and $\tau_3 = t_3 - t_2$.

$$\begin{aligned} \dot{s_2} &= \frac{\tau_3(s_2 - s_1)}{\tau_1(\tau_1 + \tau_3)} + \frac{\tau_1(s_3 - s_2)}{\tau_3(\tau_1 + \tau_3)} \\ \ddot{s_2} &= \frac{2(s_3 - s_2)}{\tau_3(\tau_1 + \tau_3)} + \frac{2(s_2 - s_1)}{\tau_1(\tau_1 + \tau_3)} \end{aligned} \tag{12}$$

Subsequently, we must determine the value for ρ, but this is contingent upon knowing r. To ascertain both values, we employ an iterative solving approach. Initially, we make an educated guess for r, then compute ρ, which is subsequently utilized to refine the estimation of r. In cases where a viable orbit exists, the value for ρ tends to converge, providing a reasonably accurate assessment of the Earth-asteroid distance. Once ρ is established, the final value for $\vec{r}$ is obtained through simple vector addition of the vectors $\vec{R}$ and $\rho\hat{s}$. The equations necessary for the iterative solving process are detailed in Equation 13. This iterative procedure enables us to determine the rates of change of both ρ and $\vec{r}$, as delineated by the relations in Equation 14.

$$\rho = k^2 \left(\frac{1}{R^3} - \frac{1}{r^3}\right) \frac{\dot{\hat{s}} \cdot (\vec{R} \times \hat{s})}{\dot{\hat{s}} \cdot (\ddot{\hat{s}} \times \hat{s})}, \qquad r = \sqrt{\rho^2 + R^2 + 2\rho\vec{R} \cdot \hat{s}} \tag{13}$$

$$\dot{\rho} = \frac{k^2}{2} \left(\frac{1}{R^3} - \frac{1}{r^3}\right) \frac{\ddot{\hat{s}} \cdot (\vec{R} \times \hat{s})}{\ddot{\hat{s}} \cdot (\dot{\hat{s}} \times \hat{s})}, \qquad \dot{\vec{r}} = \dot{\vec{R}} + \rho\dot{\hat{s}} + \dot{\rho}\hat{s} \tag{14}$$

These calculations lead us to the final required values necessary for determining the six orbital parameters. First, we compute the specific angular momentum, which resembles the conventional angular momentum but excludes the mass factor, yielding Equation 15. This equation aids in determining all three angular parameters. Upon obtaining the semi-major axis and eccentricity, we proceed to calculate the eccentric and mean anomaly to derive the true anomaly, ν, of the orbit. Initially, we solve for E using Equation 16, then revert to Equation 7 to determine M. The true anomaly is then derived utilizing Equation 17, providing a description of the actual position of the orbiting body. Finally, in determining the epoch of perihelion, we rely on Equation 18, derived from Kepler's Third Law. Armed with the derived values from the aforementioned steps, we possess all the necessary tools for calculating the values of the six orbital parameters.

$$\vec{h} = \vec{r} \times \dot{\vec{r}} \tag{15}$$

$$E = \arccos\left(\frac{a - r}{ae}\right) \tag{16}$$

$$\nu = 2\arctan\left[\sqrt{\frac{1+e}{1-e}}\tan\left(\frac{E}{2}\right)\right] \tag{17}$$

$$n = \frac{k^2}{a^3} \tag{18}$$

With all the parameters found, we then can find the position of 26 Proserpina which requires we use the following equations listed in Equation 19 below. With all this we can find the radial distance r on that date

and coupled with the true anomaly we find the ecliptic then the equatorial coordinates which provides the needed values for the right ascension and declination on any date with Equation 20

$$r = a\left(1 - e\cos\left(E\right)\right), \qquad \theta = \nu + \omega \tag{19}$$

$$\alpha = \arctan\left(\frac{y_{eq}}{x_{eq}}\right), \qquad \delta = \arcsin\left(\frac{z_e q}{2}\right) \tag{20}$$

1.3. Uncertainty

The six Keplerian values are intricately interconnected, whereby influencing one parameter significantly affects the other five. Consequently, directly propagating uncertainties becomes challenging, necessitating a new statistical approach to ascertain uncertainties in our derived values. In this report, we endeavored to employ the Monte Carlo Markov Chains method (MCMC), which integrates Markov Chains into the Monte Carlo framework to generate random samples. Monte Carlo Markov Chains (MCMC) play a pivotal role in simulating complex systems with random variables. By leveraging Markov Chains, MCMC generates a sequence of states, each representing a sample of the system. It iteratively produces new states based solely on the current state, gradually converging to a stationary distribution. Widely applied across diverse scientific disciplines such as physics, biology, finance, and machine learning, MCMC proves particularly valuable for addressing scenarios with numerous variables, where traditional analytical methods encounter limitations. Recognized as a well-established method for handling uncertainty propagation in such contexts, MCMC stands as our recommended approach for determining accuracy. The implementation of the MCMC method is detailed in Section 7.2.4.

Collaborating with Umit Uzunboy, Maha Macknojia, Arvin Talwani, and Dilruba Yalcinakaya, this report is organized into several sections, commencing with a concise elucidation of the dataset employed in Section 2. Subsequently, Section 3 elucidates the methodology employed to determine the six orbital parameters, followed by the analysis of our derived parameters and uncertainty evaluation in Section 4. Section 5 encompasses the discussion of the obtained results and concludes our findings.

