Determining the Right Ascension and Declination of Mars

Raees Khan

November 11, 2021

Abstract

In this assignment I want to develop an algorithm that calculates the Right Ascension and Declination of Mars as a function of the date. Results obtained from this algorithm will be compared to the actual values provided in the Mars01.dat file to see how good the fit is. The values of various parameters used will be reported.

Contents

1 Preliminaries

The central goal of this assignment is to calculate the Right Ascension and Declination of Mars as a function of the date. However before I jump into outlining the algorithm which will allow me to do so, let me note some important facts.

(i) I'll be using the current standard epoch $J2000$ e.g. my orbital parameters will be tabulated w.r.t to this epoch.

(ii) I'll be using the following form of Kepler's equation

$$M = E - esin(E)$$

Where M is the Mean Anomaly, E is the eccentric anomaly and e is the eccentricity of the orbit.

(iii) Once I know the eccentric anomaly, the cartesian coordinates in the plane of the orbit can be found as

$$x = a(cos(E) - e)$$
$$y = asin(E)\sqrt{1 - e^2}$$
$$z = 0$$

Where a is the semimajor axis

(iv) During my discussion with Dr.Boudreau in office hours, he mentioned that the Euler angles should rotate both Mars and Earth onto the equatorial plane. However the values I found only rotate me onto the ecliptic plane and I have to take into account the final rotation from the ecliptic to the equatorial plane.

(v) I'll be looking up values from authoritative sources instead of doing a curve fit. This shouldn't be too hard in my discussion with Rui, using the curve fit library in Python, so maybe it is a good exercise to practice later.

2 Algorithm

2.1 Ecliptic coordinates of Mars

The first step necessary to begin implementation of the algorithm is for the user to input the date for which they would like to calculate the RA and DE of Mars. As an example March 27, 2015 is a valid input.

Once this has been provided, the first step of the algorithm is to calculate the parameter d, which is the number of days since the epoch January 1, 2000 to the provided date.

The next step is to calculate the Mean anomaly for Mars. This is given as

$$M_{mars} = M_{0,mars} + (n_{mars})(d)$$

Where n_{mars} is the mean daily motion of mars and $M_{0,mars}$ is the mean anomaly at the epoch, which is defined as

$$M_{0,mars} = L_{mars} - p_{mars}$$

As stated in [1], these represent the following

(i) L_{mars}, mean longitude of Mars at the epoch

(ii) p_{mars}, longitude of perihelion of Mars at the epoch

These values are tabulated for Mars [2]. Plugging in yields

$$M_{mars} = (0.5240613)(d) + 355.45332 - 336.04084$$

$$M_{mars} = 0.5240613 * d + 19.41248$$

However M_{mars} might exceed 360° and hence I mod it to get the correct value

$$M_{mars} = (0.5240613d + 19.41248)\%360$$

Finally the above value must be converted to radians before proceeding.

Once d has been specified, and M_{mars} has been calculated, one can directly use Kepler's equation

$$M_{mars} = E_{mars} - e_{mars}sin(E_{mars})$$

to calculate the eccentric anomaly E_{mars} for mars and with this value in hand, one can find

$$x_{mars} = a_{mars}(cos(E_{mars}) - e_{mars})$$
$$y_{mars} = a_{mars} sin(E_{mars})\sqrt{1 - e_{mars}^2}$$
$$z_{mars} = 0$$

Through these values one can define the position vector of mars in its orbit

$$\vec{r}_{mars} = [x_{mars}, y_{mars}, z_{mars}]$$

The next step is crucial and that is to rotate this position vector onto the ecliptic plane. This can be done once the Euler angles for this rotation are known. These are specified as follows

$$\phi_m = 49.5574^\circ$$
$$\theta_m = 1.8497^\circ$$
$$\psi_m = 286.5016^\circ$$

I obtained these values from the tabulated values of longitude of the ascending node, inclination, and argument of periapsis at [2] and [4].

Doing the standard ZXZ Euler rotation I get

$$\vec{r}_{mars\ ecliptic} = R_z(\phi_m)R_x(\theta_m)R_z(\psi_m)\vec{r}_{mars}$$

Where R_z and R_x are the respective rotation matrices along the z, x axis in 3 dimensions.

