Astrometry from CCD Images

Ayushi Singh, Anita Bahmanyar, Morten Nyborg Stostad

ayushi.singh@mail.utoronto.ca

Astronomy and Physics, University of Toronto, ON

4 December, 2013

Abstract

This experiment examines the five epoch of 30 Urania asteroid to obtain the orbital elements and the model of the orbit. Python programming language and DS9 software was used to inspect, calculate and model the data. The position of asteroid at each epoch, after correcting for plate constants, was obtained in lab 3. However, due to high residuals and high χ^2_{red} for x and y, the these position were incorrect. Therefore using positions from JPL Horizons all orbital elements were calculated within small errors. This was used to predict α and δ at another time: $t = 2455950.68528$, where $\alpha = 3{:}00{:}18.63$, $\Delta\alpha = 0{:}01{:}18.83$ and $\delta = +19{:}23{:}10.58$, $\Delta\delta = +0{:}00{:}53.11$. Similarly, this was done for all five epoch, which showed variation within errors indicating that the result was not very consistent.

1 Introduction

The purpose of this experiment was to find the orbital elements of the 30Urania Asteroid and model the orbit. It was done using the data collected from Dunlap Institute Telescope on five different days and the position of the 30Urania Asteroid in equatorial coordiantes calculated in lab3 [?]. The orbit of an object around sun is the path it takes over the course of its period. This object will have six orbital elements that will determine the shape and orientation of the orbit and the period of the object. These elements are: semi-major axis, eccentricity, inclination, longitude of ascending node, epoch of perihelion, and argument of perihelion. These six elements are constants and therefore, can be used to calculate the position of this object at any given time (Julian Day), by calculating true anomaly, mean anomaly and eccentric anomaly. In this lab, all six orbital elements of 30Urania Asteroid are calculated.

The report explains the step by step process used to obtain the final result with errors. After observed position of the Asteroid was corrected using plate constants for each epoch, it was converted from equatorial coordiantes into ecliptic coordiantes. Using these position and the distance of Earth from sun from JPL [?], all the six orbital elements were calculated and the orbit of asteroid was modelled.

2 Equipment

The data was collected using Dunlap Institute Telescope (DIT) that was located at New Mexico Skies on Mt. Joy site near Mayhill, NM. It was a 50-cm robotic telescope dedicated to the search for optical counterparts of gamma-ray bursts. The telescope had 4096 $\times$ 4096 pixel CCD array, with the view of 36 $\times$ 36 arc minutes. The focal length being 3454 mm and with F/ratio = F/6.8 [?]. Also, an online catalog was used for the obtaining the position of Earth from Sun at each epoch. This catalog is called JPL's Horizons web-interface [?]. It was also used to check the accuracy of our results. For the data reduction, calculations and modelling, DS9 software and Python Programming Language was used. DS9 provided the visual representation of the data set, including all the statistic about the data in the header. And Python allowed to perform calculations and modelling of the data.

3 Data Summary

The data was already collected before hand and the link to the data was provided to us at the start of this experiment on the course website [?].

Table 1: Summary of all the data collected for this experiment

Data #	Name	Date DD/MM/YYYY	Time [UTC time]	Julian Date	Comments
1	30 Urania	20/01/2012	04:28:30	2455946.68646	Asteroid
2	30 Urania	21/01/2012	04:40:27	2455947.69476	Asteroid
3	30 Urania	23/01/2012	05:43:40	2455949.73866	Asteroid
4	30 Urania	24/01/2012	04:26:48	2455950.68528	Asteroid
5	30 Urania	29/01/2012	01:27:18	2455955.56062	Asteroid

Each dataset has 3 files provided in them. This includes, raw data, dark counts due to CCD and flat-field taken at either dawn or twilight. The flat-field has already been corrected.