2. Data and Observations

Julian Date	Right Ascension [deg]	Declination [deg]
2455946.8	81.03	26.98
2455947.8	80.89	26.97
2455949.9	80.63	26.94
2455952.7	80.31	26.91

Table 2:: Over the four observed dates for 26 Proserpina, the following celestial positions are found. These values come from the JPL Horizons Ephemeris and list the date in Julian Dates as well as the right ascension and declination positions in degrees. With these values we can follow the listed process in this report to find the path 26 Proserpina follows in space.

Data for this lab is gathered from the JPL Horizons ephemeris from NASA. This collection of information is gathered from various missions conducted on ground and in space. This data base holds highly accurate data on over 1 million asteroids including 26 Proserpina. Our selected days of observation are the same as the previous Proserpina lab which determined the positions through plate solving. These days are listed in Table 2 above and when comparing the calculated results to JPL Horizons, the difference is very minimal. The data thus from the ephemeris is used in this report as they have better accuracy from advanced studies on the asteroid. The camera used in imaging 26 Proserpina leaves a pixel error of 0.53 arc seconds so this is the uncertainty we take for the above values.

3. Methodology

3.1. Ceres

As mentioned before, we want to show the use of the Keplarian parameters to describe the full path of Ceres. When given the 6 values shown in Table 4 in the Appendix, the next step is to provide an estimate of the motion in the ecliptic plane. In this plane we will not see the angles which describe the 3D position but the shape of the orbit will be well known. From here it becomes easy to see the radial distance of the orbiting body to the Sun. When iteratively solving for the mean and eccentric anomaly by using Equations 8 and 9 we find the motion. In the Data Analysis we look into the comparison of the true and the estimated values.

3.2. 26 Proserpina Parameters

Our first step to determining Proserpina's motion is to take our celestial coordinates and find the unit vector s which points from Earth to the asteroid in Cartesian coordinates. We start by using Equation 10 to find the vector in the equatorial plane. We need the position in the ecliptic plane to account for the tilt the Earth sits at on its axis. Equation 11 is used to rotate our vector where the x-axis is held fixed and rotated about by an angle $\epsilon = 23.44°$. With this we find the unit vectors pointing from the Earth to Proserpina for the 4 observation days.

The next step is to use the Equations 12 to determine s, $\dot{s}$, and $\ddot{s}$ for the second observation date. From here, we needed to gather the position of the Earth with respect to the Sun on the dates we observed 26 Proserpina's motion. By utilizing Python's astropy.coordinates.get_sun function and inputting the Julian dates of our observations we are given the exact right ascension and declination the Sun was at in the sky. Again Equation 10 is used and this gives S, the unit vector pointing from the Sun to the Earth.

Using the information we have collected, the next step is to find the distance Earth sits from Proserpina, ρ. As stated before, ρ cannot be solved directly and instead we will work to get an approximation. There are many methods that can be used to come to an answer and in this report we have chosen to work with an iterative solve. We have two required relations of ρ and r which we use for the iteration listed as Equations 13. We provide an initial guess of 1.1 AU for r and then plug this in to find a corresponding value for ρ. From here we plug this ρ to find an updated value of r. Repeating this process 100 times leads to Figure 4

With these found values, $\dot{\rho}$ can now be solved for from Equation 14 and then the next step is to find the heliocentric vector and its time derivative to the asteroid. This can be found by adding the Sun-Earth vector and the Earth-asteroid vector and then taking the derivative. This just uses the ρ magnitude multiplied

with the unit vector $\hat{s}$ which we found earlier. We label this as $\vec{r}_{helio}$ to give our confirmed distance from the Sun to 26 Proserpina.

Table 1 holds Equations 1 to 6 which provide the relations for all 6 of the Keplarian orbit parameters. There is an order of steps to follow in order to determine the parameters of the Proserpina's orbit. The easiest to find is the semi-major axis since it does not relate to any other parameters. Next to find is the argument of the perihelion which requires the specific angular momentum ($\vec{h}$) which is simply calculated by Equation 15. From here we get the longitude of the ascending node with a similar looking calculation used to also find inclination.

To find the eccentricity we now use Equation 2. From here we want the eccentric anomaly which is found in Equation 16 relating the semi-major axis, eccentricity, and heliocentric distance together. The mean anomaly is found from the eccentric anomaly with Equation 7. We quickly solve for the mean motion n with Equation 18 which allows us to solve for our epoch of perihelion with Equation 5. Now using the mean motion and anomaly alongside the Julian data of the second observation we find the true anomaly of the orbit with Equation 17. Lastly a relation between Ω and ν is found in Equation 6 to give us the argument of perihelion.

3.3. Uncertainty Calculations

The uncertainties are done in the MCMC method and we take small perturbations off the known positions and find the parameter values these new values provide us. We take 1000 samples and then plot them into histograms to show the uncertainty values they sit at. We only want to look at bounded orbits so we are required to remove any iterations which do not follow that. We determine these by finding parameters that do not make any physical sense and removing them before they can get added to the histogram. The rest are then used to find the relative deviation off the center value which gives us the uncertainty in our values. The histograms are found in the Appendix in Figure 6

3.4. Estimating Position on a 4^{th} Epoch

Now with the 6 values we are able to find the position of 26 Proserpina at any given time. We plug in our fourth observation day and find calculate the mean anomaly followed by the eccentric anomaly. This allows us to find the radius and true anomaly associated with that day. Using our radius and then the polar coordinate of the object using Equation 19, the ecliptic Cartesian coordinates are found and we convert these into equatorial coordinates. Using Equation 20 we have the last required relations to find the right ascension and declination.