2.2 Ecliptic coordinates of Earth

The ecliptic coordinates of Mars obtained in the previous section are unfortunately heliocentric and to calculate the RA and DE values, one must first convert these into a geocentric reference frame. In this regard, the procedure is to calculate the heliocentric ecliptic coordinates of the Earth and to subtract them from the heliocentric ecliptic coordinates of Mars to get the geocentric ecliptic coordinates of Mars.

The first step is then to calculate the heliocentric ecliptic coordinates for the Earth, which is detailed as follows.

Using the parameter d specified previously , calculate the mean anomaly for the Earth. This is given as

$$M_{earth} = M_{0,earth} + (n_{earth})(d)$$

Where n_{earth} is the mean daily motion of earth and $M_{0,earth}$ is the mean anomaly at the epoch, which is defined as

$$M_{0,earth} = L_{earth} - p_{earth}$$

As stated in [1], these represent the following

(i) L_{earth}, mean longitude of Earth at the epoch

(ii) p_{earth}, longitude of perihelion of Earth at the epoch

These values are tabulated for Earth [3]. Plugging in yields

$$M_{earth} = (0.9856)(d) + 100.46435 - 102.94719$$

$$M_{earth} = (0.9856 * d - 2.48284)\%360$$

Once M_{earth} has been calculated, one can directly use the relation

$$M_{earth} = E_{earth} - e_{earth} sin(E_{earth})$$

to calculate the eccentric anomaly E_{earth} for Earth and with this value in hand, one can find

$$x_{earth} = a_{earth}(cos(E_{earth}) - e_{earth})$$
$$y_{earth} = a_{earth} sin(E_{earth})\sqrt{1 - e_{earth}^2}$$
$$z_{earth} = 0$$

Through these values one can define the position vector of Earth in its orbit

$$\vec{r}_{earth} = [x_{earth}, y_{earth}, z_{earth}]$$

Now I want to rotate onto the ecliptic plane. I can do this using the Euler angles for this rotation for earth tabulated at [3] and [4]

$$\phi_e = 18.272^\circ$$
$$\theta_e = 0^\circ$$
$$\psi_e = 85.901^\circ$$

Doing the standard ZXZ Euler rotation I get

$$\vec{r}_{earth\ ecliptic} = R_z(\phi_e)R_x(\theta_e)R_z(\psi_e)\vec{r}_{earth}$$

2.3 Calculating Right Ascension and Declination

So far I have specified how to obtain the following position vectors

$$\vec{r}_{mars\ ecliptic}$$

$$\vec{r}_{earth\ ecliptic}$$

These are the Heliocentric positions for the Earth and Mars in the ecliptic plane. Subtracting I can get the geocentric position for Mars in the ecliptic plane

$$\vec{r}_{mars\ ecliptic\ geocentric} = \vec{r}_{mars\ ecliptic} - \vec{r}_{earth\ ecliptic}$$

The final step before calculating the RA and DE is to rotate from the ecliptic plane to the equatorial plane. This corresponds to an x rotation by the axial tilt of the earth which is 23.4393°.

$$\vec{r}_{mars\ equatorial\ geocentric} = R_x(23.4393^\circ)\vec{r}_{mars\ ecliptic\ geocentric}$$

Now let me denote the components of $\vec{r}_{mars\ equatorial\ geocentric}$ as x, y, z. Then using basic geometry, RA and DE is given as [1]

$$\boxed{RA = tan^{-1}\left(\frac{y}{x}\right)}$$

$$\boxed{DE = 90 - tan^{-1}\left(\frac{z}{\sqrt{x^2+y^2}}\right)}$$

3 Mars01.dat

Before implementing the Algorithm, it seems like a good idea to parse the Mars01.dat file and obtain the relevant information. This includes the following lists:

(i) Dates of observations

(ii) Observed values of RA

[1] The DE calculation takes the (90 - stuff) form since I want DE = 0 at the north pole.

