Assignment 2

Mars orbit

Submitted By: Nirmalya Gayen, Sr. No. 19464

Initial Steps:

1. **Get time as days**: the function **get_times(data)** is used to convert, $[\text{'}year\text{'}, \text{'}month\text{'}, \text{'}day\text{'}, \text{'}hour\text{'}, \text{'}minute\text{'}]$ as days. The 1st date is considered as 0
2. **Convert oppositions to degrees**: from given $[\text{'}zodiacIndex\text{'}, \text{'}degree\text{'}, \text{'}minute\text{'}, \text{'}second\text{'}]$, converted it to degrees and every zodiacIndex covers 30 degrees and 1st zodiacIndex is 0

Q1. MarsEquantModel(c, r, e1, e2, z, s, times, oppositions)

1. **Centre**: we have the angle 'c' of centre from sun, and the distance is 1. We calculate c_x and c_y from this.
2. **Equant**: we have e1: the distance of equant from sun and e2: the angle between 'aries', 'sun' and 'equant'. So, we calculate e_x and e_y with the angle $(e2 + z)$ as z is the angle from 'aries' for equant 0.
3. **Observation Offset Angle (d)**: we convert the days of observation to angle using angular velocity 's' of mars. And add 'z' because 'z' is the angle between 'aries' and 1st sample day of mars.
4. **Get intersection point of prediction line on circle**: using the formulas:
$$(x - c_x)^2 + \left(y - C_y\right)^2 = r^2$$
$$\left(y - e_y\right) = (x - e_x) * \tan(d)$$
solving the equations we get $(x_1, y_1), (x_2, y_2)$ we get the correct point using coordinate of d. if d is in 1st or 4th coordinate we select positive x and corresponding y value or else do otherwise.
5. **Get the angle**: using $\tan^{-1}\frac{y}{x}$ we get the angle (calculated angle) and we subtract this with given angle.
6. **Max Error**: max error is the absolute max error for all 12 data points.

Q2. bestOrbitInnerParams(r, s, times, oppositions)

1. **Finding $\boldsymbol{c, e1, e2, z}$**: using discretised exhaustive search (with given r, s, times, oppositions) using **MarsEquantModel** maxError and choose the minimum error parameters.
2. **Minimize**: then we are calling scipy.minimize with maxError from Q1

Q3. bestS(r, times, oppositions)

1. **Discretise s in range of $\boldsymbol{(360\ /\ 686, 360\ /\ 688)}$:** then call the **MarsEquantModel** and similarly choose the minimum error 's'
2. **Continue:** continue the same with the neighbours of best 's'

Q4. bestR(s, times, oppositions)

1. **Take initial 'r'**: take initial 'r' as some value between 1 – 10 (7)
   a. **Find Error**: find error for the selected 'r' and store it
   b. **Get new 'r'**: get new 'r' by getting all the distances of intersection points and getting average of them.
   c. **Exit loop**: exit loop after some iteration (10) or error range exceeding (7-9)
2. **Update initial 'r'**: update initial 'r' by 0.15
3. **Exit loop**: exit loop after some fixed iteration number (6)

Q5. bestMarsOrbitParams(times, oppositions)

1. **Initialize**: $s = 360 / 687$
2. **Until error is less than 4/60 degrees**: we do the update of variables until we get acceptable maxError.
   a. **Get Best 'r'**: using the function of Q4 we get best 'r' and send the initialized 's' value.
   b. **Get Best 's'**: using the function in Q3 we get best 's' and send the 'r' value we get in previous line.
3. **Get other values**: get other parameters (c, e1, e2, z) from **'bestOrbitInnerParams'**. And return the 'errors' and 'maxError' with it.

Output:

Fit parameters: r = 8.2000, s = 0.5241, c = 148.9097, e1 = 1.5231, e2 = 92.9734, z = 55.8350

The maximum angular error = 0.0464