4. Data Analysis

4.1. Analyzing the Ceres Parameter Estimation

When using the data in Table 4 to find the orbital motion of Ceres and plotting them we get the following plots shown in Figures 1 and 5. There is differences noticeable through the estimated value and this arises from the iterative solving being used. As we iterate more and more, small differences in parameter

values add up and quickly diverge from the known values. We see in that the values disperse but then do shift back.

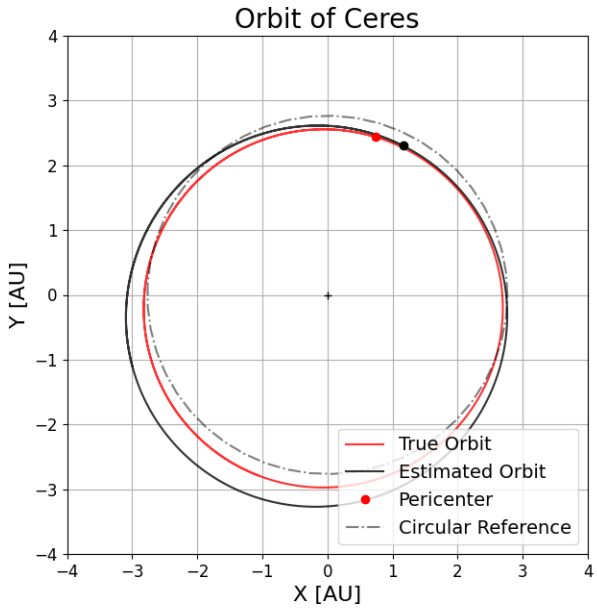

Fig. 1.—: This plot shows the motion of Ceres in the 2 dimensional orbital plane using the estimated and the true values. Only the eccentricity, semi-major axis, and the epoch of perihelion are used as they define the motion in the orbital plane. The remaining parameters are the angles which define the position of Ceres in 3 dimensional space. Near the pericenter, both the estimated and true values follow closely but on the opposite side it is clear that the difference in values creates a bigger difference in the two plots.

4.2. Determined Orbital Parameters of 26 Proserpina

Parameter	Determined Value	Real Value
a [AU]	$2.57^{+0.07}_{-0.07}$	2.652
e	$0.092^{+0.02}_{-0.02}$	0.087
Ω [degrees]	$46.9^{+8.6}_{-12.0}$	45.9
i [degrees]	$3.55^{+0.57}_{-0.57}$	3.56
τ [JD]	2455258^{+125}_{-131}	2454868
ω [degrees]	$210.9^{+15.9}_{-14.7}$	193.1

Table 3:: This table holds our found values for 26 Proserpina's orbital parameters. The values appear to be close to the values found by NASA (NASA 2024) and this tells us our program is a good estimate for determining the orbital motion of not only 26 Proserpina but other orbiting objects as well.

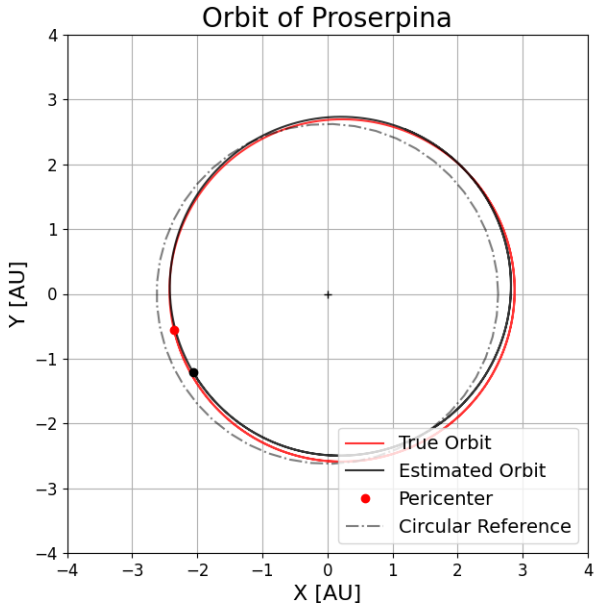

Fig. 2.—: This plot shows the motion of 26 Proserpina in the 2 dimensional orbital plane using the estimated and the true values. The paths follow close to one another and deviate by very small amounts and do align again after one full rotation.

We present the following values for the parameters of Proserpina in Table 3, alongside the real values determined by NASA. Notably, our determined values closely approximate those of NASA, which is encouraging. This alignment is particularly significant given our prior observations with Ceres plots, indicating that even minor deviations can significantly alter the orbit path of any celestial body. Consequently, our derived values offer an accurate depiction of the true orbit. However, it's essential to acknowledge that over extended periods, deviations may become more pronounced, leading to a decrease in accuracy as we move away from our estimated value for the epoch of perihelion. Figure 2 illustrates the motion we've determined for 26 Proserpina alongside its true motion. The discrepancy in pericenters suggests some disparities in our analysis, potentially stemming from differences in values.

Some uncertainties, unfortunately, exceed our desired threshold. Specifically, the longitude of the ascending node exhibits the largest percent error at 26% of the found value. While the true values for all parameters generally fall within the range we've calculated, there are exceptions. Firstly, the epoch of perihelion deviates by approximately 400 days from the true value. This discrepancy likely explains the differences in pericenter locations observed in Figure 2, as the epoch of perihelion specifies our reference time. Additionally, the argument of perihelion slightly exceeds the calculated range. Since these two parameters are calculated last among the six, errors in the preceding four calculations may contribute to this issue. Examining the histograms presented in the Appendix as Figure 6, we observe that the semi-major axis and eccentricity exhibit relatively good fits, while the longitude of the ascending node and inclination could benefit from further refinement. Notably, the plots for the epoch and argument of perihelion display poor fits, with the τ histogram revealing two peaks, indicating errors in the parameter solving process.