(iii) Observed values of DE

To obtain these I converted the given file to a .txt format and eliminated the heading and ran the following script

```python
#Packages used
import numpy as np
import matplotlib.pyplot as plt
from numpy import cos,sin,arccos,sqrt
from scipy.optimize import fsolve
from datetime import date


file = open("Mars01.txt", "r")
data = file.readlines() #Reading in the file
file.close()

ra_obs = [] #List to store all observed Right Ascension Values
de_obs = [] #List to store all observed Declination Values
dates = [] #List to store dates at which values were observed

for val in data:

    time = (val[1:13]).replace(" ","") #Extracting the date

    ra_h = float(val[23:25]) #Extracting the RA by hours,minutes,seconds
    ra_m = float(val[26:28])
    ra_s = float(val[29:34])

    de_sign = float(str(val[35])+str(1))
    de_de = float(val[36:38]) #Extracting the DE by degrees,minutes,seconds
    de_m = float(val[39:41])
    de_s = float(val[42:46])

    phi = (ra_h + ra_m/60 + ra_s/3600)*15 #Converting RA to azimuthal angle between 0 and 360


    if(de_sign != -1):
        theta = 90.0 - (de_de + (60*de_m+de_s)/3600) #Converting DE to polar angle between 0 and 180
    else:
        theta = 90.0 - (-de_de - (60*de_m+de_s)/3600)

    ra_obs.append(phi) #Adding values to my list for each date
    de_obs.append(theta)
    dates.append(time)
```

Figure 1: Parsing Script

4 Implementing the Algorithm

4.1 Useful Functions

I'll use Python to implement the devised algorithm. Let me define some important functions and parameters to this end

```
def deg(rad): #Function to convert radians to degrees
    return rad*180/np.pi

def rad(deg):#Function to convert degrees to radians
    return deg*np.pi/180

def sub_vec(r1,r2): #Function to subtract two vectors
    x = r1[0]-r2[0]
    y = r1[1]-r2[1]
    z = r1[2]-r2[2]

    r=np.array([x,y,z])

    return r


def day(val): #Function to return number of days 'd' since Epoch
    Y,M,D = val
    d0 = date(2000, 1, 1)
    d1 = date(Y,M,D)
    delta = d1 - d0
    return (delta.days)


def mean_anomaly_mars(d): #Function to compute mean anomaly of Mars
    return rad((19.41248 + 0.5240207766*d)%360)

def mean_anomaly_earth(d): #Function to compute mean anomaly of Earth
    return rad((-2.48284 + 0.9856002585*d)%360)

def E_A(data): #Function to calculate Eccentric anomaly
    root = fsolve(func, 1,args = data)
    return root[0]
```

Figure 2: Useful Functions

```
def func(x,*data): #Function used in calculation of Eccentric anomaly
    e,M = data
    return (x-e*np.sin(x)-M)

def position(a,e,E): #Function to calculate position vector in orbit
    x = (a)*(cos(E)-e)
    y = (a)*(sin(E)) * (sqrt(1 - e**2))
    z = 0

    r = np.array([x,y,z])
    return r


def Rx(angle): #Rotation matrix around x
    return np.matrix([[ 1, 0           , 0           ],
                      [ 0, np.cos(angle),-np.sin(angle)],
                      [ 0, np.sin(angle), np.cos(angle)]])


def Rz(angle): #Rotation matrix around z
    return np.matrix([[ np.cos(angle), -np.sin(angle), 0 ],
                      [ np.sin(angle), np.cos(angle)  , 0 ],
                      [ 0            , 0              , 1 ]])
```

Figure 3: Useful Functions

```
def euler_rotation(phi,theta,psi): #Euler rotation matrix

    R = np.matmul(np.matmul(Rz(phi),Rx(theta)),Rz(psi))
    return R

def equator_rotation(): #Rotation matrix to bring vector to equator
    obliquity = rad(23.4393)
    return Rx(obliquity)


def calculate_RA_DE(r): #Function which calculates RA and DE
    x = r[0]
    y = r[1]
    z = r[2]

    RA = deg(np.arctan2(y,x))

    if(RA<0):
        RA = RA+360
    DEC = 90 - deg(np.arctan2(z,sqrt(x**2+y**2)))
    return (RA,DEC)

def matrix_prod(R,r):
    prod = np.asarray(np.matmul(R,r)).reshape(-1)
    return prod
```