4 Data Reduction and Method

4.1 Positions of the Asteroids

The first was to extract the position of the asteroid at each epoch, Table 1, from the data file using the software DS9. It was done by compatring the patten between each dataset and finding one object that seem to move with respct to the backgound pattern. Then using python centroids were found for each epoch and was compares with USNO calalog. Using the method descrided in detail in AST356 Lab 3 [**?**], plate constants were derived for each file. These constants were derived again because in the lab 3, it was wrongly assumed that one plate constants can be used for all five epoch. Therefore, in this lab, each one were calculated seperatly and used to get the correct position of the 30 Urania at each day.

Table 2: Comparing Right Ascension and Declination of Urania over 5 epochs.

Date	α [hms J2000]	δ [dms J2000]	α (JPL) [hms J2000]	δ(JPL) [dms J2000]	$\Delta\alpha$ [s]	$\Delta\delta$ [s]
20	2:57:49.32	+19:14:30.75	2:57:49.20	+19:14:30.6	0.12	0.69
21	2:58:44.60	+19:16:27.53	2:58:44.52	+19:16:46.1	0.08	18.57
23	3:00:42.06	+19:21:40.13	3:00:41.30	+19:21:39.6	0.76	0.53
24	3:01:37.54	+19:23:44.52	3:01:37.46	+19:23:03.7	0.08	40.82
29	3:07:02.81	+19:38:13.85	3:07:01.64	+19:38:19.4	1.17	5.55

Table 2, has results from extimated position with corrected plate constants. There are compared with the position from JPL. For $\Delta\alpha$, the values are very small that most are less than a second. However, for $\Delta\delta$, the values are as high as 40.8 s. This is probably the result of the high resuides while calcualting plate constants. The χ^2_{red} value was quite big. For example, for data number 3 from Table 1, χ^2_{red} values are for $\delta = 63.30$ and $\alpha = 0.52$, which are quite large. But, it can't be reduced any further.

4.2 Converting coordiantes

The position obtained are in eqatorial coordinates. First, using the following calculations, α and δ were converted in cartesian coordinate.

$$\begin{aligned} x_{eq} &= \cos\alpha\cos\delta \\ y_{eq} &= \sin\alpha\cos\delta \\ z_{eq} &= \sin\delta \end{aligned} \tag{1}$$

Then, they are transformed into ecliptic cartesian coordiantes for further calcualtion using the folling transformation.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\ \epsilon & \sin\ \epsilon \\ 0 & -\sin\ \epsilon & \cos\ \epsilon \end{pmatrix} = \begin{pmatrix} x_{eq} \\ y_{eq} \\ z_{eq} \end{pmatrix} \tag{2}$$

where ϵ is 23.4392911°. This then forms a unit vertor $\mathbf{s} = [x, y, z]$, repersnting the distance between asteroid and the Earth.

4.3 Finding r and ρ

Using $\mathbf{s}$ for first 3 epoch and following equations $\dot{\mathbf{s}}$ and $\ddot{\mathbf{s}}$ for second epoch was calculated.

$$\begin{aligned}\dot{\mathbf{s}}_2 = & \ \frac{\tau_3(\mathbf{s_2}-\mathbf{s_1})}{\tau_1(\tau_1+\tau_3)} + \frac{\tau_1(\mathbf{s_3}-\mathbf{s_2})}{\tau_3(\tau_1+\tau_3)} \\ \ddot{\mathbf{s}}_2 = & \ \frac{2(\mathbf{s_3}-\mathbf{s_2})}{\tau_1(\tau_1+\tau_3)} - \frac{2(\mathbf{s_2}-\mathbf{s_1})}{\tau_3(\tau_1+\tau_3)}\end{aligned} \tag{3}$$

where $\tau_1 = t_2 - t_1$ and $\tau_3 = t_3 - t_2$, and these t values were used from table 1. This method was derived from taylor expension. Then distance of Earth from Sun on the second epoch was recored to be $\mathbf{R} = [-5.005639235484095e{-}01, 8.476786115155864e{-}01, -2.723171022212407e{-}05]$ from JPL Horizons [**?**].