4.3. 4^{th} Epoch Values

The 4^{th} epoch input provides us with a right ascension of 75.7°and a declination of 27.9°. Although uncertainties in these values remain uncertain based on our parameters, we do note a deviation of approximately 5 degrees from the true values specified by NASA. This proximity suggests that while we possess some estimation of the parameter's location, it's not sufficiently accurate for precise observations. Addressing the source of these deviations is paramount for enhancing the utility of our calculation values in accurately determining positions of orbiting bodies. Primarily, it appears that errors in our parameter calculations are the likely culprit. Therefore, refining this process holds the best promise for achieving accurate measurements.

5. Discussion and Conclusion

While our current method for determining uncertainties is functional and yields valid errors for our values, there's certainly room for improvement in refining the Monte Carlo Markov Chain (MCMC) method. The most effective approach would involve delving into the parameter-solving code and exploring avenues to enhance its calculations. This endeavor would help mitigate larger errors stemming from smaller deviations, thereby bolstering confidence in our results. It is advisable to allocate additional time to develop this function for error propagation in this lab, as it would yield more robust answers. Furthermore, this process extends beyond the confines of this particular scenario, proving beneficial in various scientific inquiries. Hence, it is advantageous to delve deeper into studying and refining this method, ensuring readiness for future applications. The attempted implementation of this code is elucidated in Appendix Section 7.2.4.

In addition to the iterative solving method for ρ, an alternative approach we could have explored is utilizing the scipy.optimize function fsolve. This root-solving function, when applied to our functions, could assist in pinpointing convergence points for the values of ρ and r. However, the implementation did not yield the desired outcome during the course of this report, resulting in the return of the same inputted initial value. Nonetheless, refining the implementation of this method may yield closer approximations for the two values.

The primary objective of this lab is to determine the six parameters required to describe the position of any orbiting object at any given time. This capability proves invaluable in numerous astronomical scenarios, where scientists seek to observe specific celestial objects in the night sky. Events such as the appearance of Halley's comet and solar eclipses can be predicted based on knowledge of the orbits of each celestial body. This process has been instrumental in initiating many scientific breakthroughs. For instance, Einstein's theory positing that light bends in the presence of gravity was corroborated by observing star positions during total solar eclipses. Scientists were able to anticipate their observation opportunities by accurately calculating the Moon's position over time. The combined knowledge of the Sun's distance from the Earth and the Earth's distance from the object furnishes us with sufficient information to predict the object's appearance on any given day.

6. Bibliography

REFERENCES

NASA. 2024, Small-body database lookup

7. Appendix

7.1. Extra Figures

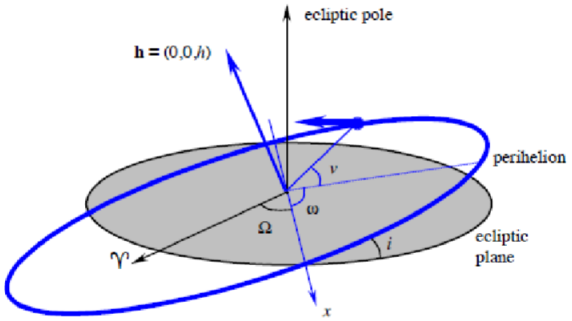

Fig. 3.—: The above picture is useful for providing a visual to the 3 angular parameters as it can be visually difficult to put an image to them. The orbital plane is in the path that the observed object follows while the plane of reference is where the Earth and Sun sit as they go around one another.

	a	Ω	i	e	ω	τ
	[AU]	[deg]	[deg]		[deg]	[Julian Day]
True	2.766	80.72	10.61	0.079	73.12	2454868
Estimated	2.946	80.65	10.56	0.125	63.20	2454833
Error	-0.18	0.07	0.05	-0.05	9.9	35

Table 4:: Provided from the Lab document we are provided with the following orbital parameters for Ceres' orbit. Using the values found we can see how estimated values and the true values compare when measuring the orbital path of a moving body.

This preprint was prepared with the AAS LaTeX macros v5.0.

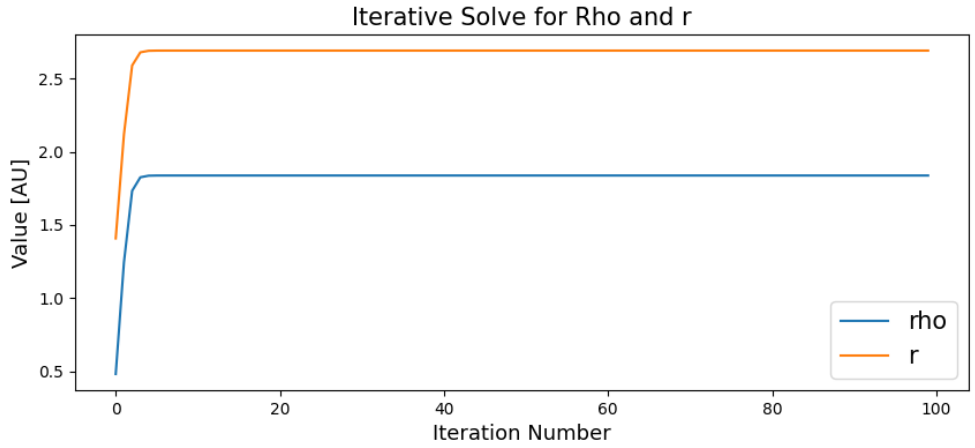

Fig. 4.—: The above plots show the change in the values for (a) ρ and (b) r over the 100 iterations. They both converge to a singular value which we can take to be the approximate value for both