Figure 4: Useful Functions

```
e_earth = 0.0017 #Earth Parameters
a_earth = 1
earth_phi = rad(18.272)
earth_theta = rad(0)
earth_psi = rad(85.901)

e_mars = 0.093 #Mars parameters
a_mars = 1.52371
mars_phi = rad(49.57854)
mars_theta = rad(1.85061)
mars_psi = rad(286.231)


#Dictionary helpful when calculating days since Epoch
months = {"Jan": 1, "Feb":2, "Mar":3, "Apr":4,
          "May": 5, "Jun":6,"Jul": 7, "Aug":8,
          "Sep":9, "Oct":10, "Nov": 11, "Dec":12}
```

Figure 5: Orbital Parameters

4.2 Main Calculation

I'll implement the algorithm described for each day in the list of dates obtained from the Mars01.dat file. This is shown as follows

```
for ele in dates: #Looping over dates

    Y = int(ele[0:4]) #Extracting Year,Month,Date
    M = months[ele[5:8]]
    D = int(ele[9:11])

    d = day((Y,M,D)) #Calculating days since Epoch

    M_mars = mean_anomaly_mars(d) #Calculating Mean anomaly for Mars
    E_mars = E_A((e_mars,M_mars)) #Calculating Eccentric anomaly for Mars
    pos_mars = position(a_mars,e_mars,E_mars) #Calculating orbit position vector
    R_euler_mars = euler_rotation(mars_phi,mars_theta,mars_psi) #Defining euler rotation
    eclipitic_pos_mars = matrix_prod(R_euler_mars,pos_mars) #Applying euler rotation


    M_earth = mean_anomaly_earth(d) #Calculating Mean anomaly for Earth
    E_earth = E_A((e_earth,M_earth)) #Calculating Eccentric anomaly for Earth
    pos_earth = position(a_earth,e_earth,E_earth) #Calculating orbit position vector
    R_euler_earth = euler_rotation(earth_phi,earth_theta,earth_psi) #Defining euler rotation
    eclipitic_pos_earth = matrix_prod(R_euler_earth,pos_earth) #Applying euler rotation


    r_eclip = sub_vec(eclipitic_pos_mars,eclipitic_pos_earth) #FInding geo-centric position of Mars
    R_equat = equator_rotation() #Defining rotation from ellipitic to equatorial plane
    r_equat = matrix_prod(R_equat,r_eclip) #Applying the rotation


    RA,DE = calculate_RA_DE(r_equat) #Calculating RA and DE
    ra_alg.append(RA) #Appending RA and DE to respective lists-
    de_alg.append(DE)
```

Figure 6: Algorithm Implementation

5 Results

5.1 Code

To interpret the results I'll plot them, the code for this is given as follows

```
#Plotting results

time_scale = list(range(len(data)))

plt.style.use('dark_background')
plt.scatter(time_scale,ra_obs,color="r", label = "Actual data given")
plt.plot(time_scale[0:50],ra_alg[0:50],color = 'b', label = 'Prediction from algorithm')
plt.plot(time_scale[50:],ra_alg[50:],color = 'b')
plt.grid()
plt.legend(loc = 'upper center',fontsize='xx-large')
plt.title("Right Ascension vs Time", fontsize="xx-large")
plt.xlabel("Time",fontsize= 'xx-large')
plt.ylabel("Right Ascension",fontsize= 'xx-large')
plt.show()

plt.style.use('dark_background')
plt.scatter(time_scale,de_obs,color="r", label = "Actual data given")
plt.plot(time_scale,de_alg,color = 'b', label = 'Prediction from algorithm')
plt.grid()
plt.legend(fontsize= 'xx-large')
plt.xlabel("Time",fontsize= 'xx-large')
plt.ylabel("Declination",fontsize= 'xx-large')
plt.title("Delicnation vs Time", fontsize="xx-large")
plt.show()
```

Figure 7: Code to Plot

5.2 Plots

I obtain the following plots:

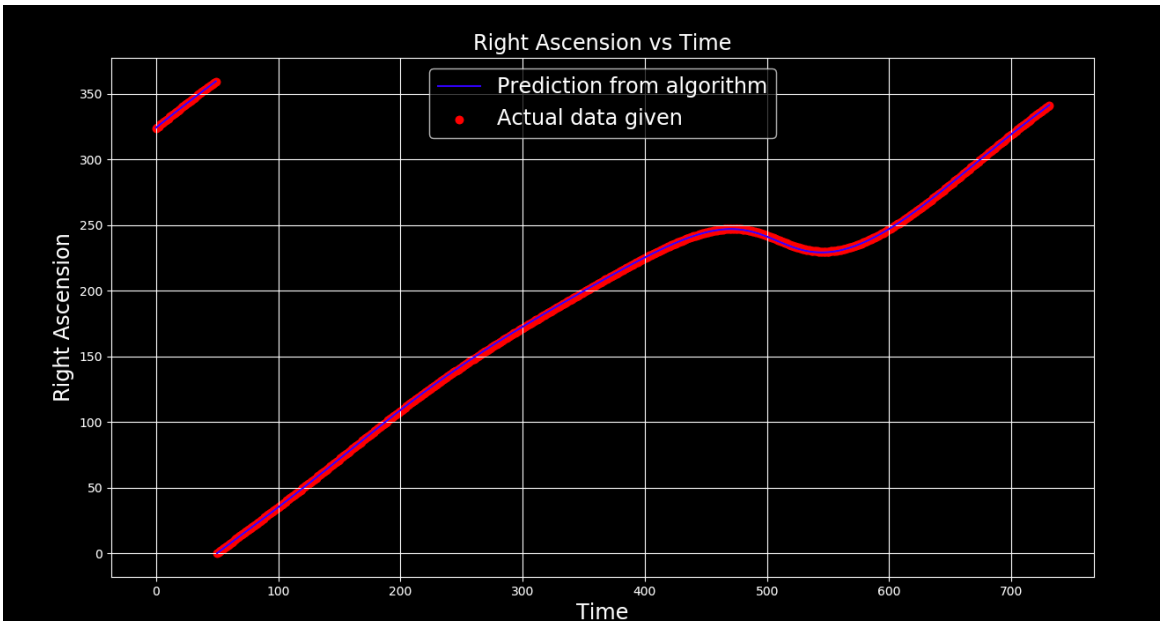

Figure 8: Right Ascension vs Time

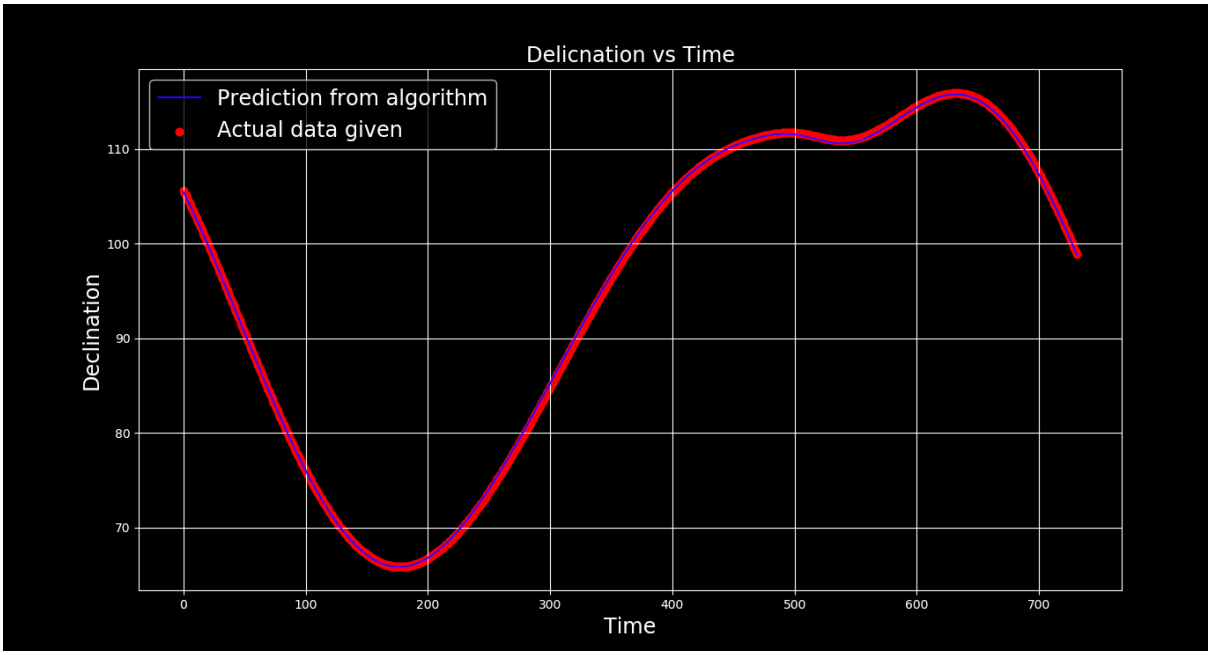

Figure 9: Declination vs Time

5.3 Interpreting the Results

The fit seems to be reasonably well looking at the plots. The values of parameters giving this fit are