$$\begin{aligned}\rho = & \ k^2\left(\frac{1}{R^3} - \frac{1}{r^3}\right)\frac{\dot{\mathbf{s}}\cdot(\mathbf{R}\times\mathbf{s})}{\dot{\mathbf{s}}\cdot(\ddot{\mathbf{s}}\times\mathbf{s})} \\ r^2 = & \ \rho^2 + R^2 + 2\rho\mathbf{R}\cdot\mathbf{s}\end{aligned} \tag{4}$$

In both of these equation there is r and ρ. Therefore, to calculate r and ρ a loop in python was created and iterated over 100 increments, see appendix A for the code. And the result was the following:

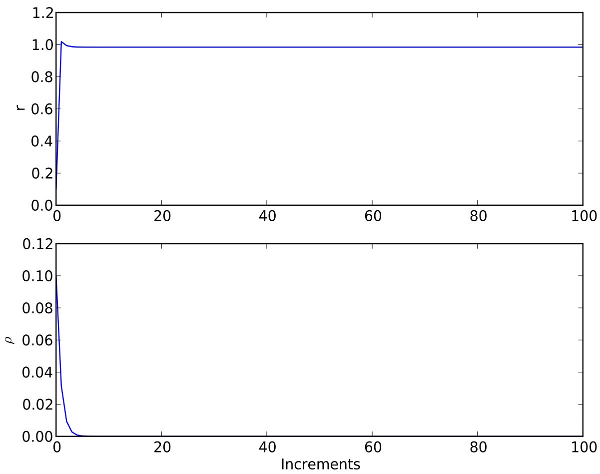

Figure 1: r and ρ value derived using estimated position of Urania.

Figure 1, r and ρ converge to 1 and 0, respectively. However, this is not correct for rho being 0 implies that distance better asteroid and Earth is zero. This is because the distace between two is given by $\rho\mathbf{s}$. Even after many trials, the errors were consistent. They possible reason is discussed in the Section 6.1. So, to continue with the lab and caculate orbital elements, JPL positions were considered, which gave the following result.

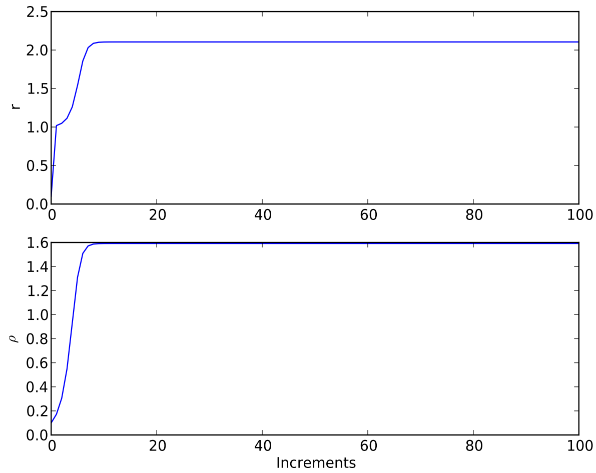

Figure 2: r and ρ value derived using JPL position of Urania.

Figure 2, clearly give us a convergence of r to 2.105 and ρ to 1.591. The true value for r and ρ are 2.119 and 1.605, respectively. And the estimated results are within 1% error.

5 Calculation and Modeling

5.1 Calculating orbital elements for Urania

From $\mathbf{s}, \mathbf{R}$ and ρ, the following can be calculated

$$\begin{aligned} \mathbf{r} = & \quad \mathbf{R} + \rho\mathbf{s} \\ \ddot{\mathbf{r}} = & \quad \ddot{\mathbf{R}} + \rho\dot{\mathbf{s}} + \dot{\rho}\mathbf{s} \\ \mathbf{h} = & \quad \mathbf{r} \times \dot{\mathbf{r}} \end{aligned} \tag{5}$$