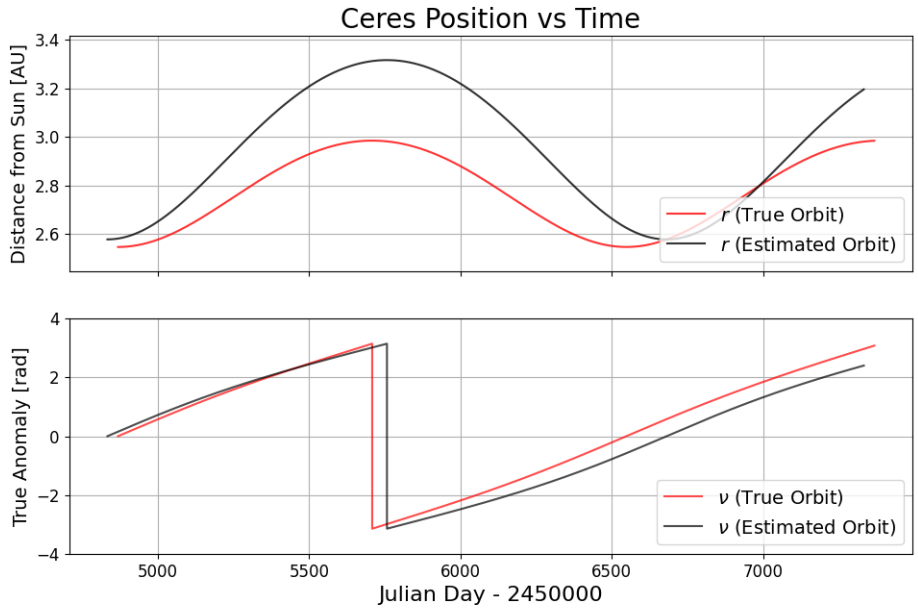

Fig. 5.—: The two plots show (a) radial distance Ceres is from the Sun and (b) the true anomaly. By using the parameters given from Table 4 to determine the motion we see how the estimates although off by no more than

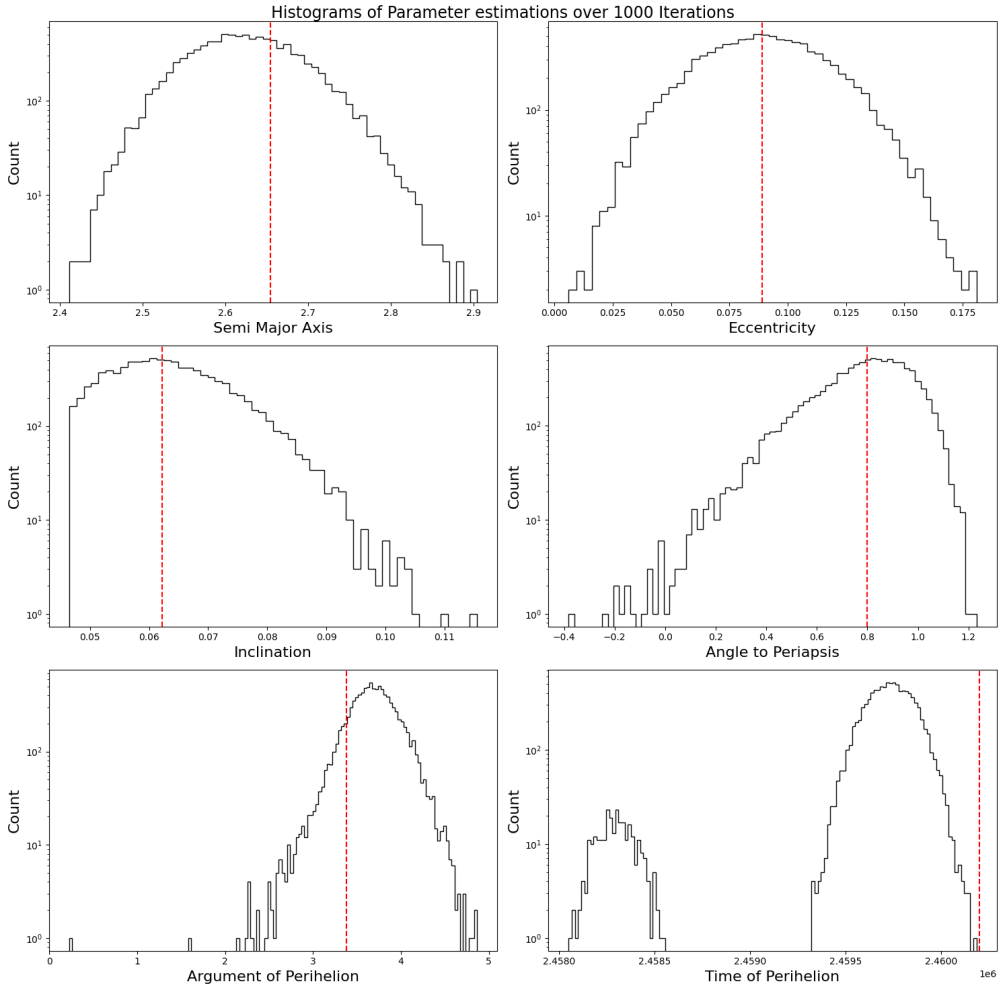

Fig. 6.—: The histograms of each parameter is shown above. The red lines represent where the found value is and the spread depicts how close the 1000 iterations sat in relation to that value. The semi-major axis and eccentricity appear well and the middle row is alright but the bottom row depict very weak uncertainties.

