Mohit Kumar
SR No. 04-01-03-10-51-21-1-19825
MTech Artificial Intelligence

Assignment 2: Mars Orbit

---

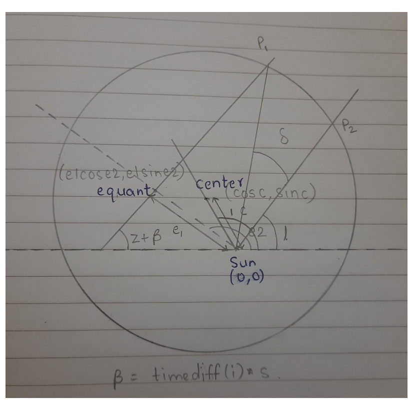
python packages used

1. pandas
2. numpy

Mars Orbit Model Assumptions

1. The Sun is at the origin.
2. Mars's orbit is circular, with the centre at a distance 1 unit from the Sun and at an angle c (degrees) from the Sun-Aries reference line, $(\cos c, \sin c)$ in cartesian coordinates.
3. Mars's orbit has radius r (in units of the Sun-centre distance).
4. The equant is located at (e1, e2) in polar coordinates with centre taken to be the Sun, where e1 is the distance from the Sun and e2 is the angle in degrees with respect to the Sun-Aries reference line, $(e_1 \cos e_2, e_1 \sin e_2)$ in cartesian coordinates.
5. The 'equant 0' angle z (degrees) which is taken as the earliest opposition, also taken as the reference time zero, with respect to the equant-Aries line (a line parallel to the Sun-Aries line since Aries is at infinity).
6. The angular velocity of Mars around the equant is s degrees per day.

Calculation of angular error(δ)



The equation of the Mars circular orbit(assumed) centre at $(\cos c, \sin c)$ is given by:

$$(x - \cos c)^2 + (y - \sin c)^2 = r^2 \quad or \quad x^2 + y^2 - 2x\cos c - 2y\sin c + 1 - r^2 = 0 \tag{1}$$

Let the equation of the line representing the ith opposition be given by:

$$y = m_1 x + c_1 \qquad or \qquad y = \tan(z+\beta) * x + c_1 \tag{2}$$

where $\beta = timediff(i) * s$
Now, since the equant lies on this line, it's coordinates should satisfy 1.i.e.
$e_1 \sin e_2 = \tan(z+\beta) * e_1 \cos e_2 + c_1$
where z is the zero of the equant longitudes, s is the angular velocity of Mars around the equant in degrees per day, timediff(i) is the time difference in days between the ith opposition and the zeroth opposition.

Therefore we get, $c_1 = e_1 \sin e_2 - \tan(z+\beta) * e_1 \cos e_2$
To get the point of intersection of this line with the orbit, P_1. We substitute $y = \tan(z+\beta) * x + c_1$ in the equation of the Mars orbit.
$x^2 + (\tan(z+\beta) * x + c_1)^2 - 2x \cos c - 2(\tan(z+\beta) * x + c_1) \sin c + 1 - r^2 = 0$
We get a quadratic equation ($ax^2 + bx + c = 0$),

$$x^2(1+\tan^2(z+\beta)) + x(2c_1 \tan(z+\beta) - 2\cos c - 2\tan(z+\beta)\sin c) + (c_1^2 - 2c_1 \sin c + 1 - r^2) = 0 \tag{3}$$

with

$$\begin{aligned} a &= 1 + \tan^2(z+\beta) \\ b &= 2c_1 \tan(z+\beta) - 2\cos c - 2\tan(z+\beta)\sin c \\ c &= c_1^2 - 2c_1 \sin c + 1 - r^2 \end{aligned}$$

Solving the above quadratic equation we get the coordinates of the point P_1. The angle $(l+\delta)$ is $\tan^{-1}\frac{y}{x}$. δ therefore comes out to be:

$$\delta = \tan^{-1}\frac{y}{x} - l \tag{4}$$

We get two roots of the above quadratic equation since a circle and a line can intersect at atmost 2 points. We get two angular errors δ_1 and δ_2, and we take the smaller of the two as our angular error δ.

Implementation details

1. The get_times function computes the time difference in days between consecutive oppositions.
2. The get_oppositions function computes the longitude angles for each opposition.
3. The compute_delta function takes the point of intersection and longitude angle as input and computes the angular error *delta*.
4. The function MarsEquantModel takes c,r,e1,e2,z,s,times,oppositions as input and returns the angular error for each opposition and the maximum angular error.
5. The function bestOrbitInnerParams takes a fixed r and s as input along with times and oppositions and does a discretised exhaustive search over c, e = (e1,e2), and z to minimise the maximum angular error for the given r and s. It outputs the best c,e1,e2,z, the angular error for each opposition, and the maximum angular error.

6. The function bestS takes a fixed r as input along with times and oppositions and does a discretised search for s to minimise the maximum angular error for the given r. It outputs the best s the angular error for each opposition, and the maximum angular error.

7. The function bestR takes a fixed s as input along with times and oppositions and does a discretised search for r to minimise the maximum angular error for the given s. It outputs the best r the angular error for each opposition, and the maximum angular error.

8. The function bestMarsOrbitParams takes times and oppositions as input and does a discretised search for r and s to minimise the maximum angular error. It internally uses the bestOrbitInnerParams function. It outputs the best r,s,c,e1,e2,z the angular error for each opposition, and the maximum angular error.

Results The best fit parameters of the assumed Mars orbit were found to be $r = \mathbf{8.8}$, $s = \mathbf{0.524}$, $c = \mathbf{146.2}$, $e_1 = \mathbf{1.7}$, $e_2 = \mathbf{148.5}$, $z = \mathbf{56.2}$. The errors for each opposition were found to be(in degrees) [**0.683, 0.508, 0.223, 0.106, 0.280, 0.236, 0.262, 0.410, 0.265, 0.014, 0.329, 0.581**]. The maximum opposition error was found to be **0.683 degrees**.