Here, $\mathbf{r}$ is the vector distance between Sun and the asteroid and $\ddot{\mathbf{r}}$ is the velocity. $\mathbf{h}$ represents the vector perpendicular to the orbital plane of the asteroid. Now, there are few equations that uses variables from equation 5 to calcualte the orbital elements.

$$\begin{aligned} a = & \quad (k^2 r)/(2k^2 - rV^2), V = |\dot{\mathbf{r}}|^{\mathbf{2}} \\ e = & \quad \sqrt{1 - (h^2/ak^2)} \\ \cos i = & \quad h_z/h \\ \tan\Omega = & \quad -h_x/h_y \end{aligned} \tag{6}$$

where a is semi-major axis, e is eccentricity, i is inclination, $Omega$ is longitude of ascending node. These gives four of the six elements. Furthermore, using a period of the asteroid can be calculated by $p = 2\pi/n$. Where, $n^2a^3 = k^2$ and $k = \sqrt{GM_\odot} = 0.017202098950$ AU$^{3/2}$ d^{-1}. For Urania, 1263.45 days is the estimated period. Using same method, true period comes to be 1328.44 days. Therefore, there is a error of 64.70 days between true and estimated values.

$$
\begin{aligned}
cosE &= a - r/ae \\
M &= E - esinE \\
\tau &= t - (M/n) \\
\nu &= 2\arctan\left[\sqrt{\tfrac{1+e}{1-e}}\tan E/2\right] \\
\cos(\nu - \omega) &= (x\cos\Omega + y\sin\Omega)/r
\end{aligned}
\tag{7}
$$

here, the last two orbital elements are calculated, τ is epoch of perihelion and ω is argument of perihelion. Furthermore, three more equations are also defined in equation 7. E is eccentric anomaly, M is mean anomaly and ν is true anomaly. The sign on E is important and need to be placed seperatly. If the object is going from perihelion to aphelion then E ¿ 0. However, if object is going from aphelion to perihelion then E ¡ 0. In this case, the vaule of E is kept positive because using the $\dot{\mathbf{r}}$, it is deduced that asteroid is going from from perihelion to aphelion.

Table 3: Comapring true and estimated orbital elements for Urania.

	a [AU]	Ω [deg]	i [deg]	e	ω [deg]	τ [Julian Day]
True	2.365	307.93	2.098	0.127	86.27	
Estimate	2.287	308.26	2.084	0.116	74.24	2455802.67
Error	0.022	-0.33	0.014	0.011	12.03	

This table discribes the data estimated and compare it with the true vaules obtained from JPL.

5.2 Modelling the orbit of Urania

Using the orbital elements and python coding, a model of the asteroid's orbit was created. The time ellapse was from τ to (τ + period of the orbit + 500 days).

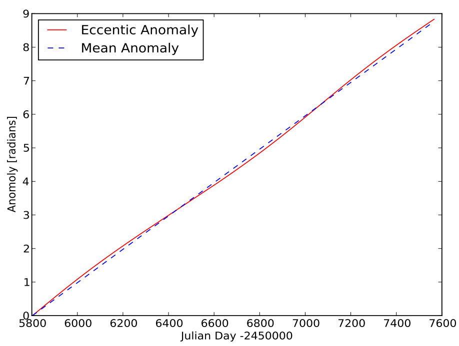

Figure 3: Mean anomaly and eccentric anomaly vs. time

Figure 3 shows the procession of mean anomaly and eccentric anomaly over time for Urania.

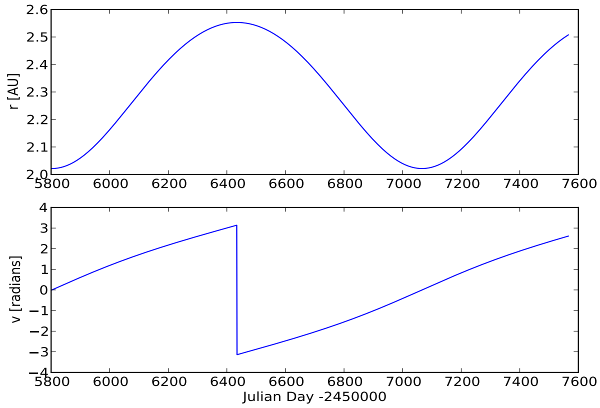

Figure 4: Radial sepration and true anomaly

Figure 4 shows the radial sepration from Sun and the true anomaly, which is the angular distance from ω as the asteroid moves in it's orbit.