7.2. Python Functions

7.2.1. Newton Iterator

This function performs the Newtonian iteration to find the mean and eccentric anomaly when provided with the orbital parameters of the object.

```
def newton_iterator(k, e, a, tau, dt = 0.01):
    """
    Performs Newtonian iteration on the given parameters to find the eccentric and mean
    anomaly of orbit
    :param k: constant with value of 0.0172
    :param e: eccentricity of orbit
    :param a: semi major axis of orbit
```

```
    :param tau: epoch of perihelion
    :param dt: time step
    :return: eccentric anomaly array, mean anomaly array, associated times
    """

    # Initialize arrays
    t = np.arange(tau, tau + 2500, dt, dtype=float)
    E = np.zeros_like(t, dtype=float)
    M = np.zeros_like(t, dtype=float)

    # Initial values
    i = 0

    # Iterate over and over to find the converging values for E and M
    while i < len(t) - 1:
        E[i + 1] = E[i] + (M[i] - E[i] + e * np.sin(E[i])) / (1 - e * np.cos(E[i]))
        M[i + 1] = M[i] + np.sqrt((k ** 2) / (a ** 3)) * dt
        i = i + 1

    return E, M, t
```

7.2.2. Parameter Solving

The following code goes through the process of taking the Julian dates and the associated right ascension and declination in degrees to find the 6 orbital parameters. First is the iteration function which iterated 100 times to find the converging values for ρ and r looking for their convergence. The second function is the full solve function which utilizes that iteration to find the parameter values.

```

def iterative_rho(r, R, k, s, s_dot, s_ddot, N):
    """
    Iteravley solve for rho by setting an initial r value and finding the associated rho value, using that rho to
    find a new r we can loop this process over and over to find where the values converge. If they do not then we would
    have an unbounded orbit where the object is moving away from Earth never to return.

    :param r: input distance from Sun to orbiting object
    :param R: Position vector of Sun from Earth
    :param k: k constant
    :param s: unit vector from Earth to object
    :param s_dot: first time derivative of s
    :param s_ddot: second time derivative of s
    :param N: Number of iterations to go through
    :return: arrays containing the interated r and rho values found
    """
    # Initialize the lists
    rho_list, r_list = [], []

    # Find magnitude of the R vector
    R_mag = np.linalg.norm(R)

    # Iteravley solve for rho by finding r then using that to find rho and keep looping until the values converge
    for i in range(0, N):
```

```
        denom = np.dot(s_dot, np.cross(s_ddot, s))
        rho = k ** 2 * (1 / R_mag ** 3 - 1 / r ** 3) * (np.dot(s_dot, np.cross(R, s))) / denom
        rho_list.append(rho)

        r = np.sqrt(rho**2 + R_mag**2 + 2*rho*np.dot(R, s))
        r_list.append(r)

    return np.array(r_list), np.array(rho_list)

def full_solve(jd, ra, dec):

    Compute all 6 orbital parameters of an orbiting object given at least 3 observation dates and the corresponding right
    asencsion and declination
    """
    :param jd: array holding the dates of observation is Julian Days
    :param ra: array holding right ascension values in radians
    :param dec: array holding declination values in radians
    :reutrn: 6 orbital parameters in the following order:
            semi-major axis, eccentricity, inclination, longitude from ascending node,
            argument of perihelion, epoch of perihelion
    """


    x_eq = cos(dec) * cos(ra)
    y_eq = sin(ra) * cos(dec)
    z_eq = sin(dec)

    """Using Equations 61 from the lab document to find the cartesian equatorial positions.
    """

    # Equatorial unit vector
    s_eq = np.array([x_eq, y_eq, z_eq])

    # Next Equation 62/63 is used to rotate the positions of the equitorial frame to Proserpinas plane of orbit
    eps = np.radians(23.43929111)
    T = np.array([[1, 0, 0], [0, cos(eps), sin(eps)], [0, -sin(eps), cos(eps)]])

    # Get the s unit vector for Proserpinas movement. A good check is to see if the magnitude equals to 1
    s = np.dot(T, s_eq)

    ## Finding the first and second time derivates of s

    # First we get the s unit vectors separately for the first, second and third measurement
    s1 = s.T[0]
    s2 = s.T[1]
    s3 = s.T[2]

    # Calculate the first and second time derivatives for s by using this function
    s2_dot, s2_ddot = pf.taylor_expand(jd, s1, s2, s3)

    # Collected the Julian day time of the first observation of Proserpina

    # Convert the julian day into an astropy Time object
    obs_time = Time(jd, format="jd")
```

```
78
79      # Get the solar coordinates at the observing times
80      solar_coordinates = get_sun(obs_time)
81      solar_ra = solar_coordinates.ra.deg
82      solar_dec = solar_coordinates.dec.deg
83
84      # Using Eq 61 we convert the celestial coordinates into equatorial coordinates
85      ra_sun = np.deg2rad(solar_ra)
86      dec_sun = np.deg2rad(solar_dec)
87
88      X, Y, Z = pf.celestial_to_equatorial(ra_sun, dec_sun)
89      S_eq = -np.array([X, Y, Z])
90