$$axial\_tilt = 23.4393^\circ$$
$$a_e = 1.0000$$
$$a_m = 1.52371$$
$$e_e = 0.0017$$
$$e_m = 0.093$$
$$\phi_m = 49.5574^\circ$$
$$\theta_m = 1.8497^\circ$$
$$\psi_m = 286.5016^\circ$$
$$\phi_e = 18.272^\circ$$
$$\theta_e = 0^\circ$$
$$\psi_e = 85.901^\circ$$

I can also calculate the average percentage error to get a quantitative sense of the fit

Computing on Python I get

```
#Lists to append percent error to
RA_perc_error_list = []
DE_perc_error_list = []

for i in range(len(ra_alg)):
    #calculating percent error
    RA_perc_error = 100*(np.abs(ra_obs[i]-ra_alg[i]))/ra_alg[i]
    DE_perc_error = 100*(np.abs(de_obs[i]-de_alg[i]))/de_alg[i]

    #appending percent error to lists for all points
    RA_perc_error_list.append(RA_perc_error)
    DE_perc_error_list.append(DE_perc_error)

#computing avg percent error
mean_RA_perc_error = np.mean(RA_perc_error_list)
mean_DE_perc_error = np.mean(DE_perc_error_list)

print(mean_RA_perc_error,mean_DE_perc_error)
```

Figure 10: Error Calculation

I get 0.35% error on average in RA values and 0.16% error on average in DE values. This seems reasonable.

6 Further Analysis

I also tested my algorithm on Jupiter with observational data taken from [5] and orbital parameters taken from [4] and [6].

To get the plots to match up correctly, I noted all the tabulated values , but rounded the time period up to 12 years from 11.86 years in calculating the daily motion. I'm not exactly sure why there is this discrepancy, thus further investigation is needed.