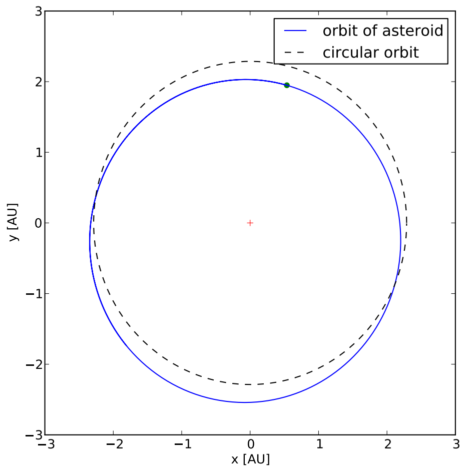

Figure 5: Full orbit of Urania compared with a circular orbit, which has Sun in the center. The green dot on the orbit of Urania is the point where perihelion occues.

In figure 5, it is clear that the perihelion occurs where the orbit is closer to the center, that is, the Sun.

5.3 Predicting positions

Using the equation from equation 5, 6 and 7, the postion was predicted for the same five epoch so that it was easy to comapre and check for the accuracy, see Appendix B for code. Following is the table that shows the predicted results.

Table 4: Comparing predicted Right Ascension and Declination of Urania over 5 epochs with JPL.

Date	α	δ	$\Delta\alpha$	$\Delta\delta$
	[hms J2000]	[dms J2000]	[hms J2000]	[dms J2000]
20	2:53:38.01	+19:55:29.73	0:04:11.18	0:19:00.87
21	2:55:17.43	+19:02:28.03	0:03:27.03	0:14:18.07
23	2:58:37.80	+19:16:18.51	0:02:03.50	0:05:21.09
24	3:00:18.63	+19:23:10.57	0:01:18.82	0:00:53.11
29	3:07:06.53	+19:50:16.47	0:00:04.89	0:11:57.07

From Table **??**, the results show that error in α is decreasing where as error in δ are much random. After ploting the JPL postion and predicted postions over time, the results are as following:

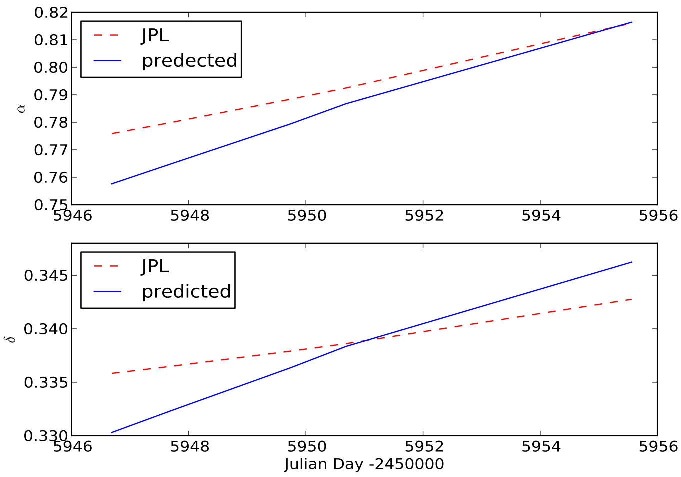

Figure 6: Predicted postion and JPL position vs. time

Here, it is clear that there is lot of variation in error. However, the asnwer is still quite close to the actual results. Here the angles was converted using the loops in python, see Appendix C for code.