91      # Using the T epsilon matrix from before, we find the S unit vector in the eccliptic
        plane
92      S = np.dot(T, S_eq)
93
94      S1 = S.T[0]
95      S2 = S.T[1]
96      S3 = S.T[2]
97
98      ## Now we work to find $r$, $\rho$ and $\dot{\rho}$
99
100     # We first need the k constant found on Page 2 of the lab document
101     k = np.sqrt(const.G * const.M_sun).to(u.AU ** (3 / 2) / u.day).value
102     R = S2
103     r = 1.1  # Initial guess of r
104
105     rerpina = 1.1
106     N = 50
107     rs = []
108     rhos = []
109
110     for i in range(N):
111         rho = pf.rho_from_r(S2, r, s2, s2_dot, s2_ddot, k)
112         rhos.append(rho)
113         r = pf.r_from_rho(rho, S2, s2)
114         rs.append(r)
115
116
117     # Now with our rho value, we find r using R as well in Equation 48
118     R_mag = np.linalg.norm(R)
119
120     # Then using Equation 49, we can find drho/dt
121     denom = np.dot(s2_ddot, np.cross(s2_dot, s2))
122     rho_dot = (k ** 2) / 2 * (1 / R_mag ** (3) - 1 / r ** (3)) * (np.dot(s2_ddot, np.cross(R
        , s2)) / denom)
123
124     ## Now we must find $r$ in the heliocentric position and $\dot{r}$ as well
125
126     # First using Equation 41 we can find the heliocentric position
127     r_helio = R + rho * s2
128
129     # We use Equation 50 again to find the first time derivative for R. The second time
        derivative is not needed
130     R_dot, _ = pf.taylor_expand(jd, S1, S2, S3)
131
132     # We now use Equation 46 to find the first time derivative of r
```

```
    r_dot = R_dot + rho * s2_dot + rho_dot * s2

    ## Now we are going to find the Keplarian Orbital elements

    r_vec = r_helio

    # First we use Equation 51 to find the semi major axis
    # Eq 51: At the midpoint of our three asteroid observations we know the magnitude
    # of the radius vector, r, and the total velocity, V.
    r_mag = np.linalg.norm(r_vec)
    V_mag = np.linalg.norm(r_dot)

    # Semimajor axis
    a = (k ** 2 * r_mag) / (2 * k ** 2 - r_mag * V_mag ** 2)

    # Eq 5: Compute h the specific angular momentum
    h = np.cross(r_vec, r_dot)

    # Eq 54: Get Omega
    omega = np.arctan(-h[0] / h[1])

    # Eq 55: Get i
    h_mag = np.linalg.norm(h)
    inclination = np.arccos(h[2] / h_mag)

    # Eq 56: Get e
    eccentricity = np.sqrt(1 - ((h_mag ** 2) / (a * k ** 2)))

    # Eq 57: Get eccentric anomaly, the first few lines here help with the sign issue
    theta = np.arccos(r_vec[2] / r_mag)
    phi = np.arctan2(r_vec[1], r_vec[0])
    e_r = np.array([np.sin(theta) * np.cos(phi), np.sin(theta) * np.sin(phi), np.cos(theta)
    ])
    v_r = np.dot(r_dot, e_r)
    ecc_anomaly = (np.arccos((a - r_mag) / (a * eccentricity)) * np.sign(v_r))
    # Eq 25: Get the mean anomaly
    mean_anomaly = -(ecc_anomaly - eccentricity * sin(ecc_anomaly))

    # Get Mean Motion
    n = np.sqrt((k ** 2) / a ** 3)

    # Eq 26: Get tau
    tau = jd[1] - mean_anomaly / n

    # Orbital period
    p = 2 * pi / n

    # Eq 31: True Anomaly
    true_anomaly = 2 * np.arctan((np.sqrt((1 + eccentricity) / (1 - eccentricity))) * np.tan
    (ecc_anomaly / 2))

    # Eq 58: Get Argument of Perihelion
    arg_perihelion = np.arccos((s2[0] * cos(omega) + s2[1] * sin(omega)) / r_mag) -
    true_anomaly

    return np.array([a, eccentricity, inclination, omega, arg_perihelion, tau])
```

7.2.3. Finding Position on Future Date

Here we can use the found parameters to then find the right ascension and declination of the asteroid given any date.

```
def get_ra_dec(params, t):

    """
    Calcualte the position of the object based on the epoch and the 6 orbital parameters
    :param params: array of length 6 holding the parameters of the orbiting body
    :param t: time to determine position at
    :return: right ascension and declination in degrees
    """

    a, e, i, omega, arg_perihelion, tau = params

    # Gather mean anomaly and iteravly solve for the eccentric anomaly
    mean_anomaly = n * (t - tau)
    ecc_anomaly = mean_anomaly
    for j in range(100):
        ecc_anomaly = ecc_anomaly + (mean_anomaly - ecc_anomaly + eccentricity * np.sin(
    ecc_anomaly)) / (1 - eccentricity * np.cos(ecc_anomaly))

    # Use the eccentric anomaly and eccentricity to determine position and true aniomaly
    r = a * (1 - e * np.cos(ecc_anomaly))
    nu = 2 * np.arctan(np.sqrt((1 + e) / (1 - e)) * np.tan(ecc_anomaly / 2))
    theta_epoch = nu + arg_perihelion

    # Find the x, y, z positions first in ecliptic then convert to equatorial frame
    x = r * (cos(omega) * cos(theta_epoch) - sin(omega) * cos(inclination) * sin(theta_epoch
    ))
    y = r * (sin(omega) * cos(theta_epoch) + cos(omega) * cos(inclination) * sin(theta_epoch
    ))
    z = r * np.sin(inclination) * np.sin(theta_epoch)

    x_eq = x
    y_eq = y * np.cos(epsilon) - z * np.sin(epsilon)
    z_eq = y * np.sin(epsilon) + z * np.cos(epsilon)

    r_eq = [x_eq, y_eq, z_eq]

    # Find the right ascension and declination
    ra = np.arctan(y_eq / x_eq)
    dec = np.arcsin(z_eq / 2)

    return np.rad2deg(ra), np.rad2deg(dec)
```