Results are attached below

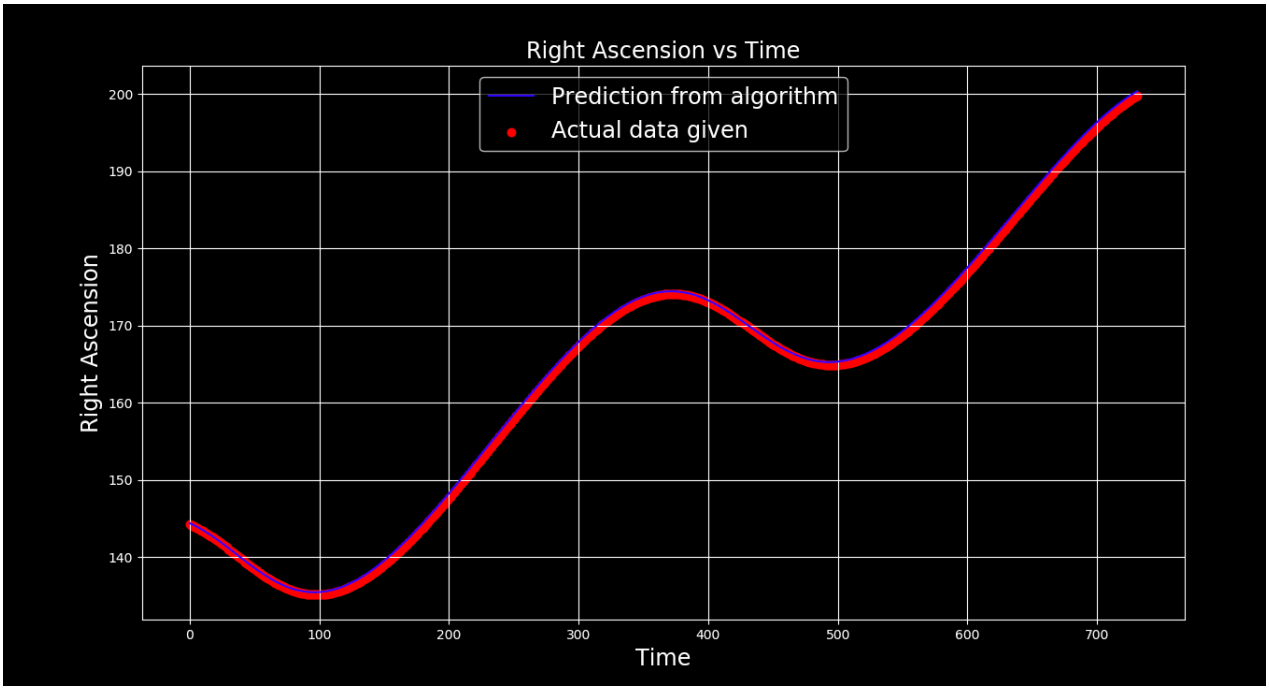

Figure 11: Right Ascension vs Time

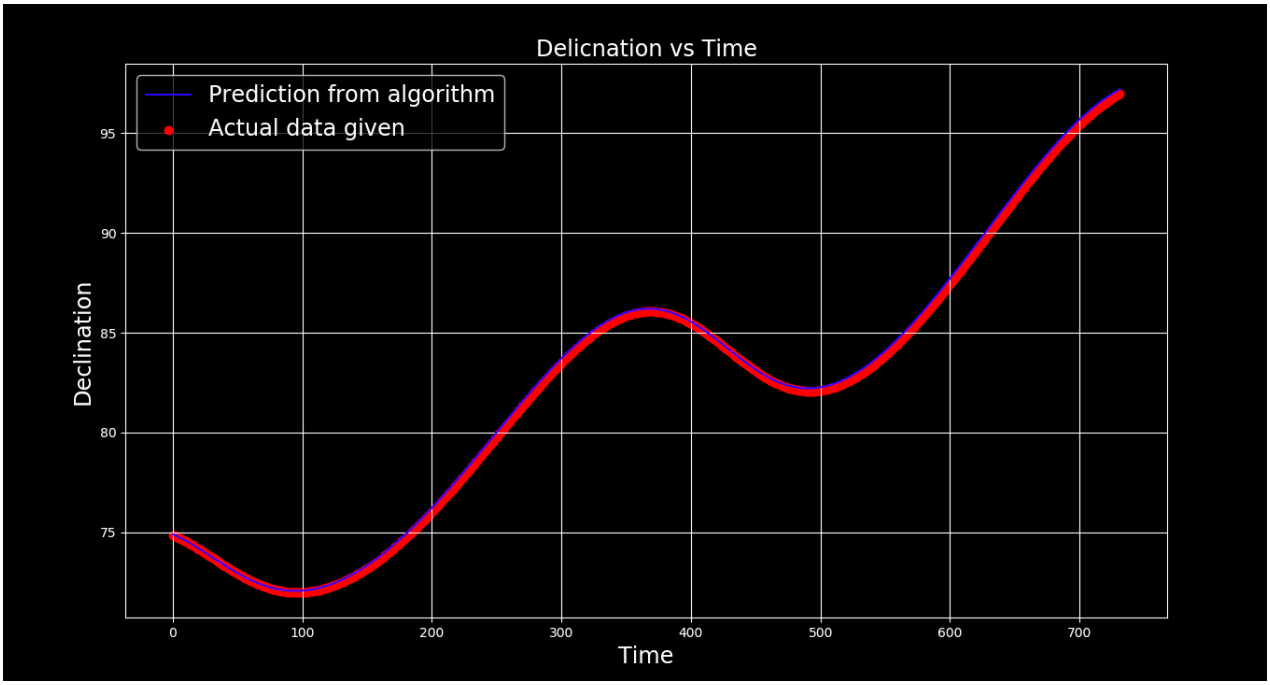

Figure 12: Declination vs Time

References

[1] https://en.wikipedia.org/wiki/Mean_anomaly

[2] https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html

[3] https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

[4] https://simple.wikipedia.org/wiki/Argument_of_periapsis

[5] https://ssd.jpl.nasa.gov/horizons/app.html#/

[6] https://nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html