6 Discussion

6.1 Saturated Data

6.2 Error in Orbital elements

6.3 Predictions

Appendices

A Calcualting ρ and r

```
# function for calculating rho
def rho_calc (k, RR, r,R,s,sd,sdd):
    rho = (k**2)*((1/RR**3)-(1/r**3))*(np.dot(sd,(np.cross(R,s))))
          /(np.dot(sd,(np.cross(sdd,s))))
    return rho

# function for calculating r
def r_calc (rho, RR, R, s ):
    r = np.sqrt((rho**2)+(RR**2) + (2*rho*(np.dot(R,s))))
    return r

# this is used to get the values for rho and r
rho = 0.1 # initial value
r = 0.1 # initial value

m = 0
rlist = np.zeros(101)
rlist[0] = r
rholist = np.zeros(101)
rholist[0] = rho
while m < 100:
    r = r_calc (rho, RR, R, s )
    rlist[m+1] = r
    rho = rho_calc (k, RR, r,R,s,sd,sdd)
    rholist[m+1] = rho
    m+=1
```

This piece of code was used to calculate the r and ρ value by iterating over 100 increments to get a convergence. This failed for the position values obtained from the data set but worked for JPL positions.

B Predicting the position at given time

```
timehold = [2455946.68646, 2455947.69476, 2455949.73866,
            2455950.68528, 2455955.56062]
r_vec = []
r_eq = []
RA = [] # this will store right accession values
DEC = [] # this will store declination values
for time in timehold:
    holdindex = np.nonzero(t<=time) # store all the index
                                    #from array t,until the element
                                    #value is less than or equal to time
    MAnomaly = n*(time - tau)
    EAnomaly = E[holdindex[0][-1]]
    trueAnomaly = find_v(e,EAnomaly)
    theta_predict = trueAnomaly+w
    r_predict = a*(1-e*np.cos(EAnomaly))
    rvec = np.matrix([(r_predict*np.cos(theta_predict)),
                     (r_predict*np.sin(theta_predict)),
                     0])
    recliptic = (TzTx*rvec.T)
    Te = np.matrix([[1,0,0],
                   [0, np.cos(epislon), -np.sin(epislon)],
                   [0, np.sin(epislon), np.cos(epislon)]])
    requatorial = np.array((Te*recliptic).T)
    recliptic = np.array(recliptic.T)

    Rvec = np.array((Te*np.matrix(R).T).T)
    rho_s = recliptic - R
    rho_s2 = Te*np.matrix(rho_s).T
    rho_s = np.array(rho_s2.T)
    rho = np.linalg.norm(rho_s)

    r_a = np.arctan(rho_s[0][1]/rho_s[0][0])
    dec = np.arcsin(rho_s[0][2]/rho)
    RA.append(r_a)
    DEC.append(dec)
    r_vec.append(rvec)
```

The method is straight forward. Except obtaining the value for EAnomaly. This was done by finding the index of where that time occurs in array t (see the code for holdindex). And then find the corresponding Eccentric Anomaly using the last value of holdindex.

C Converting time to radian and visevera

This loop was used to convert standard position to radians

```
ra = []
dec = []
for i in list:
    r = 15*(float(i[0]) + float(i[1])/60.
        + float(i[2])/3600.)*np.pi/180
    d = (float(i[3]) + float(i[4])/60.
        + float(i[5])/3600.)*np.pi/180
    ra.append(r)
    dec.append(d)
```

This loop was used to convert radians to standard position

```
ra_new = []
dec_new = []
j = 0
while j < len(RA):
    a1 = RA[j]*180/(np.pi*15)
    a2 = a1%1
    a3 = a2*60%1
    a4 = a3*60
    alpha = [a1-a2,a2*60-a3,a4]

    d1 = DEC[j]*180/np.pi
    d2 = d1%1
    d3 = d2*60%1
    d4 = d3*60
    delta = [d1-d2,d2*60-d3,d4]

    ra_new.append(alpha)
    dec_new.append(delta)
    j+=1
```