7.2.4. Uncertainty Propagation with Monte Carlo Makrov Chain

This code performs the MCMC calculations on our parameters to find their uncertainties

```
    """Perturb our inputs to our asteroid"""
np.random.seed(1003)
NUM_ARCSEC = 0.53
position_err_deg = NUM_ARCSEC/3600

```

```
trial_ra = ra_deg + np.random.normal(0, position_err_deg, len(ra_deg))
trial_dec = dec_deg + np.random.normal(0, position_err_deg, len(dec_deg))

trial_solution = mf.full_solve(jd, np.radians(trial_ra), np.radians(trial_dec))

"""print("\nSemi Major Axis: {:.2f}".format(trial_solution[0]))
print("Eccentricity: {:.3f}".format(trial_solution[1]))
print("Inclination: {:.2f}".format(trial_solution[2]))
print("Angle to Periapsis: {:.2f}".format(trial_solution[3]))
print('Argument of Perihelion {:.2f}'.format(trial_solution[4]))
print("Time of Perihelion: {:.2f}".format(trial_solution[5]))"""

N = 1000
mc_solutions = np.empty((N, 6))
for i in range(N):
    trial_ra = ra_deg + np.random.normal(0, position_err_deg, len(ra_deg))
    trial_dec = dec_deg + np.random.normal(0, position_err_deg, len(dec_deg))
    mc_solutions[i,:] = mf.full_solve(jd, np.radians(trial_ra), np.radians(trial_dec))

a_filter = (mc_solutions[:, 0] > 0.1) & (mc_solutions[:, 0] < 10)
# peri_filter = (mc_solutions[:,3] > 1.25) & (mc_solutions[:,3] < 1.5)

_filter = a_filter

mc_solutions[:, 4] = mc_solutions[:, 4] % 2 * np.pi

print(f"Number of closed orbit {np.sum(_filter)}")
parameters_labels = ["Semi Major Axis", "Eccentricity",
                     "Inclination", "Angle to Periapsis",
                     "Argument of Perihelion", 'Time of Perihelion']

#true_values = [2.766, 0.079, np.deg2rad(10.61), np.deg2rad(80.72), np.deg2rad(73.12),
    2454868.0]

"""for i in range(6):
    param = mc_solutions[_filter, i]
    plt.figure(figsize=(12, 5))
    plt.hist(param.flatten(), bins='fd', histtype='step', color='k')
    plt.axvline(x=true_values[i], color='red', linestyle='--')
    plt.xlabel(parameters_labels[i], size=16)
    plt.ylabel("Count", size=16)
    plt.yscale('log')"""
    #plt.show()

import corner
_range = []
for i in range(6):
    param = mc_solutions[_filter,i]
    _range.append((np.percentile(param,1),np.percentile(param,99)))

import emcee

def gauss_lk(x, mu, sigma):
    N = x.size
    return -0.5 * N * np.log(2 * np.pi) - N * np.log(sigma) - 0.5 * np.sum(( (x - mu) /
    sigma ) ** 2)

def lk_input(p):
```

```
    mu = np.hstack((ra_deg, dec_deg))
    #Here we are setting the measurement error on our inputs
    NUM_ARCSEC_ERR = 0.5
    sigma = 1/3600 * NUM_ARCSEC_ERR
    return sum([gauss_lk(p[i], mu[i], sigma) for i in range(p.size)])

def lk_prior(p):
    """
    Here we are setting some reasonable estimates for how what we think the output should looks like
    These can come from previous experiments, other kinds of prior knowlede, here I just used the known truth
    Values with some massive errorbars and assumed gaussian probability around that point.
    """
    mu = np.array([2.766, 0.079, np.deg2rad(10.61), np.deg2rad(80.72), np.deg2rad(73.12),2454868.0])
    sigma = [mu[0]*0.5, 0.05, np.pi/12, np.pi/2, np.pi/2, np.pi/2, 365]
    lk = sum([gauss_lk(p[i], mu[i], sigma[i]) for i in range(p.size)])
    return lk

def lnprob(p):
    x = mf.full_solve(jd, p[:3],p[-3:])
    #Throw out unbound orbits
    if x[0] == -1:
        return -np.inf
    return lk_input(p) + lk_prior(x)

nwalkers = 250
ndim = 6
p0 = np.empty((nwalkers, ndim))
for i in range(nwalkers):
    p0[i,:] = np.hstack((ra_deg, dec_deg)) + np.random.normal(0, 1/3600 * 0.25,6)
# print(lnprob(np.hstack((lamdba_ceres + 1,beta_ceres))))
sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob)

print("Running burn-in...")
p0, _, _ = sampler.run_mcmc(p0, 100)
sampler.reset()

niter = 200
print("Running production...")
pos, prob, state = sampler.run_mcmc(p0, niter)

print(np.mean(sampler.acceptance_fraction))
for j in range(6):
    for i in range(sampler.chain.shape[0]):
        plt.plot(sampler.chain[i,:,j], color = 'k', alpha = 0.2)
    plt.show()
```